Institute for Theoretical Physics
University of Cologne

Prof. Dr. Simon Trebst
Dr. Yoshito Watanabe

Computational Many-Body Physics

Exercise Sheet 2

Summer Term 2024

Due date: Wednesday, **8th May**, 2024

Website: thp.uni-koeln.de/trebst/Lectures/2024-CompManyBody.shtml

Exercise 2: Lattices

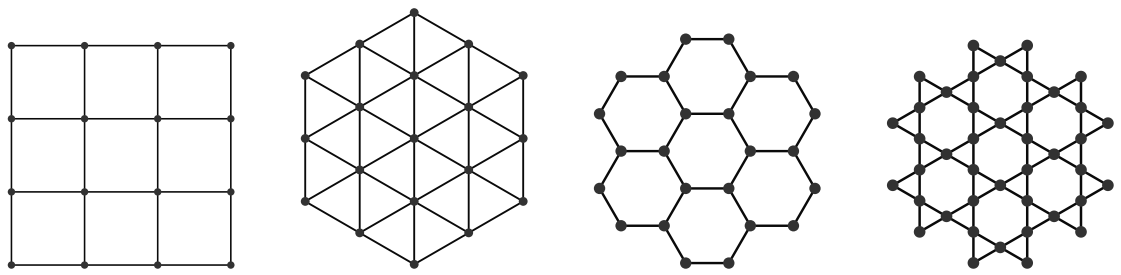
In this first exercise we want to introduce the concept of lattices underlying various models in (quantum) statistical physics.

Your overall task in this exercise is to implement a **lattice library**. Such a library contains type definitions and functions that can be used in various applications, which all use lattices underlying a model. Building your future numerical codes on the foundation of such a library has the benefit of a high versatility to study different model systems without much further effort.

In general, a lattice library should help in constructing models whose elements are defined for each lattice site. Therefore the minimum requirements that a lattice library has to fulfill are:

- access the real space position for every site of the lattice,
- access the nearest neighbors (connected sites) of a given site within the lattice,
- plot a given lattice.

To implement such a lattice library, we suggest that you to proceed along the following steps:



a) First, go back to your solid state theory course (Bachelor or Master) and look up the definition of what a *lattice* is and what components it contains. Make a list of everything that seems to be important for a lattice. You might also consider reading through the entire exercise sheet to know what your lattice in principle should be able to do.

Hint: You can document the list within the julia notebook containing your solution for this exercise. In this case you can also use markdown and LaTeX formatted text as well as formulas.

b) In a second step, implement a new *type* that represents the lattices in your code. If you are new to julia programming, we suggest you to read up on how to define your own types e.g. in the julia documentation before proceeding.

In general, you should at least define a type *Lattice* which contains information on all sites of the lattice as well as on how these sites are connected. After all, the idea behind a lattice library is that the connectivity information of the lattice can easily be accessed. If possible, use the list of part a) in the process to not miss any necessary details.

c) After you defined the necessary types, you can now start implementing the needed *functions* for the library. Start with functions that are needed for constructing a specific lattice object (e.g. adding connections or adding sites) and finish by implementing a plotting function which plots a constructed *Lattice* object. This plotting function should be your tool of choice to check if you implemented a specific lattice correctly.

d) To finally test your lattice implementation, try generating a honeycomb lattice of size $L \times L$ unit cells and plot it to see if your code works for $L = 10$ as well as $L = 15$.

Bonus - What other lattices can you think of? Implement functions to generate these lattices with periodic boundary counditions as well.

Exercise 3: Metropolis algorithm for classical spin models

The purpose of this exercise is to implement the Metropolis algorithm for three different classical spin models, the **Ising** model, the **Heisenberg** model as well as the **Kitaev** model, which were introduced in the lecture. All three spin models are formulated in terms of localized classical spins, which are located on the sites of an underlying lattice and which interact along the bonds of this lattice. Thus, we suggest that you to use the lattice library of the previous exercise. This way, you will also be able to generalize the codes to arbitrary lattices without loss of generality.

In this exercise, we want to focus on the **honeycomb** lattice – however you are free to use another lattice of your choice as well.

Ising Model

We want to start the exercise with the simplest of the three models, the **Ising** model. Its Hamiltonian can be defined as

$$\mathcal{H} = J \sum_{\langle i,j \rangle} \sigma_i \sigma_j \tag{1}$$

It contains classical Ising spins σ_i which can only take the values $\sigma_i = \pm 1$ and which are located on lattice sites i, j of an underlying lattice. The notation $\langle i, j \rangle$ denotes a sum over connected sites i and j of this lattice.

a) In a first step towards a numerical description of the thermodynamics of the Ising model, we want you to **implement** the Metropolis algorithm for the Ising model and a given *Lattice* object. Therefore, we suggest that you to proceed as follows:

(i) Implement a function which returns a random configuration of Ising spins for a given *Lattice* object. The spins can be initialized randomly and should not have a predefined order.

(ii) Implement a function to plot one configuration on the passed *Lattice* object. This should help in checking the physics of the algorithm later on. Test your function by plotting multiple of the random configurations on the honeycomb lattice.

(iii) Implement a function which performs a single Metropolis sweep, i.e. a sequence of **Metropolis updates** for N randomly chosen spins (where N is the number of spins in the complete lattice). You should pass this function the current spin configuration, as well as your lattice and the temperature of choice.

(iv) Test your Metropolis sweep function by running it around 10.000 times on a random initial configuration of a 10×10 honeycomb lattice with $J = \pm 1$ and $T = 0.1$. Plot the final configuration and check if it agrees with your expectations.

(v) If you have not done so already, try to improve the performance of your Metropolis algorithm. One obvious improvement is to calculate only the energy of a configuration *once*, pass it with the temperature into the sweep function, and update it *locally* every time a spin is flipped. By using this technique, the sweep function should be able to perform much better and obtain identical results in much faster times.

Hint: Besides plotting the configuration, we suggest to check the energy against the correctly calculated energy after all sweeps to test if the incremental method indeed works as expected.

After finishing all of the above tasks, you are now ready to implement thermodynamic observables in your code and measure expectation values of various quantities.

b) In a second step, we want to visualize the concept of **thermalization**. Plot the energy of a spin configuration at temperature $T = 0.4$ versus the number of sweeps you have performed on this configuration. What do you observe?

The slow convergence of the energy during the sweeping is due to the convergence of the Markov chain towards the Boltzmann distribution. To compensate this fact in the calculation of observables, one usually performs a number of sweeps without measuring at all before the measurements start in a second stage, the so called **thermalization sweeps**. In your future calculations, you should always use thermalization sweeps equal to at least $1/3$ of the number of measurement sweeps, better more than less.

c) After having discussed the technical parts of Monte Carlo we want to focus on the **thermodynamics** of the model. We want you to implement a function to calculate the magnetization $M = \sum_i \sigma_i$ of a given spin configuration, as well as making sure you are able to obtain the energy E and the square of the energy E^2 after every sweep in your calculations.

Calculate and plot the expectation value of the **energy per spin**, $\langle E(T)\rangle/N$, as well as of the **magnetization per spin**, $\langle M(T)\rangle/N$), versus the temperature in a range of $T \in [0.1, 4.0]$ using your Monte Carlo algorithm. Check if the $T \to 0$ limits of both observables behave as expected and if you can identify the thermal phase diagram. In general, the $T \to 0$ limit is a useful way of debugging as the system approaches the often well known ground state and all observables can be compared with their expected values.

Furthermore, we want to investigate the **specific heat capacity**

$$c_V(T) = \frac{1}{N}\frac{\partial\langle E(T)\rangle}{\partial T} \tag{2}$$

Show that it can be calculated on every temperature point as

$$c_V(T) = \frac{1}{N\,T^2}(\langle E(T)^2\rangle - \langle E(T)\rangle^2) \tag{3}$$

and use the formula to calculate the specific for the above temperature region. Can you identify the location of the phase transition?

Last but not least plot Monte Carlo averages for the **Binder cumulant** of the magnetization

$$U(T) = 1 - \frac{\langle M(T)^4\rangle}{3\langle M(T)^2\rangle^2} \tag{4}$$

for the full temperature range, locate the phase transition again and discuss the graph.

d) In a next step, we turn to the system size dependence of thermodynamic observables and perform **finite-size scaling**. Plot the energy per spin, magnetization per spin, and specific heat each in a figure with linear system sizes $L = 5, 10, 15$ and investigate how the behavior of the various graphs changes.

Heisenberg model

The **Heisenberg** model is a generalization of the Ising model to spins of higher dimension. Its Hamiltonian can be defined as

$$\mathcal{H} = J\sum_{\langle i,j\rangle} \vec{S}_i\vec{S}_j \tag{5}$$

It contains classical Heisenberg spins $\vec{S}_i$ which are classical vectors of length $\vec{S}_i^2 = 1$ and which are located on lattice sites i, j of an underlying lattice. The notation $\langle i, j\rangle$ again denotes a sum over connected sites i and j of this lattice.

In principle, you now have to go back to the part of the exercise about the Ising model and perform all steps analogous for the Heisenberg model – you should notice that all you have to change are the *format of spins* and the *energy calculation* within the sweep method. However, before you dive into the thermodynamics of the Heisenberg model, let's consider yet another technical difficulty to overcome:

e) Since you are now dealing with constrained three-dimensional vectors as spins, you have to make sure that you are **sampling** an individual spin from a uniform distribution. As a vector on the unit sphere can be constructed from two angles θ and ϕ in polar coordinates, one might be tempted to sample those uniformly as $\phi \in [0, 2\pi]$ and $\theta \in [0, \pi]$.

Plot a sample of spins, constructed by sampling these two angles, in a common 3D plot and investigate the distribution. Are the spins really uniformly distributed on the sphere?

From your observations, it should be clear that sampling spins with two angles is not feasible. Convince yourself by means of a similar plot that spins are indeed uniformly distributed when sampling the z-component and the angle ϕ.

f) After being able to correctly sample uniformly distributed Heisenberg spins, go through the steps a) to d) of the previous exercise and implement a Metropolis algorithm for the Heisenberg model on the honeycomb lattice.

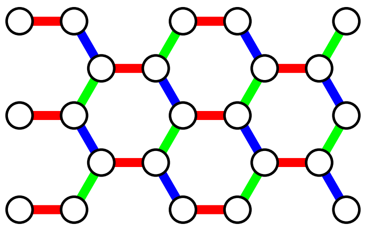
Plot the respective observables versus temperature and perform a suitable finite size scaling. Compare your results with the results from the Ising model.

Kitaev model

In the third part of this exercise, we want to familiarize you with yet another classical spin model, the **Kitaev** model. Motivated by its rich quantum behavior, it can also be formulated for classical spins by means of the following Hamiltonian for Heisenberg spins $\vec{S}_i$ on lattice sites of the honeycomb lattice:

$$\mathcal{H} = J_x \sum_{\langle i,j\rangle_x} S_i^x S_j^x + J_y \sum_{\langle i,j\rangle_y} S_i^y S_j^y + J_z \sum_{\langle i,j\rangle_z} S_i^z S_j^z \tag{6}$$

Although this model looks like a complicated Heisenberg model at first glance, it is yet entirely different! The coupling of the individual spin components only takes place along certain directions of the lattice. Take e.g. the x-component of the spin. It is only coupled along x-bonds of the lattice (which are shown in red in the neighboring sketch).



g) Before implementing the Metropolis algorithm for the Kitaev model, you have to implement the directional dependence of bonds within your lattice representation. Make sure that you can distinguish the three bond types when accessing neighbors and try to reproduce the lattice plot above.

h) Implement the Metropolis algorithm for the Kitaev model on the honeycomb lattice. Make a plot of the specific heat for various system sizes and the following parameter sets:

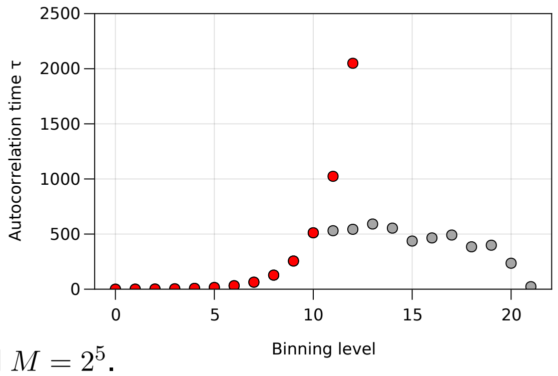
- $J_x = J_y = J_z = 1$ (isotropic case)
- $J_x = J_y = 1/2$, $J_z = 2$ (anisotropic case I)
- $J_x = J_y = 4/3$, $J_z = 1/3$ (anisotropic case II)

Compare these three parameter sets and discuss whether there is a phase transition at finite temperature.

Exercise 4: Binning analysis

In this exercise we want to analyze **time series data with intrinsic autocorrelations** – a situation which we will routinely encounter in the statistical analysis of observable measurements from Markov chain Monte Carlo simulations. To this end, we will perform a binning analysis on a tailor-made data set (where we control the level of autocorrelations).

a) We start with the generation of two different 'artificial' data sets of size N. The first data set consists of two randomly generated numbers that are repeated $N/2$ times. Thus, the data is almost fully correlated. For the second data set, you sample M different random numbers, each of which is repeated N/M times. This creates M blocks containing N/M copies of the same number. The blocks are internally correlated, while they are not correlated with each other. Plot the data series for visualization. You can use $N = 2^{15}$ and $M = 2^5$.



b) Implement a binning function that reduces a given data set $A^{(0)}$ of size N to a data set $A^{(l)}$ of size $N/2^l$. The individual elements of the "binned" data set $A^{(l)}$ result from the mean value of the elements of the previous data set $A^{(l-1)}$.

$$A_i^l = \frac{1}{2}\left(A_{2i-1}^{l-1} + A_{2i}^{l-1}\right) \tag{7}$$

c) Finally, determine the standard deviation and the integrated autocorrelation time τ_A^{int} for both data series. Do this upto the highest possible binning level and afterwards plot the calculated values against the binning level. Describe you observations.

$$\tau_A^{int} = \frac{1}{2}\left[\left(\frac{\Delta A}{\Delta A^{(0)}}\right)^2 - 1\right] \tag{8}$$

d) You may now go back to the Monte Carlo simulations of the previous exercise and perform a binning analysis for the time series of different observables (e.g., the energy or absolute magnetization) for varying temperatures and/or system sizes.