1 HW 5: Multigrid Simulation of the Gaussian Model

Your total number of points for this homework is (10/20 P). If you have any questions feel free to write me an email (s6nnschl@uni-bonn.de).

Exercise 5.1: Analytical Derivation

Not correct. (0/1 P) The correct results are

$$\langle m \rangle = 0 \tag{1}$$

$$\langle m^2 \rangle = \frac{a}{4\beta N^3} \sum_{k=\text{odd}}^{N-1} \frac{\cos^2\left(\frac{\pi k}{2N}\right)}{\sin^4\left(\frac{\pi k}{2N}\right)} \tag{2}$$

$$\langle H_a \rangle = \frac{N-1}{2\beta} \tag{3}$$

We will discuss the corresponding derivation in the next tutorial.

Exercise 5.2: Standard Metropolis-Hastings Algorithm

In principle, your implementation is correct. Regarding your questions "Why is the magnetization varying so much with each iteration? Why is the energy increasing?":
Right now, you can't say anything for sure about your results, since you didn't calculate/ plot any error bars. More precisely, a calculation of the blocked-bootstrap mean/error would be necessary (-1 P). In itself, your Metropolis-Hastings implementation looks correct and I assume that the results are OK. According to equation (1), we expect $\langle m \rangle$ to be constant 0 so your measurements fluctuate around 0. Whether your results are compatible with this would become apparent when the errors are taken into account. In this context, please also note how extremely your $\langle m \rangle(N)$ plots vary when repeatedly applying the Metropolis-Hastings algorithm! According to equation (2), if we plot $\langle m^2 \rangle \beta / a$ as a function of N, we expect a more or less linear increase in a relevant x-axis range (e.g., up to $N = 40$). However, one can define an intensive variable $\langle m^2 \rangle = \langle m^2 \rangle / N$. This variable approaches a constant value in the large N limit. The whole thing looks similar for $\langle H_a \rangle(N)$ (linear increase) and $\langle \epsilon \rangle = \langle H_a \rangle / N$ (convergence to constant value), see equation (3). In total (4/5 P)

Exercise 5.3: Subgrids

Exercise 5.3.1: Explicit Form of $\phi^{(2a)}$

Not complete (-0.5 P). If you take all contributions proportional to $u_i^{(2a)}$ you will find the coarse level field

$$\phi_i^{(2a)} = \frac{1}{4}\left(\phi_{2i-1}^{(a)} + 2\phi_{2i}^{(a)} + \phi_{2i+1}^{(a)}\right) - \frac{1}{2a^2}\left(\tilde{u}_{2i-2}^{(a)} - 2\tilde{u}_{2i-2}^{(a)} + \tilde{u}_{2i+2}^{(a)}\right), \tag{4}$$

where $\tilde{u}_i^{(a)}$ is the non-interpolated lattice. In total (0.5/1 P).

Exercise 5.3.2: Generalized Hamiltonian

In principle correct, but it would be more efficient to use the `sum()` functions since python for loops are expensive. (1/1 P)

Exercise 5.3.3: Fine-To-Coarse Restriction Function

Here, it would be clearer to write a function that only coarsens the lattice, i.e. something like

```
def coarse(u):
    '''Restriction function'''
    return u[::2]
```

(1/1 P)

Exercise 5.3.4: Coarse-To-Fine Prolongation Function

Proposal for a function that interpolates the field to a finer grid:

```
def uncoarse(u):
    '''Prolongation function'''
    u_half = np.zeros(2*len(u))
    for i in range(2*len(u)-1):
        if i%2!=0 :
            u_half[i] = (u[int((i-1)/2)] + u[int((i+1)/2)])/2
        else:
            u_half[i] = u[int(i/2)]
    return u_half
```

(0.5/1 P)

Exercise 5.4: Multigrid Algorithm

Your function does not work yet. Equipped with the `coarse()`, `uncoarse()` and generalized Hamiltonian functions described above, you can implement `Multigrid()` as a recursive routine:

- do a certain number of pre-sweeps
- calculate the effective external field for the courser level
- for g in range(gamma): (gamma determines if it is a V, W cycle and so on)

  use the Multigrid() function itself and double the lattice spacing a by setting $a \to 2a$
- interpolate for refinement
- do a certain number of post-sweeps
- return an updated field

+2 P for following the algorithm description in your implementation, so in total (2/7 P).
I will present a proposal code in detail in the next tutorial.

Exercise 5.5: Testing

Does not work yet but +1 P for the implementation of the autocorrelation function (1/3 P).