1 HW 4: Error analysis of a Markov Chain

Your total number of points for this homework is (13.5/20 P). If you have any questions feel free to write me an email (s6nnschl@uni-bonn.de).

Exercise 4.1: Fluctuation

Your implementation is correct but in the x-axis range you chose, you can't see much (-0.5 P). My suggestion is to choose `N_steps = 2500` and `ax.set_xlim(2000,2500)`. Then, the following becomes clear: In the $N_{md} = 4$ case there are more short term oscillations. However, because of the relatively low acceptance rate, there are more extensive regions without changes (plateaus). In case of $N_{md} = 100$ you can see a large wave-length behavior due to high correlation. In total (1.5/2 P)

Exercise 4.2: Autocorrelation

Exercise 4.2.1: Implementation

Your estimator for the normalized autocorrelation function is correct. (4/4 P)

Exercise 4.2.2: Plot

Here, you cannot distinguish between the two different curves (e.g. use `ax.set_xlim(0,20)` to view the interesting x-axis range). Thus, you cannot see that the $N_{md} = 100$ case has a slower fall off due to the higher acceptance rate, and so on. An example for the plotting routine is as follows:

```
tau_arr = np.arange(0,100,step=1)

c_tau_arr_1 = []
c_tau_arr_2 = []
C_0_1 = autocorr(m1, 0)
C_0_2 = autocorr(m2, 0)

for tau in tau_arr:
    # for m1:
    c_tau_tmp_1 = autocorr(m1, tau) / C_0_1
    c_tau_arr_1.append(c_tau_tmp_1)
    # for m2:
    c_tau_tmp_2 = autocorr(m2, tau) / C_0_2
    c_tau_arr_2.append(c_tau_tmp_2)


fig, ax = plt.subplots(figsize=(9,6))

ax.plot(tau_arr, c_tau_arr_1, label='$N_{md}=100$', ms=3.0, ls=":", marker="
    o")
ax.plot(tau_arr, c_tau_arr_2, label='$N_{md}=4$', ms=3.0, ls=":", marker="o"
    )

ax.set_xlim(0,40)
ax.set_xlabel(r"$\tau$", fontsize=14)
```

```
ax.set_ylabel(r"$C(\tau) = \bar\Gamma^{{(m)}}(\tau)   /   \bar\Gamma^{{(m)}}(0)$", fontsize=14)

ax.grid()
ax.legend()
```

Description, attempted explanation and comparison are missing. (0.5/1 P)

Exercise 4.3: Blocking

Exercise 4.3.1: Implementation

Correct. (2/2 P) Also your plots look ok and you can see that binning makes the autocorrelation go down (e.g. use `ax.set_xlim(0,20)` to view the interesting x-axis range).

Exercise 4.3.2: Naive Standard Error

When I run your code, I get a `NameError: name 'ctaus_arr1' is not defined`. But otherwise your code for the naive standard error (defined as $\sigma/\sqrt{N/b}$) looks correct. However, the standard deviation should have been computed on the magnetization data and not on the autocorrelation `ctau_arr1` and `ctau_arr2` (−2 P). In total (1/3 P)

Exercise 4.4: Bootstrapping

Exercise 4.4.1: Implementation

Your for loops are a bit messed up (−2 P). For example, you could use something like

```
for k in range(N_bs):
    for j in range(N):
        r = np.random.randint(N)
        m_bs[j] = m_b_8_2[r]
    m_mean[k] = np.mean(m_bs)
```

Also, you do not return the bootstrap mean but only the bootstrap error. In total (4/6 P)

Exercise 4.4.2: Error Stability

Correct that you computed the bootstrap over the magnetization. It would have been nice to plot the bootstrap mean with error bars (i.e. bootstrap errors) as a function of bootstrap sample size. What exactly does it mean that stability is given by an error (−0.5 P)? In total (0.5/1 P)

Exercise 4.4.3: Comparison

Here you should find that the bootstrap error agrees well with the naive estimate, cf. also our discussion in the previous tutorial. (0/1 P)