Non-linear estimation problem using Markov Chain Monte-Carlo and Simulated Annealing

Joël M. Fonseca

February 20, 2019

Abstract

The aim of this project is to experiment the Markov Chain Monte-Carlo (MCMC) method for a generalized non-linear estimation problem. In addition, it is required to implement the Simulated Annealing (SA) technique in order to approximate the global optimum.

1 Introduction

1.1 Problem description

The non-linear estimation problem is as follows. Suppose we have a known matrix $\boldsymbol{W}$ of dimension $m \times n$ with entries $W_{ij} \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$ for $1 \leq i \leq m$ and $1 \leq j \leq n$. Then, one would like to recover a vector

$$\boldsymbol{X} = (X_1, X_2, ..., X_n)^T \in \mathcal{S} = \{-1, 1\}^n \tag{1}$$

from the m observations:

$$Y_i = \varphi\left(\sum_{j=1}^{n} \frac{W_{ij}}{\sqrt{n}} X_j\right) \qquad 1 \leq i \leq m \tag{2}$$

where φ is the Rectifier Linear Unit (ReLU) function defined as

$$\varphi(X) = \max(0, X). \tag{3}$$

Note that $\boldsymbol{Y} = (Y_1, Y_2, ..., Y_n)^T$ hereafter. As the search space is exponential (of size 2^n), a sound approach is to use the Markov chain Monte-Carlo (MCMC) method. The idea behind MCMC is to construct a Markov chain on the state space S whose stationary distribution is *Gibbs-Boltzmann* probability distribution. It gives the probability that a system will be in a certain state as a function of that state's energy and the temperature of the system. It is defined as

$$p_{\boldsymbol{Y}}(\boldsymbol{X}) = \frac{e^{-\beta H_{\boldsymbol{Y}}(\boldsymbol{X})}}{Z_\beta}, \qquad \boldsymbol{X} \in \mathcal{S} \tag{4}$$

where $p_{\boldsymbol{Y}}(\boldsymbol{X})$ is the probability of state $\boldsymbol{X}$, $H_{\boldsymbol{Y}}(\boldsymbol{X})$ is the energy of state $\boldsymbol{X}$ described as

$$H_{\boldsymbol{Y}}(\boldsymbol{X}) = \sum_{i=1}^{m} \left| Y_i - \varphi\left(\left[\frac{\boldsymbol{W}\boldsymbol{X}}{\sqrt{n}} \right]_i \right) \right|^2, \tag{5}$$

where β is the inverse temperature of the system and

$$Z_\beta = \sum_{\boldsymbol{X} \in \mathcal{S}} e^{-\beta H_{\boldsymbol{Y}}(\boldsymbol{X})} \tag{6}$$

is the normalizing factor also called *partition function*. As we take the parameter $\beta \to +\infty$, the obtained sample should converge to a vector $\boldsymbol{X}$ that minimizes the energy function described in (5).

1.2 Implementation using Python

The whole project is implemented using Python. The high-level decomposition of the code allows anyone to quickly add a new transition function, a schedule function for the simulated annealing method or even play with some parameters. The `tqdm` library is used to follow the progress of the computations in the experiments. The comments and readability of the code make it easy to use.

2 MCMC algorithms

In this project, two MCMC algorithms will be used for performance comparison, namely the Metropolis-Hastings (MH) and the Glauber dynamics (Gd) or heat bath algorithm.

2.1 Metropolis-Hastings algorithm

One of the benefits of this algorithm is that the required function to draw samples from the probability distribution $p_{\boldsymbol{Y}}(\boldsymbol{X})$ must only be proportional to the density $p_{\boldsymbol{Y}}$. This makes the algorithm particularly useful as calculating the necessary normalization factor can be extremely difficult (think of n big). Its pseudocode is described in the next figure.

Algorithm 1 Pseudocode of Metropolis-Hastings algorithm

```
X ← initialize random vector ∈ S
while H_Y(X) > threshold do
  i ← select random coordinate
  X' ← (X_1, ..., X_{i-1}, −X_i, X_{i+1}, ..., X_n)^T
  if previous move is accepted with probability: a_β(X, X') then
    X ← X'
  end if
end while
```

The acceptance probability is described as

$$a_\beta\left(\boldsymbol{X}, \boldsymbol{X'}\right) = \min\left(1, e^{-\beta\left(H_{\boldsymbol{Y}}(\boldsymbol{X'}) - H_{\boldsymbol{Y}}(\boldsymbol{X})\right)}\right). \tag{7}$$

Note that the base chain is periodic. Nevertheless, MH algorithm adds self-loops, leaving us with aperiodic, transitive chain satisfying detailed balance equations.

2.2 Glauber dynamics or heat bath algorithm

Also known as the Gibbs sampling, this algorithm represents one of the most popular MCMC algorithms. It corresponds to the MCMC analog of coordinate descent. The pseudocode is presented in the next figure.

Algorithm 2 Pseudocode of Glauber dynamics algorithm

```
X ← initialize random vector ∈ S
while H_Y(X) > threshold do
  i ← select random coordinate
  Z = X_i
  X' ← (X_1, ..., X_{i-1}, -X_i, X_{i+1}, ..., X_n)^T
  if positive-move is accepeted with probability: p_{β,+}(X, X', Z) then
    X_i ← +1
  else
    X_i ← -1
  end if
end while
```

The acceptance probability is defined as

$$p_{\beta,+}\left(\boldsymbol{X}, \boldsymbol{X}', Z\right) = \frac{1 + Z \tanh\left(\beta\left(H_{\boldsymbol{Y}}(\boldsymbol{X'}) - H_{\boldsymbol{Y}}(\boldsymbol{X})\right)\right)}{2}. \tag{8}$$

For both of these algorithms, the idea is to run the `while` loop until the energy $H_{\boldsymbol{Y}}(\boldsymbol{X})$ is sufficiently low (in the best case scenario, we get a perfect reconstruction) or until we get no more improvements after a specific number of iterations (if we get stuck in a local optimum).

3 Experiments

3.1 Setup

For all the experiments we chose the following values for the fixed parameters:

- dimension of the vector space: `n=500`,
- number of experiments to average: `num_exp=10`,
- minimal number of MCMC samplings: `num_iter_mcmc=8000`.

These numbers have been chosen so that we can observe interesting results and save our poor laptops from potentially fatal overheating. In addition, some results will be shown using the reconstruction error function, defined as

$$e\left(\hat{\boldsymbol{X}}, \boldsymbol{X}\right) = \frac{1}{4n}\left\|\hat{\boldsymbol{X}} - \boldsymbol{X}\right\|^2 \tag{9}$$

where $\hat{\boldsymbol{X}}$ represents the estimate for the ground truth vector $\boldsymbol{X}$ and $\|\cdot\|$ is the Euclidean norm. We can write

$$e\left(\hat{\boldsymbol{X}}, \boldsymbol{X}\right) = \frac{1}{n}\left\|\frac{1}{2}(\hat{\boldsymbol{X}} - \boldsymbol{X})\right\|^2,$$

which corresponds exactly to the fraction of missclasified coordinates, since the difference vector takes values in $\{-2, 0, 2\}^n$.

3.2 First observations

To get our hands dirty, we will first compare the two MCMC algorithms defined above for fixed values of the *inverse temperature* β and a specific range of α values where $\alpha = \frac{m}{n}$ represents the ratio between the number of observations and the number of unknowns. The range for each parameter is:

- $\alpha : [0.5, 1.0, 1.5, 2.0, 3.0, 4.0]$,
- $\beta : [0.5, 1.0, 1.5, 2.0, 2.5, 3.0]$.

In order to get a better understanding of the results and be able to compare both transition functions in best possible way, the normalized energy for each configuration will be presented. In order to simplify readability, we placed a marker when there was a perfect reconstruction of the signal.

3.2.1 Results

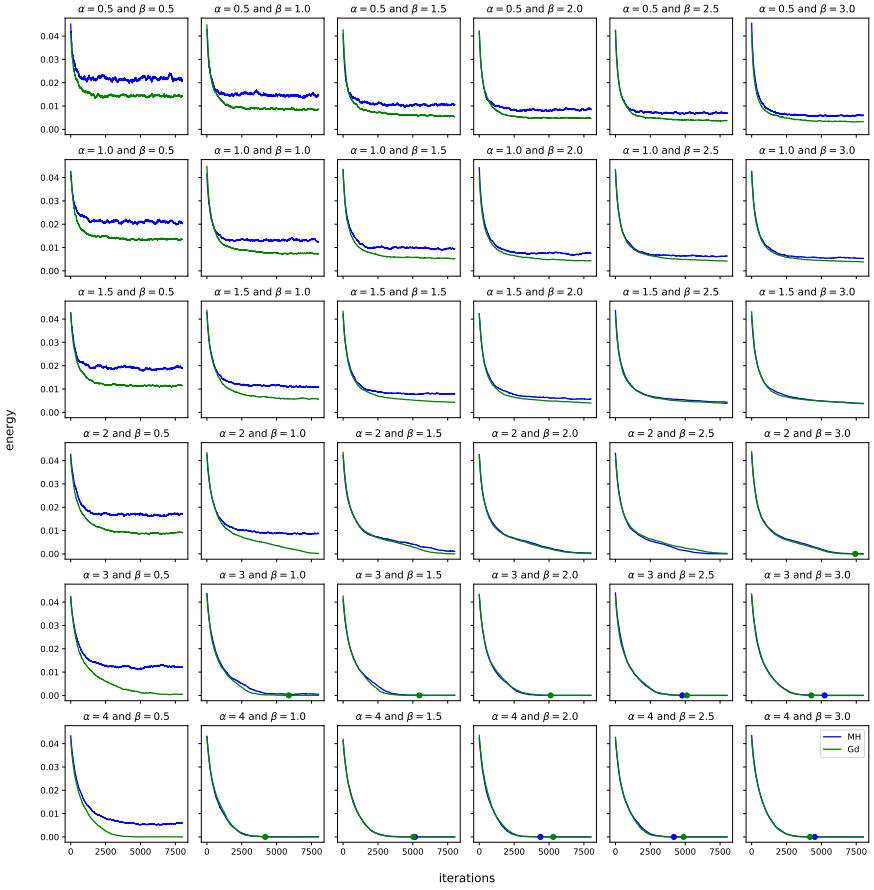

Figure 1: Normalized energy comparison between Metropolis-Hastings (MH) and Glauber dynamic (Gb) for the α and β configurations.

From Figure 1 we can observe the following points:

- small values of α: if the number of observations is not sufficiently high, we observe that for the range of β values we tested, we get stuck in a local minimum with high energy. It is known in practice that in order to get good performance, we should have $m >> n$.
- large values of α: on the other hand, if we get a lot of observations with respect to the number of unknowns, then we see that a big β helps to achieve slightly faster convergence. We also see, in a more general manner, that a bigger α helps achieve faster convergence with the β parameter fixed.

3.3 Simulated Annealing

3.3.1 Description

The technique of Simulated Annealing (SA) is taken from statistical physics. The technique is known as a thermal process: it involves heating and controlled cooling of a material to allow particles to arrange in a highly structured lattice. The difficulty remains in the cooling schedule where the cooling should be performed sufficiently slowly to obtain the ground state of the material. The analogy of the slow cooling from a probabilistic perspective is interpreted as a slow decrease in the probability of accepting worse solutions as the solution space is explored.

3.3.2 Cooling schedules

Therefore, the only feature changing along the iterations is the temperature. For our problem, this corresponds to the inverse temperature β. Two cooling schedules, both introduced by [1], will be tested as part of this project. The first one is the linear schedule,

$$\beta(t) = \beta_0 + Ct \tag{10}$$

where C is set to $C = \frac{5}{8000}$. This choice of value allows to increase the temperature by 5 each 8000 iterations. The second one is the exponential schedule,

$$\beta(t) = \beta_0 R^t \tag{11}$$

where R is set to $R = e^{\frac{\ln 4}{8000}}$ such that the temperature is multiplied by a factor of 4 every 8000 iterations.

Also, notice that the order of magnitude of the energy decreases as the number of observations increases. Hence, the β_0 should depend on the α value. For this reason, we fixed it as

$$\beta_0 = \frac{1}{\alpha} \tag{12}$$

for both schedules.

3.3.3 Results

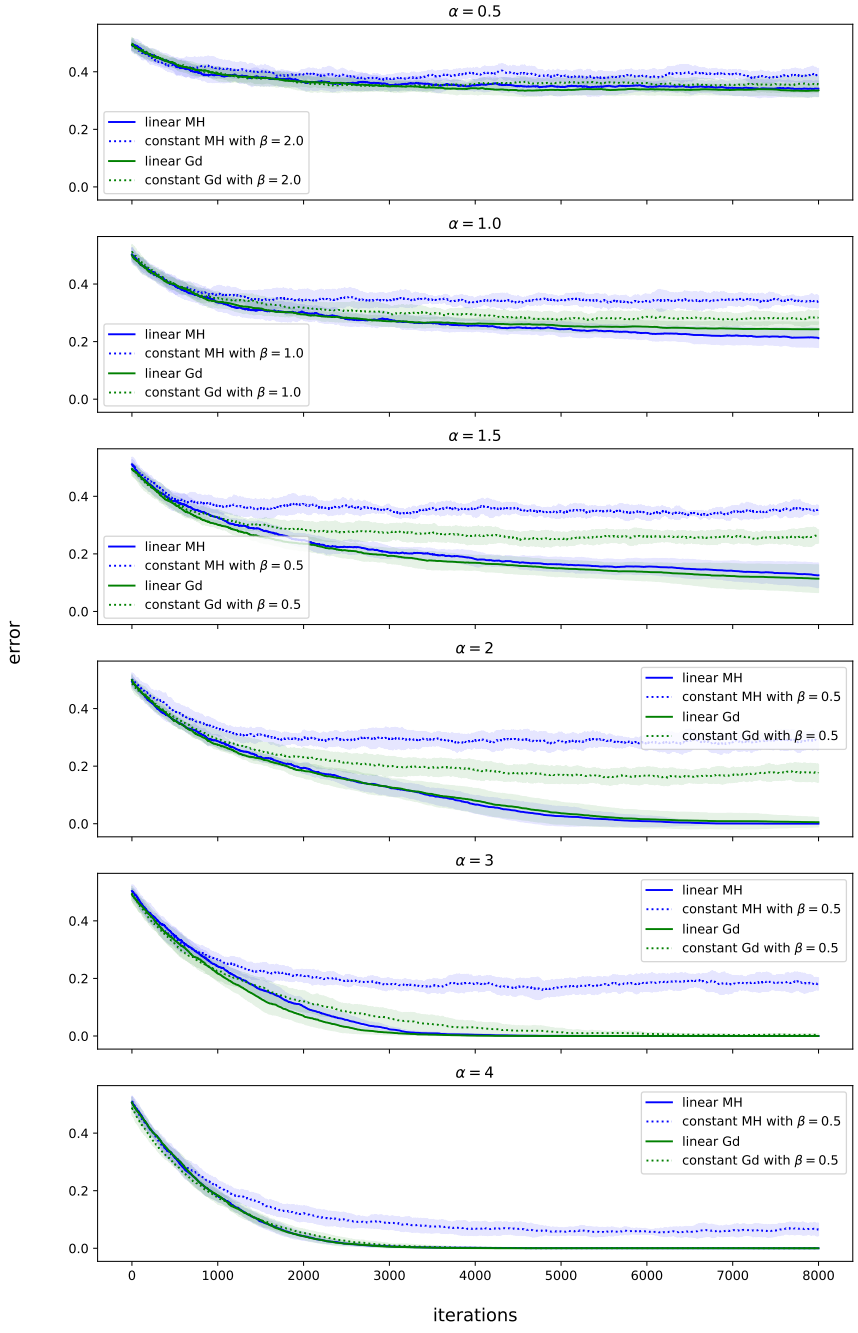

Figure 2: Error comparison between Metropolis-Hastings (MH) and Glauber dynamic (Gb) for the simulated annealing with linear schedule opposed to constant schedule (benchmark).

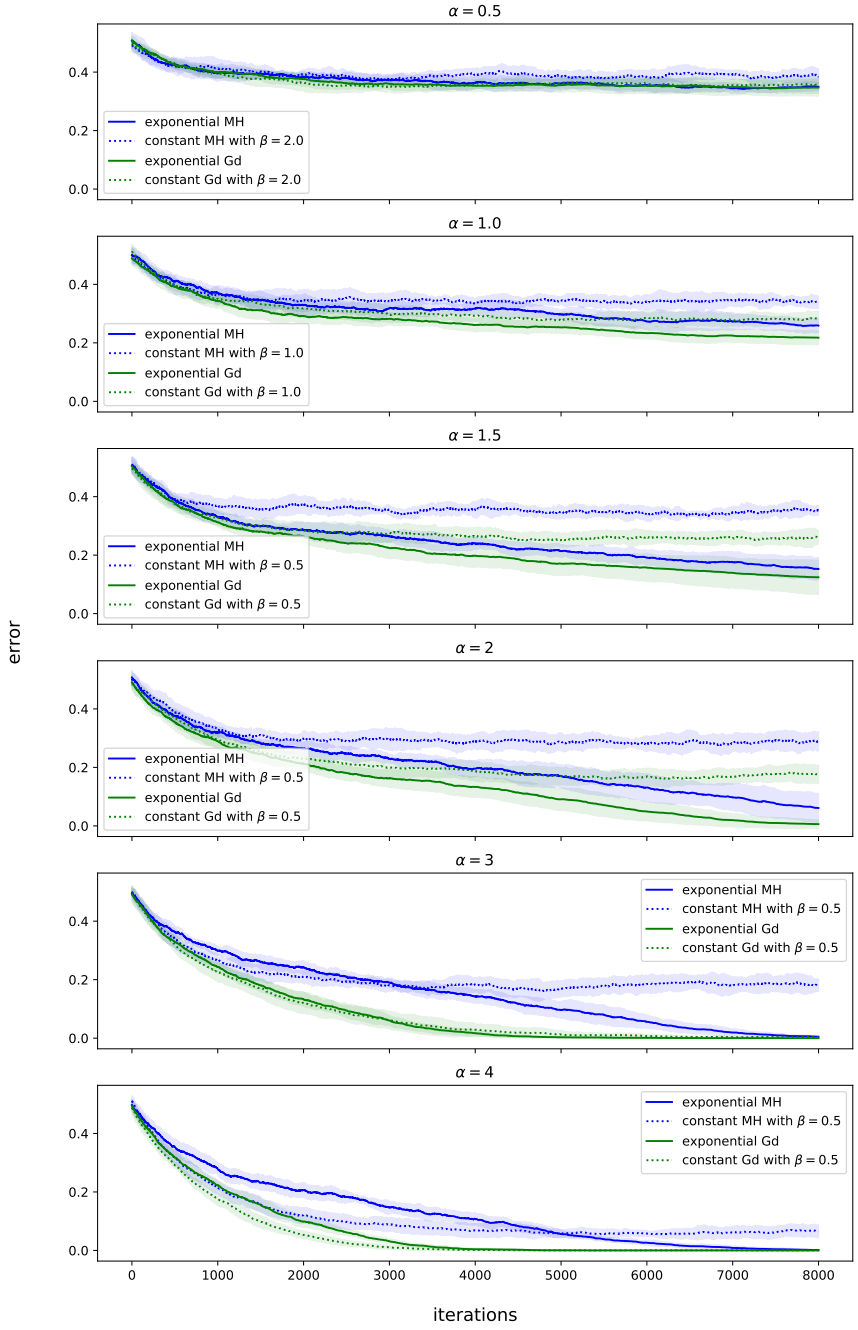

Figure 3: Error comparison between Metropolis-Hastings (MH) and Glauber dynamic (Gb) for the simulated annealing with exponential schedule opposed to constant schedule (benchmark).

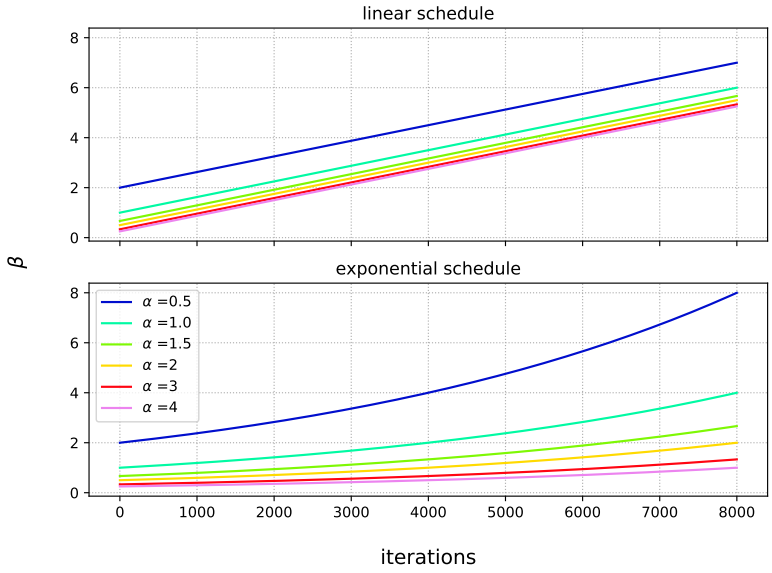

Figure 4: Schedules for linear and exponential simulated annealing implementations.

Overall, we observe that the Glauber dynamic algorithm performs better.

For the linear schedule, we see that for values $\alpha \in \{0.5, 4\}$, we get a small increase in performance with sometimes the same convergence rate. However, for the remaining values of alpha, we get a dramatic change, notably for the case $\alpha = 2$ where we have a perfect reconstruction before reaching the 8000 iterations.

For the exponential schedule, the same observations apply unless for $\alpha = 4$ for the Glauber dynamic algorithm where the constant schedule performs better. One of the reasons could be that the corresponding β_0 is very small and in contrast to the linear schedule, the change of the inverse temperature increases very slowly in the exponential schedule as we can see it from Figure 4. A way to prevent this problem, that is, to have a β_0 too small, would be to define a minimum threshold such that we avoid a change in inverse temperature that is too slow.

In comparison to a purely random guess, we see in Figure 5 and 6 that the α value should be very close to 0 for the Metropolis-Hastings and Glauber dynamic to behave the same way.

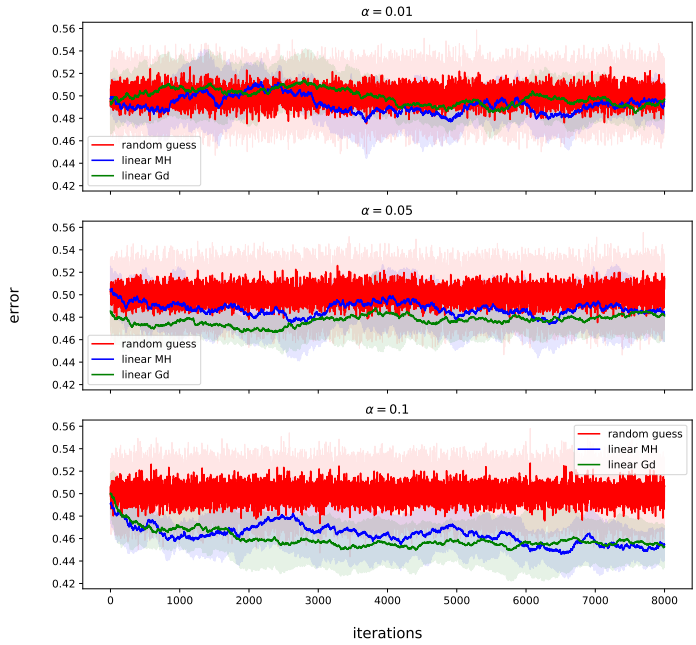

Figure 5: Limiting behaviour near $\alpha = 0$ of linear schedule.

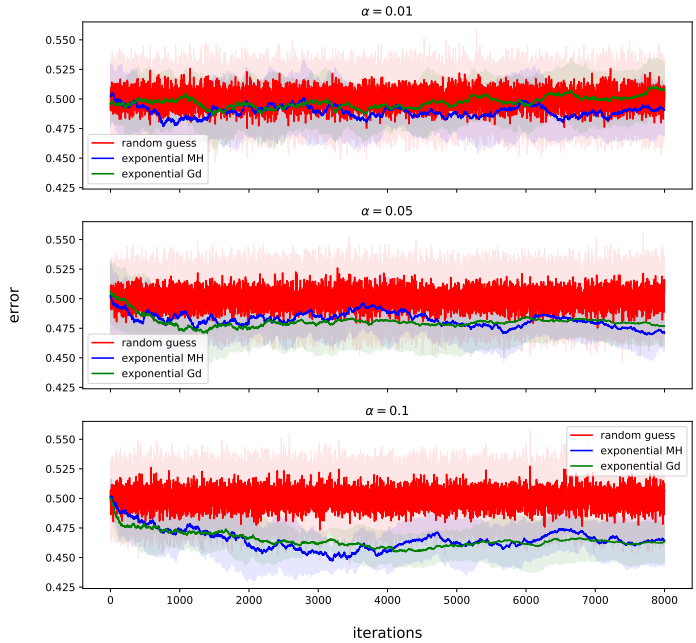

Figure 6: Limiting behaviour near $\alpha = 0$ of exponential schedule.

4 Conclusion

Finally, we applied the technique of Simulated Annealing for the Markov Chain Monte-Carlo method. We were able to compare performance between the Metropolis-Hastings and the Glauber dynamic algorithm for the MCMC method as well as measure the behaviour among a linear and exponential cooling schedule for the SA. We found that a rather simple setup for the SA technique can already outperform the rate of convergence of a constant schedule where the inverse temperature β is fixed.

5 Further work

For this type of project, a lot of resources are required in relation to the results we want to demonstrate. It becomes interesting to use multiprocessing in order to reduce the calculation time. An attempt to implement the project by exploiting multiprocessing quickly failed due to a resource conflict between the `multiprocessing` and `numpy` libraries. Indeed, `numpy` parallelizes its matrix multiplication to accelerate calculations. According to the parameters defined by default, the `multiprocessing` library which also exploits the parallelization of cores/threads by definition, conflicts with the same resources that `numpy` uses. Due to time constraints, we were unable to solve the problem. Indications for further investigation are commented in the code as well as the attempt itself.

There are many other ways to optimize the code. One of them is to define a stop condition during MCMC iterations. You will find in the code an optimized version of the MCMC. Note that it has not been used to simplify the comparison between the different configurations, otherwise we would have to compare and average experiments of different lengths, which can be problematic.

Of course, we can try different step sizes for both cooling schedules, and we can also try completely different ones.

References

[1] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulatied annealing. *Science*, 220(4598):671–680, 1983.