Spin some spins

- **Topic:** Metropolis Monte Carlo, Ising model, importance sampling, magnetism, phase transitions, critical points
- **Task:**
  1. Implement the Metropolis algorithm.
  2. Run a simulation at constant temperature below and above $T_C \approx 2.27$. Check that the system looks reasonable.
  3. Study how long does it take for the magnetization to stabilize. This is the thermalization period you need to wait before you start collecting data.
  4. Run a series of simulations at different temperatures and plot magnetization as a function of temperature. Can you estimate T_C?
- **Template:** ising.py
- **Further reading:**
  - https://en.wikipedia.org/wiki/Metropolis-Hastings_algorithm
  - https://en.wikipedia.org/wiki/Ising_model
  - https://en.wikipedia.org/wiki/Critical_phenomena
  - https://en.wikipedia.org/wiki/Markov_chain_Monte_Carlo

ising.py

`ising.`**`animate`**`(history)` [source]

Animate the simulation.

Parameters: **history** (*list*) – list of systems at different times

`ising.`**`calculate_Binder_cumulant`**`(magnetizations)` [source]

Calculates the Binder cumulant.

The transition from an ordered, ferromagnetic state to a disordered, paramagnetic state should be discontinuous at the thermodynamic limit, but in a finite system, the transition is gradual. This makes it difficult to pinpoint the critical temperature T_C from a finite simulation.

This can be done very precisely using the cumulant method. The quantity

$$U_4 = 1 - \frac{\langle M^4 \rangle}{3\langle M^2 \rangle^2},$$

known as the Binder cumulant, is also dependent on system size. At temperatures below T_C, it approaches 0 as the system size grows. At temperatures above T_C, it approaches 2/3 for the Ising model. At T_C, it has a non-trivial system dependent value, which is approximately independent of system size.

Therefore one can pinpoint the transition temperature by calculating the cumulant as a function of temperature at different system sizes and plotting the results. The cumulants will cross approximately at T_C.

Parameters: **magnetizations** (*array*) – sequence of magnetization values
Returns: the cumulant U_4
Return type: float

`ising.`**`calculate_flip_energy`**`(grid, i, j)` [source]

Calculates how much the total energy of the system would change, if the spin (i, j) was flipped.

The system is not changed in any way. The function only calculates how much the energy would change, if a spin was flipped.

The total energy of the 2D Ising model is

$$E = -\sum_{a,b} J_{a,b} s_a s_b,$$

where $a = (i, j)$ and $b = (k, l)$ are spin coordinates, s_a, s_b are the spin values at these coordinates (either 1 or -1), and $J_{a,b}$ is the coupling parameter between these two spins.

- If $J > 0$, the spins interact ferromagnetically, i.e., they prefer to point in the same direction.
- If $J < 0$, the spins interact antiferromagnetically and prefer to point in opposite directions.
- If $J = 0$, there is no interaction between the spins.

In our model, $J = 1$ for all spins which are directly next to each other (vertical and horizontal neighbors), and $J = 0$ for all other spins (in some suitable units). This means that if only one spin, $s_{i,j}$, changes its sign, there are exactly four terms in the energy sum that change: the terms with $s_{i,j}$ and its neighbors up, down, right and left, i.e., $s_{i+1,j}$, $s_{i-1,j}$, $s_{i,j+1}$, $s_{i,j-1}$. Furthermore they change by changing their sign. For instance, if $s_{i,j,\text{before}} = 1$ and $s_{i+1,j} = -1$, initially the energy of this pair is $-J_{(i,j),(i+1,j)} s_{i,j,\text{before}} s_{i+1,j} = 1$. After the flip, we would have $s_{i,j,\text{after}} = -1$, and the energy of the pair would become -1. The *change* in energy would therefore be

$$\Delta E_1 = 2J_{(i,j),(i+1,j)} s_{i,j,\text{before}} s_{i+1,j} = -2.$$

The energies for the three other affected pairs are calculated similarly, so substituting $J = 1$ we get the total energy change

$$\Delta E = 2s_{i,j,\text{before}}(s_{i+1,j} + s_{i-1,j} + s_{i,j+1} + s_{i,j-1}).$$

Parameters:
- **grid** (*array*) – the system as an integer array
- **i** (*int*) – row of the spin to flip
- **j** (*int*) – column of the spin to flip

Returns: energy change ΔE
Return type: int

`ising.`**`calculate_magnetic_moment`**`(grid)` [source]

Calculates the total magnetization of the system.

A magnetic moment is associated with electron spins, $\vec{\mu} = -g\frac{e}{2m_e}\vec{s}$. In the Ising model, spins only point either up or down and get values of +1 or −1, and so also the magnetic moment points only up or down. Furthermore, by choosing a unitless scale $g\frac{e}{2m_e} = 1$, the magnetic moment becomes the same as the spin (only reversed), and we can calculate the total magnetization as a sum over all spins in the system,

$$M = -\frac{1}{N^2}\sum_{i,j} s_{i,j}.$$

Here $N \times N$ is the number of spins in the system. Thus M is scaled so that it only gets values between −1 and 1. If $|M| = 1$, the magnetization is complete and all spins point in the same direction. If $M = 0$, there are equally many spins pointing up and down, and there is no macroscopic magnetic moment.

Parameters: **grid** (*array*) – the system as an integer array
Returns: magnetization absolute value $|M|$
Return type: float

`ising.`**`calculate_statistics`**`(samples)` [source]

Calculates and returns the sample mean, variance, standard deviation and error of mean for the given set of samples.

For a sequence of N values for a random variable X, the mean is estimated by the average

$$\mu_X = \langle X \rangle = \frac{1}{N}\sum_i X_i$$

and the variance by the sample variance

$$\sigma_X^2 = \frac{1}{N-1}\sum_i (X_i - \mu_X)^2.$$

Standard deviation is the square root of variance

$$\sigma_X = \sqrt{\sigma_X^2}$$

and the standard error of mean is

$$\Delta X = \frac{\sigma_X}{\sqrt{N}}.$$

Parameters: **samples** (*list*) – sequence of random results, X_i
Returns: mean, variance, standard deviation, error of mean
Return type: float, float, float, float

`ising.`**`draw`**`(frame, history)` [source]

Draws the system for animation.

Parameters:
- **frame** (*int*) – index of the frame to draw
- **history** (*list*) – list of systems at different times

`ising.`**`flip`**`(grid, i, j)` [source]

Flips the spin $s_{i,j}$ in the system.

Since spin values up and down are represented by values of +1 and −1, respectively, the function simply changes the sign of the given element in the array.

Parameters:
- **grid** (*array*) – the system as an integer array
- **i** (*int*) – row of the spin to flip
- **j** (*int*) – column of the spin to flip

`ising.`**`generate_grid`**`(lattice_size, randomly=False)` [source]

Creates the system as $N \times N$ array.

Each value in the array represents a spin with two possible values, up and down. The value up is represented by the value 1 and the value down by -1.

Parameters:
- **lattice_size** (*int*) – system size N
- **randomly** (*bool*) – If True, spins will be randomly oriented. Otherwise all spins point up.

`ising.`**`main`**`()` [source]

Main program.

`ising.`**`metropolis_step`**`(grid, temperature)` [source]

Tries to flip one randomly chosen spin using the Metropolis algorithm.

The flip is either accepted or discarded according to a Boltzmann-like probability:

- If the spin flip leads to *lower* total energy, it is always accepted.
- If the spin flip leads to *higher* total energy, the new configuration is accepted with probability $P = e^{-\frac{\Delta E}{k_B T}}$, where ΔE is the energy change, k_B is the Boltzmann constant and T is the temperature.

If the spin flip is accepted, the change is applied to the given system.

The function uses a unitless temperature scale with $k_B = 1$.

Note
This function is incomplete!

Parameters:
- **grid** (*array*) – the system as an integer array
- **temperature** (*float*) – temperature

`ising.`**`print_progress`**`(step, total)` [source]

Prints a progress bar.

Parameters:
- **step** (*int*) – progress counter
- **total** (*int*) – counter at completion

`ising.`**`run_cumulant_series`**`(n_steps=10000, thermalization_steps=1000, lattice_sizes=[5, 10, 15], min_temperature=1.0, max_temperature=3.0, t_steps=11)` [source]

Runs a series of simulations at different temperatures and lattice sizes.

The function calculates Binder cumulants for different systems and plots these. If the statistics are good enough, these cumulants should intersect approximately at the critical point T_C.

Parameters:
- **n_steps** (*int*) – total simulation cycles (each cycle is N^2 spin flip attempts)
- **thermalization_steps** (*int*) – thermalization cycles in the beginning of the simulation
- **lattice_sizes** (*list*) – list of lattice sizes N
- **min_temperature** (*float*) – the inital temperature
- **max_temperature** (*float*) – the final temperature (assumed larger than min_temperature)
- **t_steps** (*int*) – the number of different temperatures to simulate

`ising.`**`run_metropolis_algorithm`**`(grid, temperature, run_steps, thermalization_steps, recording_interval=10, animated=False)` [source]

Runs the Metropolis algorithm at a contant temperature.

The algorithm repeatedly chooses random spins in the system and tries to flip them using `metropolis_step()`. The changes are applied directly on the given grid.

After a thermalization period, the algorithm also calculates the magnetization of the system. By default, the recording is not done at every step because too closely measured values will be correlated.

Parameters:
- **grid** (*array*) – the system as an integer array
- **temperature** (*float*) – the temperature
- **run_steps** (*int*) – total simulation cycles (each cycle is N^2 spin flip attempts)
- **thermalization_steps** (*int*) – thermalization cycles in the beginning of the simulation
- **recording_interval** (*int*) – number of cycles between data recording
- **animated** (*bool*) – if True, shows an animation of the simulation in the end

Returns: sequence of magnetizations
Return type: array

`ising.`**`run_temperature_series`**`(grid, min_temperature, max_temperature, temp_steps, run_steps, thermalization_steps)` [source]

Runs a series of simulations at different temperatures.

The function calculates the average magnetization and its error estimate (as the standard error of mean) at each temperature and returns these as arrays.

Parameters:
- **grid** (*array*) – the system as an integer array
- **min_temperature** (*float*) – the inital temperature
- **max_temperature** (*float*) – the final temperature (assumed larger than min_temperature)
- **temp_steps** (*int*) – the number of different temperatures to simulate
- **run_steps** (*int*) – total simulation cycles (each cycle is N^2 spin flip attempts)
- **thermalization_steps** (*int*) – thermalization cycles in the beginning of the simulation

Returns: temperatures, magnetizations, error estimates, cumulants
Return type: array, array, array, array

`ising.`**`save_grid_image`**`(grid, index)` [source]

Saves an image of the system as a $N \times N$ pixel image.

The image is saved in a file named "ising_X.png" where X is the given index.

Parameters:
- **grid** (*array*) – the system as an integer array
- **index** (*int*) – index for naming the output file

`ising.`**`show_grid`**`(grid)` [source]

Draws the system as a $N \times N$ pixel image.

Parameters: **grid** (*array*) – the system as an integer array

`ising.`**`write_data_to_file`**`(lattice_size, temps, mags)` [source]

Writes a data series in a file.

The data is written in file "ising_data_N.txt" where N is the lattice size.

The data is written so that each column holds a temperature and the calculated magnetization at that temperature.

Parameters:
- **lattice_size** (*int*) – system size N
- **temps** (*list*) – a list of temperatures
- **mags** (*list*) – a list of magnetizations