École Polytechnique Fédérale de Lausanne

University of Cambridge

Master's degree in Physics

Quantum-Enhanced Monte Carlo Markov Chain Optimization

Supervisors:

Prof. Giuseppe Carleo (EPFL)

Prof. Crispin Barnes (University of Cambridge)

Candidate:

Daniele Cucurachi

02 June 2023 - Academic year 2022/2023

Acknowledgments

Throughout the entire duration of my studies, I have received a great deal of support and assistance.

I would like to thank Prof. Giuseppe Carleo (EPFL), Prof. Crispin Barnes and the whole *Quantum Information Group* (University of Cambridge) for their help and assistance with my thesis work.

Furthermore, I would like to thank Arthur Strauss (EPFL/NUS), Baptiste Claudon (EPFL) and Efe Ersoy (ETH) for making my entire master's degree experience very fun and interesting, and Dr. Erika Schraner (Stanford) for regularly providing me with advice and support.

Lastly, I would like to thank Fabrizio Forte (EPFL/Harvard) for accompanying me on this entire journey not only as a colleague but also as a friend.

Abstract

Quantum-enhanced Monte Carlo Markov chain optimization

The combination of classical Monte Carlo Markov chains (MCMC) methods with quantum computers showed potential for achieving quantum advantage in sampling from the Boltzmann probability distribution, a computationally hard task arising in many diverse fields. Quantum-enhanced proposal distributions, defined by parameterized unitaries, can outperform classical strategies in proposing effective MCMC moves. However, it is crucial to carefully tune the values of the parameters defining these proposal distributions, as they determine the resulting advantage over the classical counterpart. A general optimization method becomes essential when considering parameterized unitaries for which is not possible to identify a reasonable parameter set. This could happen when adopting complicated proposal strategies depending on a large number of parameters, or simply when no prior or relevant information is available.

In the present thesis, we propose a general optimization algorithm that exploits estimates of the autocorrelation function of a certain observable, calculated over a set of samples, to optimize the parameters defining the proposal distribution. In chapter 1 we briefly review relevant concepts in the theory of discrete-time Markov chains, which are fundamental to understanding the optimization algorithm. Chapter 2 introduces quantum-enhanced Monte Carlo Markov chains algorithms, focusing on the findings presented in a recent paper. In chapter 3 the optimization algorithm is described in great detail, followed by the presentation and analysis of classical simulations in chapter 4. To conclude, the current limitations of the algorithm and its potential scalability with respect to the problem's size are discussed.

Contents

Chapter 1

Monte Carlo Markov chains

Monte Carlo Markov Chain (MCMC) methods are a class of algorithms constituting a fundamental tool in statistical physics and other fields for sampling from complex probability distributions. In the following, we briefly review the theory of discrete-time Markov chains in order to provide the reader with the essential concepts needed to understand the results presented in this thesis. Interested readers can refer to the original sources for a more detailed explanation [1, 2, 3, 4, 5, 6, 7].

1.1 Markov-chain sampling

When talking about sampling, we must distinguish between direct sampling and Markov-chain sampling. Despite the final goal being the same, these two approaches are fundamentally different. Direct sampling consists of being able to generate samples according to a target probability distribution, without the need for any intermediate step. When dealing with complex probability distributions, direct sampling can fail as these distributions may be computationally intractable. In other words, it may be too hard to calculate the individual probabilities and moments of the distribution. For instance, this is a well-known problem in statistical physics (and not only) where often the calculation of probability distributions' moments involves high-dimensional integrals. When direct sampling is not possible, Monte Carlo Markov chain methods come into play. However, the possibility of sampling from complex probability distribution comes with a cost. Markov chains do not sample from the target probability distribution straight away, they converge towards it and, even when convergence is reached, there can be a correlation between successive samples.

1.1.1 Discrete-time Markov chains

Let's consider a discrete set S, called *state space*, and a sequence of n random variables $\{X_i\}_i^n = \{X_0, X_1, ..., X_n\}$ with values in S. We call this sequence *discrete-time stochastic process* with state space S. In the following, we will refer to the elements of S by using $\boldsymbol{s}$ for a generic element and $\boldsymbol{s}^i, \boldsymbol{s}^j, \boldsymbol{s}^k$,... for specific elements. If a certain random variable belonging to the sequence is $X_t = \boldsymbol{s}^i$, the process is said to be in state $\boldsymbol{s}^i$ at time t and we represent it with $\boldsymbol{s}_t^i$.

Definition 1.1.1. Let $\{X_i\}_i^n$ be a discrete-time stochastic process with state space S. If, for all $t \geq 0$ and all the states $\boldsymbol{s}_0, \boldsymbol{s}_1, ..., \boldsymbol{s}_{t-1}, \boldsymbol{s}_t, \boldsymbol{s}_{t+1}$ we have:

$$P(X_{t+1} = \boldsymbol{s}_{t+1} | X_t = \boldsymbol{s}_t, X_{t-1} = \boldsymbol{s}_{t-1}, ..., X_0 = \boldsymbol{s}_0) = P(X_{t+1} = \boldsymbol{s}_{t+1} | X_t = \boldsymbol{s}_t) \tag{1.1}$$

we call the discrete-time stochastic process a *Markov chain*. The Markov chain is called a *homogeneous Markov chain (HMC)* if $P(X_{t+1} = \boldsymbol{s}_{t+1} | X_t = \boldsymbol{s}_t)$ does not depend on t.

Intuitively, Equation 1.1 defines Markov chains as stochastic processes with no memory of the past (Markov property). The next step depends only on the previous one, regardless of the past history of the chain.

1.1.2 The transition matrix

Definition 1.1.2. The matrix $P = \{p_{ij}\}$, where $\boldsymbol{s}^i, \boldsymbol{s}^j \in S$, whose entries are defined by:

$$p_{ij} = P(X_{t+1} = \boldsymbol{s}^j | X_t = \boldsymbol{s}^i) \tag{1.2}$$

is called *transition matrix*.

Transition matrices are *stochastic matrices*, meaning that:

$$p_{ij} \geq 0 \qquad and \qquad \sum_{\boldsymbol{s}^j \in S} p_{ij} = 1 \tag{1.3}$$

as the probabilities to visit all the possible states $\boldsymbol{s}^j \in S$ must sum up to 1.
A Markov chain at time t can be described using a discrete probability distribution over the elements of S, which we will call ν_t. Each entry $\nu_t(\boldsymbol{s}^k)$ represents the probability that the chain visits state $\boldsymbol{s}^k$ at time t:

$$\nu_t(\boldsymbol{s}^k) = P(X_t = \boldsymbol{s}^k) \tag{1.4}$$

We call *initial distribution* the probability distribution ν_0 defined as:

$$\nu_0(\boldsymbol{s}^k) = P(X_0 = \boldsymbol{s}^k) \tag{1.5}$$

where X_0 is the starting point of the chain's discrete sequence and $\boldsymbol{s}^k$ the initial state. The probability distribution describing a discrete-time homogeneous Markov chain at time t is given by:

$$\nu_t^T = \nu_0^T P^t \tag{1.6}$$

A Markov chain is therefore fully specified by two ingredients only: the transition matrix P and the initial distribution ν_0.

Definition 1.1.3. A probability distribution π is called *stationary distribution* of the transition matrix P if:

$$\pi^T = \pi^T P \tag{1.7}$$

A Markov chain whose initial distribution is its stationary distribution is called *stationary Markov chain.*

As we will see in the next section, Markov chains converge in variation to their stationary distribution π regardless of the initial distribution ν_0. However, the initial distribution has an important role in determining how fast the chain approaches the stationary distribution.

1.2 Markov-chains convergence

We will consider, moving forward, only *ergodic* (i.e. aperiodic, irreducible and positive recurrent) homogeneous Markov chains, meaning that they explore the full state space S without ending up trapped in a subset of it and without following a periodic pattern (interested readers can refer to [1] for a more formal definition). Before addressing the concept of convergence, we ought to define a metric to quantify the distance between probability distributions, which will come in handy later on to understand the type of convergence of interest.

Definition 1.2.1. Let S be a discrete state space, and let $\xi(\boldsymbol{s})$ and $\nu(\boldsymbol{s})$ be two probabilities distributions defined on S. The *total variational distance* is defined by:

$$||\xi - \nu||_{TV} = max_{\boldsymbol{s}^i \in S} |\xi(\boldsymbol{s}^i) - \nu(\boldsymbol{s}^i)| \tag{1.8}$$

Thanks to Definition 1.2.1, we can now formally define convergence in Markov-chains:

Theorem 1.2.1 (Convergence Theorem)**.** Let P be the transition matrix of an ergodic Markov chain, let π be its stationary distribution and S the state space. There exist two constants, $\alpha \in (0, 1)$ and $C \geq 0$, such that:

$$||\nu^T P^t - \pi^T||_{TV} \quad \leq \quad C\alpha^t \tag{1.9}$$

for all probabilities distributions ν defined on S.

1.2.1 The spectral gap

The Convergence Theorem tells us that the probability distribution describing the state of a Markov chain gets closer, step by step, to its stationary distribution in variational distance. This process can be explained by analyzing the spectrum of the transition matrix. Let us consider a transition matrix P. Transition matrices are stochastic matrices, therefore the eigenvalues λ_i always satisfy:

$$|\lambda_i| \;\leq\; 1 \tag{1.10}$$

In particular, ergodic chains always feature 1 as a simple eigenvalue:

$$\pi_1 P = 1P \qquad 1 = \lambda_1 > \lambda_2 \geq \lambda_3 \geq ... \geq -1 \tag{1.11}$$

where π_1 is the stationary distribution and λ_i are the eigenvalues sorted in decreasing order. Given a certain initial distribution ν_0, we can express it in terms of the eigenvectors π_i of P:

$$\nu_0^T = a_1\pi_1^T + a_2\pi_2^T + a_3\pi_3^T + ... = \sum_i a_i\pi_i^T \tag{1.12}$$

After t steps we have:

$$\nu_0^T P^t = a_1 1\pi_1^T + a_2(\lambda_2)^t\pi_2^T + a_3(\lambda_3)^t\pi_3^T + ... = \sum_i a_i(\lambda_i)^t\pi_i^T \tag{1.13}$$

Because of Equation 1.11, all the terms in the summation will go to zero except for λ_1, therefore leading to Equation 1.9. The slowest decaying mode corresponds to the one with the largest eigenvalue after $\lambda_1 = 1$, usually called λ_{SLEM}, where "$SLEM$" stands for *second largest eigenvalue magnitude.*

Definition 1.2.2. Let P be a transition matrix, the *absolute spectral gap* is defined as:

$$\delta \ = \ 1 - |\lambda_{SLEM}| \tag{1.14}$$

where:

$$\lambda_{SLEM} \ = \ sup\{|\lambda| : \lambda \ \in \ spec(P) \ \wedge \ \lambda \neq 1\} \tag{1.15}$$

Despite Markov chains' convergence being a complex process, the spectral gap provides a simple and effective way to describe it. Together with the initial distribution, it determines how fast the chain approaches the stationary distribution. Moreover, it is directly related to other important quantities such as the mixing time τ_ϵ. The mixing time represents the minimum time needed for a Markov chain to get within a certain distance (in variation) from the stationary distribution, regardless of the initial distribution ν_0.

Definition 1.2.3. Let P be a transition matrix with stationary distribution π. The *mixing time* τ_ϵ is given by:

$$\tau_\epsilon \ = \ inf\{t : \ sup_{\nu_0}||\nu_0^T P^t - \pi^T||_{TV} \leq \epsilon\} \tag{1.16}$$

τ_ϵ is bounded by the spectral gap:

$$(\delta^{-1} - 1)log(\frac{1}{2\epsilon}) \ \leq \tau_\epsilon \ \leq \delta^{-1}log(\frac{1}{\epsilon \ inf_{\boldsymbol{s}}(\pi(\boldsymbol{s}))}) \tag{1.17}$$

1.2.2 Detailed balance

We now know that Markov chains converge to their stationary distribution. However, if we want to leverage Markov chains to sample from complex probability distributions, we need to devise a strategy to define a transition matrix such that its stationary distribution is the target probability distribution we want to sample from. Let π^* be the target probability distribution and $p_{ij} = P(j|i)$ the probability to transition from state i to state j, defining the entries of the transition matrix. We can rewrite π^* in the following way:

$$\begin{aligned} \pi^*(\boldsymbol{s}^k) &= \sum_{\boldsymbol{s}^i \in S \setminus \boldsymbol{s}^k} \pi^*(\boldsymbol{s}^i)p_{ik} + \pi^*(\boldsymbol{s}^k)p_{kk} \\ \pi^*(\boldsymbol{s}^k)(1 - p_{kk}) &= \sum_{\boldsymbol{s}^i \in S \setminus \boldsymbol{s}^k} \pi^*(\boldsymbol{s}^i)p_{ik} \end{aligned} \tag{1.18}$$

From the normalization condition defined in Equation 1.3, we have that $1 - p_{kk} = \sum_{\boldsymbol{s}^i \in S \setminus \boldsymbol{s}^k} p_{ki}$. Introducing it in the previous equation, we obtain:

$$\pi^*(\boldsymbol{s}^k)(\sum_{\boldsymbol{s}^i \in S \setminus \boldsymbol{s}^k} p_{ki}) = \sum_{\boldsymbol{s}^i \in S \setminus \boldsymbol{s}^k} \pi^*(\boldsymbol{s}^i)p_{ik} \tag{1.19}$$

One way to satisfy the equation is the following:

$$\pi^*(\boldsymbol{s}^k)p_{ki} \ = \ \pi^*(\boldsymbol{s}^i)p_{ik} \quad \forall \ \boldsymbol{s}^i, \boldsymbol{s}^k \ \in \ S \tag{1.20}$$

Also known as the *detailed balance condition.* In summary, if we manage to define a transition matrix P whose entries satisfy Equation 1.20 for a desired probability distribution π^*, we know the corresponding Markov chain will converge in variation to that distribution. A Markov chain whose transition matrix satisfies the detailed balance condition is called *reversible.*

1.2.3 The Metropolis-Hastings algorithm

Given a certain desired probability distribution π^* and a transition matrix $P = \{p_{ij}\}$, they can always be reconciled with the detailed balance condition by adopting the following strategy:

$$p_{ij} \;=\; \underbrace{Q(j|i)}_{\text{propose } \boldsymbol{s}^i \to \boldsymbol{s}^j} \; \underbrace{A(j|i)}_{\text{accept } \boldsymbol{s}^i \to \boldsymbol{s}^j} \tag{1.21}$$

where we split the probabilities p_{ij} into proposal probability $Q(j|i)$ and acceptance probability $A(j|i)$. As the p_{ij} must satisfy Equation 1.20, the acceptance probabilities become:

$$\frac{A(j|i)}{A(i|j)} = \frac{\pi^*(\boldsymbol{s}^j)Q(i|j)}{Q(j|i)\pi^*(\boldsymbol{s}^i)} \tag{1.22}$$

leading to the so-called *Metropolis-Hastings algorithm*:

$$A(j|i) \;=\; min(1,\; \frac{\pi^*(\boldsymbol{s}^j)Q(i|j)}{Q(j|i)\pi^*(\boldsymbol{s}^i)}) \tag{1.23}$$

The Metropolis-Hastings algorithm is a general and fairly simple approach to ensure convergence to the desired probability distribution. However, it does not give any information about the convergence rate or the correlation between successive samples, which are fundamental aspects of Markov chain sampling. They both depend on the quality of the proposal distribution Q. Therefore, even though in principle we could use any Q and still satisfy the detailed balance condition, we want to aim for a proposal distribution that resembles the target distribution. A random proposal strategy could lead to many rejections and consequently increase the correlation between successive samples.

1.3 Statistical errors and autocorrelations in MCMC

As already mentioned, Markov chains are a powerful tool to evaluate high-dimensional integrals. Let's consider the following integral:

$$\langle f \rangle = \int_S f(\boldsymbol{s})p(\boldsymbol{s})d\boldsymbol{s} \tag{1.24}$$

where $f : S \to \mathbb{R}$ is a real-valued function (also called observable) on the state space S and $p(\boldsymbol{s})$ a probability distribution defined on S as well. By sampling n times from $p(\boldsymbol{s})$ we can approximate the integral:

$$\langle f \rangle = \int_S f(\boldsymbol{s})p(\boldsymbol{s})d\boldsymbol{s} \quad\approx\quad \bar{f} = \frac{1}{n}\sum_i^n f(X_i) \tag{1.25}$$

where X_i is the i-th sample. If these samples are independent, we know from the Central Limit Theorem (CLT) that the estimator variance is given by:

$$Var(\bar{f}) = \frac{1}{n}Var_{p(\boldsymbol{s})}(f) \tag{1.26}$$

meaning that the standard error of the estimator $\sqrt{Var}(\bar{f})$ goes down as $1/\sqrt{n}$. However, Markov chain samples are not independent in general. Therefore, when using Markov chain sampling to approximate complex integrals we need to take into account the correlation by including an additional factor in the estimator variance: the integrated autocorrelation time. Equation 1.26 then becomes:

$$Var(\bar{f}) = \frac{\tau_{int,f}}{n} Var_{p(\boldsymbol{s})}(f) \tag{1.27}$$

where $n/\tau_{int,f}$ is in practice an estimate of the number of effectively independent samples. In simple words, the integrated autocorrelation time tells us how many Markov chain steps we need after sampling X_i such that the state sampled at X_i is forgotten.

1.3.1 The autocorrelation time

Before giving a formal definition of the integrated autocorrelation time, we ought to define the autocorrelation function (ACF) and the exponential autocorrelation time. We consider a stationary reversible Markov chain $\{X_i\}_i^n$, with transition matrix P. As the chain is stationary, we have:

$$\langle f_i \rangle = \langle f_{i+l} \rangle = \mu \qquad \forall \; i, l \in \mathbb{N} \tag{1.28}$$

where $f_i = f(X_i)$ and X_i the i-th Markov chain sample. The *unnormalized autocorrelation function* is then given by:

$$C_{ff}(l) = \langle (f_i - \langle f_i \rangle)(f_{i+l} - \langle f_{i+l} \rangle) \rangle = \langle (f_i - \mu)(f_{i+l} - \mu) \rangle = \langle f_0 f_l \rangle - \mu^2 \tag{1.29}$$

while the *normalized autocorrelation function*:

$$\rho_{ff} = \frac{C_{ff}(l)}{C_{ff}(0)} \qquad C_{ff}(0) = Var_\pi(f) \tag{1.30}$$

Intuitively, the ACF measures the correlation between Markov chain samples separated by a lag l. For large enough lags, the ACF typically decays exponentially:

$$C_{ff}(l) \sim e^{-\frac{l}{\tau_{exp,f}}} \tag{1.31}$$

where $\tau_{exp,f}$ is the relaxation time of the slowest decaying mode on which f has a non-zero projection (interested readers are encouraged to refer to [5] for a more detailed discussion). We can now define the *exponential autocorrelation time* τ_{exp}:

$$\begin{gathered} \tau_{exp,f} = \lim_{l \to +\infty} \frac{l}{-log(|\rho_{ff}(l)|)} \\ \tau_{exp} = sup_f(\tau_{exp,f}) \end{gathered} \tag{1.32}$$

where τ_{exp} is the relaxation time of the slowest mode in the system, whose corresponding eigenvalue is λ_{SLEM}. It can be shown that:

$$R \; = \; e^{-1/\tau_{exp}} \tag{1.33}$$

with R being the spectral radius of P:

$$R \; = \; inf\{r : \; spec(P) \setminus 1 \subset \{\lambda : \; |\lambda| \leq r\}\} \tag{1.34}$$

As the considered Markov chain is reversible, i.e. satisfies the detailed balance condition, the spectrum of P is real and the eigenvalues lie in the interval $[-1, 1]$. We can therefore rewrite Equation 1.33 as:

$$\tau_{exp} \;=\; \frac{-1}{log(\lambda_{SLEM})} \tag{1.35}$$

which provides a direct link between the autocorrelation time and the spectral gap $\delta = 1 - \lambda_{SLEM}$. Finally, the *integrated autocorrelation time* is defined as:

$$\tau_{int,f} = \frac{1}{2}\sum_{l=-\infty}^{+\infty} \rho_{ff}(l) = \frac{1}{2} + \sum_{l=1}^{+\infty} \rho_{ff}(l) \tag{1.36}$$

Despite being related, $\tau_{int,f}$ and τ_{exp} play different roles. On the one hand, $\tau_{int,f}$ determines the statistical error affecting MCMC estimates, as we saw in Equation 1.27. On the other hand, τ_{exp} provides insights into the chain convergence process and gives a measure of the initialization bias, i.e. the initial transient of the chain whose samples do not represent the stationary distribution. As they both are fundamental quantities in Monte Carlo Markov chain simulations, several estimators can be found in the literature [8, 9, 5].

1.3.2 Estimators

Sokal proposed a simple approach to estimate the integrated autocorrelation time based on a set of samples [5]. It consists in estimating the ACF first, in order to then calculate τ_{int} as per definition. We consider a set of samples $\{f_i\}_i^n$, where $f_i = f(X_i)$ and X_i the i-th (stationary) Markov chain sample. The unnormalized ACF estimator is given by:

$$\hat{C}_{ff}(l) = \frac{1}{n-l}\sum_{i}^{n-l}(f_i - \mu)(f_{i+l} - \mu) \tag{1.37}$$

where μ is the observable mean:

$$\mu = \langle f \rangle \tag{1.38}$$

In case the observable mean is unknown, the estimator is redefined as:

$$\hat{\hat{C}}_{ff}(l) = \frac{1}{n-l}\sum_{i}^{n-l}(f_i - \bar{f})(f_{i+l} - \bar{f}) \tag{1.39}$$

where:

$$\bar{f} = \frac{1}{n}\sum_{i}^{n} f_i \tag{1.40}$$

introducing a bias which decays as $1/n$. Both $\hat{C}_{ff}(l)$ and $\hat{\hat{C}}_{ff}(l)$ are consistent estimators:

$$\lim_{n\to\infty} \hat{C}_{ff}(l) = \lim_{n\to\infty} \hat{\hat{C}}_{ff}(l) = C_{ff}(l) \tag{1.41}$$

however, it is important to underline that $\hat{C}_{ff}(l)$ and $\hat{\hat{C}}_{ff}(l)$ are random variables, affected therefore by a statistical error which is usually quantified using the standard deviation (standard error). The estimator variance is given by [5, 10, 11]:

$$Var(\hat{C}_{ff}(l)) = (\frac{1}{n}) \sum_{m=-\infty}^{\infty} [C_{ff}(m)^2 + C_{ff}(m+l)C_{ff}(m-l) + \kappa(l,m,m+l)] + o(\frac{1}{n})$$

$$\begin{aligned} Cov(\hat{C}_{ff}(l), \hat{C}_{ff}(u)) = (\frac{1}{n}) \sum_{m=-\infty}^{\infty} [C(m)_{ff}C_{ff}(m+u-l) + C_{ff}(m+u)C_{ff}(m-l) + \\ \kappa(l,m,m+u)] + o(\frac{1}{n}) \end{aligned} \tag{1.42}$$

where $l, u \geq 0$ and κ is the connected 4-points autocorrelation function:

$$\begin{aligned} \kappa(r,s,l) = \langle (f_i - \mu)(f_{i+r} - \mu)(f_{i+s} - \mu)(f_{i+l} - \mu) \rangle \\ -C_{ff}(r)C_{ff}(l-s) - C_{ff}(s)C_{ff}(l-r) - C(l)_{ff}C(s-r)_{ff} \end{aligned} \tag{1.43}$$

As the terms in the summation in Equation 1.42 do not depend on n, we have that the variance is inversely proportional to the number of samples (as the CLT suggests), leading to a standard error that goes down as $1/\sqrt{n}$ and dominates over the bias for large n. We can now estimate the normalized ACF:

$$\hat{\rho}(l) \;=\; \hat{\hat{C}}_{ff}(l)/\hat{\hat{C}}_{ff}(0) \tag{1.44}$$

Finally, the integrated autocorrelation time estimator is given by:

$$\hat{\hat{\tau}}_{int,f} \;=\; \frac{1}{2} \sum_{l=-(n-1)}^{+(n-1)} \hat{\rho}(l) \tag{1.45}$$

Even though Equation 1.45 it seems a natural way to estimate $\tau_{int,f}$, the estimator variance does not go to zero as n (number of samples) diverges [11]. This happens because the noise coming from the terms $\hat{\rho}(l)$ at large lags ($l >> \tau_{exp}$) dominates over the $\tau_{int,f}$ estimate. A practical solution consists in including a function $h(l)$ that cuts off large lag terms, redefining the estimator in the following way:

$$\hat{\hat{\tau}}_{int,f} \;=\; \frac{1}{2} \sum_{l=-(n-1)}^{+(n-1)} h(l)\hat{\rho}(l) \qquad h(l) \begin{cases} \approx 1 & |l| \lesssim \tau_{exp} \\ \approx 0 & |l| >> \tau_{exp} \end{cases} \tag{1.46}$$

A common choice for $h(l)$ is :

$$h(l) \begin{cases} = 1 & |l| \leq M \\ = 0 & |l| > M \end{cases} \tag{1.47}$$

where M is a parameter that has to be tailored to the specific problem. Equation 1.47 makes the estimator consistent but introduces a bias at the same time. The value of the parameter M controls the trade-off between the two:

$$bias(\hat{\hat{\tau}}_{int,f}) \;=\; -\frac{1}{2} \sum_{|l|>M} \rho(l) + o(\frac{1}{n}) \qquad Var(\hat{\hat{\tau}}_{int,f}) \;\approx\; \frac{2(2M+1)}{n} \tau_{int,f}^2 \tag{1.48}$$

Chapter 2

Quantum-enhanced Monte Carlo Markov chain

In the following chapter, we present a hybrid algorithm, developed by Layden et al. [12], that samples from the Boltzmann probability distribution of classical Ising models outperforming classical Monte Carlo Markov chain methods at low temperatures. We start by defining the specific sampling problem addressed by the algorithm, followed by a concise description of the classical strategies commonly adopted. We then present the quantum-enhanced Monte Carlo Markov chain (QMCMC) algorithm and, finally, report the simulations and experimental results from the paper. The concepts and findings presented in this chapter are fundamental to understanding the content of chapter 3.

2.1 The sampling problem

Before diving into the sampling problem, we ought to briefly introduce the Ising model. The Ising model is a statistical physics model introduced by E. Ising in 1925 to study phase transitions and critical phenomena. It consists of a number N of spins s_i, binary variables assuming either the value $+1$ or -1. The spins, arranged on a lattice, interact with each other according to the coupling strengths $\{J_{ij}\}_{i>j=1}^{N}$. They can also interact individually with an external magnetic field, represented by the coefficients $\{h_i\}_{i=1}^{N}$. An instance of the model is fully defined by the couplings J_{ij} and the fields h_i. The energy function is defined as:

$$E(\boldsymbol{s}) = -\sum_{i>j=1}^{N} J_{ij} s_i s_j - \sum_{i=1}^{N} h_i s_i \tag{2.1}$$

assigning some scalar value E to every spin configuration $\boldsymbol{s} = \{s_1, s_2, ..., s_N\}$. The corresponding Boltzmann probability, representing the probability of observing a configuration $\boldsymbol{s}$ given a certain value of the parameter T (usually called temperature), is defined as:

$$\mu(\boldsymbol{s}) = \frac{1}{Z} e^{-E(\boldsymbol{s})/T} \qquad\qquad Z = \sum_{\boldsymbol{s} \in S} e^{-E(\boldsymbol{s})/T} \tag{2.2}$$

where S is the state space containing all the possible 2^N spins configurations and Z is the partition function normalizing the probabilities.

Sampling from the Boltzmann distribution of classical Ising models turns out to be a useful task in many diverse fields. However, due to the exponentially large state space ($|S| =$

2^N), the partition function is in general computationally intractable. To overcome this issue, various approaches have been developed, with Monte Carlo Markov Chain (MCMC) methods being the most widely adopted solution.

2.2 Classical MCMC proposal strategies

In subsection 1.2.3 we highlighted how the proposal strategy, defining the transition matrix of the chain, is crucial in determining the convergence rate of a Markov chain to the target probability distribution. Among many classical proposal strategies, the most popular and commonly used are:

- *local proposal strategy:* new configurations $\boldsymbol{s'}$ are proposed by simply flipping a single uniformly random spin in the current spin configuration $\boldsymbol{s}$ ($s_i \to -s_i$)
- *uniform proposal strategy:* new configurations $\boldsymbol{s'}$ are selected randomly among all the possible $\boldsymbol{s} \in S$

These two strategies are not suited for sampling from the Boltzmann distribution of Ising models at low temperatures, in particular when couplings J_{ij} and fields h_i are random variables following no specific pattern (the so-called spin glasses). Indeed, in these cases, Equation 2.1 defines a rugged energy landscape with several local minima that can be far away in Hamming distance, as shown in Figure 2.1b. Both the *local* and the *uniform* proposal strategies fail to quickly explore the full landscape and get often stuck in local minima, slowing down the convergence. The first one tends to propose states that are close in Hamming distance, featuring similar energies. On the one hand, they are likely to be accepted, on the other a large number of steps are required in order to escape local minima.

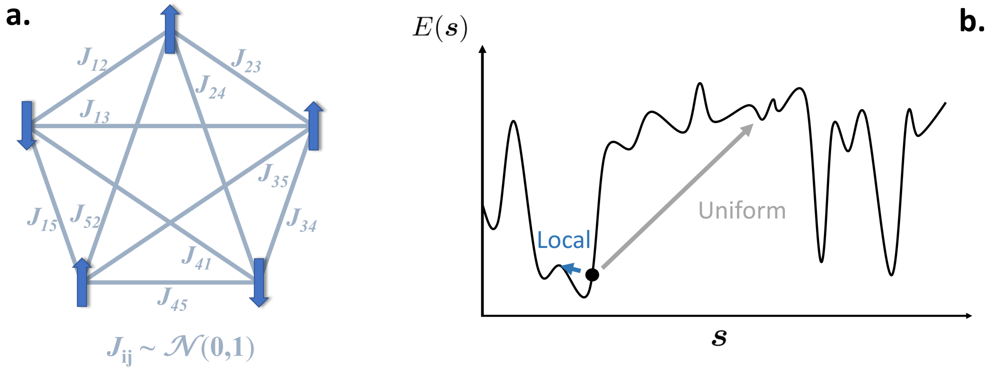


Figure 2.1: a. Example of a spin glass model: $N = 5$ fully connected Ising model instance with normally distributed couplings. The blue arrows represent the spins while the edges are the couplings J_{ij}. **b.** Example of a typical spin glass model energy landscape. Classical proposal strategies are depicted for illustration purposes.

The *uniform* strategy is able to reach faraway configurations in Hamming distance. However, the random proposal does not prioritize configurations corresponding to energy minima. Since, at low temperatures, high-energy state probabilities are much smaller than low-energy ones, this strategy leads to many rejections and consequently a slow convergence and a high correlation.

2.3 Quantum-enhanced MCMC algorithms

Layden et al. recently proposed a hybrid quantum-classical algorithm (Algorithm 1)[1] that samples from the Boltzmann probability distribution of Ising models outperforming classical methods at low temperatures ($T \leq 10$). The algorithm exploits the properties of certain quantum spin glass Hamiltonians, which have been shown to exhibit eigenvectors with concentrated components on low-energy spin states, in order to improve the proposal strategy [13, 14]. Indeed, these Hamiltonians are likely to promote transitions between low-energy states even when far away in Hamming distance, allowing to effectively navigate the energy landscape while prioritizing low-energy states.

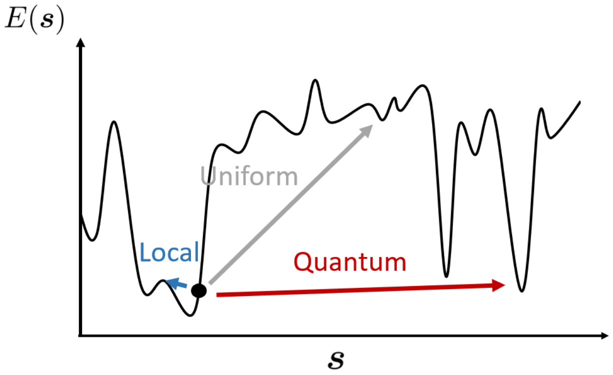


Figure 2.2: Quantum and classical proposal strategies comparison. In the quantum case, the proposal strategy prioritizes low-energy states, resembling the Boltzmann probability distribution at low temperatures.

The algorithm starts, like its classical counterpart, by selecting a random initial spin configuration $\boldsymbol{s}$. The spin configuration $\boldsymbol{s}$ is mapped onto a computational basis state $|\boldsymbol{s}\rangle$, eigenstate of the quantum Hamiltonian:

$$H_{prob} = -\sum_{i>j}^{N} J_{ij} Z_i Z_j - \sum_{i}^{N} h_i Z_i \tag{2.3}$$

The quantum state is then evolved under the following parametrized Hamiltonian:

$$H(\gamma) = (1-\gamma)\alpha H_{prob} + \gamma H_{mix} \qquad H_{mix} = \sum_{i}^{N} X_i \tag{2.4}$$

where $\gamma \in [0,1]$ is a parameter controlling the weight of H_{mix}, $\alpha = \sqrt{N}/(\sum_{i>j}^{N} J_{ij}^2 + \sum_{i}^{N} h_i^2)$ is a factor ensuring that H_{prob} and H_{mix} share the same order of magnitude, and $Z_i = I_1 \otimes ... \otimes \hat{Z} \otimes ... \otimes I_N$ are Pauli operators acting on the i-th spin (same for X_i). Note that H_{mix} is responsible for generating transitions between different H_{prob} eigenstates. Finally, the evolved state $|\psi\rangle$:

$$|\psi\rangle = U(\gamma, t)|\boldsymbol{s}\rangle = e^{-iH(\gamma)t}|\boldsymbol{s}\rangle \tag{2.5}$$

is measured in the computational basis. The measurement result $|\boldsymbol{s'}\rangle$, mapped back onto the corresponding spin configuration $\boldsymbol{s'}$, constitutes the proposal for the successive step,

[1] We have chosen to employ a universal pseudocode approach throughout this thesis. The reader can easily interpret the general algorithms and translate them into their preferred programming language.

which can be either accepted or rejected depending on the associated Metropolis-Hastings probability. The aforementioned routine is then repeated until convergence is reached.

Although this algorithm has the potential to outperform classical methods, it is fundamental to carefully choose the values of the parameters γ and t as they determine the quality of the quantum proposal strategy affecting the resulting advantage. In Algorithm 1, in each iteration, γ and t are sampled uniformly at random from the intervals [0.25, 0.6] and [2, 20] respectively. This approach has been adopted because the optimal parameter values feature a complicated dependence on the specific model instance and no straightforward optimization method is available. While random sampling over some reasonable intervals already enables quantum advantage, more advanced strategies may lead to even better results. In particular, an optimization method becomes essential when considering cases where is not possible to identify reasonable intervals a priori, for example when adopting more complicated proposal strategies depending on a large number of parameters. This topic will be explored in the next chapter.

Algorithm 1 Quantum-enhanced Monte Carlo Markov chain

```
$\boldsymbol{s} \leftarrow$ random($\{\boldsymbol{s} : \boldsymbol{s} \in S\}$)    ▷ selecting initial configuration uniformly at random
**repeat**
    **Propose state**
    $\gamma \leftarrow$ random uniform ($[0.2, 0.6]$)
    $t \leftarrow$ random uniform ($[2, 10]$)
    $|\boldsymbol{s}\rangle \leftarrow \boldsymbol{s}$    ▷ mapping spin configuration onto computational basis state
    $|\psi\rangle \leftarrow U(\gamma, t)|\boldsymbol{s}\rangle$    ▷ where $U(\gamma, t) = e^{-iH(\gamma)t}$
    $|\boldsymbol{s'}\rangle \leftarrow$ measure $|\psi\rangle$    ▷ measuring $|\psi\rangle$ in the computational basis
    $\boldsymbol{s'} \leftarrow |\boldsymbol{s'}\rangle$    ▷ mapping computational basis state onto spin configuration
    **Accept or reject state**
    $A \leftarrow \min(1, e^{(E(\boldsymbol{s})-E(\boldsymbol{s'}))/T})$
    **if** $A \geq$ random uniform ($[0, 1]$) **then**
        $\boldsymbol{s} \leftarrow \boldsymbol{s'}$
    **end if**
**until** converged
```

The transition matrix describing the quantum-enhanced Markov chain is given by:

$$P(\boldsymbol{s'}|\boldsymbol{s}) = A(\boldsymbol{s'}|\boldsymbol{s})Q_{(\boldsymbol{s'}|\boldsymbol{s})}(\gamma, t) + \delta_{\boldsymbol{ss'}} \sum_{\boldsymbol{s''} \in S} (1 - A(\boldsymbol{s''}|\boldsymbol{s}))Q_{(\boldsymbol{s''}|\boldsymbol{s})}(\gamma, t) \tag{2.6}$$

where $Q_{(\boldsymbol{s'}|\boldsymbol{s})}(\gamma, t)$ is the quantum proposal distribution, with $U(\gamma, t)$ the proposed parametrized anstaz:

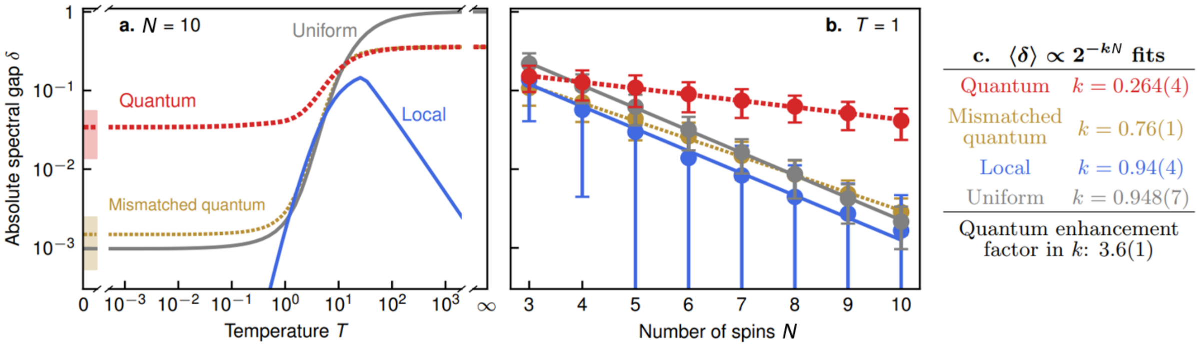
$$Q_{(s'|s)}(\gamma, t) = |\langle \boldsymbol{s'}|U(\gamma, t)|\boldsymbol{s}\rangle|^2 \qquad U(\gamma, t) = e^{-iH(\gamma)t} \tag{2.7}$$

and $A(\boldsymbol{s'}|\boldsymbol{s})$ is the Metropolis-Hastings acceptance probability:

$$A(\boldsymbol{s'}|\boldsymbol{s}) = min(1, \frac{\mu(\boldsymbol{s'})Q_{(\boldsymbol{s}|\boldsymbol{s'})}(\gamma, t)}{\mu(\boldsymbol{s})Q_{(\boldsymbol{s'}|\boldsymbol{s})}(\gamma, t)}) \tag{2.8}$$

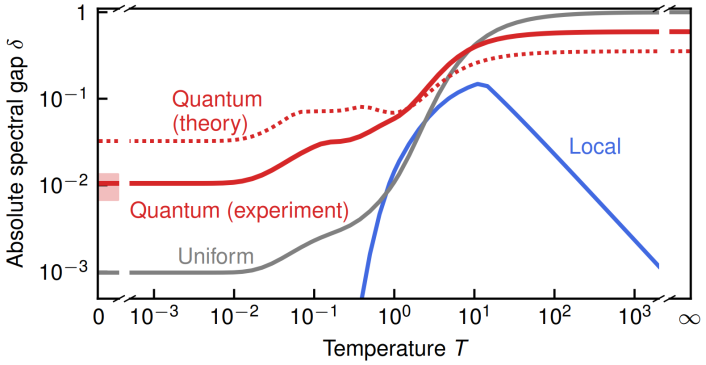
Note that the factor $\sum_{\boldsymbol{s''} \in S}(1 - A(\boldsymbol{s''}|\boldsymbol{s}))Q_{(\boldsymbol{s''}|\boldsymbol{s})}(\gamma, t)$ in Equation 2.6 is needed to ensure that probabilities in the transition matrix columns sum up to 1. The unitary $U(\gamma, t)$ defining the quantum proposal distribution is chosen to be symmetric such that is possible to easily calculate the acceptance probabilities:

$$Q_{(\boldsymbol{s'}|\boldsymbol{s})}(\gamma,t) = Q_{(\boldsymbol{s}|\boldsymbol{s'})}(\gamma,t) \quad \Rightarrow \quad A = min(1, \frac{\mu(\boldsymbol{s'})}{\mu(\boldsymbol{s})}) = min(1, e^{(E(\boldsymbol{s})-E(\boldsymbol{s'}))/T}) \tag{2.9}$$



adapted from [12]

Figure 2.3: Average spectral gap δ simulations for both classical and quantum proposal strategies. In the quantum case, the spectral gap is numerically calculated by diagonalizing the transition matrix defined in Equation 2.6. The plots show the average over 500 Ising model instances, fully connected with normally distributed couplings $J_{ij} \sim \mathcal{N}(0,1)$ and fields $h_i \sim \mathcal{N}(0,1)$. The colored error bands represent the standard deviations while solid and dashed lines simply help with visibility. **a.** Comparison of the temperature dependencies of quantum and classical proposal strategies. At low temperatures ($T \leq 10$) the quantum proposal shows a clear advantage over the classical competitors. **b.** Scaling of the spectral gap with respect to the problem size (number of spins N) for the various proposal strategies. **c.** Spectral gap's exponential decay rate fit results.



adapted from [12]

Figure 2.4: Average spectral gap δ experimental results. In the quantum case, a large number $(5,76 \times 10^7)$ of quantum transitions $|\boldsymbol{s}\rangle \rightarrow |\boldsymbol{s'}\rangle$ has been recorded in experiments and used to estimate $Q_{(\boldsymbol{s}|\boldsymbol{s'})}(\gamma,t)$. The experiments verified the simulations' results, demonstrating that quantum proposal distributions have the potential to bring a quantum advantage.

Reported in Figure 2.3, numerical simulations of the spectral gap averaged over 500 different instances of the Ising model (fully connected with randomly distributed couplings

$J_{ij} \sim \mathcal{N}(0,1)$ and fields $h_i \sim \mathcal{N}(0,1)$) showed that the quantum proposal distributions lead to significantly faster convergence at low temperatures ($T \leq 10$) compared to classical methods. Subsequent experiments conducted on IBM's quantum computers verified these results (Figure 2.4), demonstrating that quantum proposal distributions are able to bring an advantage in the NISQ era. The relatively shallow circuits required to implement the algorithm render it suitable for modern noisy devices.

Chapter 3

Quantum-enhanced MCMC optimization

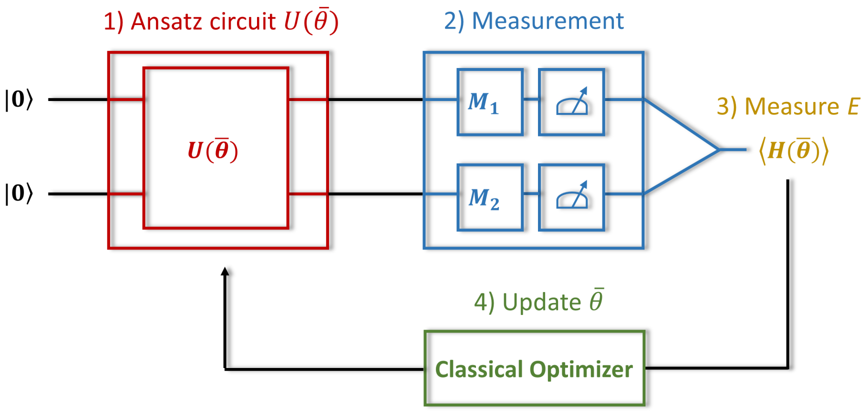
In the previous chapter, we presented a hybrid algorithm combining classical MCMC methods with quantum-enhanced proposal distributions defined by a parameterized unitary $U(\gamma, t)$ (Equation 2.7). Apart from the symmetry requirement (Equation 2.9), which is necessary to easily calculate the acceptance probabilities, the choice of the unitary is unconstrained. There are several promising options, such as exploiting the reverse quantum annealing [15], however, it is likely they will depend on a certain set of parameters $\bar{\theta}$. The unitary $U(\gamma, t)$ is a perfect example: it depends on γ, controlling the relative weights of the two Hamiltonians H_{prob} and H_{mix}, and t, the evolution time. It is, therefore, crucial to develop a general approach to optimize these parameters. In the following, we present a general algorithm that addresses this problem and we demonstrate its effectiveness by optimizing the parametrized unitary $U(\gamma, t)$. Firstly, we outline the general structure of the algorithm, followed by a thorough analysis of each individual step. We then compare the performances of the optimized quantum proposal strategy with the results obtained by Layden et al. [12]. To conclude, we discuss the limitations of the optimization algorithm and suggest potential areas of future research.

3.1 QMCMC optimization algorithm

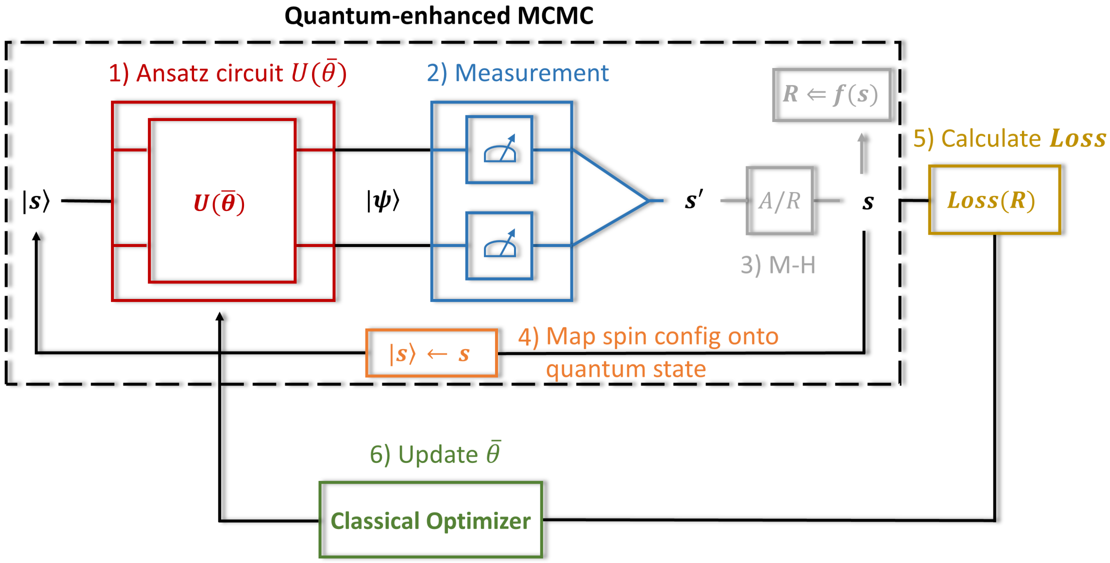
The optimization algorithm takes inspiration from the Variational Quantum Eigensolver (VQE) algorithm. As depicted in Figure 3.1a, the VQE iteratively updates the parameters of a parametrized circuit until the minimum of a loss function (the expectation value of the energy in this case) is reached. The same strategy is adopted by our algorithm, whose high-level representation is shown in Figure 3.1b, while the pseudocode[1] can be found in Algorithm 2.

Before diving into the details of the optimization routine, we ought to define the notation we will be using. Let S be the state space of all the 2^N possible spin configurations $\boldsymbol{s}$, with N the number of spins in the considered system. Let $f : S \rightarrow \mathbb{R}$ be a real-valued function defined on S. $\bar{\theta}$ will be the generic set of parameters, defining the quantum proposal distribution $Q_{(\boldsymbol{s'}|\boldsymbol{s})}(\bar{\theta}) = |\langle \boldsymbol{s}|U(\bar{\theta})|\boldsymbol{s'}\rangle|^2$, we aim to optimize. With *Loss*, instead, we will refer to a generic loss function. Several suitable options are available and more details will be given in subsection 3.1.1. Finally, let R be an array where we can store partial results during the computation.

[1] We have chosen to employ a universal pseudocode approach throughout this thesis. The reader can easily interpret the general algorithms and translate them into their preferred programming language.



a. Variational Quantum Eigensolver algorithm



b. Quantum-enhanced MCMC optimization algorithm

Figure 3.1: **a.** High-level representation of the Variational Quantum Eigensolver algorithm **b.** High-level representation of the Quantum-enhanced MCMC optimization algorithm. The algorithm starts by running a Quantum-enhanced Markov chain: it evolves the current computational basis state $|\boldsymbol{s}\rangle$ under $U(\bar{\theta})$ (1), the evolved state $|\psi\rangle$ is then measured in the computational basis (2) and the measurement result is either accepted or rejected according to the Metropolis-Hastings probabilities (3). An observable f is evaluated over the new spin configuration and, finally, the spin configuration is mapped onto a computational basis state (4) that constitutes the starting point for the successive step. Once a certain number n of samples has been gathered, a *Loss* function is evaluated over these samples (5) and the resulting value is fed to the classical optimizer which provides then new parameter values for the parametrized quantum circuit implementing the quantum state proposal (6).

The optimization algorithm iteratively updates the parameters defining the quantum proposal distribution until the *Loss* function value converges to a minimum. It consists of a feedback loop that runs quantum-enhanced Markov chains to evaluate a *Loss* function and optimize the proposal strategy. At each iteration, we run a quantum-enhanced Markov chain proposing moves using the current proposal distribution $Q_{(\boldsymbol{s'}|\boldsymbol{s})}(\bar{\theta}) = \langle \boldsymbol{s}|U(\bar{\theta})|\boldsymbol{s'}\rangle$, and we record the visited states $\{\boldsymbol{s_0}, \boldsymbol{s_1}, \boldsymbol{s_2}, ...\}$. For each visited state $\boldsymbol{s}$, an observable $f(\boldsymbol{s})$ is evaluated and stored in the register R. The observable choice depends on the specific problem we are dealing with and on the *Loss* function we want to use. Once a set number n of MC steps has been performed, we then evaluate the *Loss* function over the n samples and pass the value to the classical optimizer. The classical optimizer provides then new parameter values for the parametrized quantum circuit implementing the quantum state proposal $U(\bar{\theta})$. This routine is repeated until either the *Loss* converges or we reach a maximum number of iterations.

Note that in Algorithm 2, the function ClassicalOptimizer(Loss(R), $\bar{\theta}_{guess}$) is not defined as any minimization algorithm could be used. In subsection 3.1.3, we will present the one we used for numerical simulations, proposing also promising alternatives. The function Loss(R) is not defined as well. Several suitable options are available and will be presented in subsection 3.1.1.

Algorithm 2 QMCMC optimization algorithm

```
function RunQMCMC($\bar{\theta}$)
    $U(\bar{\theta}) \leftarrow e^{-iH(\bar{\theta})t}$
    $R \leftarrow \emptyset$
    $\boldsymbol{s} \leftarrow$ last visited state in previous iteration
    for set number $n$ of steps do
        propose next state
        $|\boldsymbol{s}\rangle \leftarrow \boldsymbol{s}$        ▷ mapping spin configuration onto computational basis state
        $|\psi\rangle \leftarrow U(\bar{\theta})|\boldsymbol{s}\rangle$
        $|\boldsymbol{s'}\rangle \leftarrow$ measure $|\psi\rangle$        ▷ measuring $|\psi\rangle$ in the computational basis
        $\boldsymbol{s'} \leftarrow |\boldsymbol{s'}\rangle$        ▷ mapping computational basis state onto spin configuration
        accept or reject state
        $A \leftarrow min(1, e^{(E(\boldsymbol{s})-E(\boldsymbol{s'}))/T})$
        if $A \geq$ random uniform $([0, 1])$ then
            $\boldsymbol{s} \leftarrow \boldsymbol{s'}$
        end if
        $R \leftarrow R \cup \{f(\boldsymbol{s})\}$        ▷ evaluating observable $f$ over the sampled state $\boldsymbol{s}$
    end for
    return $R$
end function

optimization algorithm
$\bar{\theta} \leftarrow$ initialize params guess
repeat
    $\bar{\theta}^* \leftarrow$ ClassicalOptmizer(Loss(RunQMCMC()), $\bar{\theta}$)
    $\bar{\theta} \leftarrow \bar{\theta}^*$
until converged or max iteration reached
return $\bar{\theta}$        ▷ return optimal parameters
```

3.1.1 Loss function

The algorithm described in Algorithm 2 is loss function agnostic. Several options turned out to be suitable choices, such as the convergence of a certain observable (energy or magnetization), or the mean first passage time (MFPT), defined as the average time the system takes to reach a particular state for the first time. In the end, since we aimed for a general and potentially scalable optimization approach, the autocorrelation function (ACF) proved to be the optimal candidate. The ACF of a given observable f can be heuristically estimated using Markov chain samples, regardless of the problem considered, the proposal strategy used or the chosen observable f. As already showed in subsection 1.3.1, considering a certain stationary Markov chain in the space state S, represented by a set of successive states $\{\boldsymbol{s}_0, \boldsymbol{s}_1, \boldsymbol{s}_2, ...\} = \{\boldsymbol{s}_i\}_i^n$, the ACF function of a given observable f is defined as:

$$C(l)_{ff} = \langle (f_i - \mu)(f_{i+l} - \mu) \rangle \qquad f_i = f(\boldsymbol{s}_i) \quad \boldsymbol{s}_i \in S \tag{3.1}$$

where μ is the observable mean:

$$\mu = \langle f \rangle \tag{3.2}$$

and typically decays exponentially for large enough l:

$$C_{ff}(l) \sim e^{-\frac{l}{\tau_{exp}}} \tag{3.3}$$

where τ_{exp} is the exponential autocorrelation time and l is the lag. The ACF provides a measure of the correlation between Markov chain samples separated by a lag l. Under the assumption that the chosen observable f has a non-zero projection on the transition matrix eigenvector corresponding to λ_{SLEM}, the exponential autocorrelation time τ_{exp} is directly related to the spectral gap δ through:

$$\tau_{exp} = -\frac{1}{ln(\lambda_{SLEM})} = -\frac{1}{ln(1-\delta)} \tag{3.4}$$

By minimizing $C_{ff}(l)$ at a certain lag l, we are effectively minimizing the exponential autocorrelation time τ_{exp} and consequently maximizing the spectral gap δ. In other words, we are maximizing the convergence rate of the Markov chain (for a more detailed discussion, see subsection 1.2.1). For the sake of brevity, in the following we will use the term "autocorrelation time" and the Greek letter τ to refer to the exponential autocorrelation time.

Several estimators for either C_{ff} or τ are available in the literature, however, they all share the same limitation: they all depend on one or more parameters, which strongly influence the quality of the final estimate. We consider these as hyperparameters of our optimization algorithm, which have to be carefully tuned in order to obtain satisfying results.

After a thorough evaluation of several possible solutions, the estimator proposed by A. Sokal [5] was selected due to its simplicity (more computationally efficient with respect to the others) and ease of optimization of the related hyperparameters, discussed in detail in subsection 3.1.2. The estimator, *Loss* function in our optimization algorithm, is defined as:

$$\hat{C}_{ff}(l) = \frac{1}{n-l}\sum_{i}^{n-l}(f_i - \mu)(f_{i+l} - \mu) \tag{3.5}$$

where n is the number of samples we use for the estimate, in other words, the number of steps in the chain we run to evaluate the estimator.

Equation 3.1 assumes that we are considering a stationary Markov chain, meaning that it reached convergence and we are effectively sampling from the target probability distribution:

$$\mu = \langle f \rangle = \langle f_i \rangle = \langle f_{i+l} \rangle \qquad \forall i, i+l < n \tag{3.6}$$

Therefore, in order to be able to use experimental data to estimate $C_{ff}(l)$ through $\hat{C}_{ff}(l)$, we should wait for convergence first. From a practical point of view, we do not need to sample exactly from the target probability distribution in order to obtain a good estimate of $C_{ff}(l)$, a probability distribution close enough to the desired one will suffice.

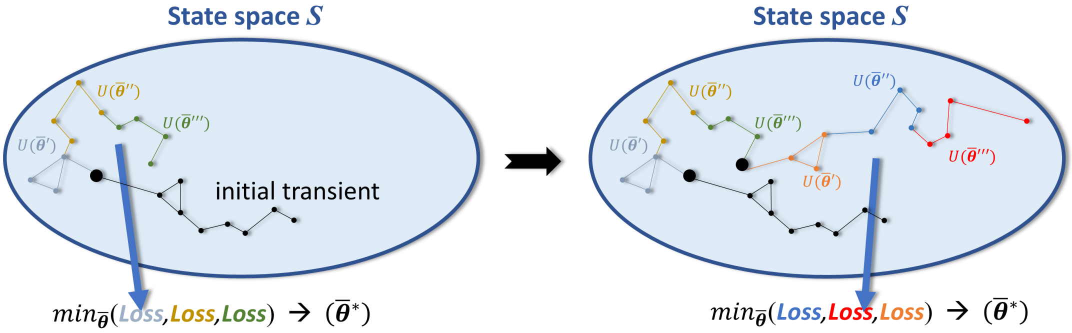

State space S
$U(\bar{\boldsymbol{\theta}}'')$
$U(\bar{\boldsymbol{\theta}}''')$
$U(\bar{\boldsymbol{\theta}}')$
initial transient
$min_{\bar{\boldsymbol{\theta}}}$(Loss,Loss,Loss) → ($\bar{\boldsymbol{\theta}}^*$)
State space S
$U(\bar{\boldsymbol{\theta}}'')$
$U(\bar{\boldsymbol{\theta}}'')$
$U(\bar{\boldsymbol{\theta}}''')$
$U(\bar{\boldsymbol{\theta}}')$
$U(\bar{\boldsymbol{\theta}}')$
$U(\bar{\boldsymbol{\theta}}''')$
$min_{\bar{\boldsymbol{\theta}}}$(Loss,Loss,Loss) → ($\bar{\boldsymbol{\theta}}^*$)

Figure 3.2: Pictorial representation of the QMCMC optimization algorithm. The colored straight lines represent Markov chains jumping from one state to another, where the states are the colored points. We start by discarding the data from the initial transient (black lines and dots), such that during the first optimization step we are already sampling from a probability distribution close to the target one. We run several Markov chains characterized by different proposal distributions (colored lines and dots) and calculate the *Loss* for each one. The minimum *Loss* value determines the new parameter set, which becomes the initial guess for the second optimization step (big black dot).

This poses a problem: every time we want to evaluate the quality of a certain proposal distribution $Q_{(\boldsymbol{s'}|\boldsymbol{s})}(\bar{\theta}) = \langle \boldsymbol{s}|U(\bar{\theta})|\boldsymbol{s'}\rangle$ by calculating $\hat{C}_{ff}(l)$ (the *Loss*) over a set of samples we have to discard a large number of Markov chain steps before we can start gathering "good" samples. However, there is a practical solution. An optimization step consists in evaluating several proposal distributions by running as many Markov chains and calculating the *Loss* for each one. The way these proposal distributions are selected converging to a minimum depends on the specific minimization algorithm adopted (see subsection 3.1.3 for more details). In order to avoid discarding data every time the *Loss* is evaluated, the following approach can be used. We start the optimization by discarding an initial transient, such that during the first optimization step we are already sampling from a probability distribution close to the target one. Then, we start each Markov chain from the last state visited by the previous one, concatenating different chains, as illustrated in Figure 3.2. Even though this results in a non-Markovian chain as the optimization process retains memory of the past, this approach leads to individual chains having an effective initial distribution. As explained in chapter 1, the initial distribution has an important role in determining the convergence speed. Thus the chains are able to quickly get close enough to the stationary probability distribution and produce "good" samples. This strategy turned out to work well in practice, significantly reducing the running time of the algorithm.

3.1.2 Hyperparameters: number of samples n and lag l

$\hat{C}_{ff}(t)$ provides a consistent and unbiased estimate of $C_{ff}(t)$, but requires prior knowledge of the observable mean μ. Given that, in general, we do not have access to this information, we have to settle for:

$$\mu \approx \bar{f} = \frac{1}{n}\sum_{i}^{n} f_i \tag{3.7}$$

redefining the estimator as:

$$\hat{\hat{C}}_{ff}(l) = \frac{1}{n-l}\sum_{i}^{n-l}(f_i - \bar{f})(f_{i+l} - \bar{f}) \tag{3.8}$$

However, relying on Equation 3.7 introduces a bias, which decays as $1/n$. To leading order in $1/n$, $\hat{C}_{ff}(l)$ (Equation 3.5) and $\hat{\hat{C}}_{ff}(l)$ (Equation 3.8) share the same behaviour. Therefore, the larger the number of samples n, the more accurate is our estimator. In addition to the bias, the second source of error comes from the fact that $\hat{C}_{ff}(l)$ and $\hat{\hat{C}}_{ff}(l)$ are random variables. We can therefore look at the variance to estimate the error scaling with respect to n (for a more detailed discussion, see subsection 1.3.2):

$$Var(\hat{C}_{ff}(l)) = (\frac{1}{n})\sum_{m=-\infty}^{\infty}[C(m)^2 + C(m+l)C(m-l) + \kappa(l, m, m+l)] + o(\frac{1}{n}) \tag{3.9}$$

where κ is the connected 4-points autocorrelation function:

$$\begin{aligned} \kappa(r, s, l) = \langle(f_i - \mu)(f_{i+r} - \mu)(f_{i+s} - \mu)(f_{i+l} - \mu)\rangle \\ -C(r)C(l-s) - C(s)C(l-r) - C(l)C(s-r) \end{aligned} \tag{3.10}$$

Leading to a standard error $\sqrt{Var}(\hat{C}_{ff})$ that goes down as $1/\sqrt{n}$.
In summary, both the bias error and the standard error, which constitute the noise η affecting $C_{ff}(l)$ estimates, go down when increasing the number of samples. Therefore, in general, is desirable to maximize n in order to get more accurate results. The larger n, the less noisy is the optimization landscape that the classical optimizer explores. As for large n the standard error term dominates, we consider the noise to decrease approximately as $1/\sqrt{n}$.

The second hyperparameter to deal with is the lag l, i.e. the distance at which we are looking if there is a correlation between Markov chain samples. It is crucial to choose l carefully as it affects the classical optimizer's ability to distinguish between two different sets of parameters. Consider two different parameters sets $\bar{\theta}_1$ and $\bar{\theta}_2$ characterized by different proposal distributions, respectively $Q_{(\boldsymbol{s'}|\boldsymbol{s})}(\bar{\theta}_1) = \langle \boldsymbol{s}|U(\bar{\theta}_1)|\boldsymbol{s'}\rangle$ and $Q_{(\boldsymbol{s'}|\boldsymbol{s})}(\bar{\theta}_2) = \langle \boldsymbol{s}|U(\bar{\theta}_2)|\boldsymbol{s'}\rangle$. The optimizer uses the value of the corresponding ACFs at a certain lag l, called $C_{ff}(l)_1$ and $C_{ff}(l)_2$, to determine which set of parameters defines a better proposal distribution. Since the optimizer objective is to minimize the ACF:

$$\Delta_C(l) = C_{ff}(l)_1 - C_{ff}(l)_2 \quad \rightarrow \quad \begin{cases} \Delta_C(l) < 0 & \rightarrow \quad \bar{\theta}_1 \ better \\ \Delta_C(l) > 0 & \rightarrow \quad \bar{\theta}_2 \ better \end{cases} \tag{3.11}$$

Given that we do not have access to the true values $C_{ff}(l)_1$ and $C_{ff}(l)_2$, the optimizer evaluates the corresponding estimators $\hat{\hat{C}}_{ff}(l)_1$ and $\hat{\hat{C}}_{ff}(l)_2$ affected by noise, which decreases with the number of samples n as shown previously. If $\Delta_C(l)$ ends up having the same order of magnitude of the noise, either because we did not gather enough samples or we made a poor choice for the lag l, the algorithm is not able to distinguish between $\bar{\theta}_1$ and $\bar{\theta}_2$. The hyperparameters define how accurate is the optimization.

We developed several approaches to select a proper lag value l:

- **Lag integration** ***Loss*****:**
  Considering a set of Markov chain samples, $C_{ff}(l)$ assumes positive values as long as there is correlation between samples separated by a lag l. If we pick a value of l large enough such that the samples are not correlated anymore, then $C_{ff}(l)$ assumes random values, as shown in Figure 3.3.

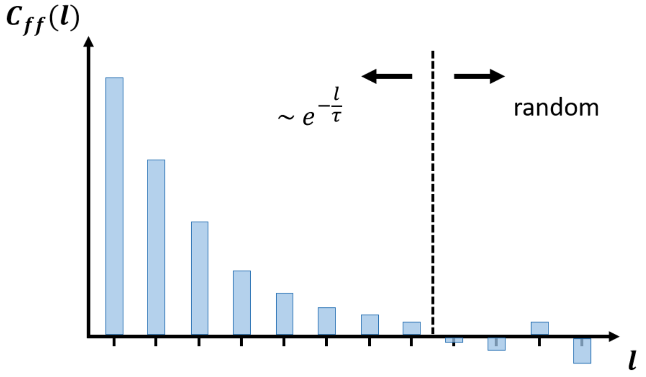


Figure 3.3: Typical ACF trend. It features an exponential decay followed by a plateau as the degree of correlation decreases. At large l it assumes random values. Note that, for the sake of simplicity, we assumed that the ACF is governed by a single exponential autocorrelation time even at small l.

In order to remove the burden of selecting a good l, we can integrate $\hat{\hat{C}}_{ff}(l)$ in l and stop the integration as soon as $\hat{\hat{C}}_{ff}(l) \leq 0$, as described in Algorithm 3. The integral becomes the *Loss* function used by the optimization algorithm (Algorithm 2).

Algorithm 3 lag integration *Loss*

```
function Loss($R$)
    $n \leftarrow count(R)$                    ▷ $n$ is the number of samples in the set $R$
    $\bar{f} \leftarrow mean(R)$
    $l, C_{int}, C \leftarrow 0, 0, 0$
    repeat
        $C_{int} \leftarrow C_{int} + C$
        $C \leftarrow 0$
        $l \leftarrow l + 1$
        for $f_i \in R$ and $i + l \leq n$ do
            $C \leftarrow C + (f_i - \bar{f})(f_{i+l} - \bar{f})$
        end for
        $C \leftarrow C/(n - l)$
    until $C \leq 0$
    return $C_{int}$
end function
```

- **Fixed lag** ***Loss*****:**
  A simple and heuristic method to select a reasonable lag value l_{fix} is through a trial and error approach. $\hat{\hat{C}}_{ff}(l_{fix})$ becomes then the *Loss* function, as shown in Algorithm 4. The effectiveness and applicability of this approach depend on the problem at hand.

Algorithm 4 fixed lag *Loss*

```
function Loss($R$)
    $n \leftarrow count(R)$                          ▷ $n$ is the number of samples in the set $R$
    $\bar{f} \leftarrow mean(R)$
    $C \leftarrow 0$
    for $f_i \in R$ and $i + l_{fix} \leq n$ do
        $C \leftarrow C + (f_i - \bar{f})(f_{i+l_{fix}} - \bar{f})$
    end for
    $C \leftarrow C/(n - l_{fix})$            ▷ can be omitted, it does not add any information
    return $C$
end function
```

- **Lag optimization** ***Loss*****:**
  Another option consists in including a subroutine in our algorithm that updates l at each iteration by tracking the maximum of $\Delta_C(l)$, as described in Algorithm 5. In this way, the algorithm automatically adjusts the lag value based on the current state of the optimization. At each optimization step, the algorithm keeps track of $\hat{\hat{C}}_{ff}$ at three different lag values: $\hat{\hat{C}}_{ff}(l - \xi)$, $\hat{\hat{C}}_{ff}(l)$, $\hat{\hat{C}}_{ff}(l + \xi)$, where l is the current lag value and ξ is a parameter controlling the magnitude of the lag update. Only $\hat{\hat{C}}_{ff}(l)$ is actually used as *Loss* function to carry out the optimization.

Algorithm 5 lag optimization *Loss*

```
function Loss($R, l$)
    $n \leftarrow count(R)$                          ▷ $n$ is the number of samples in the set $R$
    $\bar{f} \leftarrow mean(R)$
    $\Upsilon \leftarrow \{l - \xi,\ l,\ l + \xi\}$            ▷ $\xi$ determines the magnitude of the lag update
    $\Lambda \leftarrow \emptyset$
    for $\upsilon \in \Upsilon$ do
        $C_\upsilon \leftarrow 0$
        for $f_i \in R$ do
            $C_\upsilon \leftarrow C_\upsilon + (f_i - \bar{f})(f_{i+\upsilon} - \bar{f})$
        end for
        $C_\upsilon \leftarrow C_\upsilon/(n - l)$
        $\Lambda \leftarrow \Lambda \cup \{C_\upsilon\}$
    end for
    return $\Lambda$, $C_l$                          ▷ $\Lambda$ is an array containing $C_{l-\xi}, C_l, C_{l+\xi}$
end function

function ClassicalOptimizer(Loss(RunQMCMC(), $l$), $\bar{\theta}$)
    start from $\bar{\theta}$, find $\bar{\theta}^*$ that minimizes $C_l$          ▷ minimize only with respect to $C_l$
    return $\bar{\theta}^*$, $\Lambda^*$        ▷ return best parameters set $\bar{\theta}^*$ and corresponding $C$ array $\Lambda^*$
```

```
end function

function OptimizeLag($\Lambda$)
    $\Delta_v \leftarrow \emptyset$
    for $C_v \in \Lambda$ and $C_v^{pre} \in \Lambda^{pre}$ do
        $\Delta_v \leftarrow \Delta_v \cup \{|C_v^{pre} - C_v|\}$
    end for
    if $\Delta_{l-\xi} - \eta > \Delta_l$ then          ▷ update $l$ if visibility $\Delta$ is larger despite noise $\eta$
        $l \leftarrow l - \xi$
    else if $\Delta_{l+\xi} - \eta > \Delta_l$ then     ▷ update $l$ if visibility $\Delta$ is larger despite noise $\eta$
        $l \leftarrow l + \xi$
    end if
    $\Lambda^{pre} \leftarrow \Lambda$
    return $l$
end function

Optimization algorithm
$\bar{\theta} \leftarrow$ initialize params guess
repeat
    $\bar{\theta}^*$, $\Lambda^* \leftarrow$ ClassicalOptmizer(Loss(RunQMCMC(), $l$), $\bar{\theta}$)
    $\bar{\theta} \leftarrow \bar{\theta}^*$
    $l \leftarrow$ OptmizeLag($\Lambda^*$)
until converged or max iteration reached
return $\bar{\theta}$                                   ▷ return optimal parameters
```

Once the optimization step is concluded, it calculates:

$$\begin{aligned} |\Delta_{\hat{\hat{C}}}(l-\xi)| &= |\hat{\hat{C}}_{ff}^{pre}(l-\xi) - \hat{\hat{C}}_{ff}(l-\xi)| \\ |\Delta_{\hat{\hat{C}}}(l)| &= |\hat{\hat{C}}_{ff}^{pre}(l) - \hat{\hat{C}}_{ff}(l)| \\ |\Delta_{\hat{\hat{C}}}(l+\xi)| &= |\hat{\hat{C}}_{ff}^{pre}(l+\xi) - \hat{\hat{C}}_{ff}(l+\xi)| \end{aligned} \tag{3.12}$$

where $\hat{\hat{C}}_{ff}^{pre}$ are the ACFs values resulting from the previous optimization step. If there is a clear increase in Δ_C for either $l+\xi$ or $l-\xi$, despite the noise η, the lag value l is updated accordingly. Note that the value of ξ has to be tailored to the specific problem considered.

- **Fitting $\hat{\hat{C}}_{ff}(l)$ to estimate the autocorrelation time τ:**
  Finally, we can also estimate $C_{ff}(l)$ and fit the final curve to infer the autocorrelation time τ directly. As in the integration case, we stop estimating $C_{ff}(l)$ as soon as we hit the first negative value. This approach, however, turned out to perform significantly worse with respect to the others.

3.1.3 Classical optimizer

Since C_{ff}, the *Loss* function we sought to minimize, is very noisy unless a large number of samples n are drawn at each iteration, we opted for a gradient-free algorithm, the Nelder-Mead algorithm. The Nelder-Mead algorithm is a direct search method widely used for non-convex optimization problems. It is a heuristic algorithm that seeks to minimize an objective function by iteratively adapting a simplex S, a geometrical object consisting of $m+1$ vertices in a m-dimensional parameters space. The vertices of the simplex are evaluated according to the objective function, and then the simplex is transformed into a new set of vertices based on their values. This process is repeated until the algorithm converges to a minimum. The pseudocode describing the algorithm is reported in Algorithm 6. Despite the standard Nelder-Mead algorithm does not perform well in high dimensional spaces, alternative approaches to overcome this problem have been developed [16]. This gives us the possibility to use the Nelder-Mead even with complicated ansatzes $U(\bar{\theta})$ depending on a large number of parameters.

Algorithm 6 Nelder-Mead algorithm

```
function ClassicalOptimizer(Loss(RunQMCMC()), $\bar{\theta}_{guess}$)
    initialize simplex S with $m+1$ vertices        ▷ $\bar{\theta}_{guess}$ being one of S vertices
    repeat
        sort S vertices by function Loss(RunQMCMC($\bar{\theta}$)) value
        calculate centroid $\bar{\theta}_c$ of the $m$ best vertices
        reflect worst vertex $\bar{\theta}_n$ through $\bar{\theta}_c$ to obtain $\bar{\theta}_r$
        if Loss(RunQMCMC($\bar{\theta}_r$)) < Loss(RunQMCMC($\bar{\theta}_n$)) then
            expand $\bar{\theta}_n \leftarrow \bar{\theta}_r + \Delta\bar{\theta}$ in the direction of $\bar{\theta}_c$
            if Loss(RunQMCMC($\bar{\theta}_r + \Delta\bar{\theta}$)) < Loss(RunQMCMC($\bar{\theta}_r$)) then
                $\bar{\theta}_n \leftarrow \bar{\theta}_r + \Delta\bar{\theta}$
            else
                $\bar{\theta}_n \leftarrow \bar{\theta}_r$
            end if
        else
            contract worst vertex $\bar{\theta}_n \leftarrow \bar{\theta}_c + \gamma(\bar{\theta}_n - \bar{\theta}_c)$        ▷ $\gamma$ weights the update
            if Loss(RunQMCMC($\bar{\theta}_c + \gamma(\bar{\theta}_n - \bar{\theta}_c)$)) < Loss(RunQMCMC($\bar{\theta}_n$)) then
                $\bar{\theta}_n \leftarrow \bar{\theta}_c + \gamma(\bar{\theta}_n - \bar{\theta}_c)$
            else
                shrink entire simplex towards best vertex $\bar{\theta}_1$
            end if
        end if
    until converged or max iteration reached
    return best vertex $\bar{\theta}_1$
end function
```

However, the algorithm has some limitations. It is sensitive to the initial simplex and the convergence rate is relatively slow. More suited minimization algorithms may be able to outperform the Nealder-Mead, like the ones proposed by Lavrijsen et al. [17], which have been proven to perform well in noisy settings.

3.1.4 Observable f

Finally, the last point to address is how to choose an observable for the optimization algorithm. Ideally, the ACF of the chosen observable should be dominated by a single exponential autocorrelation time, associated with the slowest mode in the system, such that:

$$C_{ff}(l) \approx Be^{-\frac{l}{\tau}} \tag{3.13}$$

where B is a constant depending on the specific function and τ is directly related to the spectral gap δ through Equation 3.4. In such a case, minimizing $C_{ff}(l)$ directly corresponds to maximizing δ. If an observable does not fit this description it may work as well, however, it is not guaranteed as is not clear if by minimizing $C_{ff}(l)$ we are effectively minimizing the autocorrelation time related to δ. In the simulations presented in chapter 4, both the energy and the magnetization turned out to be suitable choices for the specific optimization problem we addressed.

Chapter 4

Simulations results and discussion

In order to analyze the performance of our algorithm, we run state vector simulations on a classical computer, monitoring the spectral gap δ and the *Loss* function value during the optimization (specs of the hardware used for simulations can be found in Table 4.1). Despite Markov chains' convergence being a complex and convoluted process, the spectral gap provides a simple and effective way to describe it. The spectral gap value is defined by the slowest Markov chain's mode and is directly related to important quantities, such as the mixing time τ_ϵ (see subsection 1.2.1 for more details). We numerically calculated the spectral gap by diagonalizing the transition matrix $P(\boldsymbol{s'}|\boldsymbol{s})$ using *scipy.linalg* library in Python. $P(\boldsymbol{s'}|\boldsymbol{s})$ was found by computing entry by entry as per definition:

$$P(\boldsymbol{s'}|\boldsymbol{s}) = A(\boldsymbol{s'}|\boldsymbol{s})Q_{(\boldsymbol{s'}|\boldsymbol{s})}(\gamma, t) + \delta_{\boldsymbol{ss'}} \sum_{\boldsymbol{s''}\in S} (1 - A(\boldsymbol{s''}|\boldsymbol{s}))Q_{(\boldsymbol{s''}|\boldsymbol{s})}(\gamma, t) \tag{4.1}$$

where $Q_{(\boldsymbol{s'}|\boldsymbol{s})}(\gamma, t)$ is the symmetric quantum proposal distribution:

$$Q_{(\boldsymbol{s'}|\boldsymbol{s})}(\gamma, t) = Q_{(\boldsymbol{s}|\boldsymbol{s'})}(\gamma, t) = |\langle \boldsymbol{s'}|U(\gamma, t)|\boldsymbol{s}\rangle|^2 \qquad U(\gamma, t) = e^{-i((1-\gamma)\alpha H_{prob} - \gamma H_{mix})t} \tag{4.2}$$

and $A(\boldsymbol{s'}|\boldsymbol{s})$ is the Metropolis-Hastings acceptance probability:

$$A(\boldsymbol{s'}|\boldsymbol{s}) = min(1, \frac{\mu(\boldsymbol{s'})Q_{(\boldsymbol{s'}|\boldsymbol{s})}(\gamma, t)}{\mu(\boldsymbol{s})Q_{(\boldsymbol{s}|\boldsymbol{s'})}(\gamma, t)}) = min(1, \frac{\mu(\boldsymbol{s'})}{\mu(\boldsymbol{s})}) = min(1, e^{(E(\boldsymbol{s}) - E(\boldsymbol{s'}))/T}) \tag{4.3}$$

The specific problem we choose to address, is the optimization of the parametrized ansatz $U(\gamma, t)$ proposed by Layden et al., presented in chapter 2 [12]. The temperature range we are interested in is $T \in [0, 10]$, as the quantum-enhanced MCMC showed quantum advantage at low temperatures. We start by presenting simulations at $T = 10$, discussing the results in detail. We analyze the spectral gap landscape and test the proposed *Loss* functions on different models. We then present simulations at $T < 10$ and, finally, we discuss the limitations of the optimization algorithm and its potential scalability.

Computer Name	CPU(s)	Cores	RAM/GB
hitachi	AMD EPYC 7502 32-Core Processor (×2)	64	2000
danieleHP	Intel(R) Core(TM) i7-1065G7 CPU	4	16

Table 4.1: Hardware used for the numerical simulations.

4.1 Optimization algorithm simulations

We run simulations adopting both the *fixed lag Loss* and *lag integration Loss*. Note that we omitted the results of the simulations using the *lag optimization Loss*. Given that we were able to select a good l_{fix} value for the problem at hand through a trial and error on a small number of model instances, the lag optimization did not bring any advantage. However, when dealing with problems where is not possible to identify a reasonable l_{fix} value, the *lag optimization Loss* may turn out to be a useful resource.

model : fully connected 2D Ising model with random $J_{ij} \sim \mathcal{N}(0, 1)$ and $h_i \sim \mathcal{N}(0, 1)$

simulation type	samples n	lag l	observable f	T	average over # model instances	simulations per model instance
optimization (1)	$2,5 \times 10^3$	4	energy $E(\boldsymbol{s})$	10	5	5
colormap (1)	4×10^3	4	energy $E(\boldsymbol{s})$	10	1	1
optimization (2)	$2,5 \times 10^3$	integral	energy $E(\boldsymbol{s})$	10	5	5
colormap (2)	4×10^3	integral	energy $E(\boldsymbol{s})$	10	1	1

Table 4.2: Simulations specs used for numerical simulations reported in Figure 4.1, Figure 4.2, Figure 4.3 and Figure 4.4.

Figure 4.1 and Figure 4.3 depict the evolution of the average spectral gap and *Loss* value during the optimization process over several model instances. Each plot shows the average of 25 simulations: we used 5 randomly generated model instances and run 5 simulations per instance, where the initial γ and t values were uniformly sampled from $[0.1, 0.25]$ and $[1, 10]$ respectively. The results are compared to the ones obtained using the approach developed by Layden et al. [12] and presented in chapter 2, where γ and t are uniformly sampled from reasonable intervals, $[0.25, 0.6]$ and $[2, 20]$ respectively. Our algorithm converges to larger spectral gap values on average, indicating better proposal strategies that lead to faster convergence. Moreover, scatter plots show a clear correlation between the *Loss* function and the spectral gap, demonstrating the effectiveness of the *Loss* function. As the spectral gap reaches large values, the correlation suffers a slight decrease (around $Loss = 0$), particularly visible in simulations with *fixed lag Loss* (Figure 4.1). This suggests that the corresponding proposal distributions $Q_{(\boldsymbol{s}|\boldsymbol{s'})}(\gamma, t)$ define Markov chains with ACFs decaying so rapidly that at $l = 4$ (i.e. the l_{fix} value used) are already assuming random values. We can see this effect more pronounced in the $N = 4$ case because smaller size problems can achieve larger spectral gap values corresponding to faster ACF decay rates. The clustering of points, in both the *fixed lag* and *integration lag Loss* case, indicate that the *Loss* function has converged and the optimization must be stopped.

Colormaps of the *Loss* and spectral gap are displayed in Figure 4.2 and Figure 4.4. For this second series of simulations, we discretized the parameters space in 10^4 points and for each point, corresponding to a pair (γ, t), we calculated the *Loss* and the spectral gap of 2 randomly generated model instances 64040[1] and 64045 with respectively $N = 4$ and $N = 8$ spins. The *Loss* colormaps illustrate the landscape that the classical optimizer explores during the optimization. The *Loss* is able to replicate all the features present in the spectral gap colormap, demonstrating its effectiveness.

[1] The number represents the *numpy random seed* used to generate the model.

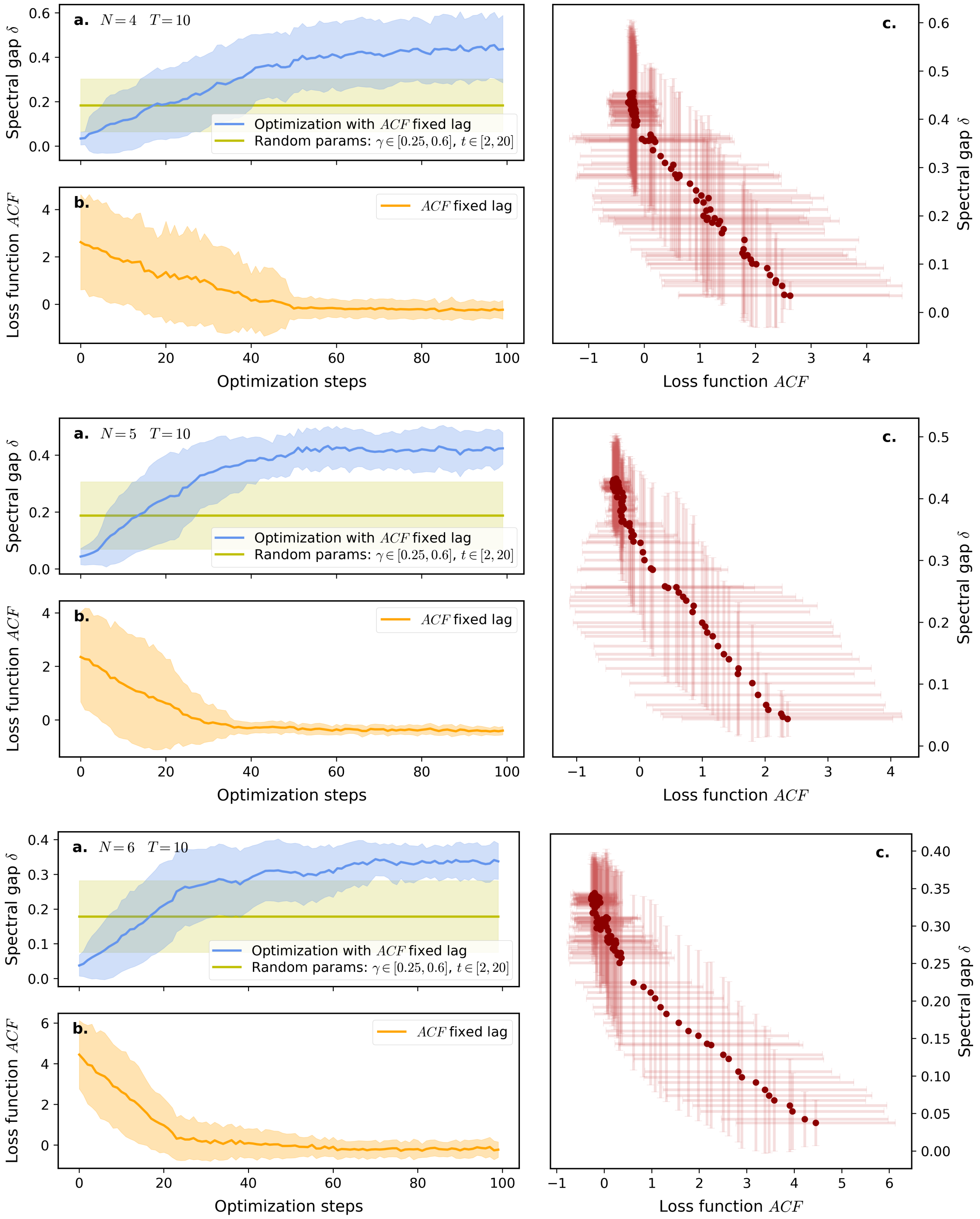


Figure 4.1: a. Optimization algorithm simulations. The average spectral gap is represented by the blue solid lines, the error bands show the population standard deviation. The results are compared with the approach used by Layden et al. (random parameters) [12]. **b.** The average fixed lag *Loss* is represented by the orange solid lines, the error bands show the population standard deviation. **c.** Correlation between average spectral gap and average *Loss* values. The error bars show the population standard deviation. Simulations specs can be found in Table 4.2 under "optimization (1)".

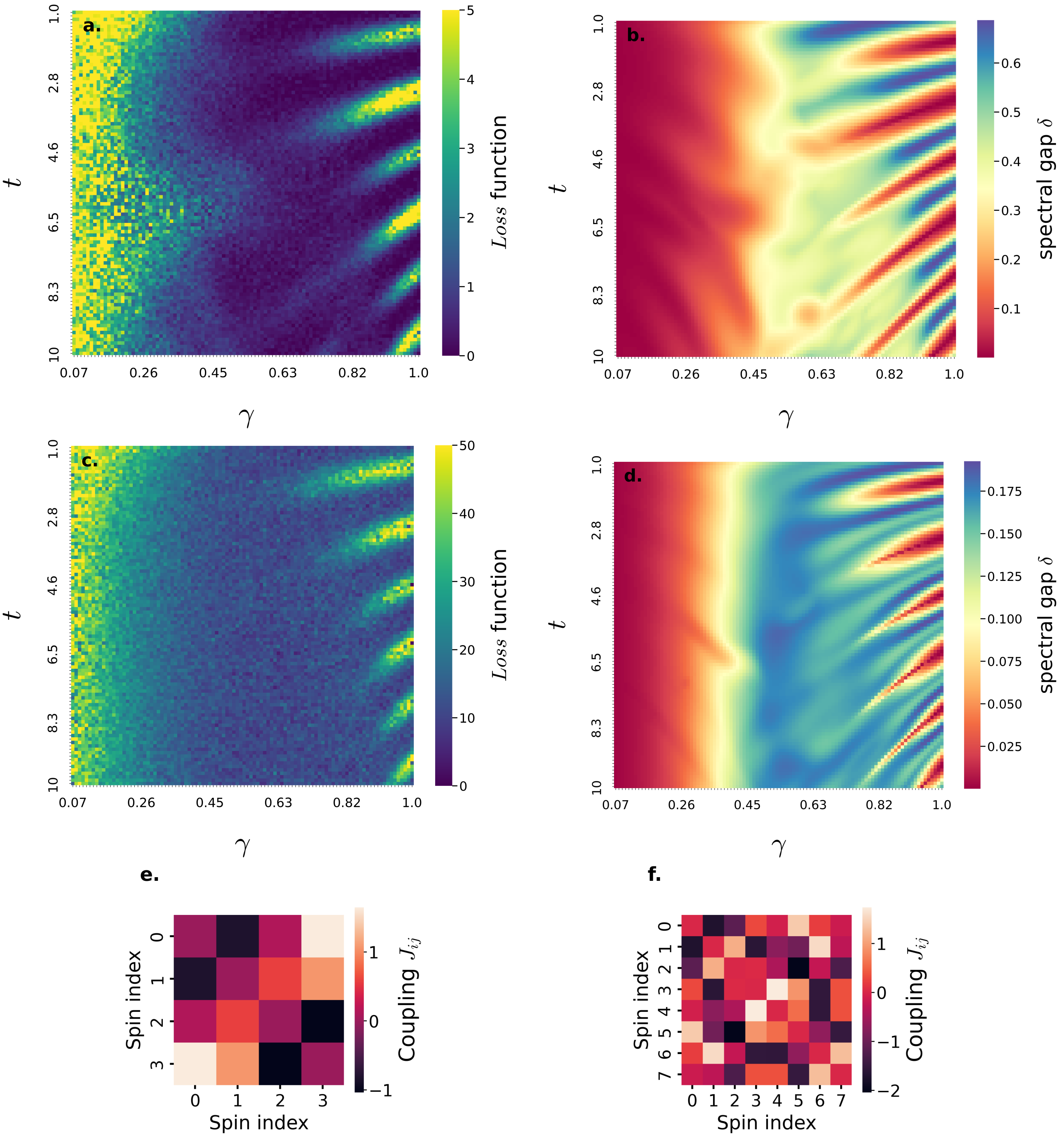


Figure 4.2: *Loss* and spectral gap δ colormaps of 2 specific model instances with 4 and 8 spins, at $T = 10$. **a.** *fixed lag Loss* colormap of the model instance 64040 ($N = 4$ spins) **b.** Spectral gap δ colormap of the model instance 64040 ($N = 4$ spins). **c.** *fixed lag Loss* colormap of the model instance 64045 ($N = 8$ spins). **d.** Spectral gap δ colormap of the model instance 64045 ($N = 8$ spins). **e.** Spins coupling J_{ij} colormap of the model instance 64040 ($N = 4$ spins). **e.** Spins coupling J_{ij} colormap of the model instance 64045 ($N = 8$ spins). Simulations specs can be found in Table 4.2 under "colormap (1)".

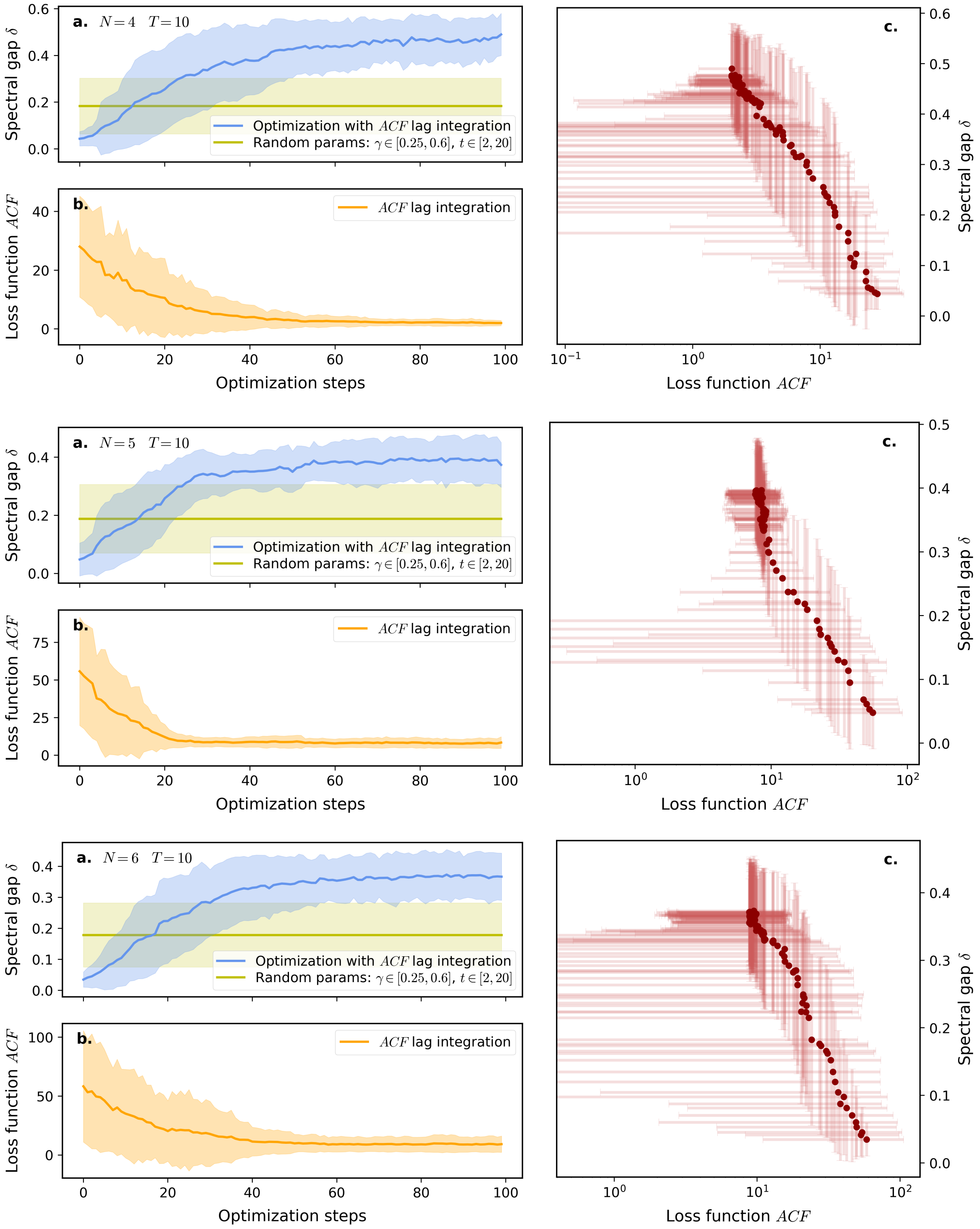


Figure 4.3: **a.** Optimization algorithm simulations. The average spectral gap is represented by the blue solid lines, the error bands show the population standard deviation. The results are compared with the approach used by Layden et al. (random parameters) [12]. **b.** The average lag integration *Loss* is represented by the orange solid lines, the error bands show the population standard deviation. **c.** Correlation between average spectral gap and average *Loss* values. The error bars show the population standard deviation. Simulations specs can be found in Table 4.2 under "optimization (2)".

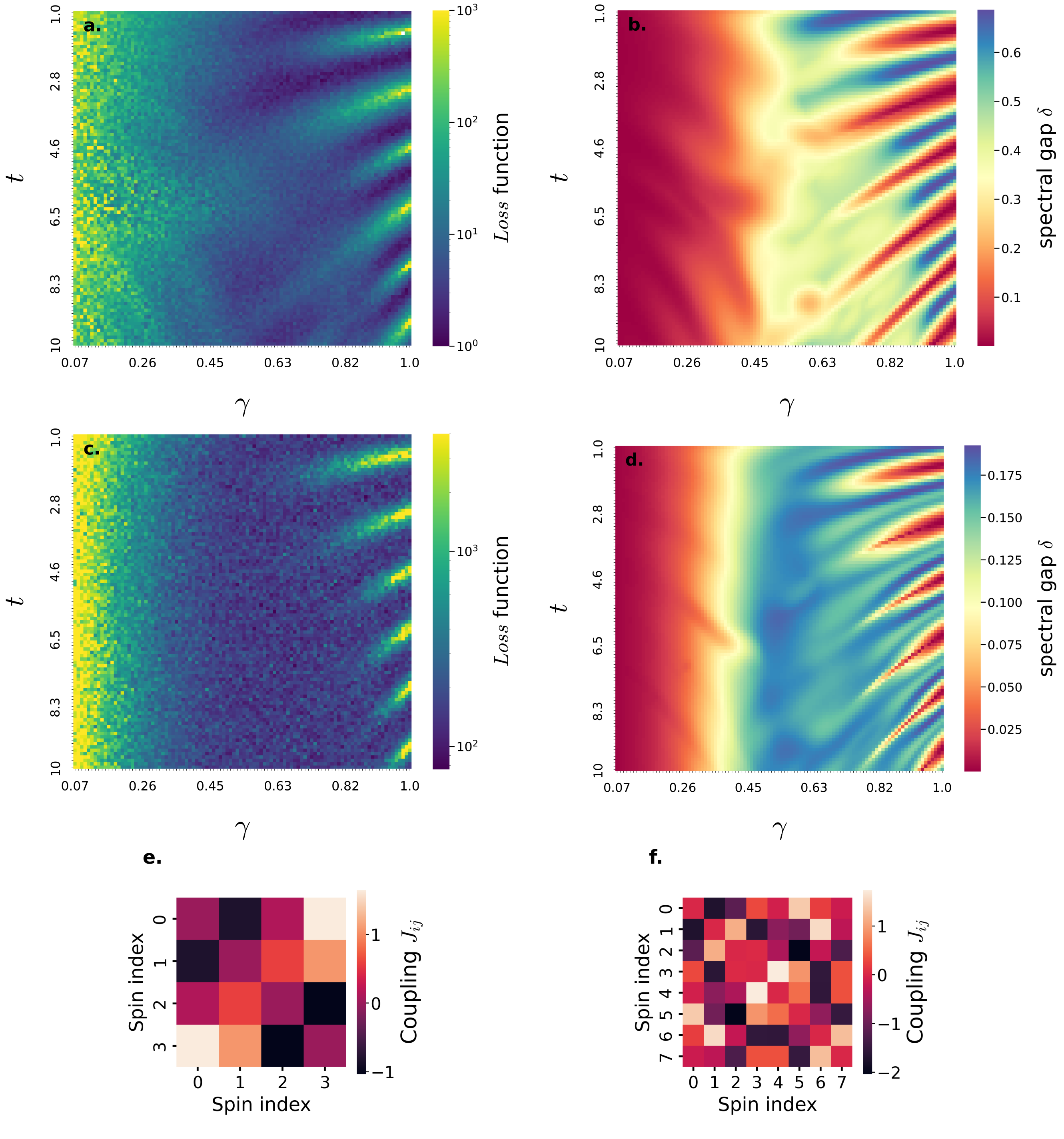


Figure 4.4: *Loss* and spectral gap δ colormaps of 2 specific model instances with 4 and 8 spins, at $T = 10$. **a.** *integration lag Loss* colormap of the model instance 64040 ($N = 4$ spins) **b.** Spectral gap δ colormap of the model instance 64040 ($N = 4$ spins). **c.** *integration lag Loss* colormap of the model instance 64045 ($N = 8$ spins). **d.** Spectral gap δ colormap of the model instance 64045 ($N = 8$ spins). **e.** Spins coupling J_{ij} colormap of the model instance 64040 ($N = 4$ spins). **e.** Spins coupling J_{ij} colormap of the model instance 64045 ($N = 8$ spins). Simulations specs can be found in Table 4.2 under "colormap (2)".

4.1.1 Spectral gap landscape analysis

Spectral gap landscapes generated by different model instances share some common features. In the following, we aim to explain the nature of these common characteristics. Firstly, we can see that at small γ values the spectral gap assumes small values as well, regardless of t. This is due to the fact that when evolving a computational basis state $|\boldsymbol{s}\rangle$ we are not effectively promoting transition between quantum states, but rather adding a random phase to it as $|\boldsymbol{s}\rangle$ states are H_{prob} eigenstates:

$$H(\gamma) = (1-\gamma)\alpha H_{prob} - \gamma H_{mix} \quad \xrightarrow{\gamma<<} \quad H(\gamma) \approx H_{prob} \quad \rightarrow \quad U(\gamma, t) \approx e^{-iH_{prob}t} \tag{4.4}$$

In other words, a Markov chain characterized by a proposal distribution $Q_{(\boldsymbol{s}|\boldsymbol{s'})}(\gamma, t)$ with small γ tends to spend a long time in a certain state before jumping onto a new one, increasing the correlation between successive samples and leading to small spectral gap values. If $\gamma = 0$, the chain is trapped in the initial state and δ goes to zero.

At large γ values instead, fringes associated with small spectral gap values appear. This second common feature turns out to have a more interesting explanation. As γ approaches 1, we have:

$$H(\gamma) = (1-\gamma)\alpha H_{prob} - \gamma H_{mix} \quad \xrightarrow{\gamma\approx 1} \quad H(\gamma) \approx H_{mix} \quad \rightarrow \quad U(\gamma, t) \approx e^{-i\gamma H_{mix}t} \tag{4.5}$$

where $H_{mix} = \sum_j X_j$ with $X_j = I_1 \otimes ... \otimes \hat{X} \otimes ... \otimes I_N$. Since $[X_j, X_i] = 0 \ \forall i, j$, the quantum proposal distribution then becomes:

$$\begin{gathered} U(\gamma, t) \approx e^{-i\gamma H_{mix}t} = e^{-i\gamma \sum_j X_j t} = \prod_j e^{-iX_j\gamma t} \\ Q_{(\boldsymbol{s'}|\boldsymbol{s})}(\gamma, t) = |\langle \boldsymbol{s}|U(\gamma, t)|\boldsymbol{s'}\rangle|^2 \approx |\langle \boldsymbol{s}| \prod_j e^{-iX_j\gamma t}|\boldsymbol{s'}\rangle|^2 = \\ |\langle \boldsymbol{s}| \prod_j (cos(\gamma t)I - i\ sin(\gamma t)X_j)|\boldsymbol{s'}\rangle|^2 \end{gathered} \tag{4.6}$$

leading to:

$$\begin{aligned} &\gamma t = \pi m \qquad m \in \mathbb{N} \qquad \rightarrow \qquad Q_{(\boldsymbol{s'}|\boldsymbol{s})}(\gamma, t) \approx 0 \quad \forall\ \boldsymbol{s'} \in S \setminus \{\boldsymbol{s}\} \\ &\gamma t = \pi \frac{m}{2} \qquad m \in \mathbb{N} \qquad \rightarrow \qquad Q_{(\boldsymbol{s'}|\boldsymbol{s})}(\gamma, t) \approx 0 \quad \forall\ \boldsymbol{s'} \in S \setminus \{\bar{\boldsymbol{s}}\} \end{aligned} \tag{4.7}$$

where $\bar{\boldsymbol{s}}$ is the opposite spin configuration with respect to $\boldsymbol{s}$, the most distant in Hamming distance. In other words, when $\gamma = 1$ and $\gamma t = \pi m$ the Markov chain is trapped in the initial state, while when $\gamma t = \pi \frac{m}{2}$ it bounces back and forth between the initial state and the one corresponding to the opposite configuration. When $\gamma \approx 1$ and $\gamma t \approx \pi m$ instead, this effect is only partial. The chain tends to spend a long time in a certain state before jumping onto a new one, generating the fringes corresponding to small spectral gap values. A similar argument can be made for the $\gamma \approx 1$ and $\gamma t \approx \pi \frac{m}{2}$ case. The chain tends to spend a long time jumping between two states before moving onto a new pair, leading to highly correlated samples. The fringes follow indeed the curves defined in the parameter space by Equation 4.7, as illustrated in Figure 4.5.

Equation 4.7 explains also the artifact present in some *Loss* colormaps, such as the one depicted in Figure 4.6. When γ is exactly 1 and $\gamma t = \pi m$ the chain stays in the initial state and the ACF (therefore the *Loss*) goes to zero, indicating a "good" set of parameters.

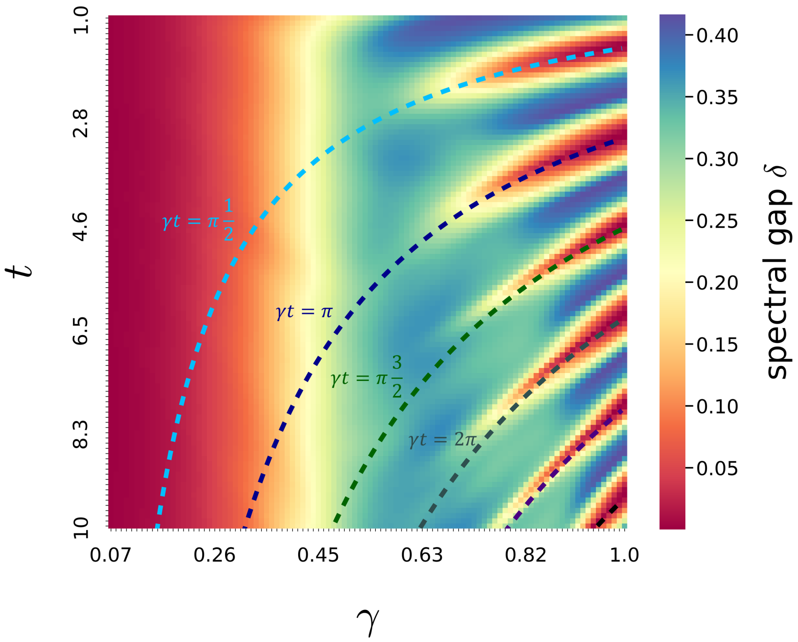


Figure 4.5: Spectral gap colormap showing the average over 10 randomly generated model instances with $N = 6$ and $T = 10$. The dashed lines indicate the parameter pairs fulfilling the conditions $\gamma t = \pi m$ or $\gamma t = \pi\frac{m}{2}$.

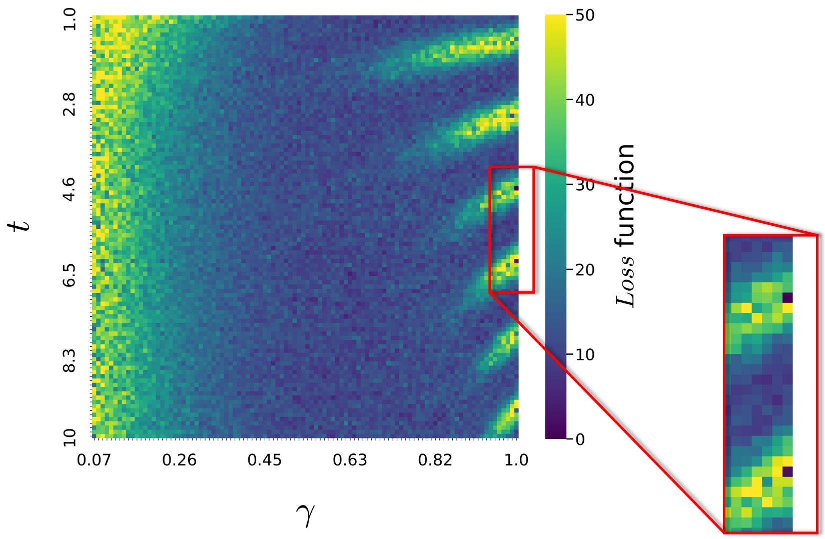


Figure 4.6: Zoomed-in view of a *Loss* colormap showing the artifact occurring when Equation 4.7 is satisfied.

However, we know these parameter pairs to be associated with a small spectral gap, as confirmed by the spectral gap colormaps. A similar argument can be made for the $\gamma t = \pi\frac{m}{2}$ case.

4.1.2 Observable f: magnetization

The simulations previously presented made use of the energy to calculate the *Loss*. The magnetization can be used as well, leading to slightly worse but still satisfying results.

However, when using the *fixed lag Loss*, a situation like the one depicted in Figure 4.7 can occur. The *Loss* assumes small or even negative values in correspondence of the $\pi\frac{m}{2}$ fringes.

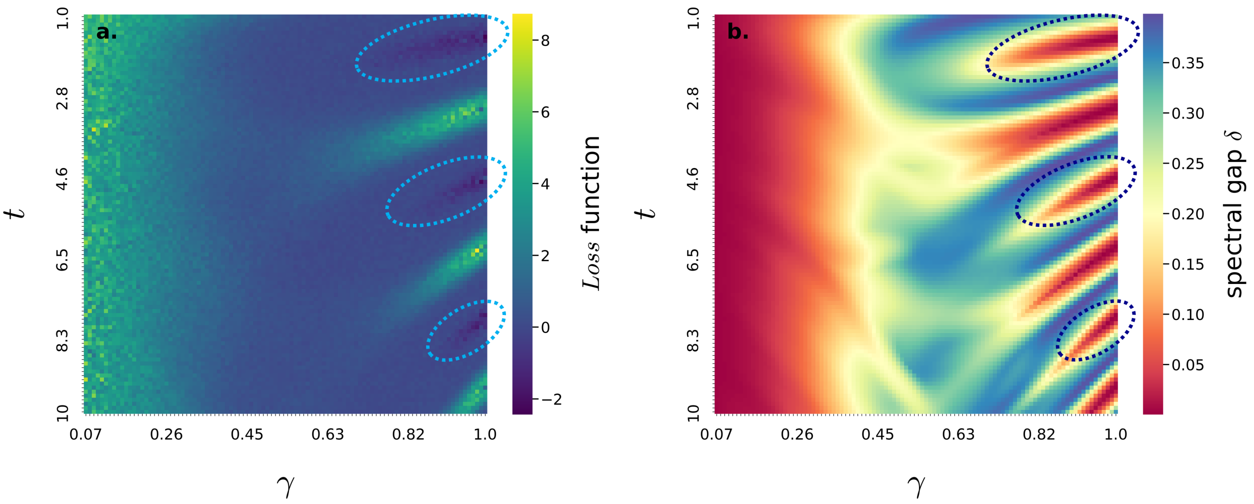


Figure 4.7: *fixed lag Loss* and spectral gap colormaps of a specific model instance (67623) with 6 spins and $T = 10$. The *Loss* uses a fixed lag $l_{fix} = 5$ and the magnetization as observable. Negative *Loss* fringes are indicated by dashed lines. **a.** *fixed lag Loss* colormap. **b.** Spectral gap δ colormap.

Let us consider the limit case $\gamma = 1$ and the $\gamma t = \pi\frac{m}{2}$. We know from Equation 4.7 that the chain jumps back and forth between two states, which feature completely opposite configurations. Therefore, the individual terms in the *Loss*:

$$\hat{\hat{C}}_{ff}(l_{fix}) = \frac{1}{n-l}\sum_{i}^{n-l}(f_i - \bar{f})(f_{i+l} - \bar{f}) \tag{4.8}$$

end up being negative if l_{fix} is odd:

$$\boldsymbol{s} \to M(\boldsymbol{s}), \quad \bar{\boldsymbol{s}} \to M(\bar{\boldsymbol{s}}) = -M(\boldsymbol{s}) \quad \Rightarrow \quad (M - \bar{f})(-M - \bar{f}) < 0 \tag{4.9}$$

given that $\bar{f} = 0$. When $\gamma \approx 1$ and the $\gamma t \approx \pi\frac{m}{2}$ instead, the chain has more freedom and does not visit only a single pair of opposite configurations. However, it still spends a long time jumping between opposite configurations, leading to a similar result. Note that the *lag integration Loss* does not suffer from this problem.

4.1.3 Alternative models

In order to test the generality of our optimization approach, the proposed *Loss* functions have been tested on different models. Once again, we discretized the parameters space in 10^4 points and for each point, corresponding to a pair (γ, t), we calculated the *Loss* (using $n = 4 \times 10^3$ samples) and the spectral gap of randomly generated model instances, plotting the data as colormaps. The various models we considered are:

- Figure 4.9: 1D Ising model with normally distributed coupling $J_{ij} \sim \mathcal{N}(0, 1)$ and $h_i = 0$

- Figure 4.10: 1D ferromagnetic Ising model with $J_{ij} = 1$ and $h_i = -0.5$
- Figure 4.11: 2D square lattice Ising model with nearest neighbour normally distributed coupling $J_{\langle ij \rangle} \sim \mathcal{N}(0, 1)$ and $h_i = 0$. The square lattice is wrapped into a torus (periodic boundary conditions).

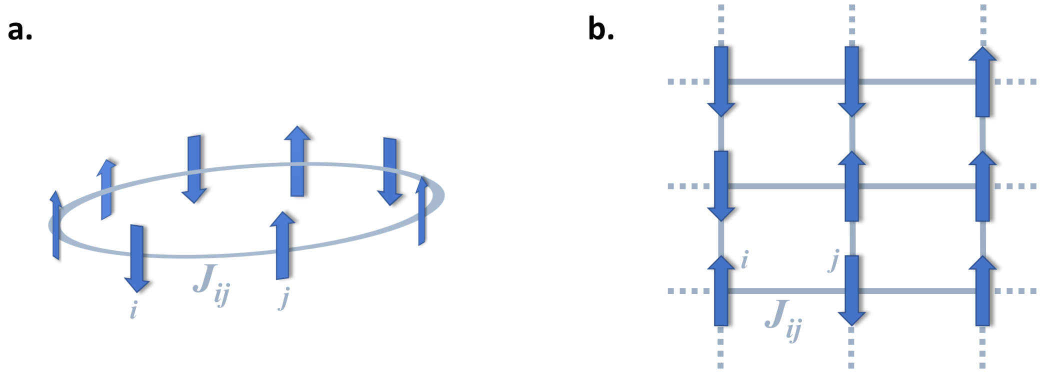


Figure 4.8: Pictorial representation of Ising models. The blue arrows represent the spins while the edges are the couplings J_{ij} **a.** 1D Ising model with periodic boundary conditions. **b.** 2D square lattice Ising model with periodic boundary conditions and nearest neighbor coupling.

We considered periodic (toroidal) boundary conditions in all three cases for the sake of simplicity. The *Loss* is able once again to replicate all the features present in the spectral gap colormap, demonstrating its effectiveness.

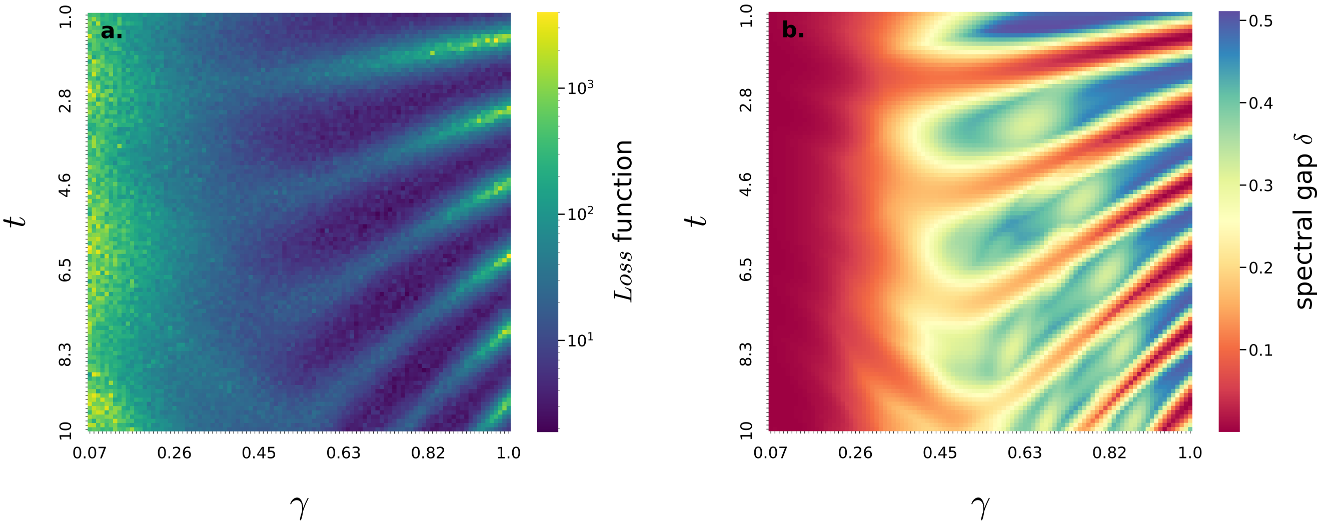


Figure 4.9: 1D Ising model with normally distributed coupling $J_{ij} \sim \mathcal{N}(0, 1)$ and $h_i = 0$. **a.** *integration lag Loss* colormap of a model instance (6723) with $N = 8$ spins and $T = 10$. **b.** Spectral gap δ colormap of the same model instance.

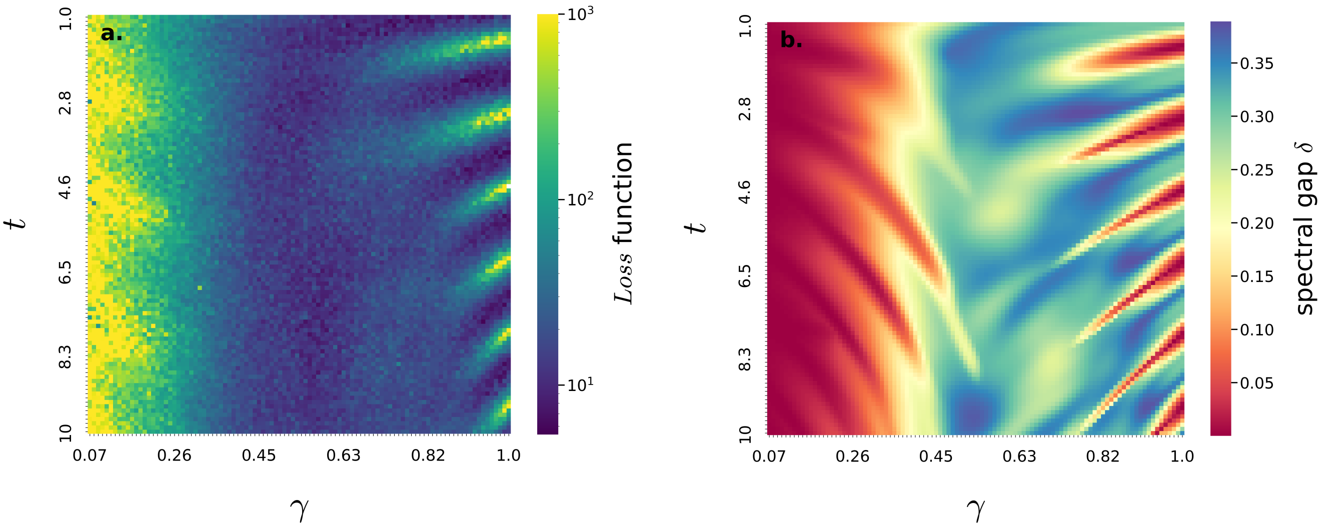


Figure 4.10: 1D ferromagnetic Ising model with $J_{ij} = 1$ and $h_i = -0.5$. **a.** *integration lag Loss* colormap of a model instance (6723) with $N = 8$ spins and $T = 10$. **b.** Spectral gap δ colormap of the same model instance.

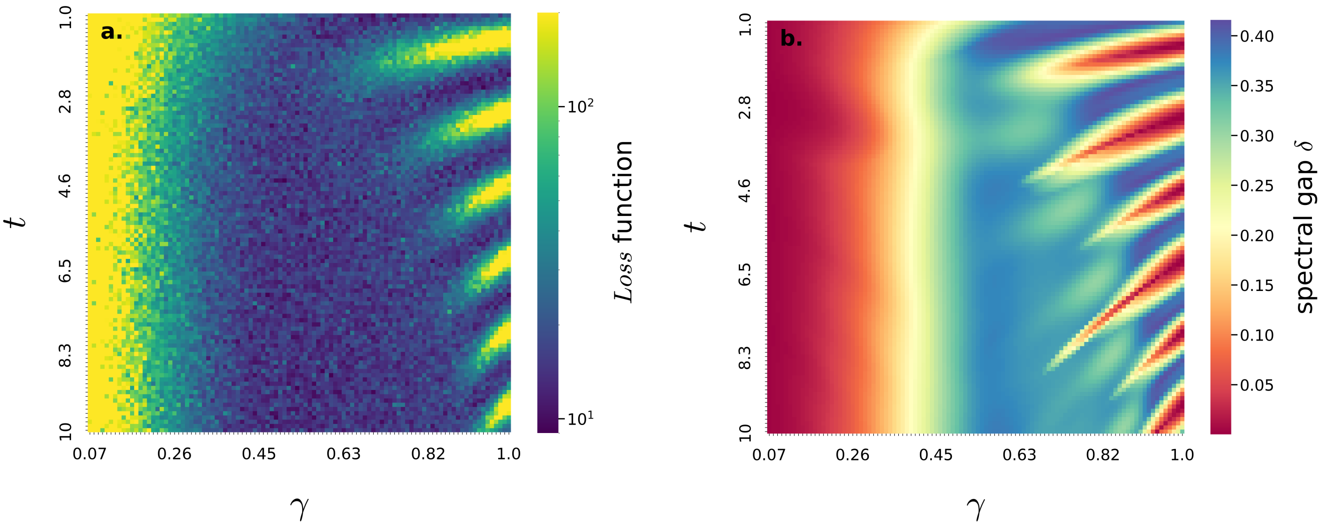


Figure 4.11: 2D square lattice Ising model with nearest neighbour normally distributed coupling $J_{\langle ij \rangle} \sim \mathcal{N}(0, 1)$ and $h_i = 0$. **a.** *integration lag Loss* colormap of a model instance (67623) with $N = 9$ spins and $T = 10$. **b.** Spectral gap δ colormap of the same model instance.

4.2 Optimization at low temperatures

When the temperature T is lowered, sampling from the Boltzmann probability distribution of Ising models with MCMC becomes more difficult. In particular, the transition matrix spectrum changes in the following way:

1. The landscape defined by the Boltzmann probability becomes more rugged as the energy barriers between different states grow exponentially fast with respect to $\beta = \frac{1}{T}$. In other words, it gets harder to explore the full state space because low-energy states have an exponentially larger probability (defined by Equation 2.2) to be visited with respect to the others and getting stuck in local minima becomes very likely. This cause the chain to have a slow mixing time, as the non-dominant modes decay is slower, leading to a clustering of the eigenvalues towards 1.

2. Consequently to this clustering, the spectral gap δ becomes smaller, indicating a slower convergence of the Markov chain.

These changes affect the optimization algorithm performance as well. Firstly, a slower convergence implies that a longer time is needed in order to sample from a probability distribution close to the target one. Since the *Loss* function is an estimator whose definition assumes using samples from a stationary Markov chain, this negatively affects the quality of the estimate. Moreover, as the temperature is lowered, Markov chains tend to spend increasingly more time, as often the proposed moves get rejected, in a smaller and smaller number of states. At low temperatures, it becomes difficult to estimate the correlation on a limited set of samples, resulting in a noisy and unreliable optimization landscape as shown in Figure 4.12.

The direct solution to this problem is running longer chains so that they have enough time to explore different states and show signs of autocorrelation which can be detected by the ACF estimator. In other words, we increase the number of samples n used to evaluate the *Loss* according to target the temperature regime. This approach reduces the noise in the optimization landscape, as shown in Figure 4.13. However, it also increases significantly the algorithm running time. Some alternative solutions, such as combining our algorithm with classical techniques developed for MCMC sampling at low temperatures, may be able to achieve satisfying results without being as computationally demanding [18, 19].

4.2.1 Low temperature simulations

model : fully connected 2D Ising model with random $J_{ij} \sim \mathcal{N}(0,1)$ and $h_i \sim \mathcal{N}(0,1)$

simulation type	samples n	lag l	observable f	T	average over # model instances	simulations per model instance
optimization	$2,5 - 10 \times 10^3$	integral	energy $E(\boldsymbol{s})$	$1 - 0.1$	5	5
colormap (1)	5×10^3	integral	energy $E(\boldsymbol{s})$	$10 - 0.1$	1	1
colormap (2)	$5 - 40 \times 10^3$	integral	energy $E(\boldsymbol{s})$	1	1	1

Table 4.3: Simulations specs used for numerical simulations reported in Figure 4.12, Figure 4.13, Figure 4.14.

Figure 4.14 depicts the evolution of the average spectral gap and *Loss* value during the optimization process over several model instances. The *lag integration Loss* has been

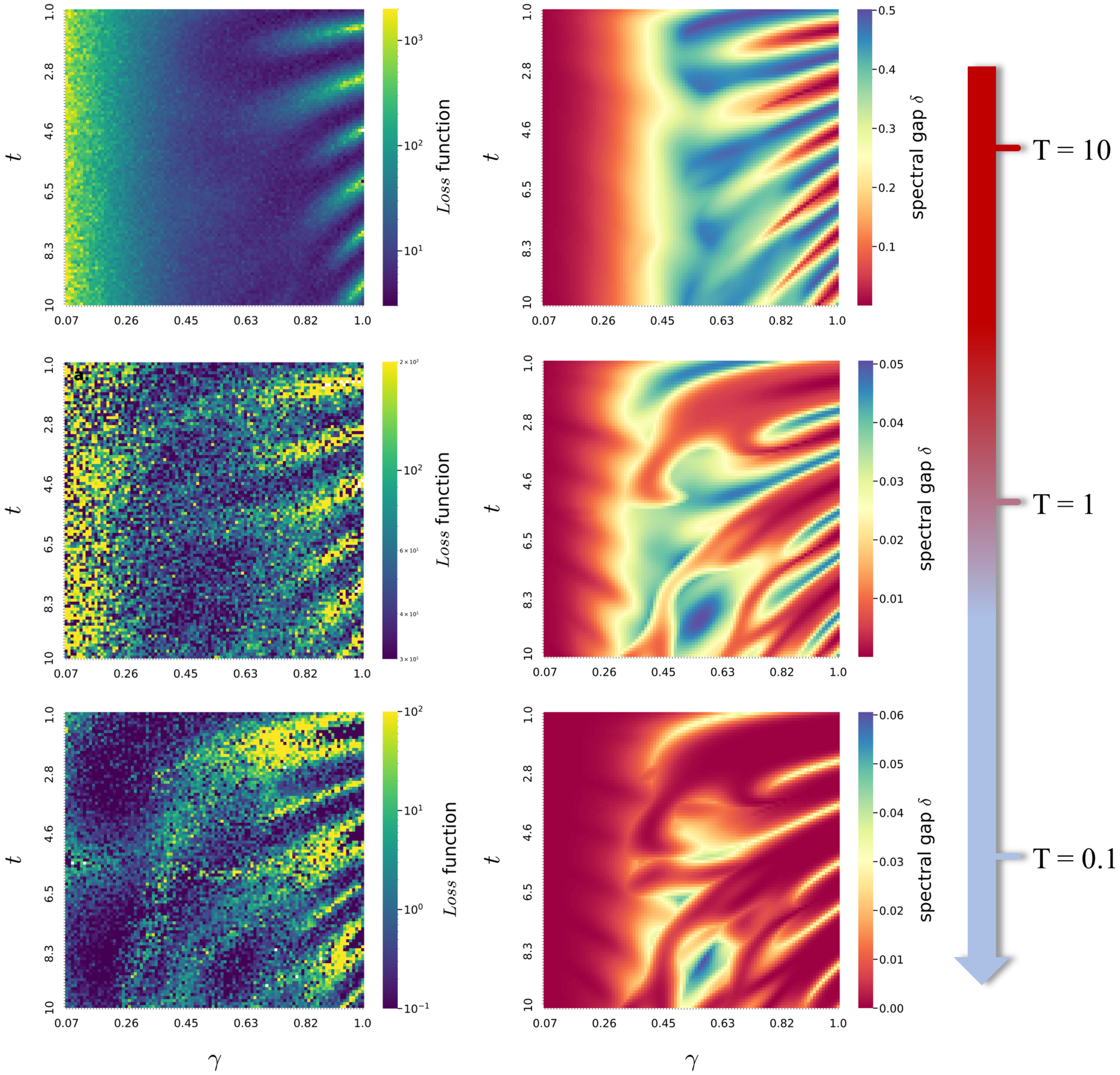


Figure 4.12: *Loss* and spectral gap temperature dependencies. The *Loss* and spectral gap of a randomly generated model instance (630201) with $N = 6$ spins have been simulated at different temperatures using $n = 5 \times 10^3$ samples. As the temperature is lowered, a slower convergence combined with growing energy barriers between different states results in a noisy and unreliable optimization landscape. At $T = 1$ the optimization landscape becomes very noisy. At $T = 0.1$ the *Loss* is not able to replicate the spectral gap landscape's features, making the optimization unfeasible. Simulations specs can be found in Table 4.3 under "colormap (1)".

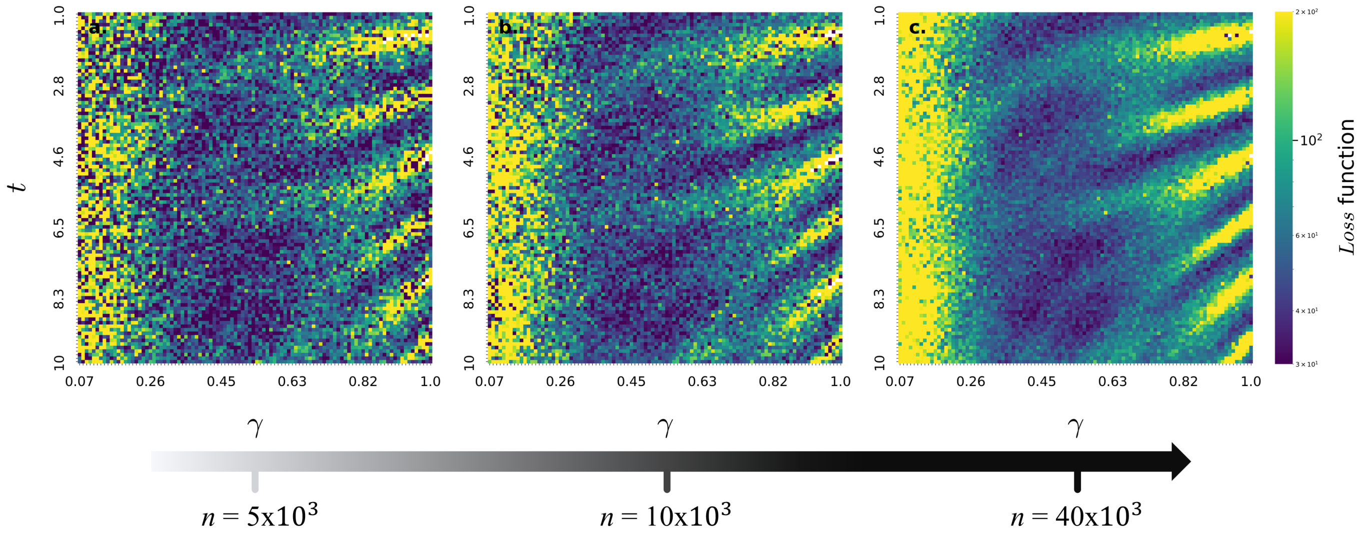


Figure 4.13: *Loss* colormap at $T = 1$ of a randomly generated model instance (630201) with $N = 6$ spins. As the number of samples n increases, the visibility in the optimization landscape is restored. Simulations specs can be found in Table 4.3 under "colormap (2)".

adopted in order to avoid the burden of selecting suitable l_{fix} values for different temperatures. Each plot shows the average of 25 simulations: we used 5 randomly generated model instances and run 5 simulations per instance, where the initial γ and t values were uniformly sampled from $[0.1, 0.25]$ and $[1, 10]$ respectively. The results are compared to the ones obtained using the approach developed by Layden et al. [12] and presented in chapter 2, where γ and t are uniformly sampled from reasonable intervals, $[0.25, 0.6]$ and $[2, 20]$ respectively. At $T = 1$, using the same number of samples $n = 2,5 \times 10^3$ we used at $T = 10$ (section 4.1), the optimization still converges to spectral gap values larger than the ones reached by randomly sampling (γ, t). However, the orange trace representing the average *Loss* value during the optimization is not monotone (Figure 4.14b, first plot), and the scatter plot (Figure 4.14c, first plot) does not show a correlation between the average spectral gap and the *Loss* as strong as in the $T = 10$ case. This is caused by the random noise that, at each optimization step, significantly modifies the landscape explored by the optimizer. At $T = 0.1$, using $n = 2,5 \times 10^3$ samples, the optimization does not succeed, as shown in the second plot of Figure 4.14. Because of the reduced mobility of the chain, the *Loss* is not able to detect correlation with only $n = 2,5 \times 10^3$ samples. The resulting unreliable optimization landscape makes the optimization unfeasible. By increasing n to 10×10^3 (Figure 4.14, third plot), therefore significantly increasing the algorithm running time, we can recover visibility in the optimization landscape and obtain satisfying results.

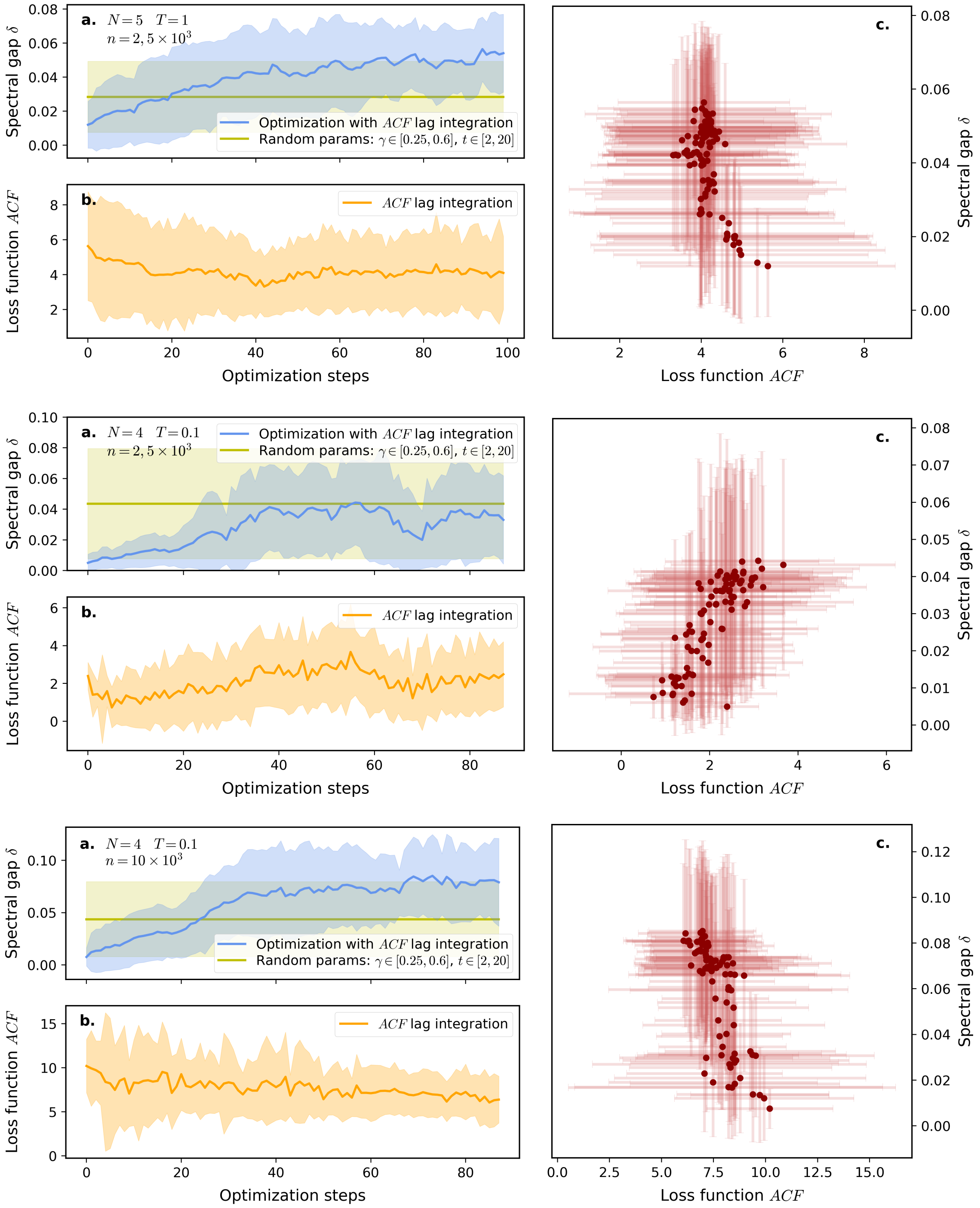


Figure 4.14: a. Optimization algorithm simulations at low temperatures. The average spectral gap is represented by the blue solid lines, the error bands show the population standard deviation. The results are compared with the approach used by Layden et al. (random parameters) [12]. **b.** The average *Loss* is represented by orange the solid lines, the error bands show the population standard deviation. **c.** Correlation between average spectral gap and average *Loss*. The error bars show the population standard deviation. Simulations specs can be found in Table 4.3 under "optimization".

4.3 Scaling with problem's size

A crucial point to address is understanding whether or not the presented algorithm constitutes a scalable approach to optimization. In particular, we want to know how to scale the hyperparameters n and l with respect to the number of spins N in order to maintain the performance. The spectral gap δ is known to decrease, meaning that the autocorrelation time τ of the slowest mode increases (Equation 3.4), as the number of spins N increases. This reduces the visibility in the optimization landscape as it becomes more difficult for the optimizer to distinguish between different proposal distributions.

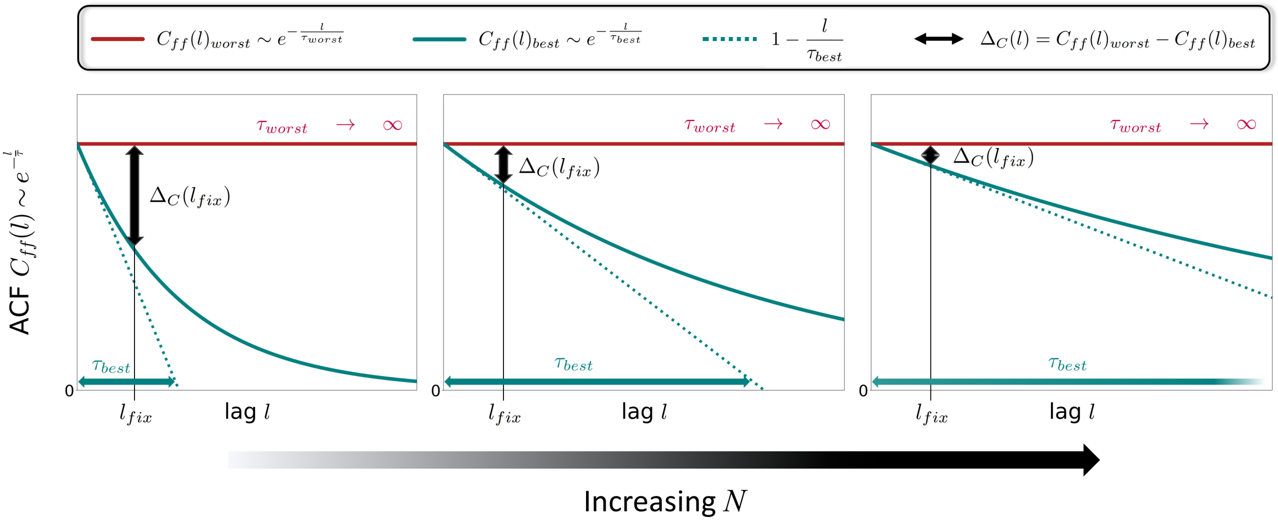

$C_{ff}(l)_{worst} \sim e^{-\frac{l}{\tau_{worst}}}$
$C_{ff}(l)_{best} \sim e^{-\frac{l}{\tau_{best}}}$
$1 - \frac{l}{\tau_{best}}$
$\Delta_C(l) = C_{ff}(l)_{worst} - C_{ff}(l)_{best}$
ACF $C_{ff}(l) \sim e^{-\frac{l}{\tau}}$
$\tau_{worst} \rightarrow \infty$
$\Delta_C(l_{fix})$
τ_{best}
0
l_{fix}
lag l
Increasing N

Figure 4.15: Pictorial representation of the visibility scaling with respect to the problem size (number of spins N). $C_{ff}(l)_{worst}$ and $C_{ff}(l)_{best}$ represent the ACFs of the Markov chains characterized by the worst and best proposal distributions respectively, given a certain parametrized unitary $U(\bar{\theta})$.

Let us consider a generic parameterized unitary $U(\bar{\theta})$. We aim to optimize it using the *fixed lag Loss*. We define the visibility in the optimization landscape as:

$$\Delta_C(l_{fix}) = C_{ff}(l_{fix})_{worst} - C_{ff}(l_{fix})_{best} \tag{4.10}$$

where $C_{ff}(l)_{worst}$ is the ACF, with decay rate τ_{worst}, of the Markov chain characterized by the worst possible proposal distribution $Q_{(\boldsymbol{s}|\boldsymbol{s}')}(\bar{\theta}_{worst})$, featuring the smallest possible spectral gap. On the other hand, $C_{ff}(l_{fix})_{best}$ represents the ACF, with decay rate τ_{best}, of the Markov chain characterized by the best possible proposal distribution $Q_{(\boldsymbol{s}|\boldsymbol{s}')}(\bar{\theta}_{best})$, featuring the largest possible spectral gap. In other words, $\bar{\theta}_{worst}$ and $\bar{\theta}_{best}$ represent the performance limits of the chosen unitary $U(\bar{\theta})$. Let us now remember that the *fixed lag Loss* is a noisy estimator of the ACF evaluated at $l = l_{fix}$ (see subsection 3.1.1 for more details). If $\Delta_C(l_{fix})$ is reduced to the same scale of the estimator's noise η, the optimizer is not able to distinguish between the worst and best proposal distributions. Therefore, any proposal distribution is effectively indistinguishable from the others.
We assume that the chosen observable f is dominated by a single exponential autocorre-

lation time:

$$\begin{aligned} C_{ff}(l)_{best} &\approx Be^{-\frac{l}{\tau_{best}}} \\ C_{ff}(l)_{worst} &\approx Be^{-\frac{l}{\tau_{worst}}} \end{aligned} \tag{4.11}$$

where B is a constant depending on the specific observable. We consider the worst possible proposal distribution to feature a diverging autocorrelation time $\tau_{worst} \to \infty$ (corresponding to $\delta \to 0$):

$$C_{ff}(l)_{worst} \approx Be^{-\frac{l}{\tau_{worst}}} \approx B \tag{4.12}$$

redefining the visibility as:

$$\Delta_C(l_{fix}) \approx B - C_{ff}(l_{fix})_{best} \tag{4.13}$$

This constitutes the best-case scenario as $\Delta_C(l_{fix})$ is maximized. If $\Delta_C(l_{fix}) \lesssim \eta$ when considering $\tau_{worst} \to \infty$, then any other case involving a different τ_{worst} value wolud still lead to $\Delta_C(l_{fix}) \lesssim \eta$. We are now interested in estimating $C_{ff}(l)_{best}$ in the limit of large N. Given that, in general, we have:

$$\lim_{N \to \infty} \delta = 0 \tag{4.14}$$

we approximate τ_{best}, for large N, in the following way:

$$\tau_{best} = -\frac{1}{ln(1 - \delta_{best})} \approx \delta_{best}^{-1} \tag{4.15}$$

Considering once again the limit of large N, we have:

$$C_{ff}(l)_{best} \approx Be^{-\frac{l}{\tau_{best}}} \approx B(1 - \frac{l}{\tau_{best}}) \tag{4.16}$$

Finally, we obtain:

$$\Delta_C(l_{fix}) \approx B - B(1 - \frac{l_{fix}}{\tau_{best}}) = Bl_{fix}/\tau_{best} \approx Bl_{fix}\delta_{best} \tag{4.17}$$

As discussed in subsection 3.1.2, we consider the estimator's noise η to go down approximately as $1/\sqrt{n}$, where n is the number of samples. In general, depending on the specific problem at hand, the noise could feature a complicated dependency on the problem size, therefore we have $\eta \approx \Omega(N)/\sqrt{n}$. The visibility to noise ratio, in the limit of large N, is then given by:

$$\Delta_C(l_{fix})/\eta \;\approx\; \frac{B(N)\sqrt{n}l_{fix}\delta_{best}(N)}{\Omega(N)} \tag{4.18}$$

Equation 4.18 can be used to predict whether the optimization algorithm scales efficiently in the limit of large N for a specific problem. A similar but slightly more complicated argument can be made for the *lag integration Loss* as well.

Let us consider the problem we focused on in previous sections, i.e. sampling from the Boltzmann probability distribution of Ising models. We can safely assume, following Equation 3.9, that $\Omega(N)$ is on the same order of magnitude of $B(N)$ or even larger. Therefore, if the spectral gap decays exponentially fast with respect to the number of spins N, the number of samples n has to be increased exponentially in order to maintain the same visibility to noise ratio. On the other hand, if sampling from a specific Ising model instance involves a spectral gap with a slower (polynomial) decay, the algorithm may scale efficiently. In other words, the scaling of the required number of samples for optimization aligns proportionally with the scaling of the problem at hand.

Chapter 5

Conclusions

Quantum-enhanced Monte Carlo Markov chains are a class of hybrid algorithms combining classical Monte Carlo Markov chains methods with quantum-enhanced proposal distributions defined by parameterized unitaries. They perform Monte Carlo Markov chains proposing moves through a quantum computer, which are then rejected or accepted by a classical one. These algorithms showed potential for achieving quantum advantage in sampling from complex probability distributions, a computationally hard task arising in many diverse fields. However, no straightforward method is available for optimizing the parameters defining the quantum-enhanced proposal distributions, which play a fundamental role in determining the actual advantage.

The present thesis provides valuable insights into the optimization of quantum (and classical) parametrized proposal distributions for Monte Carlo Markov chains methods. We proposed a general optimization algorithm that exploits estimates of the autocorrelation function, calculated over a set of samples, to find the best proposal distribution. The optimization algorithm iteratively updates the parameters defining the quantum proposal distribution until a *Loss* function converges to a minimum. Inspired by the Variational Quantum Eigensolver algorithm, it consists of a feedback loop that runs quantum-enhanced Markov chains to evaluate a *Loss* function and pass the value to a classical optimizer. The classical optimizer provides then new parameter values to the parametrized quantum circuit implementing the quantum state proposal, thus initiating the successive step.

As a proof of concept, we classically simulated the optimization of specific quantum proposal strategies tailored for sampling from the Boltzmann probability distribution of Ising models at low temperatures. We showed how, by optimizing the parameters with our algorithm, it is possible to achieve better proposal distributions leading to a significantly faster convergence compared to other known approaches. The results were quantified in terms of the spectral gap δ. The various proposed *Loss* functions, tested on different Ising models, showed a clear correlation with the spectral gap value, proving their effectiveness. Moreover, the spectral gap landscape was analyzed, and its characteristic features were explained. We also investigated the evolution of the algorithm's performance when the temperature is lowered, proposing a possible solution to reduce the noise in the optimization landscape and obtain satisfying results at $T < 10$ at the cost of increasing the algorithm running time. Finally, the current limitations of the optimization algorithm and its potential scalability with respect to the problem's size were discussed. Under reasonable assumptions, we derived an approximate scaling law that predicts whether the algorithm's performance in the limit of large N (where N represents the problem's size).

5.1 Outlooks

While this thesis addressed many relevant questions, there are still several that remain unexplored. Moving forward, there are several promising research directions that could build on the findings presented in chapter 3. In the following, we will discuss some of these potential outlooks:

- **Adaptive ansatz:** the unitary investigated in the present work is specifically tailored to address the problem of sampling from the Boltzmann distribution of Ising models. A more general approach could be found in adaptive ansatzes. Adaptive ansatzes proved to be effective solutions in variational quantum algorithms [20], therefore they may turn out to be not only a more general, but also a better-performing approach toward sampling from complex probability distributions using quantum computers.

- **Parallel tampering and simulated annealing:** Sampling with Markov chains at low temperatures is a well-known problem, therefore effective solutions in classical settings have already been developed, such as parallel tampering and simulated annealing [18, 19]. These techniques could be combined with our algorithm in order to reduce the computational resources needed to achieve satisfying results at low temperatures.

5.2 Code availability

The code used for numerical simulations can be found in the following GitHub repository: https://github.com/DanieleCucurachi/QMCMC.git

Bibliography

[1] P. Bremaud. *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues.* Texts in Applied Mathematics. Springer New York, 2013. ISBN: 978-1-4757-3124-8. URL: `https://books.google.co.uk/books?id=jrPVBwAAQBAJ`.

[2] Werner Krauth. *Statistical Mechanics: Algorithms and Computations.* 2006.

[3] Yuval Peres David A. Levin. *Markov Chains and Mixing Times (Second Edition).* Vol. 41. Mar. 1, 2019. DOI: `10.1007/s00283-018-9839-x`. URL: `https://doi.org/10.1007/s00283-018-9839-x`.

[4] Radford M. Neal. *Probabilistic Inference Using Markov Chain Monte Carlo Methods.* 2011.

[5] A. Sokal. "Monte Carlo Methods in Statistical Mechanics: Foundations and New Algorithms". In: *Functional Integration: Basics and Applications.* Ed. by Cecile DeWitt-Morette, Pierre Cartier, and Antoine Folacci. Boston, MA: Springer US, 1997. ISBN: 978-1-4899-0319-8. DOI: `10.1007/978-1-4899-0319-8_6`. URL: `https://doi.org/10.1007/978-1-4899-0319-8_6`.

[6] Christian P. Robert and George Casella. "Monte Carlo Statistical Methods". In: *Technometrics* 47 (2005).

[7] "Markov chain Monte Carlo". In: *Wikipedia, The Free Encyclopedia* (2010). [Online; accessed 15-March-2023]. URL: `https://en.wikipedia.org/wiki/Markov_chain_Monte_Carlo`.

[8] Madeleine B. Thompson. "A Comparison of Methods for Computing Autocorrelation Time". In: (2010). DOI: `10.48550/arXiv.1011.0175`. arXiv: `1011.0175 [stat.CO]`. URL: `https://arxiv.org/abs/1011.0175`.

[9] Tjerk P. Straatsma, Herman J. C. Berendsen, and A. J. Stam. "Estimation of statistical errors in molecular simulation calculations". In: *Molecular Physics* 57 (1986), pp. 89–95. DOI: `10.1080/00268978600100071`. URL: `https://doi.org/10.1080/00268978600100071`.

[10] T. W. Anderson. "The Statistical Analysis of Time Series". In: *Wiley Series in Probability and Statistics* (June 1994), pp. 464–471. DOI: `10.1002/9781118186428`. URL: `https://onlinelibrary.wiley.com/doi/book/10.1002/9781118186428`.

[11] M. B. Priestly. "Spectral Analysis and Time Series". In: (1981), pp. 324–328. DOI: `doi.org/10.1002/for.3980010411`. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1002/for.3980010411`.

[12] David Layden et al. "Quantum-enhanced Markov chain Monte Carlo". In: (2022). DOI: `10.48550/ARXIV.2203.12497`. URL: `https://arxiv.org/abs/2203.12497`.

[13] Kostyantyn Kechedzhi et al. "Efficient population transfer via non-ergodic extended states in quantum spin glass". In: (2018). DOI: https://doi.org/10.48550/arXiv.1807.04792. arXiv: 1807.04792 [quant-ph]. URL: https://arxiv.org/abs/1807.04792v1.

[14] Vadim N. Smelyanskiy et al. "Nonergodic Delocalized States for Efficient Population Transfer within a Narrow Band of the Energy Landscape". In: *Physical Review X* 10.1 (Jan. 2020). DOI: 10.1103/physrevx.10.011017. URL: https://doi.org/10.1103%5C%2Fphysrevx.10.011017.

[15] E. J. Crosson and D. A. Lidar. "Prospects for quantum enhancement with diabatic quantum annealing". In: *Nature Reviews Physics* 3.7 (May 2021), pp. 466–489. DOI: 10.1038/s42254-021-00313-6. URL: https://doi.org/10.1038%5C%2Fs42254-021-00313-6.

[16] Fuchang Gao and Lixing Han. "Implementing the Nelder-Mead simplex algorithm with adaptive parameters". In: *Computational Optimization and Applications* 51.1 (Jan. 1, 2012), pp. 259–277. ISSN: 1573-2894. DOI: 10.1007/s10589-010-9329-3. URL: https://doi.org/10.1007/s10589-010-9329-3.

[17] Wim Lavrijsen et al. "Classical Optimizers for Noisy Intermediate-Scale Quantum Devices". In: *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, 2020. DOI: 10.1109/qce49297.2020.00041. URL: https://doi.org/10.1109%5C%2Fqce49297.2020.00041.

[18] Koji Hukushima and Koji Nemoto. "Exchange Monte Carlo Method and Application to Spin Glass Simulations". In: *Journal of the Physical Society of Japan* 65.6 (1996), pp. 1604–1608. DOI: 10.1143/JPSJ.65.1604. eprint: https://doi.org/10.1143/JPSJ.65.1604. URL: https://doi.org/10.1143/JPSJ.65.1604.

[19] Peter JM Van Laarhoven et al. *Simulated annealing*. Springer, 1987.

[20] Harper R. Grimsley et al. "An adaptive variational algorithm for exact molecular simulations on a quantum computer". In: *Nature Communications* 10.1 (July 8, 2019), p. 3007. ISSN: 2041-1723. DOI: 10.1038/s41467-019-10988-2. URL: https://doi.org/10.1038/s41467-019-10988-2.