Exercise 04: Multigrid simulation of the Gaussian Model

Computational Physics WS20/21

Marc Vöhringer Carrera und Helen Meyer

December 9, 2020

1 Introduction

In this week's exercise we want to implement a multigrid algorithm in order to compute the average magnetization, average magnetization squared and the average energy of the Gaussian model in one dimension. It is described by the Hamiltonian

$$H_a(u) = \frac{1}{a}\sum_{i=1}^{N}(u_i - u_{i-1})^2 \tag{1}$$

with a real valued field u. a is the spacing between neighbouring grid points and $N = L/a$ (here: a power of 2), where L defines the volume of the field. We use Dirichlet boundary conditions

$$u(0) = u_0 = 0 \tag{2}$$

$$u(L) = u_N = 0. \tag{3}$$

2 Metropolis-Hastings sweep

In order to derive analytic solutions for the expectation values of the average magnetization m, the average magnetization squared m^2 and the energy E (Hamiltonian), we use the Fourier decomposition

$$u_i = sum_{k=1}^{N-1} c_k \sin(k\pi i/N). \tag{4}$$

With that we obtain the Hamiltonian

$$H_a(c) = \frac{2N}{a}\sum_{k=1}^{N-1} c_k^2 \sin^2\left(\frac{k\pi}{2N}\right) \tag{5}$$

and the partition sum (with $\sigma_k^2 \equiv= a/(4\beta N \sin^2(k\pi/2N))$)

$$\begin{aligned}
Z(\beta, N, a) &= \prod_{i=1}^{N} \int_{-\infty}^{\infty} \mathrm{d}u_i \exp(-\beta H_a(u)) && (6)\\
&= \prod_{k=1}^{N} \int_{-\infty}^{\infty} \mathrm{d}c_k \exp(-\beta H_a(c)) && (7)\\
&= \prod_{k=1}^{N} \int_{-\infty}^{\infty} \mathrm{d}c_k \exp\left(-\frac{\beta 2N}{a} \sum_{j=1}^{N-1} c_j^2 \sin^2\left(\frac{j\pi}{2N}\right)\right) && (8)\\
&= \prod_{k=1}^{N} \left(\int_{-\infty}^{\infty} \mathrm{d}c_k \exp\left(-\frac{c_k^2}{2\sigma^2}\right)\right) && (9)\\
&= \left(\frac{\pi a}{2N\beta}\right)^{(N-1)/2} \prod_{k=1}^{N} \frac{1}{\sin(k\pi/2N)}. && (10)
\end{aligned}$$

1: *Using the Fourier decomposition and the formulas given on the exercise sheet, we get for the average magnetization*

$$m = \frac{1}{N}\sum_{i=1}^{N-1} u_i = \frac{1}{N}\sum_{k=1,\mathrm{odd}}^{N-1} c_k \cot\left(\frac{k\pi}{2N}\right) \tag{11}$$

and its squared

$$m^2 = \frac{1}{N}\sum_{i=1}^{N-1} u_i^2 = \frac{1}{2}\sum_{k=1}^{N-1} c_k^2 \tag{12}$$

The expectation value of an operator $O(c)$ is given by

$$\langle O \rangle = \frac{1}{Z}\prod_{k=1}^{N}\int_{-\infty}^{\infty} \mathrm{d}c_k \exp(-\beta H_a(c)), \tag{13}$$

with Z obtained in equation (6). Plugging in the functions for m, m^2 and the Hamiltonian H and again using the formulas on the sheet, we get for the expectation values

$$\begin{aligned}
\langle m \rangle &= 0 && (14)\\
\left\langle m^2 \right\rangle &= \frac{a}{8N\beta}\sum_{k=1}^{N-1}\frac{1}{\sin^2(k\pi/2N)} && (15)\\
\langle E \rangle &= \frac{N-1}{2\beta} && (16)
\end{aligned}$$

As we did in the last exercises, we want to generate a Markov Chain using a sweep through the field u defined as:

- Randomly choose a site x for an update, $x \sim U(1, ..., N-1)$

- Propose a new $u\prime(x) = u(x) + r\delta$ with $r \sim U([-1,1])$ and *delta* a fixed scale parameter.
- Use the Metropolis accept / reject step

and repeating this $N-1$ times.
2: *We tested our version of the Metropolis Hastings sweep in* `multigrid.py` *with the parameters $\delta = 2$, $N = 64$, $\beta = 1.$ and $a = 1$ and created a Markov Chain of length $N_{\text{cfg}} = 1000$ and 1000 thermalization sweeps. We obtain the expectation values (the errors being bootstrap errors)*

$$\langle m \rangle = 0{,}0008 \pm 0{,}0072 \tag{17}$$

$$\left\langle m^2 \right\rangle = 0{,}0371 \pm 0{,}0169 \tag{18}$$

$$\langle E \rangle = 30{,}786 \pm 5{,}361. \tag{19}$$

Comparing to the exact solutions $\langle m \rangle = 0$, $\langle E \rangle = 31.5$ and $\langle m^2 \rangle = 5.332$, one can see that the algorithm works fairly well for computing energy and magnetization, but not well for the magnetization squared.

3 Multigrid Algorithm

The idea of the multigrid algorithm is to do most sweeps on a coarser level (with fewer gridpoints) and update the field u in the current level using prolongation. Fine-to-coarse restriction gives us a field u with fewer grid points (coarser grid) by simpling taking every second grid point

$$u_i^{(2a)} = u_{2i}^{(a)}, \quad i = 0, ..., N/2 \tag{20}$$

Coarse-to-fine prolongation uses linear interpolation:

$$I_{(2a)}^{(a)} u^{(2a)} = \begin{cases} u_{i/2}^{(2a)} & i = 0, 2, ..., N \\ \left(u_{(i-1)/2}^{(2a)} + u_{(i+1)/2}^{(2a)}\right)/2 & i = 1, 3, ..., N-1 \end{cases} \tag{21}$$

We add a field $\phi^{(a)} = 0$ to the Hamiltonian in order to generalize it:

$$H_a(u) = \frac{1}{a} \sum_{i=1}^{N} (u_i - u_{i-1})^2 + a \sum_{i=1}^{N-1} \phi_i^{(a)} u_i \tag{22}$$

The prolongation $u^{(a)} = \tilde{u}^{(a)} + I_{(2a)}^{(a)} u^{(2a)}$ leads to

$$H_a(u^{(a)}) = H_a(\tilde{u}^{(a)}) + H_{2a}(u^{(2a)}) \tag{23}$$

$$H_{2a}(u^{(2a)}) = \frac{1}{2a} \sum_{i=1}^{N} (u_i^{(2a)} - u_{i-1}^{(2a)})^2 + 2a \sum_{i=1}^{N-1} \phi_i^{(2a)} u_i^{(2a)} \tag{24}$$

3: *Plugging in and comparing coefficients of $u^{(2a)}$ we obtain an expression for the external field on the coarser grid $\phi^{(2a)}$:*

$$\phi_i^{(2a)} = \frac{1}{4}\left(\phi_{2i+1}^a + 2\phi_{2i}^a + \phi_{2i-1}^a\right) + \frac{1}{2a}\left(2\tilde{u}_i - \tilde{u}_{2i+2} - \tilde{u}_{2i-2}\right), i = 1, ..., N-1 \quad (25)$$

It depends on the field on the finer level as well as on the field $u^{(a)}$. The restriction and prolongation functions are implemented in ***`multigrid.py`****.*

Now the goal is to implement the multigrid algorithm:

1. Do ν_{pre} sweeps at current level. Proceed to step 5, if current level is coarsest level.

2. Generate next coarser level (determine $\phi^{(2a)}$ and use the Hamiltonian H_{2a} to sweep).

3. **Recursive step:** Do γ multigrid cycles for the coarser level.

4. Update current $u^{(a)}$:
$$u^{(a)} \leftarrow u^{(a)} + I_{(2a)}^{(a)} u^{(2a)} \quad (26)$$
with $u^{(2a)}$ from step 3.

5. Do ν_{post} sweeps at current level.

4&5: *The multigrid algorithm is implemented in* ***`multigrid.py`*** *for 3 levels. We tested the algorithm with $N = 64$, $\delta = 2$, $\gamma = 1, 2$, $n = 3$ and $\nu_{\text{pre}} = \nu_{\text{post}} =$ [4,2,1] (from coarsest to finest level). Unfortunately, the values obtained for the average magnetization (squared) and the energy differ (by some orders of magnitude) from the analytic results (see **2**). We did not have time to find this mistake. The autocorrelation function for m^2 is shown in Fig. (3.1). One can see that it they are negative for a Monte Carlo time of about 35.*

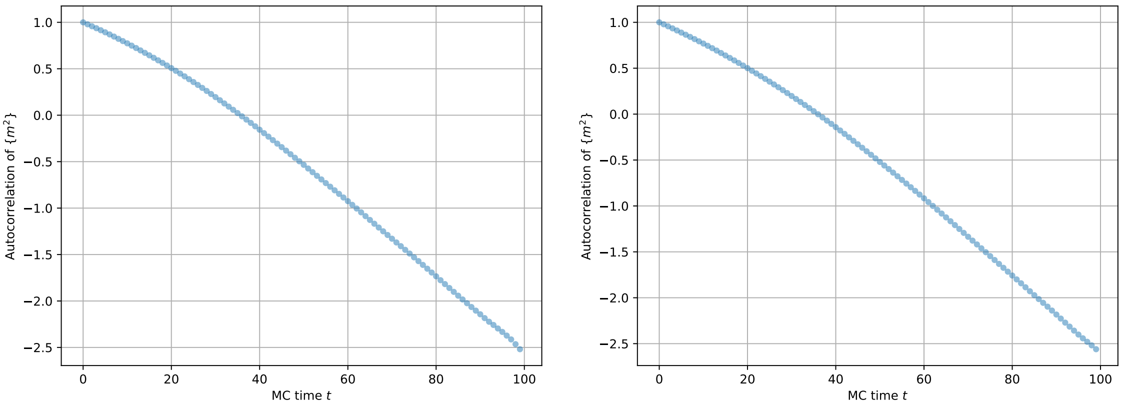

Figure 3.1: The autocorrelation functions of $\{m^2\}$ obtained from the multigrid algorithm with $\gamma = 1$ (left) and $\gamma = 2$ (right).