1. **Ising Model**
We would like to apply the Metropolis algorithm to simulate the 1D Ising chain model: N magnetic dipoles fixed on a chain. Write a program following the Metropolis algorithm. A code skeleton is available in the code section of the homepage.
   (a) Use a *warm* start configuration (all spins are set randomly) as the initial condition. Set the number of dipoles to $N = 20$.
   (b) Pick one dipole randomly and flip its spin to create the new *trial* configuration. Calculate the energy of both the old and the trial configuration, E_i and E_{tr}, by using
$$E = -J \sum_{i=1}^{N} s_i s_{i+1}. \tag{1}$$
We will set $J = 1$ to fix the energy scale. Use a periodic boundary condition for your chain.
   (c) Calculate the relative probability for the trial configuration following
$$\mathcal{P} = e^{-\Delta E/kT}, \qquad \Delta E = E_{\mathrm{tr}} - E_i. \tag{2}$$
Set $kT = 0.1$ representing the low temperature case.
   (d) The trial configuration is accepted if it satisfies the condition $\mathcal{P} \geq r_j$, where r_j is a random number in the range $[0, 1]$. Convince yourself that this single condition corresponds to the Metropolis algorithm described in the lecture.
   (e) Continue your calculation for $10N$ steps and observe your result. How is the alignment of spins? Does it change with different random start parameters?
   (f) Using now $100N$ steps, try $kT = 0.01$, $kT = 0.1$, and $kT = 1$.
   (g) Try also a *cold* start where initial spins are aligned (in this case you either have to do lots of steps or increase kT).
   (h) Extend your program to the 2D model.
      1. How many steps are necessary for the 2D model?
      2. Store the result every n steps. How do you choose n?
      3. Animate your results using the function `plotConfigurations` of `exercise_02.py` or save the frames to disk and join them to a movie with `frames_to_video.py`.
      4. Change the initial conditions and observe how your result changes.
      5. Increase kT in small steps. How does the spin alignment change?

2. **Monte Carlo Integration**
Compare the results of the different techniques for Monte Carlo (MC) integration for the following functions
$$f_1(x) = x^2 + x - 1, \quad x \in [-10, 10] \tag{3}$$
$$f_2(x) = \frac{\cos(x)\log(x)}{\sqrt{x}} \quad x \in [0, 1]. \tag{4}$$
As on the last sheet, calculate the integral by using both the rejection and the mean method and plot 1000 results (each with a different seed and 10000 sample points) for each function and each method into a histogram. Compare the mean of these histograms to the analytical values of the integrals. *(Compare the accuracy of the MC integration to that of the Simpson algorithm for the second function f_2. How many integration points should be used for the Simpson algorithm for a fair comparison?)*

3. **Monte Carlo Model of Evolution**
In [1], certain aspects of evolution are described with the help of a MC model. Read the article and verify its findings by implementing the introduced model yourself and compare your results to the results of the article.

   1. N species are arranged in 1D line.
   2. Assign a random barrier B_i in the range $[0, 1]$ to each species.
   3. Mutate the species with the lowest barrier by assigning a new barrier B_n for this species and its neighbouring species B_{n-1} and B_{n+1}.
   4. Repeat this mutation for each iteration.
   5. The real time of the mutation is proportional to $\exp(B_n/T_{\text{car}})$, were T_{car} is a characteristic time scale. (Use $T_{\text{car}} = 0.01$ as it is done by the author of the article.)
   6. Produce similar plots to those printed in the article.

References

[1] P. Bak and K. Sneppen, *Punctuated equilibrium and criticality in a simple model of evolution*, Phys. Rev. Lett. 71 (1993) 4083.