Variational Quantum Simulations of Lattice Gauge Theories

DISSERTATION

zur Erlangung des akademischen Grades
doctor rerum naturalium
(Dr. rer. nat.)
im Fach: Physik
Spezialisierung: Theoretische Physik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät
der Humboldt-Universität zu Berlin

von

Paolo Stornati

Präsidentin der Humboldt-Universität zu Berlin:
Prof. Dr.-Ing. Dr. Sabine Kunst

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät:
Prof. Dr. Elmar Kulke

Gutachter: 1. Prof. Dr. Agostino Patella
2. Dr.rer.nat.habil. Karl Jansen
3. Prof. Dr. Phiala Shanahan

Tag der mündlichen Prüfung: 10. August 2021

Abstract

Simulations of lattice gauge theories play a fundamental role in first principle calculations in the context of high energy physics. This thesis aims to improve state-of-the-art simulation methods for first-principle calculations and apply those methods to relevant physical models. We address this problem using three different approaches: machine learning, quantum computing, and tensor networks.

In the context of machine learning, we have developed a method to estimate thermodynamic observables in lattice field theories. More precisely, We use deep generative models to estimate the absolute value of the free energy. This approach can be used where Markov chain Monte Carlo methods are problematic. We have demonstrated the applicability of our method by studying the ϕ^4 theory in two dimensions. Our approach yields more precise measurements compared to the standard Markov-chain Monte Carlo method when we cross a phase transition point in the phase space. In the context of quantum computing, our goal is to improve the current algorithms for quantum simulations. In this thesis, we have addressed two fundamental issues on modern quantum computers: the quantum noise mitigation and the design of *good* parametric quantum circuits. We have developed a mitigation routine for read-out bit-flip errors that can drastically improve quantum simulations. We demonstrate the applicability of the method with numerical simulations on IBM quantum hardware. The design of efficient parametric quantum circuits is fundamental for many variational quantum algorithms. We have developed a dimensional expressivity analysis that can identify superfluous parameters in parametric quantum circuits. Using a hybrid quantum-classical approach, we show how to implement the expressivity analysis using quantum hardware efficiently. We provide a proof of principle demonstration of this procedure on IBM's quantum hardware. Moreover, we also show how to incorporate global symmetries in parametric quantum circuits using the dimensional expressivity analysis. Quantum link models are a formulation of gauge theories in terms of discrete degrees of freedom. Quantum link models naturally make the Hilbert space finite without breaking the gauge invariance. In this thesis, we have studied the U(1) quantum link model in 2+1 dimensions in a ladder geometry. Our goal is to analyze the ground-state properties of the model at finite chemical potential. The only algorithms that are known to be able to perform these tasks are tensor network algorithms. We have used the density matrix renormalization group algorithm to find the ground state of the model at different chemical potentials.

We have observed different winding number sectors when we have introduced chemical potential in the system. Moreover, we have found a correlation between the flippability of the plaquettes and the winding number sectors.

Zusammenfassung

Simulationen von Gittereichtheorien spielen eine grundlegende Rolle bei Berechnungen ausgehend von ersten Prinzipien, insbesondere im Kontext der Hochenergiephysik. Ziel dieser Dissertation ist es, modernste Simulationsmethoden für Berechnungen zu verbessern und diese Methoden auf relevante physikalische Modelle anzuwenden. Wir begegnen diesem Vorhaben mit drei verschiedenen Ansätzen: maschinelles Lernen, Quantenrechnen und Tensornetzwerken. Im Rahmen des maschinellen Lernens haben wir eine Methode entwickelt, um thermodynamische Observablen in der Gitterfeldtheorien zu berechnen. Genauer gesagt verwenden wir sogenannte tiefe generative Modelle, um den absoluten Wert der freien Energie abzuschätzen. Dieser Ansatz kann verwendet werden, wenn Markov-Ketten-Monte-Carlo-Methoden problematisch sind. Wir haben die Anwendbarkeit unserer Methode demonstriert, indem wir die ϕ^4 -Theorie in zwei Dimensionen untersucht haben. Unser Ansatz liefert wesentlich genauere Messungen im Vergleich zur Standard-Markov-Ketten-Monte-Carlo-Methode, wenn wir einen Phasenübergangspunkt im Phasenraum überqueren. Im Rahmen des Quantenrechnens ist es unser Ziel, die aktuellen Algorithmen für Quantensimulationen zu verbessern. In dieser Arbeit haben wir zwei grundlegende Probleme moderner Quantencomputer angesprochen: die Abschwächung des Quantenrauschens und das Design von gnenügend aussagekräftigen parametrischen Quantenschaltungen. Wir haben eine Abschwächungsmethode für ausgelesene Bit-Flip-Fehler entwickelt, die Quantensimulationen drastisch verbessern kann. Wir demonstrieren die Anwendbarkeit der Methode mit numerischen Simulationen auf der IBM Quantenhardware. Der Entwurf effizienter parametrischer Quantenschaltungen ist für viele Variationsquantenalgorithmen von grundlegender Bedeutung. Wir haben eine dimensionale Expressivitätsanalyse entwickelt, mit der überflüssige Parameter in parametrischen Quantenschaltungen identifiziert werden können. Mit einem hybriden quantenklassischen Ansatz zeigen wir, wie die Expressivitätsanalyse mit Quantenhardware effizient implementiert werden kann. Wir liefern einen Beweis für die prinzipielle Demonstration dieses Verfahrens auf der Quantenhardware von IBM. Darüber hinaus zeigen wir mit Hilfe der dimensionalen Expressivitätsanalyse, wie globale Symmetrien in parametrische Quantenschaltungen integriert werden können. Quantenlink Modelle sind eine Formulierung von Eichentheorien in Bezug auf diskrete Freiheitsgrade. Quantenlink Modelle machen den Hilbert-Raum in natürlicher Weise endlich, ohne die Eichinvarianz zu brechen. In dieser Arbeit haben wir das U(1) -Quantenlink Modell in 2 + 1-Dimensionen

in einer Leitergeometrie untersucht. Unser Ziel ist es, die Grundzustandseigenschaften des Modells bei endlichem chemischem Potential zu analysieren. Die einzigen Algorithmen, von denen bekannt ist, dass sie diese Aufgaben ausführen können, sind Tensornetzwerkalgorithmen. Wir haben den Dichtematrix-Renormierungsgruppenalgorithmus verwendet, um den Grundzustand des Modells bei verschiedenen Werten des chemischen Potentialen zu ermitteln. Wir haben verschiedene Wicklungszahlsektoren beobachtet, bei eingeschalteten chemischen Potenzial. Darüber hinaus haben wir eine Korrelation zwischen der Flippbarkeit der Plaquetten und den Wicklungszahlsektoren gefunden.

Contents

Chapter 1

Introduction

First principles numerical calculations are ubiquitous and of fundamental importance in physics. For example, in the last 40 years, extraordinary results have been achieved in high-energy physics with lattice quantum chromodynamics simulations. On the other hand, there are still many open problems that need to be tackled and lots of physical quantities that we cannot evaluate with the current methods. This thesis aims to improve state-of-the-art simulation methods for first-principle calculations and apply those methods to relevant physical models. We address this effort using three different approaches: machine learning, quantum computing, and tensor networks. All of these approaches are based on variational and optimization problems.

Monte Carlo algorithms are the state-of-the-art tool for lattice quantum field theory simulations. One major problem in Monte Carlo simulation is the critical slowing down. We have tackled this problem using machine learning techniques. The variational problem consists of optimizing a neural network to approximate the partition function of the theory.

Quantum computing will be able to outperform classical computers with an exponential speed-up in some instances. One physical application is the simulation of the real-time evolution of quantum states via a unitary operator. Quantum computers, in principle, will be able to perform this kind of simulation. Nowadays, it is not yet the case. Quantum computers are too noisy and small to perform physical calculations that can impact real-life problems. In this thesis, we develop a mitigation routine that improves the noise problem in quantum computing. Moreover, we develop a dimensional expressivity analysis of parametric quantum circuits to better design quantum circuits. Both methods are crucial for the development of variational quantum simulations.

Tensor network algorithms are powerful simulation tools to simulate systems in the Hamiltonian formalism. The tensor network algorithm studied in this work is based on a variational optimization of a variational ansatz. One of the main advantages of tensor simulations is the absence of the sign problem. We exploit these advantages using the density matrix renormalization group algorithm to study the different winding sectors of the $U(1)$ quantum link model.

These three topics shape the structure of this thesis into three main chapters, each of them developing one of the directions mentioned above. In the following, we briefly highlight the structure of this thesis.

chapter 2 - We exploit the use of machine learning techniques to calculate thermodynamic observables in lattice field theories. This method aims to overcome the critical slowing down problem present in standard Monte Carlo simulations. This chapter introduces basic concepts in quantum field theory and machine learning techniques. We then develop the method that we can use to compute these observables and explain how to control its convergence. We finally provide a numerical demonstration of how our method can outperform Monte Carlo simulations at specific parameter space points.

chapter 3 - We develop two methods to improve quantum simulations addressing two major issues in quantum computing: quantum noise and the design of efficient quantum circuits. A the beginning of the chapter, we present basic concepts in quantum computing, classical simulations of quantum systems, and present with the support of numerical simulations the variational quantum simulation algorithm. The mitigation routine we develop mitigates the bit-flip error in the read-out measurement in quantum computers. We present the algorithm and use numerical simulations both on classical computers and quantum hardware to demonstrate the method's reliability. Subsequently, we develop a dimensional expressivity analysis to detect redundant parametric quantum gates in a given quantum circuit. We provide a proof of principle numerical demonstration of the method's applicability.

chapter 4 - We introduce the tensor network formalism and explain the basic building blocks of the density matrix renormalization group algorithm. We present the $U(1)$ quantum link model. We use tensor network techniques to study the relations between the ground state of the theory and the winding number sectors at non-zero chemical potential in a lattice with a ladder geometry in 2+1 dimensions.

Chapter 2

Estimation of thermodynamic observables with normalizing flows

The use of machine learning techniques to improve physical simulations has stimulated a lot of interest in the scientific community in recent years. One reason for this activity in the community is the need to find new tools to overcome the difficulties present nowadays in the state-of-the-art simulation technique for lattice field theories, i.e., Markov Chain Monte Carlo simulations (MCMC). To simulate a field theory on the lattice, one needs to discretize it on a grid, usually called a lattice. The study of quantum field theory on the lattice is called Lattice Quantum Field Theory (LQFT). LQFT allows for first-principle calculations of physical quantities and, throughout history, had an extreme success, e.g. [Tanabashi et al., 2018]. For example, the properties of hadrons, which are composed of quarks and gluons, are governed primarily by Quantum Chromodynamics (QCD). To properly compute the low energy properties of QCD we need a non-perturbative formulation of the theory. Lattice gauge theory, proposed in [Wilson, 1974], allows for essential non-perturbative first-principle calculations of theoretical properties of QCD. Despite the extreme success LQCD simulations had in the years, there are still many occasions to improve. Critical slowing down is a major issue in LQCD simulations [Schaefer et al., 2011]. With critical slowing down, we mean the increase of the computational cost while approaching the critical points of a theory.
Machine learning-based algorithms can help to overcome, or at least ease, this problem. In the last two years there have been lots of works using machine learning algorithms to improve Monte Carlo simulations in the area of:

- field configuration generation [Kanwar et al., 2020; Rezende et al., 2020; Boyda et al., 2020; Kanwar et al., 2020; Shanahan et al., 2018]
- efficient computations of correlation functions or observables [Nicoli et al., 2021; Yoon et al., 2019; Zhang et al., 2020]
- sign-problem avoidance via contour deformation of path integrals with machine learning [Detmold et al., 2020]

In our work [Nicoli et al., 2021], we focus on the estimation of thermodynamic observables on the lattice. First principle calculations of thermodynamic observables are very important due to their direct link to experimental data. For example, the high-temperature phase structure of QCD is studied in the experiments at the LHC and has a fundamental role in the study of the physics of the early universe. This chapter is organized as follow: in section 2.1 we give an introductory overview of lattice field theory and Monte Carlo methods for LQFT, in section 2.2 we give a brief introduction to the reader of machine learning and in section 2.3 we present the work we have performed in the study of Thermodynamic observables in Lattice Field Theory with deep generative models.

2.1. Lattice Quantum Field Theory

This section will not be a complete review of the field. For a complete review, we refer to [Gattringer and Lang, 2009]. We will focus only on notions that will be relevant to this thesis. Quantum field theory (QFT) is the basis of the theoretical description of matter and interactions in many areas of physics. A generic quantum field theory is described by an action S (in a n-dimensional Minkowski spacetime). S is a function of the fields Φ. The partition function, in Minkowski spacetime, is defined as:

$$Z_m = \int \mathcal{D}[\phi]\, e^{(iS(\Phi))}\,. \tag{2.1}$$

The action $S(\Phi)$ is defined as:

$$S(\Phi) = \int d^n x\, \mathcal{L}(\Phi(x)) \tag{2.2}$$

where n is the dimension of the spacetime and $\mathcal{L}$ is the Lagrangian density. For lattice simulations, it is useful to define the theory in Euclidean spacetime. To do so we need to perform the Wick rotation and transform the temporal coordinate as:

$$i\,t \to \tau. \tag{2.3}$$

With the theory defined in Euclidean spacetime, we can define the Euclidean partition function as:

$$Z = \int \mathcal{D}[\Phi]\, e^{-S(\Phi)}\,. \tag{2.4}$$

We note that, if the action is real, the partition function is no longer a complex number. The expectation value of an observable $\mathcal{O}(\Phi)$ is defined as:

$$\langle \mathcal{O}(\Phi) \rangle = \frac{1}{Z} \int \mathcal{D}[\phi] \mathcal{O}(\Phi)\, e^{-S(\Phi)}\,. \tag{2.5}$$

In a quantum statistical mechanical system described by the Hamiltonian $\mathcal{H}$ and the temperature T, the partition function Z is defined as:

$$Z(T) = Tr\left[e^{-\mathcal{H}/T}\right] \tag{2.6}$$

And the expectation value of a generic quantum mechanical observable is defined as:

$$\langle \mathcal{O}(\Phi) \rangle = \frac{1}{Z(T)} Tr\left[\mathcal{O} e^{-\mathcal{H}/T}\right] \tag{2.7}$$

In the path integral formalism, we can define the theory at finite temperature. The system befinied by the continuous action is infinite-dimensional, and thus it is impossible to be simulated on a computer. We need to introduce a regularization of the theory. Let us introduce a hypercube grid Λ with spacing a in a Euclidean spacetime in n dimensions:

$$\Lambda = (x_1, \ldots, x_k) = (am_1, \ldots, am_k), : m_1, \ldots, m_k \in \mathbb{Z}^n \tag{2.8}$$

where we fix the first direction to be the temporal direction. We also consider the theory at finite volume. We can define the regularized partition function over the lattice Λ as:

$$Z = \int \prod_{x \in \Lambda} d[\Phi(x)]\, e^{-S(\Phi(x))}\,. \tag{2.9}$$

From the definition of partition function we can define also the expectation value of a certain observable $O(\Phi)$:

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \prod_{x \in \Lambda} d[\Phi(x)]\, \mathcal{O}(\Phi) e^{-S(\Phi(x))}\,. \tag{2.10}$$

From the path integral formulation of the partition function, if the action S is a positive-definite function, we can also define the probability distribution function $P(\Phi)$ as:

$$P(\Phi) = \frac{1}{Z} e^{-S(\Phi)}\,. \tag{2.11}$$

In order to obtain precise results for the thermodynamics in lattice gauge theories an accurate tuning of the temperature to the desired value is essential. The temperature is the inverse of the extent of a compactified dimension (usually chosen to be the "time" direction). To do so, we need to define β as the inverse temperature, choose one direction (x_n) in the spacetime to be the temporal direction and define the action as:

$$S(\Phi) = \int_0^\beta dx_1 \int d^{n-1}x\, \mathcal{L}(\Phi(x)) \tag{2.12}$$

Let us suppose the temporal direction to be of extension $L_t = N_t a$. If we impose periodic boundary condition in the temporal direction we can relate the temperature T of the system

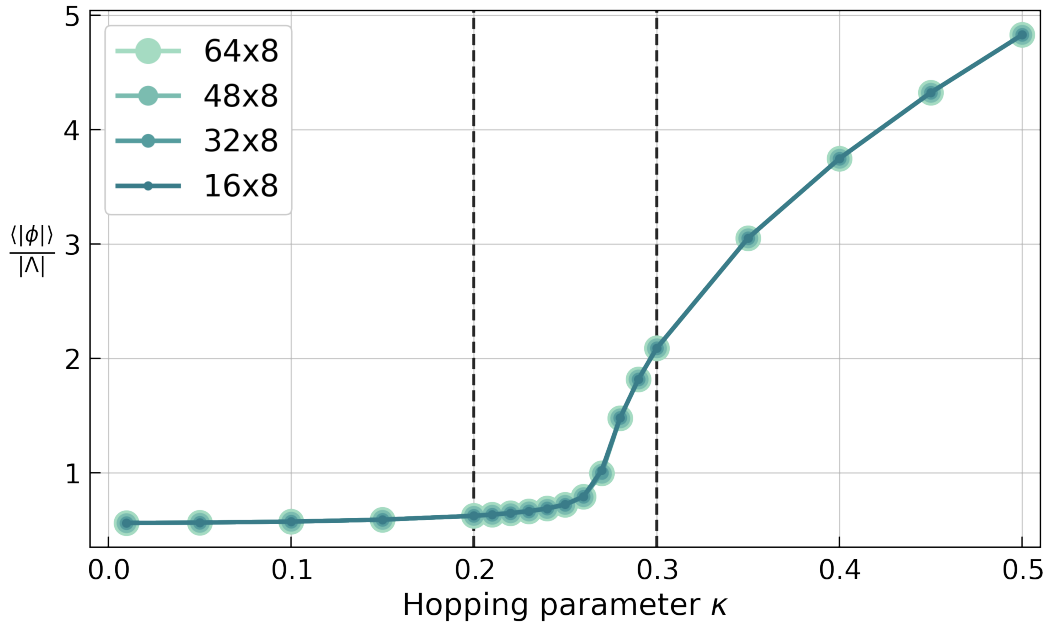


Figure 2.1: *Estimate absolute magnetization at $\lambda = 0.022$ as a function of κ. The two straight dashed lines denotes two points where further analysis will be made.*

as follows:

$$T = \frac{1}{N_t a} \tag{2.13}$$

At fixed temperature T we will then consider the system in a sufficiently large box to avoid finite size effects, i.e. :

$$L_i \gg L_t \ \forall i \in [2, n] \tag{2.14}$$

Once we have defined the finite temperature and the partition function we can define the thermodynamic observables such as the free energy F, the pressure p and the entropy S as:

$$F = -T \ln Z \tag{2.15}$$

$$p = \frac{T\partial(\ln Z)}{\partial V} \tag{2.16}$$

$$S = \frac{\partial\,(T \ln Z)}{\partial T} \tag{2.17}$$

2.1.1. Two-dimensional φ^4 model

The φ^4 theory is a real scalar field theory. The action of the theory, defined in two dimensions and discretized on the lattice, is given by:

$$S(\varphi) = a^2 \sum_{x\in\Lambda} \frac{1}{2} \sum_{\hat{\mu}=1}^{2} \frac{(\varphi(x + a\hat{\mu}) - \varphi(x))^2}{a^2} + \frac{m_0^2}{2}\varphi^2(x) + \frac{g_0}{4!}\varphi^4(x) \tag{2.18}$$

If we use the following re-definitions:

$$\varphi = (2\kappa)^{\frac{1}{2}}\phi\,, \tag{2.19}$$

$$(am_0)^2 = \frac{1-2\lambda}{\kappa} - 4\,, \tag{2.20}$$

$$a^2 g_0 = \frac{6\lambda}{\kappa^2}\,. \tag{2.21}$$

We can write the action S, setting $a = 1$, as:

$$S(\phi) = \sum_{x\in\Lambda} -2\kappa \sum_{\hat{\mu}=1}^{2} \phi(x)\phi(x+\hat{\mu}) + (1-2\lambda)\phi(x)^2 + \lambda\,\phi(x)^4$$

where κ is the hopping parameter and λ the bare coupling constant of the theory. The action is invariant under the global transformation $\phi \to -\phi$:

$$S(\phi) \to S(-\phi) = S(\phi) \tag{2.22}$$

This is a $\mathbb{Z}_2$-transformation. One important observable for the theory is the absolute magnetization $\langle|\phi|\rangle$. The absolute magnetization increase when κ increases, with fixed λ. Spontaneous magnetization is observed when κ is increased. This behavior is shown in Figure 2.1. We can identify the phase transition to be in the interval $[0.2, 0.3]$. It is useful to compute the value of the partition function Z in a point of the phase space. For $\kappa = 0$, The computation of Z decouples in independent integrals for each point of the lattice:

$$Z = \prod_{x\in\Lambda} \left(\int \mathrm{d}\phi(x)\, \exp\left(-\lambda\,\phi(x)^4 - (1-2\lambda)\,\phi(x)^2\right)\right) \tag{2.23}$$

This integral can be solved as:

$$Z(\lambda) = \sqrt{\frac{1-2\lambda}{4\lambda}}\, \exp\left(\frac{(1-2\lambda)^2|\Lambda|}{8\lambda}\right)\, K_{\frac{1}{4}}\left(\frac{(1-2\lambda)^2}{8\lambda}\right) \tag{2.24}$$

where K_n is the modified Bessel function of the second kind. We can obtain the analytic form of the free energy as:

$$F = -T\ \ln(Z)\,, \tag{2.25}$$

This result agrees with [Gattringer and Lang, 2009]. In this book one can find more informations about this calculation and the model.

2.1.2. Monte Carlo simulations

The state-of-the-art method for simulating lattice field theories is Monte Carlo simulations. Monte Carlo is a very generic term that refers to numerical integration methods that involve

random numbers. Given a field theory discretized on the lattice, we would like to compute the partition function evaluating the integral in Equation 2.9. This integral is of very high dimension (can be up to 10^{10} dimensions). Except for certain rare cases, numerical methods are needed for the computation of those integrals. Since the value of the integral function is small in almost all the space and has peaks in only small regions, standard integration routines fail, and importance sampling is needed.
Let us suppose we want to compute an integral in d dimensions over the volume V of the function $f(x)$:

$$I = \int_V dx\, f(x) \tag{2.26}$$

Given a normalized probability distribution function $p(x)$, we can rewrite the integral as:

$$I = \int_V dx\, p(x) \frac{f(x)}{p(x)} \tag{2.27}$$

If we sample N random numbers $x_i \in V$ distributed following the probability distribution $p(x)$, we can estimate I as follow:

$$I = \frac{V}{N} \sum_i \frac{f(x_i)}{p(x_i)} \tag{2.28}$$

We can use the information we know about $f(x)$ to design $p(x)$ such that we get a smoother function $f(x)/p(x)$. This is what is called importance sampling. The integrand of the integral that needs to be computed in Equation 2.9 to estimate the partition function is e^{-S}. This function is strongly peaked, and the tails of those peaks are exponentially suppressed. To solve this problem, we want to generate configurations following the probability distribution function induced by the action of the theory as defined in Equation 2.11 and use those configurations to compute observables (as defined in Equation 2.10). This can be done with Markov chain Monte Carlo methods. We refer to [Gattringer and Lang, 2009] for a deep analysis of the methods. The idea is to start with a random configuration and evolve it creating a chain of configurations. For a lattice field theory, the goal is to stochastically converge to the probability distribution function defined in Equation 2.11. A Markov chain is a series of configurations U_n that satisfy the Markov process condition. A Markov process is defined by the joint probability T defined as:

$$P(\Phi_n = \Phi' | \Phi_{n-1} = \Phi) = T(\Phi' | \Phi) \tag{2.29}$$

T does not depend on the index n and:

$$0 \leq T(\Phi' | \Phi) \leq 1 \tag{2.30}$$

$$\sum_{\Phi'} T(\Phi' | \Phi) = 1 \tag{2.31}$$

The first equation just ensures the probability to be positive and less then one, the latter is the normalization prescription. At the equilibrium, A Markov process must obey the balance condition:

$$\sum_{\Phi} T(\Phi'|\Phi)P(\Phi) \stackrel{!}{=} \sum_{\Phi} T(\Phi|\Phi')P(\Phi'). \tag{2.32}$$

Using the normalization conditions in Equation 2.31 we can derive:

$$\sum_{\Phi} T(\Phi'|\Phi)P(\Phi) = P(\Phi'). \tag{2.33}$$

This equation shows that the probability distribution $P(U)$ is the equilibrium distribution of the Markov chain, i.e., it is a fixed point of the Markov process. One can create a Markov chain by respecting the prescription in Equation 2.32 by imposing the equation to be satisfied piecewise:

$$T(\Phi'|\Phi)P(\Phi) = T(\Phi|\Phi')P(\Phi'). \tag{2.34}$$

This sufficient condition for generating a Markov process is called the *detailed balance condition*. One algorithm that respects these criteria is the Hybrid Monte Carlo (HMC) algorithm. HMC combines a molecular dynamics update [Callaway and Rahman, 1983] with a acceptance/rejection criteria introduced in [Metropolis et al., 1953].

2.1.3. Computation of thermodynamic observable with MCMC

For the computation of thermodynamic observables of lattice field theories with MCMC we mainly follow [de Forcrand et al., 2001; Philipsen, 2013]. It is not possible to compute the partition function with Markov chain Monte Carlo in a given point of the phase space [Gattringer and Lang, 2009]. What is possible is to compute the ratio of the partition function in two different points. Let us suppose we have a theory described by an action S_Ω and partition function Z_Ω that depends only on one parameter Ω. Given two points in the phase space with $\Omega = a$ and $\Omega = b$, we can compute the ratio of the partition functions as:

$$\frac{Z_a}{Z_b} = \frac{1}{Z_b} \int \mathcal{D}[\phi]\, e^{-S_b(\phi)} \frac{e^{-S_a(\phi)}}{e^{-S_b(\phi)}} = \mathbb{E}_{p_b}\left[\frac{\exp(-S_a)}{\exp(-S_b)}\right] \tag{2.35}$$

where $\mathbb{E}_{p_b}$ is the expectation value with respict to the probability distribution p_b. The last term of Equation 2.35 can be computed via MCMC. From the ratio of the partition functions we can derive the difference in free energy between the two points of the parameter space:

$$\Delta F_{ab} = F_a - F_b = -T \ln\left(\frac{Z_a}{Z_b}\right) \tag{2.36}$$

If the two distributions $p_a = e^{-S_a}/Z_a$ and $p_b = e^{-S_b}/Z_b$ are very different and do not have sufficient overlap, the variance of the Monte Carlo estimator becomes prohibitively large. Therefore, it is advisable to take small intermediate steps in the parameter space during

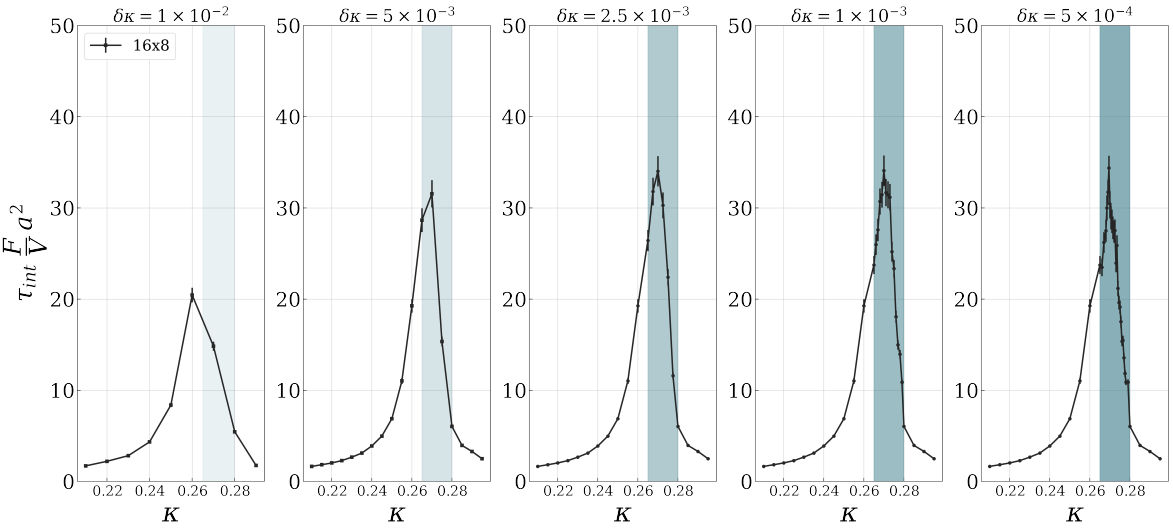


Figure 2.2: *Integrated autocorrelation time of the free energy during refinement of the step size. The lattice spacing a is set to one. The mint areas refer to the interval $\kappa \in [0.265, 0.28]$ for which the refinement is applied. Darker shades refer to smaller step sizes. The step size δ_κ is presented on top of the panels. The experiments were performed using the overrelaxed HCM algorithm.*

the integration in the parameter space, i.e., dividing the interval $[a, b]$ in small steps. When crossing a phase transformation, the problem of the overlap of the two distributions becomes more severe. For example, in [Holland et al., 2008] the probability distribution function of the $SU(4)$ Yang-Mills theory in 3 dimensions is studied in depth. The theory undergoes a weakly first-order deconfinement phase transition at $T = T_c$. It is clearly demonstrated in this work that the probability distribution function p_1 with temperature $T_1 \ll T_c$ and the probability distribution function p_2 with temperature $T_2 \gg T_c$ have basically zero overlap. In this situation, the computation of Z_1/Z_2 is completely unfeasible if we use directly Equation 2.35. If we want to compute the ratio between Z_a and Z_b we can divide the interval $[a, b]$ in N steps : $[a, a+i, a+2i, \ldots a+(n-1)i, b]$ and compute the ratio as:

$$\frac{Z_a}{Z_b} = \frac{Z_a}{Z_{a+i}} \frac{Z_{a+i}}{Z_{a+2i}} \cdots \frac{Z_{a+(n-1)i}}{Z_b} \tag{2.37}$$

The computation of the ratio done in this way is practical in modern Monte Carlo simulations. If we know, for example, the value of the partition function in the point where $\Omega = a$, we can compute:

$$F_b = F_a - \Delta F_{ab} \tag{2.38}$$

In our work we have analyzed the φ^4 model in 2 dimensions presented in subsection 2.1.1. At finite volume, for fixed $\lambda = 0.022$, the model undergoes a phase transition in the interval $\kappa = [0.2, 0.3]$.

In Figure 2.2 the integrated autocorrelation time of the free energy, defined in [Gattringer and Lang, 2009], is presented. The five different panels stand for finer step-size in the integration procedure. As we can see from this plot, a finer integration step is needed to

cross the phase transition point.

2.2. Machine learning

It is not trivial to give a unique definition of what *machine learning* algorithms are. We refer to [Goodfellow et al., 2016] for a comprehensive analysis of the field and [Carleo et al., 2019] comprehensive review of the application to physical models. We can identify in a machine learning algorithm two different elements:

- a task that need to be fulfilled,
- a performance measure.

This measure is fundamental for the learning part of the algorithm. Usually, machine learning algorithms are built as mathematical functions with learnable parameters. Probably, the most famous example and use of this kind of function are neural networks. We can identify three different kinds of learning:

- Supervised learning, where the machine learns from pre-existent data and analyzes them
- Reinforcement learning, where the learning is based on letting the machine interact with the system we want to analyze
- Unsupervised learning, where we are given unlabelled data and we want to discover properties of the mechanism that generates the data

In this thesis, we do not use and thus analyze reinforcement learning. We refer to [Sutton and Barto, 2018] for a review of the subject. In the following two sections, we briefly review supervised and unsupervised learning.

2.2.1. Supervised learning

The main goal of supervised learning algorithms is to learn a certain pattern, distribution, dependencies, or relationships between some output and input data to predict the outcome for new input data. We can identify two main categories of supervised learning: classification [Kotsiantis, 2007] and regression [Yildiz et al., 2017]. The goal of a classification algorithm is to build a classifier that, based on input classified instances, can make predictions about future instances. The output of a classification algorithm is a discrete variable. Let us make a concrete trivial example. We want to construct an algorithm that, provided a picture, can answer the question: "*Is there a cat in this picture?*". This is our task to be fulfilled. We are given the set of couples $X = \{X_i, B_i\}$ where i goes from 1 to N. X_i are pictures and the B_i are Boolean values that answer the question "Is there a cat in this picture? ". A classification machine can learn from the data how to *classify* a new input data. Given a new input picture

y, the classification algorithm c is able to compute the classifier of the picture $c(y)$. One algorithm that is widely used for this task is logistic regression. This is a hybrid algorithm between regression and classification. Logistic regression algorithms are able to predict binary variables: "yes" or "no", "true" or "false", and so on. The logistic regression algorithm can also predict the probability of dependent variable y, given independent variable X, i.e.:

$$P(y|X) = True \quad \text{or} \quad P(y|X) = False \tag{2.39}$$

For our simple example, the logistic regression algorithm can predict the probability that there is a cat in a given picture. More formally, it can predict the probability of the input variable y given the training variables X. The regression algorithm differs from the classification algorithm because the output of the algorithm is continuous variables.

The most famous and straightforward example of a regression algorithm is the one-dimensional linear regression. Simple linear regression studies relationships between two continuous (quantitative) variables that are supposed to have a linear dependence. Given a

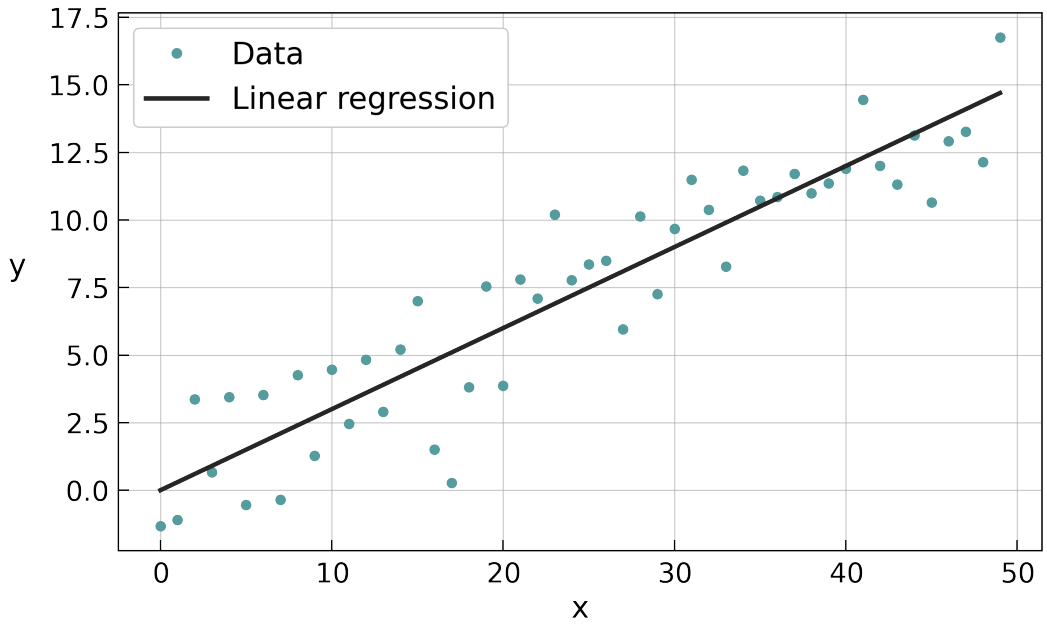


Figure 2.3: *Linear regression of random data*

set of N data $\{x_i, y_i\}$, we want to find the best set of parameters $\theta = \{\theta_1, \theta_2\}$ such that

$$f_\theta(x) = \theta_1 + \theta_2 x \tag{2.40}$$

describes in the best way as possible the data. The best way to compute the parameters θ is to minimize the cost function $C(\theta)$:

$$C(\theta) = \sum_{i=1}^{N} \left(y_i - f_\theta(x_i)\right)^2. \tag{2.41}$$

C is the sum of the squared residuals and this approach is called the method of ordinary least squares. In Figure 3.6 the example of one-dimensional linear regression is presented. This is probably the first regression algorithm presented in the literature. The procedure is easily generalizable in d dimensions and more complicated functions f and loss function C. Of course, there are regression algorithms that are much more complex than linear regression. We refer to [Uysal and Güvenir, 1999] for a full review on the subject.

2.2.2. Unsupervised learning

Unsupervised learning refers to algorithms to identify patterns in data sets containing data points that are neither classified nor labeled. The algorithms are thus allowed to classify, label and/or group the data points contained within the data sets without having any external guidance in performing that task. In other words, unsupervised learning allows the system to identify patterns within data sets on its own.

A specific problem in unsupervised learning is the following. We are given a training set D, usually in the form of samples like $x_n \sim p(x)$ with $n = 1, \dots, N$ generated from an unknown true distribution $p(x)$. The goal is to learn some properties of the distribution $p(x)$ or to be able to create a probability distribution $g(x)$ as close as possible to $p(x)$. It is not always possible to define a useful distance measure of the two distributions. We will show examples of this in the next sections. Once we have $g(x)$, we want to generate samples that are statistically similar to the observed data samples. This is often referred to as generative modeling.

Generative Adversarial Networks (GANs) are machine learning algorithms that had a great impact not only within the scientific community but their application became really popular in the mainstream media. For example, GANs are used to generate images of persons that do not exist [tpd]. GANs belong to the family of generative models in machine learning. We refer to [Wang et al., 2019] for a review of the subject. We will focus on the generation of images as an instructive example. Since images can be stored as $2d$ tensors of which every entry takes one of the three colors (g, r, b), we do not lose generality with this treatment. GANs usually have two fundamental building blocks (functions): a Discriminator D and a Generator G. The Discriminator distinguishes between real images and generated images. The Generator creates images trying to fool the Discriminator via the creation of images that are classified as real by D. Given a simple prior distribution $z \sim p_z$, G defines a probability distribution $p_g = G(z)$, with z sampled from p_z. The goal of GANs is to train the probability distribution p_g that resembles as much as possible the probability distribution defined by the real data p_r. The training of GANs is done via the minimization of a cost function C This cost function is the joint loss function for D and G:

$$C = \mathbb{E}_{x \sim p_r} \log [D(x)] + \mathbb{E}_{z \sim p_z} \log [1 - D(G(z))] \tag{2.42}$$

This loss function can be used both for the training of the discriminator and for the training

of the Generator. Usually D and G are build from deep neural networks.

2.2.3. Neural Networks

Neural networks are a machine learning algorithms that are built of layers which are defined as:

$$y^{(l)}(x) = \sigma(W^l x + b^l) \tag{2.43}$$

The weights $W^l \in \mathbb{R}^{m_l, n_l}$ and the bias $b^l \in \mathbb{R}^n_l$ define an affine map. The function σ is a non-linear function that acts element-wise on the input vector x. The layer $y^{(l)}(x)$ is thus a function from $\mathbb{R}^n_l \to \mathbb{R}^m_l$. The first layer is usually called the input layer. The last layer, i.e., the one that produces the output, is called the output layer. All the other layers are called the hidden layers. We can compose different layers to construct a neural network $g(x) : \mathbb{R}^n_1 \to \mathbb{R}^m_L$ as:

$$g(x) = (y^{(L)} \circ \cdots \circ y^{(1)})(x) \tag{2.44}$$

We can talk about a deep neural network when the number L is sufficiently large. There is no fixed threshold for L, but we can say the neural network is deep when L is bigger than 10.

2.2.4. Normalizing flows

Normalizing flows are machine learning algorithms that are defined by an invertible function. We can also define normalizing flows as tools for constructing complex distributions by transforming a probability density through a series of invertible maps. We refer to [Papamakarios et al., 2019; Rezende and Mohamed, 2016] for a review of the subject. These functions are distributions with learnable parameters. In this section, we present one example of these architectures. The machine learning architecture is called Non-linear Independent Components Estimation (NICE) and is presented in [Dinh et al., 2015]. Let us suppose that we have an element $\mathbf{x} \in \mathbb{R}^n$, and we want to build a transformation function f such that:

$$\mathbf{x} = f(\mathbf{u}), \quad \mathbf{u} \sim P_u(\mathbf{u}) \tag{2.45}$$

Where $\mathbf{u} \in \mathbb{R}^{n'}$ and P is a probability distribution function. P is usually called the base distribution of the flow-based model. The base distribution P and the transformation function f can depend on a set of parameters θ. In our work, we will restrict the dependence on the free parameters θ to the transformation function f. The characteristic and fundamental property of a normalizing flow is that f is invertible and that both f and f^{-1} are differentiable, i.e. f is a diffeomorphism. This also implies that the dimension of the two spaces needs to be equal: $n = n'$. With these assumptions, we can define a probability

distribution function for $\mathbf{x}$ as:

$$\mathbf{x} \sim P_x(\mathbf{x}) = P_u\left(f^{-1}(\mathbf{x})\right)\left|Det(J_f)\right|^{-1} \tag{2.46}$$

Where J_f is the Jacobian of the transformation function f. We can see the absolute Jacobian determinant as the relative change of volume of a small neighborhood around $\mathbf{u}$ ($\mathbf{x} = f(\mathbf{u})$) due to the transformation f. Given two transformations f_a and f_b that are invertible and differentiable, we can express the following property:

$$(f_b \circ f_a)^{-1} = f_a^{-1} \circ f_b^{-1} \tag{2.47}$$

Given the properties of the determinant under the composition of functions we can derive:

$$Det(J_{f_b \circ f_a}(\mathbf{u})) = Det(J_{f_b}(\mathbf{u}))Det(J_{f_a}(\mathbf{u})) \tag{2.48}$$

These properties are useful because they allow us to build an invertible and differentiable transformation via the composition of simpler functions. This is not enough for practical applications. We do not just want to build a normalizing flow, but we also want to use it. For this task, the Jacobian of the transformation needs to be tractable. What do we mean precisely with tractable Jacobian? Let us suppose we sample from a distribution of dimension n. It is always possible to compute the Jacobian determinant with computational cost $O(n^3)$. This is not always feasible when $n \sim O(10^{(3-4)})$. The computation of the determinant for every time we evaluate the flow would result in a huge computational cost. We thus want to construct a network such that, in principle, this evaluation is almost trivial.

2.2.5. Non-linear Independent Components Estimation

One famous example of a normalizing flow architecture that has tractable Jacobian and allows for efficient sampling is the Non-linear Independent Components Estimation (NICE) [Dinh et al., 2015]. The NICE architecture is a deep generative model build from neural networks that has the same layer functional structure of a neural network:

$$f(x) = (f_{(L)} \circ \cdots \circ f_{(1)})(x) \tag{2.49}$$

Each function f_i is called coupling layer. Let us suppose that the output of the i^{th} layer is a vector $\mathbf{x^i} \in \mathbb{R}^n$. The $(i+1)^{th}$ layer splits $\mathbf{x^i}$ in two subset $\mathbf{x_a^i} \in \mathbb{R}^{n-k}$ and $\mathbf{x_b}^i \in \mathbb{R}^k$. The output of the layer is also an element $\left(\mathbf{x_a^{(i+1)}}, \mathbf{x_b^{(i+1)}}\right) \in \left(\mathbb{R}^{n-k}, \mathbb{R}^k\right)$. The $(i+1)^{th}$ layer apply a transformation from $\left(\mathbb{R}^{n-k}, \mathbb{R}^k\right) \rightarrow \left(\mathbb{R}^{n-k}, \mathbb{R}^k\right)$ described by:

$$\begin{aligned} \mathbf{x_a^{(i+1)}} &= \mathbf{x_a^{(i)}} \\ \mathbf{x_b^{(i+1)}} &= \mathbf{x_b^{(i)}} + m\left(\mathbf{x_a^{(i)}}\right) \end{aligned} \tag{2.50}$$

where m is an arbitrary complex function. It is common to define m as a neural network. Usually, the parameters of the neural network m are the free trainable parameters of the flow. m usually take the form of Equation 2.43. One important feature of this approach is that there is a lot of freedom in the definition of m.
The Jacobian of this layer is defined as:

$$\frac{\partial x^{(i+1)}}{\partial x^{(i)}} = \begin{bmatrix} \mathbb{I}_a & 0 \\ * & \mathbb{I}_b \end{bmatrix} \tag{2.51}$$

It does not matter the value of the sub-matrix $*$. The analytical value of the determinant of the Jacobian is:

$$Det(J_f)(\mathbf{x}) = 1, \quad \forall \mathbf{x} \in \mathbb{R}^n \tag{2.52}$$

This is the best-case scenario for the computation of the determinant. It is constant and equal for all layers, and we know its analytical value. A diffeomorphism is called volume-preserving if the determinant of the Jacobian is one for any element of the domain of the transformation. The volume-preserving feature is not necessarily always wanted. To be able to map from one simple distribution, for example, the standard normal $\mathcal{N}(0,1)$, to a very complicated one, we want to be able to squeeze or stretch the volume of the prior distribution. One example of such flow is presented in [Albergo et al., 2021].

2.2.6. The Kullback–Leibler divergence

Once we have defined a normalizing flow, we want to construct a measure to optimize the flow over a target distribution. Let us suppose that we have two probability distribution functions p and q from which we can sample configurations. We want a measure of how q is similar to p. One fundamental assumption we make here is that the two probability distributions are defined in the same space χ. We need to define a measure of how close the two distributions are. The Kullback–Leibler (KL) divergence [MacKay and Mac Kay, 2003] is widely used in the literature as a measure of how two probability distributions differ. The KL divergence is defined as:

$$\mathrm{KL}(q||p) = \int_{\chi} D[x] q(x) \log\left(\frac{q(x)}{p(x)}\right) \tag{2.53}$$

The Kullback–Leibler divergence has the following properties:

- KL is non-negative given the Gibbs inequality, [I]
- KL=0 if and only if $q = p$ almost everywhere,

[I] The Gibbs inequality states that, for two different probability measures p and q, $\sum_i p_i \log(p_i) \geq \sum_i p_i \log(q_i)$, where p_i and q_i are samples from the probability distributions p and q.

– KL divergence is not symmetric:

$$\mathrm{KL}(q||p) \neq KL(p||q), \tag{2.54}$$

– KL is invariant under parameter transformation:

$$\mathrm{KL}(q||p) = \int_{\chi} D[x] q(x) \log\left(\frac{q(x)}{p(x)}\right) = \int_{\chi'} D[y] q(y) \log\left(\frac{q(y)}{p(y)}\right) \tag{2.55}$$

With $x \to y = f(x)$ and $\chi \to \chi'$.
It is easy to demonstrate this property for a one dimensional case:

$$\mathrm{KL}(q||p) = \int_{x_\alpha}^{x_\beta} dx\, q(x) \log\left(\frac{q(x)}{p(x)}\right) = \int_{y_\alpha}^{y_\beta} dy\, q(y) \log\left(\frac{q(y)\frac{dy}{dx}}{p(y)\frac{dy}{dx}}\right) \tag{2.56}$$

$$= \int_{y_\alpha}^{y_\beta} dy\, q(y) \log\left(\frac{q(y)}{p(y)}\right) \tag{2.57}$$

It is important to state that the KL divergence is not a divergence as defined in calculus. It is also important to point out that the KL divergence is not a "distance metric", as it does not satisfy the triangle inequality and it is not symmetric. The KL divergence between q and p $KL(q||p)$ is also called the relative entropy between the two distributions. If we start from the definition of the KL divergence and expand Equation 2.53, we can define:

$$\mathrm{KL}(q||p) = \int_{\chi} D[x] q(x) \log(q(x)) - \int_{\chi} D[x] q(x) \log(p(x)) = H(q,p) - H(q) \tag{2.58}$$

Where $H(q,p)$ is called the cross entropy of q and p and $H(q)$ is the entropy of q. The KL divergence is also widely used in information theory and other disciplines.

2.3. Normalizing flows model for lattice field theory

In Monte Carlo simulations, it is fundamental to be able to sample from the probability distribution function $p(\phi)$:

$$p(\phi) = \frac{1}{Z} e^{-S(\phi)}. \tag{2.59}$$

We can also build a probability distribution function using deep generative models as defined in Equation 2.46. Given a normalizing flow g_θ and a prior distribution q_z, we define the induced probability distribution function $q_\theta(\phi)$ as:

$$q_\theta(\phi) = q_z(g_\theta^{-1}(\phi)) \left|\frac{\mathrm{d}g_\theta}{\mathrm{d}z}\right| \tag{2.60}$$

We define the normalized importance weight as:

$$w(\phi) = \frac{p(\phi)}{q_\theta(\phi)}. \tag{2.61}$$

$w(\phi)$ is an index of how close the two different probability distribution functions are in a given point ϕ. We can say that the two probability distribution functions are equivalent if $w(\phi) = 1$ almost everywhere. Having access to the probability distribution function $p(\phi)$ is of great interest. We want to find the set of parameters θ that makes $p(\phi) \sim q_\theta(\phi)$. The Kullback–Leibler divergence, defined in Equation 2.53, is a measure of how two distribution functions differ from each other. For the target distribution $p(\phi)$ we can write the KL divergence as:

$$\mathrm{KL}(q_\theta||p) = \int \mathcal{D}[\phi]\, q_\theta(\phi)\, \ln\left(\frac{q_\theta(\phi)}{p(\phi)}\right) = \beta\left(F_q - F\right) \tag{2.62}$$

where the variational free energy F_q is:

$$\beta F_q = \mathbb{E}_{\phi\sim q_\theta}\left[S(\phi) + \ln q_\theta(\phi)\right] \tag{2.63}$$

and the physical free energy F:

$$F = -\tfrac{1}{\beta}\ln(Z). \tag{2.64}$$

We want to have an estimation of F_q as a function of θ. This estimation is important for the optimization problem (finding the zero of the KL divergence) and evaluating the thermodynamic observables (if the two probability distribution functions are equal almost everywhere $F_q = F$). To have an efficient sampling of the variational free energy as a function of the prior distribution, we can rewrite Equation 2.63 as:

$$\beta F_q = \mathbb{E}_{z\sim q_Z}\left[S(g_\theta(z)) - \ln\left|\frac{\mathrm{d}g_\theta}{\mathrm{d}z}\right|(z) + \ln q_Z(z)\right]. \tag{2.65}$$

The numerical efficiency of this evaluation is ensured by a efficient evaluation of the prior distribution q_z, the normalizing flow architecture g_θ and the Jacobian of the transformation $\mathrm{d}g_\theta/\mathrm{d}z$. Those properties are ensured, for example, by the NICE architecture presented in subsection 2.2.5 . We minimize the KL divergence with respect to the parameter θ of the probability distribution function q_θ using gradient descent. Since the target distribution and the free energy F do not depend on the parameters θ, we can still compute the gradient of KL divergence with respect to θ,i.e., $\nabla\mathrm{KL}_\theta$. This is one of the greatest advantages of this method. We do not need a tractable partition function for training. Moreover, we do not need to generate samples from the target distribution for training. The most efficient way to generate samples for training would be via MCMC simulations. This would kill the

method's performance because you can already compute the observables you are interested in once you have generated those configurations.

2.3.1. Estimator for the KL divergence

During the optimization process, we have access to the F_q and $\nabla \mathrm{KL}_\theta$. We do not have access to the absolute value of KL. This, in principle, is a problem because we want to control the training and have control over how close w is to 1. We want to define an estimator for the normalized importance weight w. We define the un-normalized importance weight:

$$\tilde{w}(\phi) = \frac{\exp(-S(\phi))}{q_\theta(\phi)} \tag{2.66}$$

and the variable $C(\phi)$

$$C(\phi) = S(\phi) + \ln q_\theta(\phi) = -\ln \tilde{w} \tag{2.67}$$

such that:

$$\beta F_q = \langle C \rangle_q \tag{2.68}$$

During training we can use variance of $C(\phi)$ to monitor the convergence. We can derive that, for the normalized importance weight w close to one, it holds:

$$\mathrm{KL}(q_\theta || p) = \tfrac{1}{2} \mathrm{Var}_q[C] + \mathcal{O}(\mathbb{E}_q[|w-1|^3]) \tag{2.69}$$

To demonstrate this property we start by noticing that:

$$\mathbb{E}_q\left[\tfrac{p}{q}\right] = \int \mathcal{D}[\phi] q(\phi) \frac{p(\phi)}{q(\phi)} = 1 \tag{2.70}$$

The Kullback–Leibler divergence can be written in terms of w as:

$$\begin{aligned} \mathrm{KL}(q|p) &= -\mathbb{E}_q\left[\ln\left(1-(w-1)\right)\right] = \mathbb{E}_q\left[\textstyle\sum_{j=0}^{\infty} \frac{(-1)^j}{j}(w-1)^j\right] \\ &= \underbrace{\mathbb{E}_q[w-1]}_{=0} + \tfrac{1}{2}\mathbb{E}_q[(w-1)^2] + \mathcal{O}(\mathbb{E}_q[|w-1|^3])\,. \end{aligned} \tag{2.71}$$

We can do the same expansion for $C = -\ln \tilde{w}$:

$$\begin{aligned} \mathbb{E}_q[C] &= -\mathbb{E}_q[\ln \tilde{w}] = -\mathbb{E}_q\left[\ln w + \ln Z\right] = -\ln Z - \mathbb{E}_q[\ln w] = -\ln Z + \mathrm{KL}(q|p) \\ &= -\ln Z + \tfrac{1}{2}\mathbb{E}_q[(w-1)^2] + \mathcal{O}(\mathbb{E}_q[|w-1|^3])\,. \end{aligned} \tag{2.72}$$

The variance of C is given by

$$\mathrm{Var}_q(C) = \mathbb{E}_q\left[\left(C - \mathbb{E}_q[C]\right)^2\right] = \mathbb{E}_q\left[\left(-\ln\tilde{w} + \mathbb{E}_q[\ln\tilde{w}]\right)^2\right] \tag{2.73}$$

$$= \mathbb{E}_q\left[\left(\underbrace{-\ln\tilde{w} + \ln Z}_{=-\ln w} + \mathcal{O}\left(\mathbb{E}_q[(w-1)^2]\right)\right)^2\right] \tag{2.74}$$

Expanding the logarithm around $Eq[w] = 1$ we obtain:

$$\mathrm{Var}_q[C] = \mathbb{E}_q[(w-1)^2] + \mathcal{O}\left(\mathbb{E}_q[|w-1|^3]\right) \tag{2.75}$$

We have easy access to $\mathrm{Var}_q[C]$ during training. The equation in Equation 2.75 provides a great advantage for the simulations since it gives excellent control over the optimization procedure. If, for example, the variance during training does not decrease, it probably means we are stuck in a local minimum. If this happens, we can restart the simulation from a new set of parameters θ. This control is rarely possible for lots of machine optimization problems. All of this is possible also because we know the analytical form of the target distribution. This is usually unknown, e.g., for image generation problems. We apply all of this machinery to the φ^4 model described in subsection 2.1.1. We minimize the variational

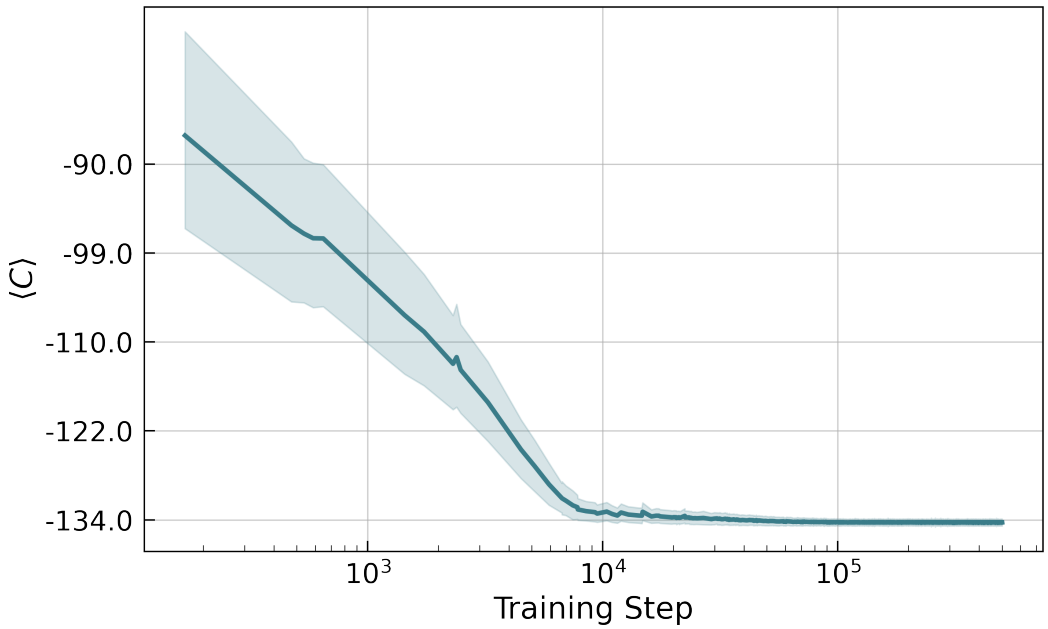


Figure 2.4: *The expectation value $\mathbb{E}_q(C)$ of C as a function of the training step for one run on the φ^4 model with $\kappa = 0.3$ and $\lambda = 0.022$ for a 16×8 lattice.*

free energy F_q during optimization of the parameters θ. F_q is directly proportional to $\langle C\rangle_q$. In Figure 2.4 $\mathbb{E}_q(C)$ as a function of the training step is presented.Even though a plateau may seem to be reached around 10^4 steps, we need a deeper analysis to attest to the convergence of the training. To do so we need to look at $\mathrm{Var}_q(C)$.

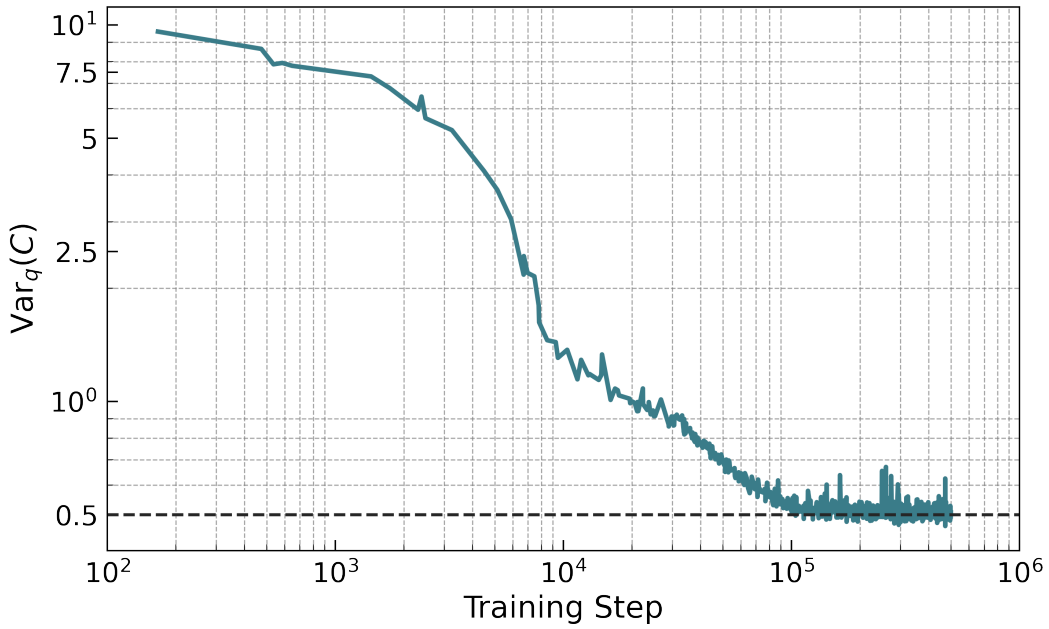


Figure 2.5: *The variance $Var_q(C)$ as a function of the training step for one simulation run on the φ^4 model with $\kappa = 0.3$ and $\lambda = 0.022$ for a 16×8 lattice.*

In Figure 2.5 the convergence of$\mathrm{Var}_q(C)$ as a function of the training step is plotted. We can see from the plot that the variance of C decreases during training but does not converge to zero. This implies the target distribution is not exactly approximated. The error we make by imperfect training is called bias and can be estimated. We refer to subsection 2.3.3 for a complete analysis of the error.

2.3.2. Thermodynamic observable evaluations

Once the training procedure converges, we can compute thermodynamic observables with generative models. We can write the partition function as:

$$Z = \int \mathcal{D}[\phi]\, q_\theta(\phi)\, \tilde{w}(\phi) \tag{2.76}$$

Since we can sample from q_θ, we can evaluate the estimator for the partition function as:

$$\hat{Z} = \frac{1}{N} \sum_{i=1}^{N} \tilde{w}(\phi_i) \qquad \text{with} \qquad \phi_i \sim q_\theta \tag{2.77}$$

From $\hat{Z}$ we can compute the estimator for the Free energy as:

$$\hat{F} = -T \ln \hat{Z}\,. \tag{2.78}$$

At this point, we want to stress the importance of equation Equation 2.77. Once the training is complete, we can sample the configuration ϕ_i in an embarrassingly parallel fashion. One

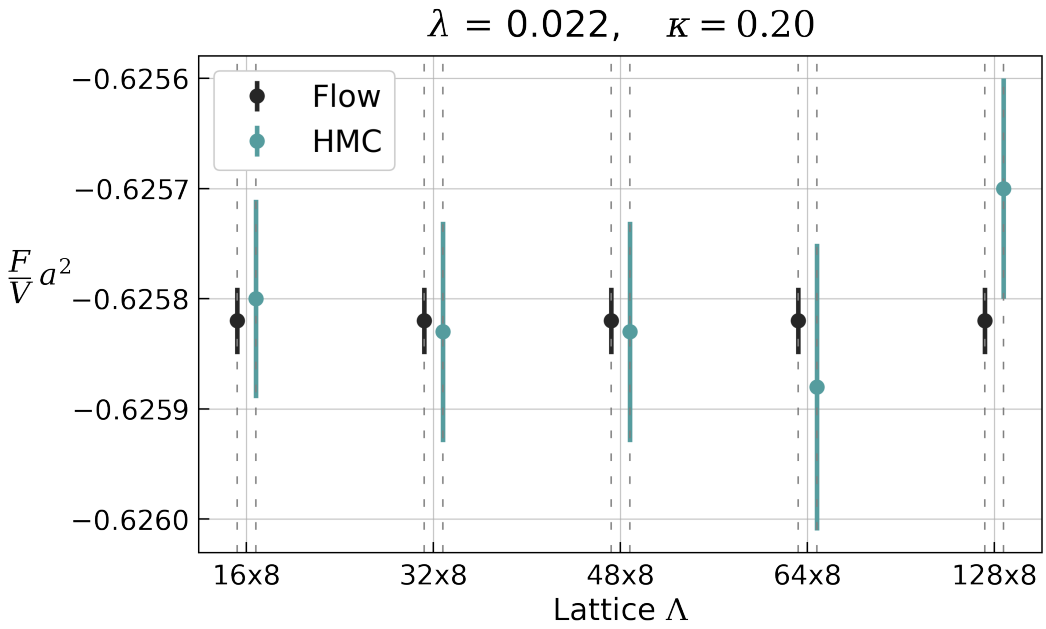


Figure 2.6: *The comparison between the estimation of the free energy density with $\kappa = 0.2$ and $\lambda = 0.022$ (i.e. in the phase without spontaneous magnetization) for the flow based algorithm and HMC algorithm for different volumes.*

can send the parameters θ of the flow and its functional form to different GPU (CPU) and do the sampling independently. Another great advantage of this method is the absence of autocorrelation time. All the configurations are sampled independently and do not need to be sampled in a chain as in Markov chain Monte Carlo.

The evaluation of thermodynamic observables via Monte Carlo simulations is more challenging when we need to cross a phase transition point during the integration in the parameters space. In Figure 2.6 and Figure 2.7, we see the comparison of the estimation of the free energy between HMC and the flow-based algorithm in two interesting points, one before the phase transition and one after.

In Figure 2.6 we estimate the free energy at $\lambda = 0.022$ and $\kappa = 0.2$. These simulations are performed at a point in the parameter space that does not require crossing the phase transition point of the theory. Both the HMC method and the flow-based method agree. Moreover, the errors have the same order of magnitude for comparable runtime. In the next paragraph, a detailed description of the numerical analysis is given.

For Figure 2.7 the situation is different. In this plot the comparison of the two method is done at $\lambda = 0.022$ and $\kappa = 0.3$. The error in the estimation of the free energy is bigger in the HMC analysis. The crossing of the phase transition causes these larger errors.

Numerical details For the HMC algorithm, we use a Hybrid Markov chain Monte Carlo Method with overrelaxation [Adler, 1981]. The overrelaxation procedure flips the sign of the field ϕ, i.e., $\phi \to -\phi$ every 10 MCMC steps. Such an overrelaxation algorithm is useful

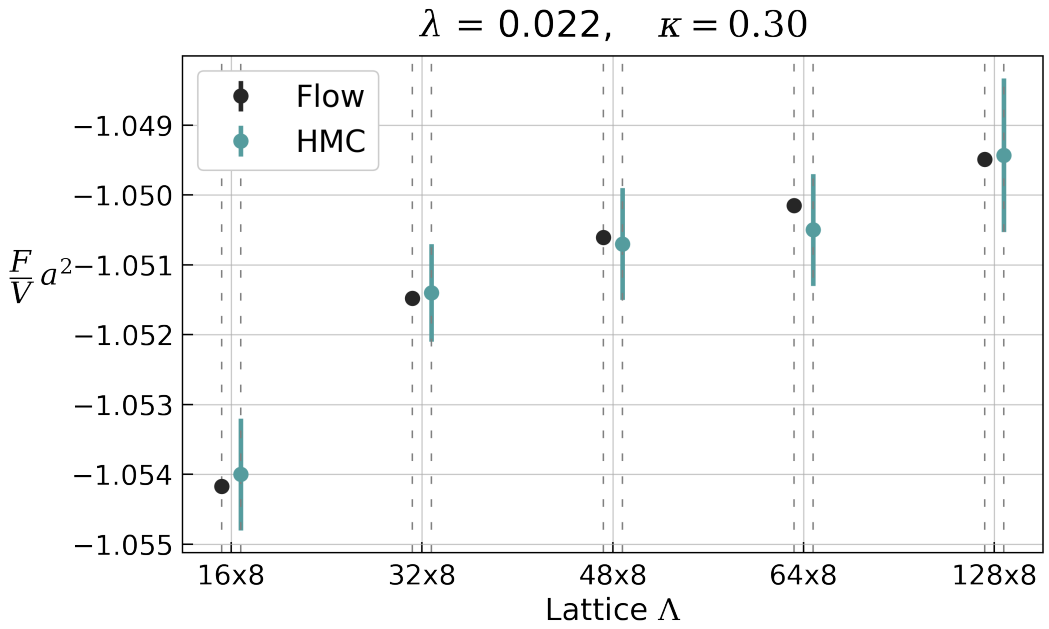


Figure 2.7: *The comparison between the estimation of the free energy density with $\kappa = 0.3$ and $\lambda = 0.022$ (i.e. in the phase with spontaneous magnetization) for the flow based algorithm and HMC algorithm for different volumes in the phase with sponta- neous magnetization.*

because MCMC has problems overcoming energy barriers between different (local or global) minima of the action. In the broken phase of our ϕ^4 theory, there are (at least classically) two degenerate minima related by $\mathbb{Z}_2$ symmetry. Without overrelaxation, the HMC would tend to be stuck in the neighborhood of one of them. With overrelaxation, it is ensured that the regions around both minima are explored. Each Markov chain consists of $5 * 10^3$ thermalization steps and $4 * 10^5$ estimation steps. For the estimation of the free energy we have used a integration step size $\delta_\kappa = 0.05$ for all the values of κ except for $\kappa \in [0.2, 0.3]$ where a refinement is needed as explained in subsection 2.1.3. In this interval the integration step is fixed to $\delta_\kappa = 0.01$. Summing up all the integration steps, we generated 14 Markov chains with a total of $5.6 * 10^6$ configurations used for the estimations. In our non-optimized implementation, the generation of the samples took about 25 hours of time.

We use a normalizing flow g_θ build with the NICE architecture with 6 coupling layers for the flow-based algorithm. We use the standard normal $\mathcal{N}(0, 1)$ for the prior distribution. To make the flow invariant under $\mathbb{Z}_2$ transformations it is enough to make the function g_θ equivariant, i.e. $g_\theta(-x) = -g_\theta(x)$. Odd functions are equivariant with respect to the $\mathbb{Z}_2$ symmetry. This is evident if we look at the definition of the probability distribution function q_θ in Equation 2.60. One way to build g_θ as an odd function is to define all the building blocks as odd functions. This is easily achievable by constructing the neural networks m as odd functions. Every coupling layer has the function m in Equation 2.51 defined by a fully connected neural network (defined in Equation 2.44) with 5 layers with bias set to zero and *Tanh* non-linearities, to ensure the $\mathbb{Z}_2$ invariance.

The hidden layers of m are built with 10^3 neurons. We always set the value of a and b in Equation 2.51 to be equal. Consecutive coupling layers have an alternating checkerboard partition to update all the lattice sites. The flow is trained for 10^6 steps with mini-batches of $8 * 10^3$. The ReduceLROnPlateau learning rate scheduler of PyTorch is used, with an initial learning rate of $0.5 * 10^{-3}$ and minimum fixed to 10^{-7}. To have a comparison with MCMC results, after training, we sample $5.6 * 10^6$ configurations for the estimation of the variational free energy. As we have already mentioned, normalizing flows allow for very efficient sampling. We can generate all the $5.6 * 10^6$ configurations in less than a minute. This is done with a significant up-front training cost that does not depend on the number of samples generated for evaluating the observables. In our non-optimized implementation, the training took 20 hours to converge for the largest volume. Even though we have a comparable runtime both for HMC and the flow-based algorithm, it does not mean that the comparison is entirely fair. Since we did not optimize both codes deeply and believe there is a significant margin to improve, those numbers should be taken as an indication. This work aims to present the method to the community and not provide a state-of-the-art method for complex LQFT calculations. This is the reason why we did not invest too much work in the optimization of the codes.

2.3.3. Bias due to imperfect training

In this section, we analyze the scaling of the bias due to imperfect training of the expectation value of the free energy with respect to the probability distribution q_θ. The discussion is based on [Nowozin, 2018]. In Figure 2.5 it is evident that the variance of C decreases during training, but it does not converge to zero. This implies the target distribution is not approximated exactly. We can estimate the error we make by imperfect training. To estimate this error, we use what is called in the literature the *delta method*. We can find an exhaustive study of the subject in [Bickel and Doksum, 1977].
Let $\hat{X}_N = \frac{1}{N}\sum_{i=1}^{N} X_i$ be the sample mean of X_i and h be a real-valued function with uniformly bounded derivatives. We can write $\mathbb{E}\left[h(\hat{X}_N)\right]$ as:

$$\mathbb{E}\left[h(\hat{X}_N)\right] = c_0 + \frac{c_1}{N} + \mathcal{O}\left(\frac{1}{N^2}\right), \tag{2.79}$$

where the random variables X_i are independent and identically distributed with $\mathbb{E}\left[X_i^{2k+2}\right] < \infty$ for $k \in \{0, 1\}$ and

$$c_0 = h(\mu), \qquad c_1 = h''(\mu)\frac{\sigma^2}{2}, \tag{2.80}$$

with $\sigma^2 = \mathbb{E}\left[(X - \mathbb{E}X)^2\right]$ and $\mu = \mathbb{E}\left[X\right]$. We can apply this method to estimate the bias of $-\beta\hat{F} = \ln\hat{Z}$ defined in Equation 2.78.

$$\mathbb{B}[-\beta\hat{F}] = \mathbb{E}_q[\ln\hat{Z}] - \ln Z\,. \tag{2.81}$$

First we remind the reader that:

$$\mathbb{E}_q\left[\tilde{w}\right] = Z\,,$$

Using the delta method for moments, with $h(x) = \ln(x)$ and second derivative $h''(x) = -\frac{1}{x^2}$, we derive that:

$$\mathbb{E}_q[\ln\hat{Z}] = \ln Z - \frac{1}{2NZ^2}\mathbb{E}_q\left[(\tilde{w} - \mathbb{E}_q[\tilde{w}])^2\right] + \mathcal{O}\left(N^{-2}\right)\,, \tag{2.82}$$

from which we can derive:

$$\mathbb{B}[-\beta\hat{F}] = -\frac{1}{2\,N}\frac{\mathbb{E}_q\left[(\tilde{w} - \mathbb{E}_q[\tilde{w}])^2\right]}{\mathbb{E}_q[\tilde{w}]^2} + \mathcal{O}(N^{-2})\,. \tag{2.83}$$

In this derivation, we have ignored the requirements of the function h for the application of the *delta method*. h needs to be bounded with uniformly bounded derivatives. This is not guaranteed for $h(x) = \ln(x)$. For realistic LFT, we can avoid this subtlety by putting the theory in a box potential. This would make $\hat{Z}$ and all its derivatives bounded from below. The effect of this solution does not have a significant impact on practical numerical experiments. This is because only very high energy states are affected by the potential and thus are extremely unlikely to be sampled.

We can also compute the variance of the estimator with the same techniques. The variance of $\hat{F}$ is given by

$$\mathrm{Var}\left[-\beta\hat{F}\right] = \frac{1}{N}\frac{\mathbb{E}_q\left(\tilde{w} - \mathbb{E}_q\left[\tilde{w}\right]\right)^2}{(\mathbb{E}_q[\tilde{w}])^2} + \mathcal{O}\left(\frac{1}{N^2}\right)\,, \tag{2.84}$$

We refer to the supplemental material of [Nicoli et al., 2021] for a full demonstration of this equation. The standard deviation of $\hat{F}$ is of order $\mathcal{O}(1/\sqrt{N})$. We can thus neglect higher order $\mathcal{O}(N^{-1})$ correction due to the bias given by imperfect training in the large N limit. In our numerical experiments, we checked that the errors show the theoretic Monte Carlo scaling of $N^{-\frac{1}{2}}$. The error estimators for general observables involving the partition function can be derived in a more general fashion as showed in [Nicoli et al., 2020].

Chapter 3

Quantum Computing

> *"I'm not happy with all the analyses that go with just classical theory, because Nature isn't classical, dammit, and if you want to make a simulation of Nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem!"*
>
> – Richard Feynman, *1981*

Quantum computing has made many important advancements in recent years due to significant improvements in algorithms and hardware technologies. One of the first steps in quantum computing goes back in history to Feynman when he said the sentence at the top of the page. Since that year, much progress has been made in the quantum computing scientific community. There has been much advancement both in the algorithms and the hardware technology. This chapter will focus on algorithm developments for quantum technologies.

We are in the Noisy Intermediate-Scale Quantum (NISQ) technology era [Preskill, 2018]. This means that soon we will have quantum computers that have the potential to outperform today's classical digital computers, but the noises of these quantum devices constitute a significant obstacle to the usage of those machines. The goal of quantum computing is to reach *quantum advantage*. With quantum advantage, we mean the possibility to perform tasks with controlled quantum systems beyond what can be achieved with ordinary digital computers. To reach a quantum advantage, we have to complete a simulation with a useful application in the real world (exponentially) faster with a quantum computer than any classical digital computer in the world. In 2019 [Arute et al., 2019] published a work demonstrating the quantum advantage era's entrance. However, they have only been able to perform a specially designed problem known to be very difficult for classical computers and suitable for quantum computers. It has not been performed a calculation that can demonstrate a quantum advantage for a real life problem. Many algorithms that are theoretically demonstrated to provide a speed-up with respect to the classical state-of-the-art best algorithm known to solve the same task have been developed. For example, the Shor algorithm [Shor, 1997] provides an exponential speed-up in the solution of the

integer-factorization problem. The algorithm presented in [Harrow et al., 2009] provides a matrix-vector-multiplication exponential speed. The reader can look up the most known algorithms in [Jordan; Montanaro, 2016]. Unfortunately, algorithms that provide a super-polynomial speed-up like, for example, require too many resources to be executed on nowadays quantum computers. The devices we have now have neither enough number of qubits neither qubits with high enough quality to perform those tasks. Nevertheless, it is still possible to do interesting physics with what we now have.

One class of algorithms that can be performed in the NISQ era is the variational hybrid quantum-classical algorithms [McClean et al., 2016; O'Malley et al., 2016; Kandala et al., 2017; Shen et al., 2017; Colless et al., 2018; Dumitrescu et al., 2018; Hempel et al., 2018; Ganzhorn et al., 2019; Kokail et al., 2019; Hartung and Jansen, 2019; Jansen and Hartung, 2020]. Those algorithms are performed on both classical and quantum computers. The idea is to evaluate computationally demanding cost functions on a quantum computer and perform a classical optimization on a classical device. Quantum computer devices are affected by certain types of errors, which can only be partially mitigated using error correction procedures [Kandala et al., 2017; Li and Benjamin, 2017; Temme et al., 2017; McClean et al., 2017; Bonet-Monroig et al., 2018; Endo et al., 2018; McArdle et al., 2019; Endo et al., 2019; Kandala et al., 2019; McClean et al., 2020; Otten and Gray, 2019a,b; Sagastizabal et al., 2019; Urbanek et al., 2019; Crawford et al., 2019; Chungheon et al., 2019; Córcoles et al., 2015; Sheldon et al., 2016; Tannu and Qureshi, 2019; Yeter-Aydeniz et al., 2019, 2020; Funcke et al., 2020]. Usually, the variational cost function depends on a certain ansatz that needs to be engineered specifically for the problem. The goal of this chapter is to develop algorithms and routines to improve the performances of quantum algorithms. This is done by developing an algorithm that mitigates bit-flip errors and a dimensional expressivity analysis of variational quantum circuits. This chapter is organized as follow: in section 3.1 we present the basis of quantum computing and variational quantum simulations, in section 3.2 we develop a mitigation routine that corrects the bit-flip errors in quantum computing [Funcke et al., 2020] and in section 3.3 we develop a dimensional expressivity analysis of quantum circuits based on the dimension of the manifold generated by a quantum circuit and its symmetries [Funcke et al., 2021].

3.1. Variational Quantum Simulations

All computing systems rely on a fundamental ability to store and manipulate information. Current computers manipulate individual bits, which store information as binary 0 and 1 states. Quantum computers leverage quantum mechanical phenomena to manipulate information. To do this, they rely on quantum bits or qubits. A qubit is a two-state quantum-mechanical system (for example, it could be a spin 1/2 atom or the polarization of a single photon). Quantum mechanics allows the qubit to be in a coherent superposition of both states simultaneously, a property that is fundamental to quantum mechanics and quantum

computing. An important distinguishing feature between qubits and classical bits is that multiple qubits can exhibit quantum entanglement. Quantum entanglement is a nonlocal property of two or more qubits that allows a set of qubits to express higher correlation than is possible in classical systems. Quantum algorithms try to use those properties to perform efficient calculations. This section is organized as follows: in subsection 3.1.1 we briefly introduce the concepts of quantum bits and quantum gates, in subsection 3.1.2 we briesfly explain the simulation tools used in this chapter, and in subsection 3.1.3 we show an example of a variational quantum simulation.

3.1.1. Quantum gates, quantum states and quantum measurements

Classical computing is based on classical bits. Classical bits can be only in a completely defined state: 0 or 1. This is not always true for Quantum bits. A qubit can be in the state $|0\rangle$, in the state $|1\rangle$ or a superposition of those. We can represent those states using two orthogonal vectors:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \; |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{3.1}$$

Given the basis states, we can construct a generic 1 qubit state as: $|\Psi\rangle = a\,|0\rangle + b\,|1\rangle$, where $a, b \in \mathbb{C}$ are complex numbers and $|a|^2 + |b|^2 = 1$. $|\Psi\rangle$ is called the wave function of the state.

Now that we have a quantum state, we want to extract information about this state. This is done via the measurement procedure. What we can extract is the probability of the qubit to be in the state $|0\rangle$ or $|1\rangle$, e.g., $P(|0\rangle)$:

$$P(|0\rangle) = |\,\langle 0|\Psi\rangle\,|^2 \tag{3.2}$$

Once we have done a measurement, we destroy the quantum nature of the state. If we have a qubit in the state $|\Psi\rangle$ and, after the measurement, we get the state $|0\rangle$; the qubit wave function is collapsed in the state $|0\rangle$. If we measure again, there is a 100% chance of finding the qubit in the state $|0\rangle$.

We can modify the state of a qubit acting on it with quantum gates. One qubit gate is represented by $2 \otimes 2$ unitary matrices. One representation based on the group $SU(2)$ of those matrices are the Pauli matrices:

$$\sigma^x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \sigma^y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \sigma^z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{3.3}$$

The matrix-vector multiplication represents the action of a gate to a qubit. The most general

single-qubit quantum gate is the $U_3(\theta, \phi, \lambda)$;

$$U_3(\theta, \phi, \lambda) = \begin{bmatrix} \cos(\theta/2) & -e^{i\lambda}\sin(\theta/2) \\ e^{i\lambda}\sin(\theta/2) & e^{i\lambda+i\phi}\cos(\theta/2) \end{bmatrix} \tag{3.4}$$

Widly used one-qubit quantum gates are the rotation gates $R_i(\theta) = \exp(-i\theta\sigma^i/2)$, where $j = \{x, y, z\}$. Those notions generalize to more than one qubit. We can use the tensor product to describe the collective state of any number of qubits. If we have two separated qubits $|\alpha\rangle$ and $|\beta\rangle$, we can describe their collective state $|\Psi\rangle$ using the tensor product:

$$|\Psi\rangle = |\alpha\rangle \otimes |\beta\rangle \tag{3.5}$$

We need 2^n complex number to store a generic n qubit state. In the literature, the concept of quantum circuit is frequently used. We define a quantum circuit as a series of quantum gate operations on quantum bits in a given order. A quantum circuit $C(\theta_1, \theta_2)$ acting on one qubit can be defined as, for example:

$$C(\theta_1, \theta_2) = R_x(\theta_1)R_y(\theta_2). \tag{3.6}$$

We can use the same rules also for the quantum circuit. For example, we can write:

$$[U_3(\theta_1, \phi_1, \lambda_1)\,|\alpha\rangle] \otimes [U_3(\theta_2, \phi_2, \lambda_2)\,|\beta\rangle] = U_3(\theta_1, \phi_1, \lambda_1) \otimes U_3(\theta_2, \phi_2, \lambda_2)\,|\alpha\rangle \otimes |\beta\rangle \tag{3.7}$$

meaning that $U_3(\theta_1, \phi_1, \lambda_1)$ acts on qubit $|\alpha\rangle$ and $U_3(\theta_2, \phi_2, \lambda_2)$ on qubit $|\beta\rangle$. A widely used two-qubit gate is the CNOT gate (also called the CX gate). The CNOT_{ij} gate acts on two qubits i and j. i is called the control qubit and j is called the target qubit. The CNOT_{ij} gate acts as follow:

$$\text{CNOT}_{ij} \begin{cases} \text{if } |i\rangle = |0\rangle \;:\; |j\rangle \to I\,|j\rangle \\ \text{if } |i\rangle = |1\rangle \;:\; |j\rangle \to \sigma^x\,|j\rangle \end{cases} \tag{3.8}$$

One reason for its popularity is that it is "a native gate" in the most common quantum computers based on superconductive qubits. A native gate is a quantum gate operation that is realized on the hardware. We define a complete set of quantum gates if we can derive all the possible unitary transformations of a n qubit gates from the composition of the native set of gates. We can demonstrate that CNOTs and unitary single-qubit operations form a universal set of quantum computing [Cirac and Zoller, 1995]. Once we have a universal set of gates, we can generate any quantum state. It is not enough to generate a state to get information. We must *measure* this state. When we defined a one-qubit state, we stated that a qubit could be in the state $|0\rangle$ or $|1\rangle$. Those states are the eigenvectors of the σ^z operator. In this case, we say that we have used the Z computational basis. This is not the only computational basis we can have. For example we can define $|+\rangle$ and $|-\rangle$ as the

eigenvectors of the σ^x operator:

$$|+\rangle = \frac{|0\rangle + |1\rangle}{2},\ |-\rangle = \frac{|0\rangle - |1\rangle}{2}. \tag{3.9}$$

Given a computation basis, we can always find a unitary operator that changes the basis of the state. In current quantum computers, it is possible only to measure in one computational basis (usually the Z base). Let us suppose we want to measure the expectation value of an operator that acts on two qubits $\mathcal{H}$:

$$\mathcal{H} = \sigma_1^x \sigma_2^x + \sigma_1^z \sigma_2^z \tag{3.10}$$

where the multiplication between σ matrices is to be understood as a direct product. The expectation value is given by:

$$\langle\psi|\,\mathcal{H}\,|\psi\rangle = \langle\psi|\,\sigma_1^x\sigma_2^x + \sigma_1^z\sigma_2^z\,|\psi\rangle = \langle\psi|\,\sigma_1^x\sigma_2^x\,|\psi\rangle + \langle\psi|\,\sigma_1^z\sigma_2^z\,|\psi\rangle \tag{3.11}$$

A quantum computer can only measure in one given basis (e.g., Z). This constraint is a limitation of the hardware. The term $\langle\psi|\,\sigma_1^z\sigma_2^z\,|\psi\rangle$ can be measured directly. Conversly, we can not measure directly $\langle\psi|\,\sigma_1^x\sigma_2^x\,|\psi\rangle$, therefore we have to add a post-rotatation to the quantum circuit. We define the operator H, called the Hadamard gate, as:

$$H = \frac{\sigma_x + \sigma_z}{\sqrt{2}} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{3.12}$$

It is trivial to derive that $\sigma^z = H\sigma^x H$. Using this property, we can write rewrite $\langle\psi|\,\sigma_1^x\sigma_2^x\,|\psi\rangle$ as:

$$\langle\psi|\,\sigma_1^x\sigma_2^x\,|\psi\rangle = \langle\psi|\,H^1H^2\sigma_1^z\sigma_2^z H^2H^1\,|\psi\rangle = \langle\psi'|\,\sigma_1^z\sigma_2^z\,|\psi'\rangle \tag{3.13}$$

where we have defined $|\psi'\rangle = H^2H^1\,|\psi\rangle$, where H^2H^1 is intended as a direct product. This relation allows for the evaluation of the expectation value of $\sigma_1^x\sigma_2^x$ with respect to $|\psi\rangle$. On a quantum mechanical level, the expectation value of $\sigma_1^x\sigma_2^x$ on the state $|\psi'\rangle$ and the expectation value of the operator $H^1H^2\sigma_1^z\sigma_2^z H^2H^1$ on the state $|\psi\rangle$ are equivalent. The latter can be done on the hardware and allows for the computation of the expectation value of different operators that are not expressed in the Z basis. This procedure can also be applied to the Y basis using as post-rotation the operator $(\sigma_y + \sigma_z)/\sqrt{2}$.

3.1.2. Simulation tools

This chapter presents numerical results obtained by using three different kinds of simulations, demonstrating the applicability and validity of the methods, algorithms, and routines we have developed. We briefly explain them in order of appearance. The easiest

and most straightforward way to perform numerical simulations of quantum computing is to use a *wave function simulator* A wave function simulator is a software that performs classical simulations of quantum circuits. It constructs exactly the wave function of a system and allows for the efficient application of quantum gates to the wave function. We can also compute any operator's exact quantum mechanical expectation value over any wave function using a wave function simulator. We need to construct the wave function as a vector and the operator as a matrix and perform the scalar products. We often use a wave function simulator to perform the first tests of the algorithm we are developing.

In this thesis, we use a self-implemented wave function simulator. In this work, we have used the hardware of IBMQ [ibm] for the simulations on real quantum devices. Since real quantum computers are noisy and expensive, it is advisable to perform classical simulations of the noisy devices before performing quantum simulations on quantum computers. We used the IBMQ software development toolkit (SDK) Qiskit [Qiskit Aer API documentation and source code] for both noisy simulations and simulations on the quantum hardware. Qiskit is a library written in the `Python` programming language. Using Qiskit SDK we can write a quantum computing program (most of the quantum gates are implemented in the library) and run our simulations on three different kinds of backends: a simple wave function simulator, a wave function simulator combined with a quantum noise model that mimics the noise of the quantum device and the real quantum device.

An exact description of the quantum noise of any quantum device is non-present at the moment. Nevertheless, noise modeling of quantum devices is quite reliable. Performing classical noisy simulations of the quantum hardware can give us good hints of the reliability of one simulation. One great advantage of Qiskit is that the same code can run on three different backends. We need to change one line of code in which we specify the back-end to pass between the three different ways of simulation.

3.1.3. Variational quantum eigensolver simulations

It is often interesting to find the lowest eigenvalue and eigenvector of a given operator (matrix). For example, the eigenvector associated with the lowest eigenvalue of the Hamiltonian of a physical system is the ground-state-wave function. We can extract a lot of physical information from the ground state of a given Hamiltonian. Due to limitations in NISQ devices, it is not yet possible to implement the *"quantum phase estimation algorithm"*(QPE) [Nielsen and Chuang, 2010] to find the lowest eigenvalue of a given Hermitian operator. The QPE algorithm can estimate an operator's lowest eigenvalue with an exponential speed-up with respect to the best classical algorithm. In NISQ devices, it is unlikely to implement this algorithm due to the too short circuit depth possible in the devices and the fact that this algorithm requires too many qubits to be implemented. We are still able to do interesting computations in NISQ devices. In 2014 Peruzzo et al. presented the idea of variational quantum eigensolver [Peruzzo et al., 2014] (VQE). This algorithm is based on attempting to

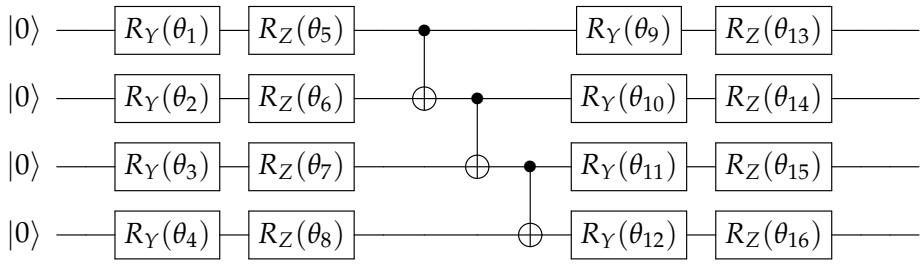


Figure 3.1: *We show a diagrammatic representation of* `EfficientSU2` *circuit for four qubits and one repetition. This graphic representation of a quantum circuit is meant to be read from left to right. On the left, the symbols* $|0\rangle$ *stands for the qubits initialized in a product state of all* $|0\rangle$*. A straight black line is associated with every qubit. A rectangular box drawn on one line represents a single qubit gate operation. The gate applied to the qubit is written inside the box. A rectangular box drawn on more than one line represents the operation of a multi-qubit gate. A black circle and a circled-square symbol connected via a vertical black line represent a CNOT gate. The black circle is drawn on the qubit line of the control qubit. The circled-square symbol is drawn on the qubit line of the target qubit.*

extract as much information as possible from the quantum computer with minimal computation effort. This is done with the help of a classical device. An algorithm that exploits both classical and quantum computations is usually called a hybrid-classical-quantum algorithm. There are two main building blocs in a VQE simulation: a variational ansatz and a cost function. The goal of a VQE is to minimize the cost function given a particular variational ansatz. The ansatz is defined by a unitary operator $U(\vec{\theta})$ that depends on a certain number of parameters $\vec{\theta}$ and an initial quantum state $|\psi_0\rangle$. The variational ansatz is a quantum circuit that produces the quantum state $\left|\psi(\vec{\theta})\right\rangle$ as:

$$\left|\psi(\vec{\theta})\right\rangle = U(\vec{\theta})\,|\psi_0\rangle \tag{3.14}$$

The form of the variational ansatz is arbitrary and up to us. It is advisable to impose the physical knowledge, e.g., symmetries, into the circuit when we perform a VQE simulation. In the IBMQ software development toolkit (SDK) Qiskit [Qiskit Aer API documentation and source code] a standard variational ansatz is proposed. The ansatz is called in the SDK Qiskit `EfficientSU2` variational ansatz. The documentation [Qiskit Aer API documentation and source code] proposes it as "a heuristic pattern that we can use to prepare trial wave functions for variational quantum algorithms or classification circuit for machine learning". Qiskit `EfficientSU2` variational ansatz has a layered structure. The EfficientSU2 circuit consists of layers of single-qubit operations and *CNOT* entanglements. For example, the circuit `EfficientSU2`(N_{qubit}=4, N_{layers}=1) is presented in Figure 3.1. The cost function, in our case, is given by the expectation value of a given operator. We suppose that the operator is a certain hamiltonian $\mathcal{H}$. $\mathcal{H}$ is a hermitian operator. This implies that the eigenvalues are

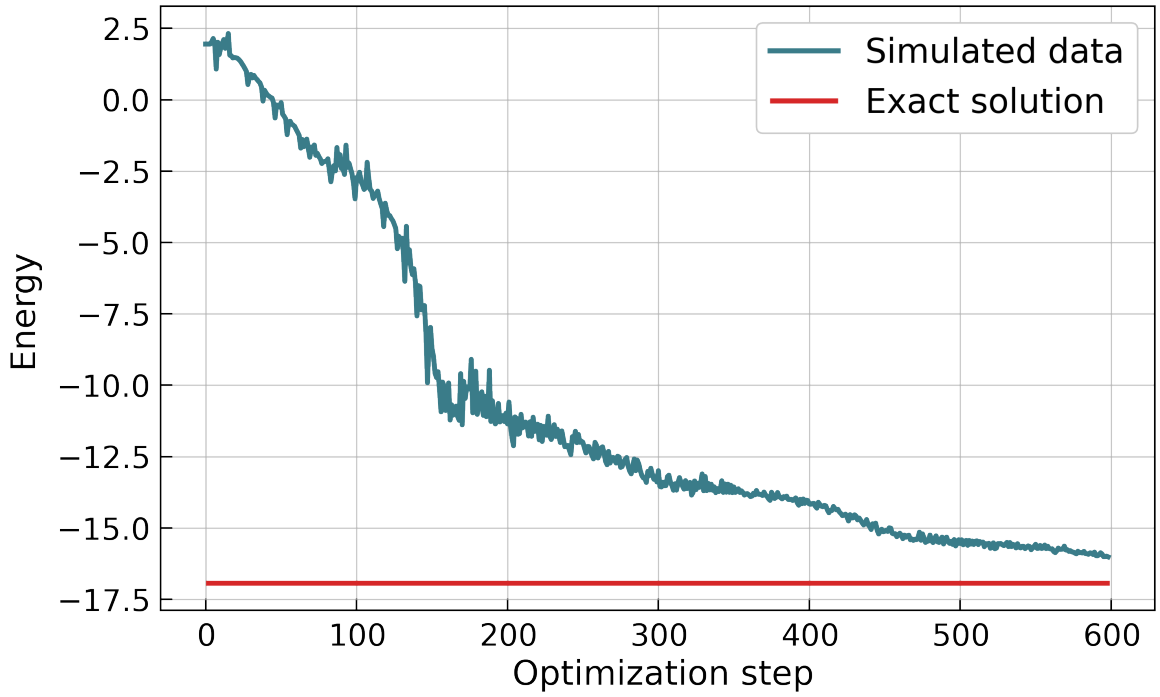


Figure 3.2: *Expectation value of the energy as a function of the optimization step for a variational quantum simulation with four qubits and 16 free parameters*

real. The cost function we want to minimize is:

$$\mathcal{C}(\vec{\theta}) = \left\langle \psi(\vec{\theta}) \right| \mathcal{H} \left| \psi(\vec{\theta}) \right\rangle \tag{3.15}$$

For a concrete and pedagogical example, we consider the Heisenberg Hamiltonian for N spins:

$$\mathcal{H} = \sum_{i=1}^{N} J_x \sigma_i^x \sigma_{i+1}^x + J_y \sigma_i^y \sigma_{i+1}^y + J_z \sigma_i^z \sigma_{i+1}^z + h_x \sigma_i^x + h_y \sigma_i^y + h_z \sigma_i^z \tag{3.16}$$

where $J = \{J_x, J_y, J_z\}$, $h = \{h_x, h_y, h_z\}$ are real coefficients. The coefficients J tune the nearest-neighbor interaction between the spins, and the coefficients h can be seen as the magnitude of an external magnetic field applied to the system. With periodic boundary condition (i.e. $\sigma_{N+1} = \sigma_1$), we can formally write the minimization problem we want to solve as:

$$\lambda_{\min} = \min_{\vec{\theta}} \left\{ \mathcal{C}(\vec{\theta}) \right\} \tag{3.17}$$

We now show a concrete numerical example that helps the reader to understand the numerical problem. We want to compute the ground state of the Heisenberg Hamiltonian. We choose the parameter $J = \{1, 1, -1\}$, $h = \{1, 1.5, 3\}$. This set of parameters does not have a particular physical meaning. They are chosen with the only intent to carefully avoid

the phase transition points in the phase diagram of the model [I] .

In Figure 3.2 we show the expectation value of the energy as a function of the optimization step during the VQE minimization. The minimization of the cost function is performed with the Nelder-Mead optimization algorithm. In red, we plot the exact value of the energy of the ground state computed via exact diagonalization. The simulations have been performed classically on a wave function simulator. The quantum circuit used as a variational ansatz is the one presented in Figure 3.1. In the variational quantum circuit, there are sixteen free parameters. These simulations can only be performed for a very limited number of qubits on a wave function simulator. This is due to the exponential growth of the simulation's computational cost with the number of qubits. Suppose we would have access to a powerful quantum computer. In that case, we could evaluate the energy on the quantum computer with a polynomial computational cost and perform the optimization of the variational parameters classically. This optimization problem is tough since it is a non-convex optimization problem [Nakanishi et al., 2020]. We suppose that the dependence on the quantum circuit's variational parameters is by the exponential of some operator. This means that we suppose that we can write every quantum gate $Q(\theta)$ as:

$$Q(\theta) = e^{i\,\theta q} \tag{3.18}$$

where the operator q has the following property: $[q]^2 = 1$. The operator q can be a generic multi-qubit operation. With this mild assumption, we can write the cost function analytically as the sum of an exponentially large number of terms that depend on the sine and cosine of all the different θ in the parametric circuit as demonstrated in [Nakanishi et al., 2020].

This class of optimization problems is very sensible to local minima. It is crucial to choose a suitable algorithm for the classical optimization routine and a good ansatz for the variational circuit. The simulations become increasingly costly on the classical level also when the number of parameters increases. Since we are simulating a physical model, we can use the symmetries of the system to improve the performance of the simulation. The Hamiltonian of the system we are studying is translationally invariant. This means that a transformation that relabels qubit $i \to i+1$ commute with the Hamiltonian. The generator of the translation transformation is the momentum operator.

We also know that the ground state of the system should have the same symmetry as the Hamiltonian. If we construct a quantum circuit that is translationally invariant, we can improve our simulations. It is challenging to build a translationally invariant quantum circuit with an entangling layer built of *CNOT* rotations. These difficulties are based on the fact that the *CNOT* entangling layer in the circuit `EfficientSU2` is not translationally invariant. To easily construct a variational quantum circuit that is translational invariant, we

[I] For example the values $J = \{0,0,1\}$, $h = \{1,0,0\}$ correspond to the phase transition point of the transversal Ising model. $J = \{1,1,1\}$, $h = \{1,1,1\}$ is also a phase transition point.

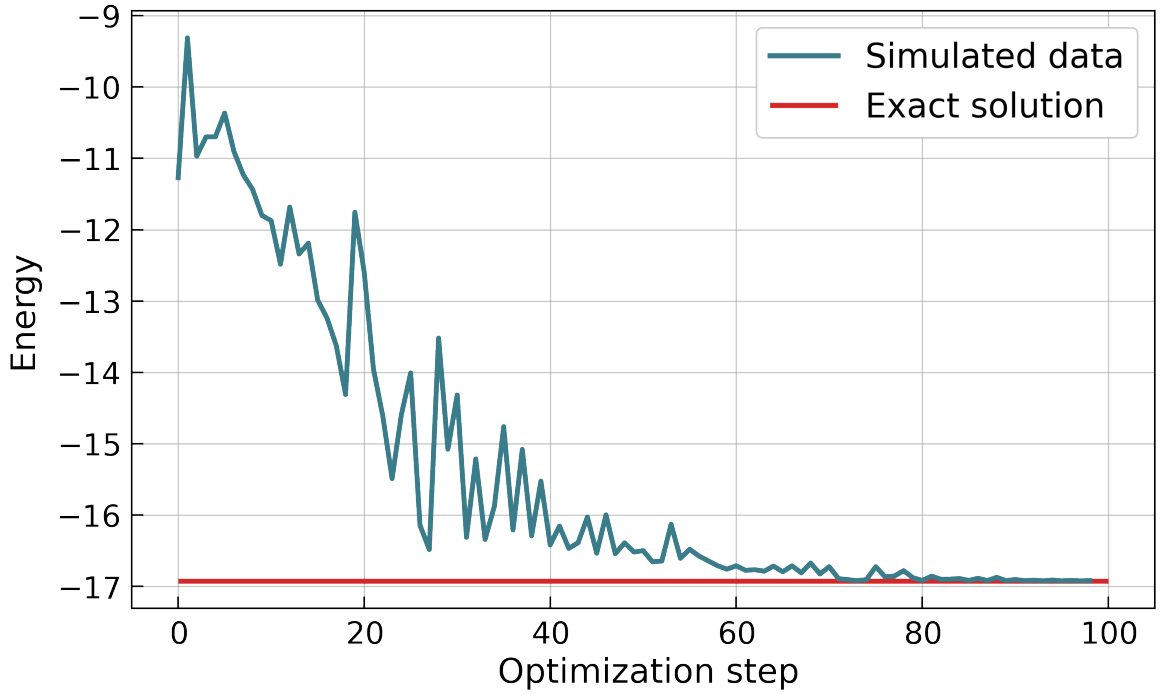


Figure 3.3: *The expectation value of the energy as a function of the optimization step for a VQE simulation with four qubits, 8 free parameters, and a translationally invariant quantum circuit presented in Figure 3.4.*

introduce the parametric XX gate as:

$$XX(\theta)_{j,k} = e^{-\frac{i}{2}\theta\sigma_j\sigma_k} \tag{3.19}$$

Since we suppose we are using a complete set of gates, we know we can build this gate from rotation gates and *CNOT* gates. We can construct a translationally invariant circuit using the same layered structure as the `EfficientSU2` quantum circuit but setting all the one qubit rotation angles in one layer to be the same and substituting the *CNOT* gates with the parametric XX gate. The XX gate-based entangling circuit also allows for a tunable entanglement inserted into the circuit. This is not possible for the *CNOT* entangling layer. The variational quantum circuit used for the simulations in Figure 3.3 is shown in Figure 3.4. The entangling layer acts with periodic boundary conditions to ensure translational invariance.

There are different advantages in using this circuit with respect to the `EfficientSU2` quantum circuit. The translationally invariant circuit has only five parameters to be optimized compared to sixteen with the `EfficientSU2` quantum circuit. Another advantage is again that the quantum circuit has a symmetry that is also present in the Hamiltonian. As we can see from the comparison between Figure 3.3 and Figure 3.2 the translationally invariant circuit performs much better in this case compared to the non-translationally invariant case. In the translational invariant case, we can reach the global minimum. This is not true for the non-translationally invariant case. In the latter, we converge to a higher

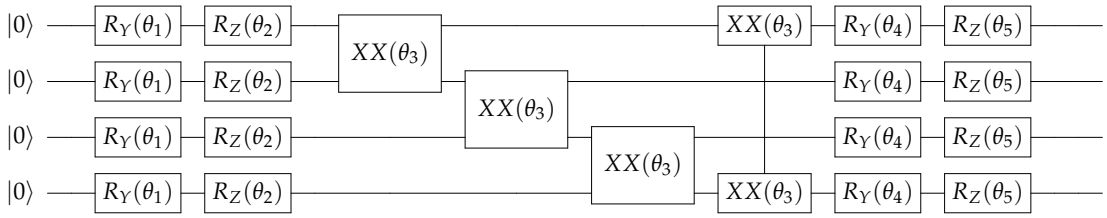


Figure 3.4: *translational invariant ansatz*

value than the exact solution, i.e., we get stuck in local minima. Moreover, the computational cost for the optimization of the translationally invariant circuit is much smaller than the non-translationally invariant case. This is because it is much easier to find the global minimum of a five-dimensional manifold with respect to a sixteen-dimensional manifold. This pedagogical example shows how important it is to incorporate the symmetry of the model we are studying in the quantum circuit ansatz. We will explore these concepts in a much more formal way in section 3.3.

3.2. Read-out Noise Mitigation

One of the big limitations of NISQ devices is the quantum noise. Quantum noise produces quantum errors that can be mitigated using error correction procedures [Nielsen and Chuang, 2010] and error mitigation procedures [Li and Benjamin, 2017; Temme et al., 2017; McClean et al., 2017; Bonet-Monroig et al., 2018; Endo et al., 2018; McArdle et al., 2019; Endo et al., 2019; Kandala et al., 2019; McClean et al., 2020; Otten and Gray, 2019a,b; Sagastizabal et al., 2019; Urbanek et al., 2019; Crawford et al., 2019; Chungheon et al., 2019; Córcoles et al., 2015; Sheldon et al., 2016; Tannu and Qureshi, 2019; Yeter-Aydeniz et al., 2019, 2020; Funcke et al., 2020]. Error correction codes require too many resources to be implemented in present hardware. In our work [Funcke et al., 2020] we have developed a mitigation routine to mitigate bit-flip errors. Bit flip errors give rise to an erroneous readout of a qubit state (you should measure 0 but get 1 instead). The readout noise is one of the most relevant noises in NISQ devices [Tannu and Qureshi, 2019].

Our method is based on a pre-calibration of the qubits and a correction of the performed noisy measurements. We have demonstrated that our approach does not introduce a significant computational overhead when measuring the expectation value of local operators. Our measurement mitigation works for any operator and any number of qubits. This procedure relies on cancellations of different erroneous measurement outcomes. In the development of the mitigation procedure, the assumption of uncorrelated readout errors is made. This assumption agrees with hardware data (see [Qiskit Aer API documentation and source code]). The goal of this section is to mitigate classical bit-flip errors. We thus neglect other sources of error, such as gate errors and decoherence Alexandrou et al. [2021].

3.2.1. Mitigation routine

We assume that we have a quantum computer that can prepare a pure state $|\psi\rangle$ for N qubits, which we measure in the computational basis (Z in this case). $|\psi\rangle$ can be written as:

$$|\psi\rangle = \sum_{i=0}^{2^N-1} c_i |i\rangle \tag{3.20}$$

where $|i\rangle$ are the elements of the computational basis and c_i are normalized complex coefficients. The mitigation routine we develop aims to mitigate the errors in the expectation value of a certain operator $\mathcal{H}$. It is fair to assume that we can write $\mathcal{H}$ as:

$$\mathcal{H} = \sum_k h_k U_k^* O_k U_k \tag{3.21}$$

where k is the number of terms in the operator $\mathcal{H}$ and h_k are their coefficients. $O_k \in \{\mathbb{1}, Z\}^{\otimes N}$ is a list of n Pauli string terms acting on N qubits. The unitary operators U_k transform this string to $U_k^* O_k U_k \in \{\mathbb{1}, X, Y, Z\}^{\otimes N}$. Since in the experiment we cannot measure directly on any basis, without loss of generality, we can assume that the operators that need to be mitigated can always be written in pauli basis. We include the operators U_k in the quantum circuit that prepares the state $|\psi\rangle$. As a result, one measurement procedure gives a bit string of 0 and 1. To obtain the expectation value of the operator, we need to reconstruct the bit string distribution. We must execute the same circuit several times. This procedure consists of preparing the state $U_k |\psi\rangle$ and measuring it many times. The number of measurements is referred to as the number of shots.

To give a meaningful description of our mitigation procedure, we need to change the point of view concerning the action of the bit-flip error. Whereas the bit flip error is usually seen as a part of the read-out procedure, we can see the read-out error as part of the operator we are measuring. We denote these operators as *noisy operators* $\tilde{Z}_q$ acting on the qubit q while the corresponding noise-free operator is called Z_q.

In this section, we use two different kinds of expectation values. $\mathbb{E}$ is the expectation value with respect to the bit-flip probability p. $O = \langle\psi| \hat{O} |\psi\rangle$ is the quantum mechanical expectation value of a generic operator $\hat{O}$ over a state $|\psi\rangle$.

Our goal is to extract the noise-free quantum mechanical expectation value $O = \langle\psi| \hat{O} |\psi\rangle$ from the noisy expectation value $\mathbb{E}\tilde{O}$ of a generic operator $\hat{O}$. To give a concrete example, we derive the noisy expectation value of a single operator. This procedure can be done explicitly and provides a good overview of how bit-flip error affects quantum measurements. We define the bit-flip probability $p_{q,0}$ as the probability of measuring a qubit q in the state $|1\rangle$ instead of the state $|0\rangle$. $p_{q,1}$ is the probability of measuring a qubit q in the state $|0\rangle$ instead of the state $|1\rangle$.

1-qubit case We consider the noisy expectation value of the operator $\tilde{Z}_q$. $\mathbb{E}\tilde{Z}_q$ can take the values:

- Z_q with probability $(1 - p_{q,0})(1 - p_{q,1})$ (no bit-flip happens),
- $-\mathbb{1}_q$ with probability $p_{q,0}(1 - p_{q,1})$(one bit-flip happens),
- $\mathbb{1}_q$ with probability $(1 - p_{q,0})p_{q,1}$(one bit-flip happens),
- or $-Z_q$ with probability $p_{q,0}p_{q,1}$(two bit-flips happen).

with respect to the bit-flip probability p. $p_{q,0}$ and $p_{q,1}$ are the bit flip probabilities for the qubit q. We can now express the noisy expectation $\mathbb{E}\tilde{Z}_q$ for the operator $\tilde{Z}_q$ as a function of the exact quantum mechanical expectation values $\{Z_q, \mathbb{1}_q\}$:

$$\mathbb{E}\tilde{Z}_q = (1 - p_{q,0} - p_{q,1})Z_q + (p_{q,1} - p_{q,0})\mathbb{1}_q \tag{3.22}$$

We can easily invert this relation to derive the quantum mechanical expectation value Z_q as a function of the noisy expectation value $\tilde{Z}_q$. The expectation value $\mathbb{1}_q$ is always 1.

General case Given a specific operator O_k, we assume that each term can be measured independently from the other. For the operator $\tilde{Z}_Q \otimes \cdots \otimes \tilde{Z}_1$ we can write:

$$\mathbb{E}\left(\tilde{Z}_Q \otimes \cdots \otimes \tilde{Z}_1\right) = \mathbb{E}\tilde{Z}_Q \otimes \cdots \otimes \mathbb{E}\tilde{Z}_1 \tag{3.23}$$

We can prove this relation by considering the expectation value of two different operators $\tilde{Z}_a$ and $\tilde{Z}_b$ acting on different qubits. We can take the expectation value of one operator and leave the others untouched. We call this expectation value the *conditional expectation value*. We can define the conditional expectation value of the operator $\tilde{Z}_a$ as $\mathbb{E}^{\tilde{Z}_a}$ and accordingly $\mathbb{E}^{\tilde{Z}_b}$ for $\tilde{Z}_b$. The expectation value of $\tilde{Z}_a$ can take the value in $T_\alpha \in \{Z_q, -Z_q, \mathbb{1}, -\mathbb{1}\}$ with probability p_α. $\mathbb{E}\left(\tilde{Z}_a \otimes \tilde{Z}_b\right)$ can be written as:

$$\mathbb{E}\left(\tilde{Z}_a \otimes \tilde{Z}_b\right) = \mathbb{E}^{\tilde{Z}_b}\left(\sum_\alpha p_\alpha T_\alpha \otimes \tilde{Z}_b\right) = \mathbb{E}\tilde{Z}_a \otimes \mathbb{E}\tilde{Z}_b \tag{3.24}$$

Our assumption is correct if the bit-flip errors are uncorrelated between the different qubits. We need to construct a matrix ω^{-1} that multiplies the noisy expectations $\mathbb{E}\langle\psi|\tilde{O}|\psi\rangle$ and gives us the noise free expectation value:

$$(O)_{O\in\{\mathbb{1},Z\}^{\otimes Q}} = \omega^{-1}\left(\mathbb{E}\tilde{O}\right)_{\tilde{O}\in\{\mathbb{1},Z\}^{\otimes Q}} \tag{3.25}$$

Given the assumption made in Equation 3.23, we can write:

$$\mathbb{E}\left(\tilde{Z}_Q \otimes \cdots \otimes \tilde{Z}_1\right) = \sum_{O\in\{\mathbb{1},Z\}^{\otimes Q}} \gamma(O_Q)O_Q \otimes \cdots \otimes \gamma(O_1)O_1 \tag{3.26}$$

where the coefficients γ can be defined as:

$$\gamma(O_q) := \begin{cases} 1 - p_{q,0} - p_{q,1} & \text{for } O_q = Z_q \\ p_{q,1} - p_{q,0} & \text{for } O_q = \mathbb{1}_q \end{cases} \tag{3.27}$$

We introduce the "lexicographic order" $\preceq$ for the noise-free operators $O \in \{\mathbb{1}, Z\}^{\otimes Q}$ and the noisy operators $\tilde{O} \in \{\mathbb{1}, Z\}^{\otimes Q}$ as:

$$\begin{aligned} &\mathbb{1}_3 \otimes \mathbb{1}_2 \otimes \mathbb{1}_1 \preceq \mathbb{1}_3 \otimes \mathbb{1}_2 \otimes Z_1 \preceq \mathbb{1}_3 \otimes Z_2 \otimes \mathbb{1}_1 \preceq \mathbb{1}_3 \otimes Z_2 \otimes Z_1 \\ \preceq &Z_3 \otimes \mathbb{1}_2 \otimes \mathbb{1}_1 \preceq Z_3 \otimes \mathbb{1}_2 \otimes Z_1 \preceq Z_3 \otimes Z_2 \otimes \mathbb{1}_1 \preceq Z_3 \otimes Z_2 \otimes Z_1 \preceq \dots \end{aligned} \tag{3.28}$$

This ordering is presented for a set of three operators. It is evident how this generalizes for N qubits. To find a generic formula to correct any operator, we need to generalize Equation 3.26. We can write the noisy expectation value as:

$$\mathbb{E}\left(\tilde{O}_Q \otimes \cdots \otimes \tilde{O}_1\right) = \sum_{O \in \{\mathbb{1}, Z\}^{\otimes Q}} \Gamma(O_Q|\tilde{O}_Q) O_Q \otimes \cdots \otimes \Gamma(O_1|\tilde{O}_1) O_1 \tag{3.29}$$

where $\Gamma(O_q|\tilde{O}_q)$ is defined as:

$$\Gamma(O_q|\tilde{O}_q) = \begin{cases} \gamma(O_q) & \text{for } \tilde{O}_q = \tilde{Z}_q \\ 1 & \text{for } O_q = \mathbb{1}_q \ \wedge \ \tilde{O}_q = \tilde{\mathbb{1}}_q \\ 0 & \text{for } O_q = Z_q \ \wedge \ \tilde{O}_q = \tilde{\mathbb{1}}_q. \end{cases} \tag{3.30}$$

From the coefficients Γ, we can construct the matrix ω that relates the noise and the noiseless expectation value. The matrix elements of ω are given by:

$$\omega\left(O|\tilde{O}\right) := \prod_{q=1}^{Q} \Gamma(O_q|\tilde{O}_q) \tag{3.31}$$

and the full matrix is defined as:

$$\omega := \left(\omega\left(O|\tilde{O}\right)\right)_{\tilde{O}, O \in \{\mathbb{1}, Z\}^{\otimes Q}} \tag{3.32}$$

The lexicographic order tells us that the matrix element $\omega\left(O|\tilde{O}\right)$ are zero if $\tilde{O} \prec O$. This property ensures that the matrix ω is lower triangular. A matrix is invertible if the determinant is non-zero. For a lower triangular matrix, it is sufficient to have the diagonal term to be non-zero. The diagonal term of ω are:

$$\omega = \prod_{q=1}^{Q} \Gamma\left(O_q|\tilde{O}_q\right) \tag{3.33}$$

We can now evaluate Equation 3.25 on an arbitrary state $|\psi\rangle$ to find the bit-flip corrected expectation values. For all $O \in \{\mathbb{1}, Z\}^{\otimes Q}$, we find:

$$\langle\psi| O |\psi\rangle = \sum_{\tilde{O}\in\{\mathbb{1},Z\}^{\otimes Q}} \omega^{-1}_{O,\tilde{O}} \mathbb{E} \langle\psi| \tilde{O} |\psi\rangle \tag{3.34}$$

Even though the generic form for the bit-flip correction of generic operators seems to have an exponential scaling in the number of qubits, this is not true for many practical applications. In subsection 3.2.6 we will discuss the scaling of the computational cost in depth.

2-qubit example For the operator $O = Z_2 \otimes Z_1$, we can derive the explicit formula that reconstructs the quantum mechanical operator as a function of the noisy expectation values. We use Equation 3.34 and explicitely derive:

$$\begin{aligned} Z_2 \otimes Z_1 = \quad & \tfrac{1}{\gamma(Z_2)\gamma(Z_1)} \mathbb{E}\left(\tilde{Z}_2 \otimes \tilde{Z}_1\right) - \tfrac{\gamma(\mathbb{1}_1)}{\gamma(Z_2)\gamma(Z_1)} \mathbb{E}\left(\tilde{Z}_2\right) \otimes \mathbb{1}_1 \\ & - \tfrac{\gamma(\mathbb{1}_2)}{\gamma(Z_2)\gamma(Z_1)} \mathbb{1}_2 \otimes \mathbb{E}\left(\tilde{Z}_1\right) + \tfrac{\gamma(\mathbb{1}_2)\gamma(\mathbb{1}_1)}{\gamma(Z_2)\gamma(Z_1)} \mathbb{1}_2 \otimes \mathbb{1}_1 \end{aligned} \tag{3.35}$$

3.2.2. Computation of the variance for the noisy expectation value

For the computation of the variance of the noisy expectation value, we start again from a one-qubit case and generalize the formula to arbitrary number of qubits and operators. The variance $\mathbb{V}\tilde{Z}_q$ of the noisy expectation value of $\mathbb{E}\tilde{Z}_q$ in Equation 3.22 is defined as:

$$\mathbb{V}\tilde{Z}_q = \mathbb{E}(\tilde{Z}_q \otimes \tilde{Z}_q) - (\mathbb{E}\tilde{Z}_q)^2 = \Phi'_{\tilde{Z}_q}(0) \otimes \Phi'_{\tilde{Z}_q}(0) - \Phi''_{\tilde{Z}_q}(0) \tag{3.36}$$

where the characteristic function $\Phi_{\tilde{Z}_q}(t)$ is defined as:

$$\Phi_{\tilde{Z}_q}(t) := \mathbb{E} \exp\left[i \operatorname{Tr}\left(t^* \tilde{Z}_q\right)\right] \tag{3.37}$$

We need to compute the derivatives $\Phi'_q(0) = i\mathbb{E}\tilde{Z}_q$ and $\Phi''_{\tilde{Z}_q}(0) = -\mathbb{E}(\tilde{Z}_q)^2$:

$$\begin{aligned} \Phi'_{\tilde{Z}_q}(0) &= i(1 - p_{q,0} - p_{q,1})Z_q + i(p_{q,1} - p_{q,0})\mathbb{1}_q \\ \Phi''_{\tilde{Z}_q}(0) &= -\left(1 - p_{q,0} - p_{q,1} + 2p_{q,0}p_{q,1}\right)Z_q \otimes Z_q - (p_{q,0} + p_{q,1} - 2p_{q,0}p_{q,1})\mathbb{1}_q \otimes \mathbb{1}_q \end{aligned} \tag{3.38}$$

We can now derive the variance in Equation 3.36:

$$\begin{aligned} \mathbb{V}\tilde{Z}_q = & \left[(p_{q,0} + p_{q,1})(1 - p_{q,0} - p_{q,1}) + 2p_{q,0}p_{q,1}\right] \times Z_q \otimes Z_q \\ & - (1 - p_{q,0} - p_{q,1})(p_{q,1} - p_{q,0})Z_q \otimes \mathbb{1}_q \\ & - (1 - p_{q,0} - p_{q,1})(p_{q,1} - p_{q,0})\mathbb{1}_q \otimes Z_q \\ & + (p_{q,0} + p_{q,1} - p^2_{q,0} - p^2_{q,1})\mathbb{1}_q \otimes \mathbb{1}_q \end{aligned} \tag{3.39}$$

We now generalize the variance to operators acting on more than one qubit, i.e., $Q > 1$. We use the notion of uncorrelated measurements Equation 3.23 to demonstrate that the covariance of two operators $\tilde{O}_1$ and $\tilde{O}_2$ acting on different qubits vanishes:

$$\mathrm{Cov}_{\otimes}(\tilde{O}_1, \tilde{O}_2) := \mathbb{E}(\tilde{O}_1 \otimes \tilde{O}_2) - \mathbb{E}(\tilde{O}_1) \otimes \mathbb{E}(\tilde{O}_2) \ = 0 \tag{3.40}$$

For the operator $\tilde{Z}_Q \otimes \cdots \otimes \tilde{Z}_1$, the variance with respect to the noisy expectation value is:

$$\begin{aligned}
&\mathbb{V}\left(\tilde{Z}_Q \otimes \cdots \otimes \tilde{Z}_1\right) = \\
&= \mathbb{E}\left(\tilde{Z}_Q \otimes \cdots \otimes \tilde{Z}_1 \otimes \tilde{Z}_Q \otimes \cdots \otimes \tilde{Z}_1\right) - \mathbb{E}\left(\tilde{Z}_Q \otimes \cdots \otimes \tilde{Z}_1\right) \otimes \mathbb{E}\left(\tilde{Z}_Q \otimes \cdots \otimes \tilde{Z}_1\right) \\
&= U^* \left(\mathbb{E}\left(\tilde{Z}_Q \otimes \tilde{Z}_Q\right) \otimes \cdots \otimes \mathbb{E}\left(\tilde{Z}_1 \otimes \tilde{Z}_1\right) - \bigotimes_{q=1}^{Q} \left(\mathbb{E}\tilde{Z}_q \otimes \mathbb{E}\tilde{Z}_q\right) \right) U \\
&= U^* \left(\left(\mathbb{V}\tilde{Z}_Q + \mathbb{E}\tilde{Z}_Q \otimes \mathbb{E}\tilde{Z}_Q\right) \otimes \cdots \otimes \left(\mathbb{V}\tilde{Z}_1 + \mathbb{E}\tilde{Z}_1 \otimes \mathbb{E}\tilde{Z}_1\right) - \bigotimes_{q=1}^{Q} \left(\mathbb{E}\tilde{Z}_q \otimes \mathbb{E}\tilde{Z}_q\right) \right) U
\end{aligned} \tag{3.41}$$

where the unitary operation U re-orders the tensor products from $|\psi_Q\rangle \otimes \cdots \otimes |\psi_1\rangle \otimes |\psi_Q\rangle \otimes \cdots \otimes |\psi_1\rangle$ to $(|\psi_Q\rangle \otimes |\psi_Q\rangle) \otimes \cdots \otimes (|\psi_1\rangle \otimes |\psi_1\rangle)$.

3.2.3. Scaling analysis and classical simulator result

In this section, we explore the scaling of the absolute error of the expectation value of different operators, simulated on classical computers, as a function of the number of shots. We have performed these simulations on a classical wave function simulator where we implemented only readout error. The expectation value of an operator is extracted from the bit-flip distribution of a state. The readout error has been implemented as a distortion of these bit-string distributions. Let us give a practical example. We want to measure the expectation value of σ_z over the state $|1\rangle$. Without any quantum noise, every measurement would give 1 as a result. If we have a bit-flip-probability of 10%, for any measurement, we flip the result from $1 \to 0$ with a probability of 10%. From the final bit-string distribution, we extract the expectation value of the operator σ_z. In the limit of an infinite number of shots, the expectation value of σ_z with 10% bit-flip probability would give the result of 0.9. To give an intuitive analysis, we fix all bit-flip probabilities $p_{q,b} = p$ to be equal. For one qubit, the expectation $\mathbb{E}\tilde{Z}_q$,defined in Equation 3.34, reduces to:

$$\mathbb{E}\tilde{Z}_q = (1 - 2p) Z_q \tag{3.42}$$

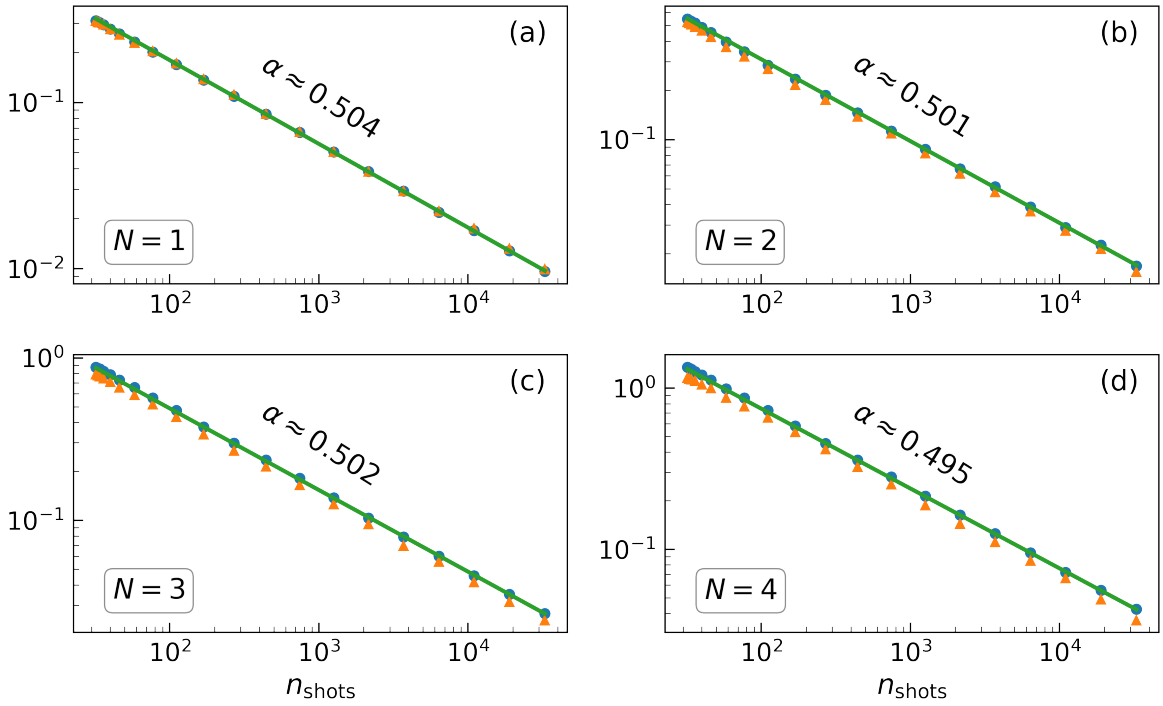


Figure 3.5: *Relative errors (blue dots) and standard deviations (orange triangles) for the bit-flip corrected expectation values of* $\langle\psi|\,\tilde{Z}_Q \otimes \cdots \otimes \tilde{Z}_1\,|\psi\rangle$*, as retrieved from histogram data using Equation 3.34 compared to the "true" bit-flip free expectation values of* $\langle\psi|\,Z_Q \otimes \cdots \otimes Z_1\,|\psi\rangle$*. Shown are the four different operators (a)* Z_1*, (b)* $Z_2 \otimes Z_1$*, (c)* $Z_3 \otimes Z_2 \otimes Z_1$*, and (d)* $Z_4 \otimes Z_3 \otimes Z_2 \otimes Z_1$*. The average relative errors are fitted with a power law in the number of shots* n*,* $y(n) \propto n^{\alpha}$ *(green lines), the slopes obtained are indicated in the different panels. The standard deviations of the relative errors are extracted from* 4096 *random states* $|\psi\rangle$ *and random bit-flip probabilities* $p_{q,b}$ *uniformly drawn between 0.05 and 0.25.*

For $Q > 1$ and equal bit-flip probabilities, the expectation value of Q different operators ($\mathbb{E}(\tilde{Z}_Q \otimes \cdots \otimes \tilde{Z}_1)$) in Equation 3.34 reduces to:

$$\mathbb{E}(\tilde{Z}_Q \otimes \cdots \otimes \tilde{Z}_1) = (1-2p)^Q\, Z_Q \otimes \cdots \otimes Z_1 \tag{3.43}$$

This implies that the matrix ω (defined in Equation 3.31) becomes diagonal with

$$\mathbb{E}(\tilde{O}_Q \otimes \cdots \otimes \tilde{O}_1) = (1-2p)^{\#Z(O)} O_Q \otimes \cdots \otimes O_1 \tag{3.44}$$

where $\#Z(O)$ is the number of terms $O_q = Z_q$ in the tensor product $O = O_N \otimes \cdots \otimes O_1$

In particular, ω is invertible as long as $p \neq 1/2$. In Figure 3.5, we show the relative error for the bit-flip corrected expectation value of $\langle\psi|\,\tilde{Z}_Q \otimes \cdots \otimes \tilde{Z}_1\,|\psi\rangle$, as retrieved from histogram data using Equation 3.34, compared to the quantum mechanical expectation value $\langle\psi|\,Z_Q \otimes \cdots \otimes Z_1\,|\psi\rangle$:

$$\frac{\left|\langle\psi|\,\tilde{Z}_Q \otimes \cdots \otimes \tilde{Z}_1\,|\psi\rangle - \langle\psi|\,Z_Q \otimes \cdots \otimes Z_1\,|\psi\rangle\right|}{\left|\langle\psi|\,Z_Q \otimes \cdots \otimes Z_1\,|\psi\rangle\right|}. \tag{3.45}$$

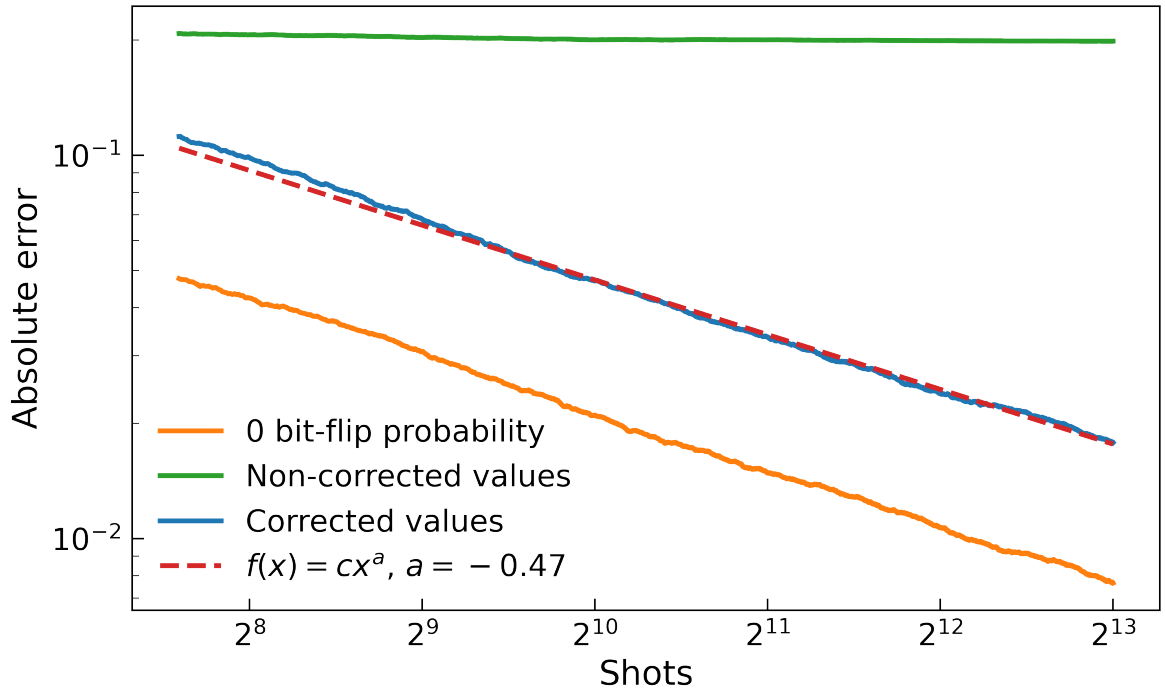


Figure 3.6: *Benchmark of the scaling of the absolute error of the bit flip corrected expectation value of the operator ZZ acting on two qubits. The values of the bit flip probabilities are* $p_{1,0} = 0.1$, $p_{1,1} = 0.13$, $p_{2,0} = 0.15$ *and* $p_{2,1} = 0.2$.

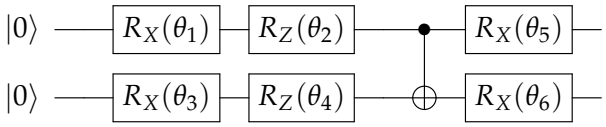


Figure 3.7: *Quantum circuit used for the data in Figure 3.6.*

We also plot the standard deviation of this relative error instead of plotting the error bars to visualize the plot better. Figure 3.5 also contains a fit $y(n) = Cn^{-\alpha}$ of the relative error in Equation 3.45, where n is again the number of shots, i.e., the number of $\langle\psi| \tilde{Z}_Q \otimes \cdots \otimes \tilde{Z}_1 |\psi\rangle$ evaluations to produce the histogram. In particular, the fit indicates Monte-Carlo type convergence $\alpha \approx 1/2$ for $Q \in \{1, 2, 3, 4\}$. Figure 3.5 has been generated using 4096 random states $|\psi\rangle$ satisfying $|\langle\psi| Z_Q \otimes \cdots \otimes Z_1 |\psi\rangle| \geq 0.25$ to avoid dividing by small numbers when computing relative errors.

In Figure 3.6 we analyze the scaling of our routine in the context of states generated by quantum circuits on a wave function simulator algorithm. We plot the absolute error of the measurement of the expectation value of the operator $Z_2 \otimes Z_1$. We prepare the state ψ using the parametric circuit shown in Figure 3.7. The circuit is composed of six single-qubit gates, one two-qubit operation, and allows for generating arbitrary two-qubit wave functions starting from $|00\rangle$. We have used different bit-flip probabilities for the two qubits. Our

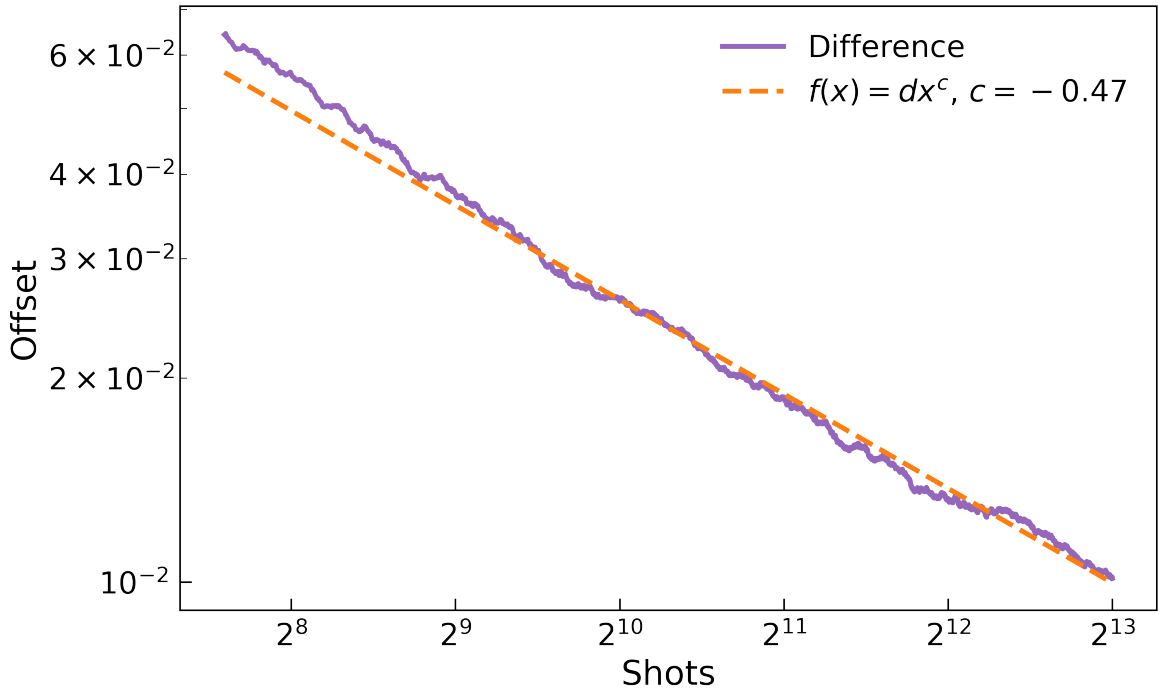


Figure 3.8: *Scaling of the difference between the bit-flip corrected data and the values with 0 bit-flip probability data in Figure 3.6.*

results demonstrate that the absolute error of the corrected data shows a power-law decay with an exponent of ~ -0.5, which one would expect for the case without readout error. In Figure 3.6 it seems that there is a constant offset between the bit-flip corrected values and the values with 0 bit-flip probability. This offset is not constant since this is a logarithmic plot. This offset decreases following a power-law with the power of ~ -0.5 as shown in Figure 3.8. The offset is zero in the limit of an infinite number of shots.

3.2.4. Noise mitigation of the noisy expectation value of generic $\mathcal{H}$

In practical applications, we want to measure the energy of the ground state of a given Hamiltonian. If we measure the distribution of $|\psi\rangle$, the measurements of $\langle\psi| U^* O U |\psi\rangle$ comprising $\langle\psi| \mathcal{H} |\psi\rangle$ are no longer independent. However, it has no impact on the expectation subject to bit flips since the linearity of the expectation value implies:

$$\mathbb{E}\,\langle\psi|\,\tilde{\mathcal{H}}\,|\psi\rangle = \mathbb{E}\,\langle\psi|\sum_\alpha \lambda_\alpha U_\alpha^* \tilde{O}_\alpha U_\alpha\,|\psi\rangle = \langle\psi|\sum_\alpha \lambda_\alpha U_\alpha^* \left(\mathbb{E}\tilde{O}_\alpha\right) U_\alpha\,|\psi\rangle\,, \tag{3.46}$$

which is precisely the expression we would obtain from summing the independently measured operators $\tilde{O}_\alpha$. In order to correct for bit-flips in this setting, we need to keep in mind that the general case requires measurements of all operators $O \preceq O_\alpha$ (with respect to the lexicographic order $\preceq$ on $\{\mathbb{1}, Z\}^{\otimes N}$) for all operators O_α in $\mathcal{H} = \sum_\alpha \lambda_\alpha U_\alpha^* O_\alpha U_\alpha$. Hence, the histogram for $\langle\psi|\,\tilde{\mathcal{H}}\,|\psi\rangle$ does not contain sufficient information. However, we can use

the classical bit-flip correction method to find coefficients $\omega_{\alpha,O}$ such that:

$$O_\alpha = \sum_{O \preceq O_\alpha} \omega_{\alpha,O} \mathbb{E}\tilde{O} \tag{3.47}$$

Inserting this into $\mathcal{H}$, we can express $\mathcal{H}$ as

$$\mathcal{H} = \sum_\alpha \lambda_\alpha U_\alpha^* \sum_{O \preceq O_\alpha} \omega_{\alpha,O} \mathbb{E}\tilde{O} U_\alpha. \tag{3.48}$$

In other words, we can replace the operator $\mathcal{H}$ by the bit-flip corrected noisy operator:

$$\tilde{\mathcal{H}}_{\text{bfc}} := \sum_\alpha \lambda_\alpha U_\alpha^* \sum_{O \preceq O_\alpha} \omega_{\alpha,O} \tilde{O} U_\alpha \tag{3.49}$$

and obtain

$$\mathbb{E} \langle\psi| \tilde{\mathcal{H}}_{\text{bfc}} |\psi\rangle = \langle\psi| \mathcal{H} |\psi\rangle . \tag{3.50}$$

We define the variance of the expectation value with respect to the bit-flip distribution as $\mathbb{V}_{\text{bsd}}$. In general, if $\tilde{\mathcal{H}} = \sum_\alpha \lambda_\alpha U_\alpha^* \tilde{O}_\alpha U_\alpha$, we are still able to predict the variance $\mathbb{V}_{\text{bsd}} \langle\psi| \tilde{\mathcal{H}} |\psi\rangle$ using the same method as above althought the covariance terms no longer vanish (each $\tilde{O}_\alpha$ is a tensor product $\tilde{O}_{\alpha,Q} \otimes \cdots \otimes \tilde{O}_{\alpha,1}$). For $\tilde{O}_{\alpha,q} = \tilde{Z}_q$, $\tilde{O}_{\alpha,q}$ takes one of the possible values $\{Z_q, -\mathbb{1}_q, \mathbb{1}_q, -Z_q\}$. For $\tilde{O}_{\alpha,q} = \tilde{\mathbb{1}}_q$, $\tilde{O}_{\alpha,q}$ always takes the value $\mathbb{1}_q$. Using these replacements for all summands in $\tilde{\mathcal{H}}$, we obtain that $\tilde{\mathcal{H}}$ takes finitely many (up to 2^N) values $\mathcal{H}_\alpha$ with probability p_α. The characteristic function $\Phi_{\tilde{\mathcal{H}}}$ is then given by

$$\Phi_{\tilde{\mathcal{H}}}(t) := \mathbb{E} \exp\left(i \operatorname{tr}\left(t^* \tilde{\mathcal{H}}\right)\right) = \sum_\alpha p_\alpha \exp\left(i \operatorname{tr}\left(t^* \mathcal{H}_\alpha\right)\right). \tag{3.51}$$

As such, we can directly conclude

$$\Phi'_{\tilde{\mathcal{H}}}(0) = \sum_\alpha p_\alpha i \mathcal{H}_\alpha = i \mathbb{E}\tilde{\mathcal{H}}, \tag{3.52}$$

$$\Phi''_{\tilde{\mathcal{H}}}(0) = -\sum_\alpha p_\alpha \mathcal{H}_\alpha \otimes \mathcal{H}_\alpha, \tag{3.53}$$

and find the variance operator

$$\begin{aligned} \mathbb{V}_{\text{bsd}} \tilde{\mathcal{H}} &= \Phi'_{\tilde{\mathcal{H}}}(0) \otimes \Phi'_{\tilde{\mathcal{H}}}(0) - \Phi''_{\tilde{\mathcal{H}}}(0) \\ &= \left(\sum_\alpha p_\alpha \mathcal{H}_\alpha \otimes \mathcal{H}_\alpha\right) - \left(\mathbb{E}\tilde{\mathcal{H}}\right) \otimes \left(\mathbb{E}\tilde{\mathcal{H}}\right) \\ &= \left(\sum_\alpha p_\alpha \mathcal{H}_\alpha \otimes \mathcal{H}_\alpha\right) - \left(\sum_{\alpha,\beta} p_\alpha p_\beta \mathcal{H}_\alpha \otimes \mathcal{H}_\beta\right). \end{aligned} \tag{3.54}$$

Similarly, we can measure the operator $\tilde{\mathcal{H}}$ on the state $|\psi\rangle$ and obtain the variance

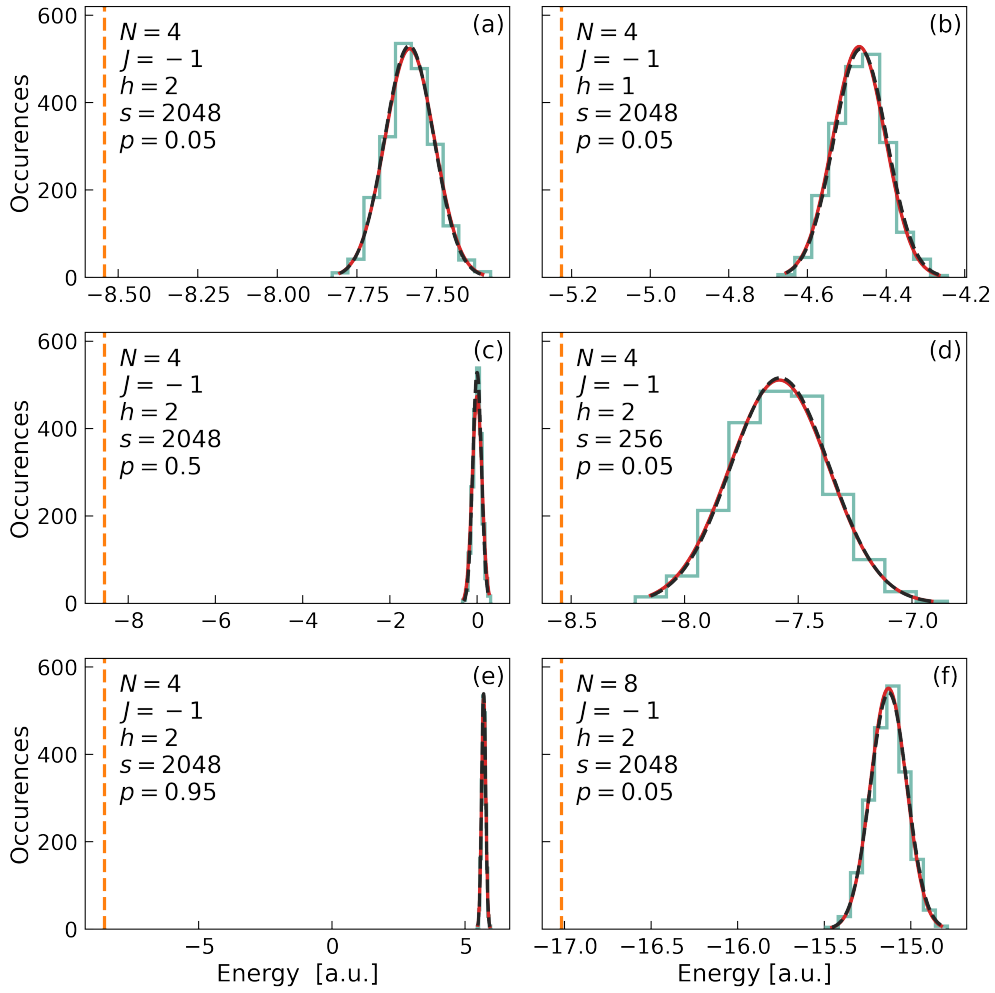


Figure 3.9: *Energy histograms for the transversal Ising model in light green. The vertical dashed orange line indicates the true ground-state energy, the solid red line the prediction and the dashed black line a fit to the data. The left column corresponds to $N = 4$, $J = -1$, $h = 2$, $n = 2048$ with (a) $p = 0.05$, (c) $p = 0.50$, and (e) $p = 0.95$. The right column shows varied N, h, and n: (b) $h = 1$, (d) $n = 256$, and (f) $N = 8$.*

$$\mathbb{V}_{\text{bsd}} \langle\psi| \tilde{\mathcal{H}} |\psi\rangle = \left(\sum_{\alpha} p_\alpha \langle\psi| \mathcal{H}_\alpha |\psi\rangle^2 \right) - \left(\sum_{\alpha,\beta} p_\alpha p_\beta \langle\psi| \mathcal{H}_\alpha |\psi\rangle \langle\psi| \mathcal{H}_\beta |\psi\rangle \right). \tag{3.55}$$

The analysis of the bit-flip error above assumed that we measure general operators $\mathcal{H}$ by expressing them as linear combinations of operators $U^* O U$ with $O \in \{\mathbb{1}, Z\}^{\otimes N}$ on an N-qubit machine, and by measuring each O independently (U being the transformation into the Z basis).

Example for a simple Hamiltonian We are interested in measuring the expectation value of:

$$\mathcal{H}_{ZZ} = \sum_{i=1}^{N} Z_i Z_{i+1} \tag{3.56}$$

with $N = 3$ qubits. We generate independent histograms for $\langle\psi| \mathbb{1}_3 \otimes Z_2 \otimes Z_1 |\psi\rangle$, $\langle\psi| Z_3 \otimes Z_2 \otimes \mathbb{1}_1 |\psi\rangle$, and $\langle\psi| Z_3 \otimes \mathbb{1}_2 \otimes Z_1 |\psi\rangle$, extract their expectation values, and recover $\langle\psi| \mathcal{H}_{ZZ} |\psi\rangle$ accordingly.

Transversal Ising model We consider the transversal Ising model Hamiltonian defined as:

$$\mathcal{H}_{\text{TI}} = \sum_{i=1}^{N} Z_i Z_{i+1} + h \sum_{i=1}^{N} X_i \tag{3.57}$$

where h is the transversal field strength. We cannot directly recover the full expectation value $\langle\psi| \mathcal{H}_{\text{TI}} |\psi\rangle$ from measuring the distribution of $|\psi\rangle$. However, we can compute $\mathcal{H}_{ZZ} = \sum_{i=1}^{N} Z_i Z_{i+1}$ that can be measured directly by using the bit-string distribution of the state $|\psi\rangle$, and $\mathcal{H}_X = h \sum_{i=1}^{N} X_i$ can be measured by using $h \sum_{i=1}^{N} Z_i$ and the bit-string distribution of the state $H^{\otimes N} |\psi\rangle$, i.e., after applying a Hadamard gate H on each qubit. In other words, we only have to measure two bit-string distributions instead of measuring each of the $2N$ Pauli-terms separately.
In Figure 3.9 the energy histograms for the transversal Ising model are presented. We have performed those simulations on a classical machine with equal bit-flip probability for all qubits. The histograms are generated by the bit-flip distribution of the energy as defined above. The expectation values are evaluated on the exact ground state (computed via exact diagonalization). The corrected measure can be recovered from Equation 3.29 and the variance from Equation 3.55. The solid red line is the prediction of the histogram distribution computed using Equation 3.55 and the dashed black line a fit to the data. The two lines are almost perfectly overlapped, showing great agreement between the prediction and the data.

This agreement shows that the correction procedure can predict both the mean energy value and the bit-flip distribution variance. This ability to predict the exact value (mean) is of great interest for practical applications. Let us suppose we have computed the ground state via a variational quantum simulation, and we want to calculate the expectation value of a specific operator on the ground state. If we know the bit-flip probabilities, we can recover the quantum mechanical expectation value of this operator precisely and extract physical information. It is important to note that, for non-equal bit-flip probabilities, there is always a constant offset between the exact value and the noisy expectation value. This constant offset would prevent the extraction of any physical values even with an infinite number of shots.

3.2.5. Experimental results

We perform quantum simulations on IBMQ hardware to demonstrate our mitigation routine's applicability using the Qiskit software development kit (SDK) [Qiskit Aer API doc-

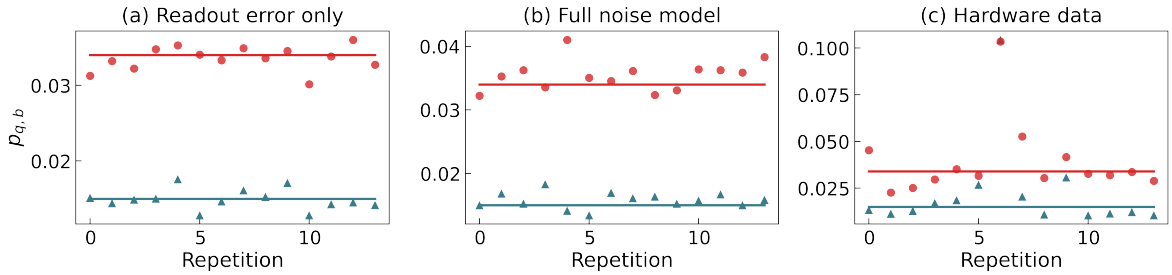


Figure 3.10: *Bit-flip probabilities $p_{0,0}$ (blue triangles) and $p_{0,1}$ (orange dots) for the single qubit case measured with the calibration procedure as a function of the repetition for (a) classically simulating ibmq_burlington with readout error only, (b) the full noise model, and (c) data obtained on the hardware. The solid lines correspond to the data provided by the noise model.*

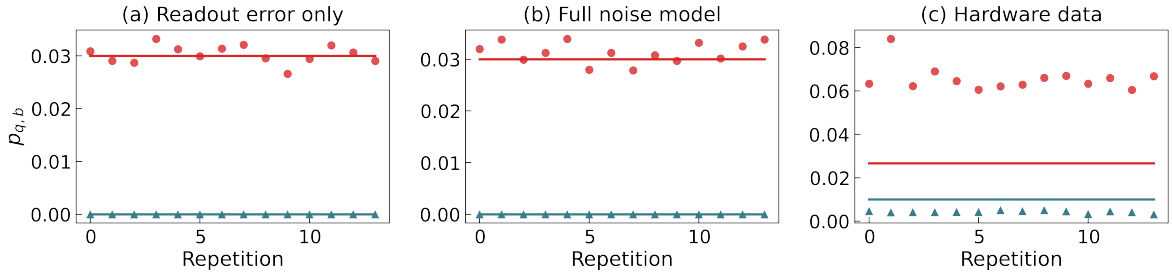


Figure 3.11: *Bit-flip probabilities $p_{0,0}$ (blue triangles) and $p_{0,1}$ (orange dots) for the single qubit case measured with the calibration procedure as a function of the repetition for (a) classically simulating ibmq_london with readout errors only, (b) the full noise model, and (c) data obtained on the quantum hardware. The solid lines correspond to the data provided by the noise model.*

umentation and source code]. Although the Qiskit SDK provides values for the bit-flip probabilities for the different qubits on the different chips, we choose to calibrate $p_{q,0}$ and $p_{q,1}$ ourselves. To obtain $p_{q,0}$, we measure the initial state using $n_{\text{calibration}}$ shots and record the number of $|1\rangle$ outcomes. Similarly, we determine $p_{q,1}$ by first applying an X gate to the qubit q, thus preparing the state $|1\rangle$ and measure the resulting state again $n_{\text{calibration}}$ times and record the number of 0 outcomes. We use $n_{\text{calibration}} = 8192$, which is the maximum number of repetitions possible on the real quantum hardware for all data shown in the text. Moreover, to acquire some statistics on how the obtained values for the bit-flip probabilities fluctuate, we repeat this procedure multiple times. Subsequently, we average all the data obtained for $p_{q,b}$. The resulting bit-flip probabilities are the ones used for correcting the data.

The corresponding results for the calibration of the imbq_burlington quantum hardware are shown in Figure 3.10. The classical simulation of the chip using the noise model produces, as expected, bit-flip probabilities in agreement with the values provided. Looking at the data from the quantum hardware in Figure 3.10(c), we see that these fluctuate over a wide range between different repetitions. Thus, in this case the bit-flip probabilities cannot be extracted very reliably. The corresponding results for the calibration of the ibmq_london quantum hardware are shown in Figure 3.11. Looking at the data resulting from simulating

ibmq_london classically with readout noise only in Figure 3.11(a), we observe that the bit-flip probabilities of our calibration procedure yields scatter around the value provided by the noise model. Using the full noise model does not change the picture a lot, only the values for $p_{0,1}$ scatter slightly more around the value of the noise model, as Figure 3.11(b) reveals. The data generated on the actual ibmq_london quantum hardware in Figure 3.11(c) does not agree very well with the values of the noise model. Even the values for $p_{0,0}$, which does not involve a single gate, are in general lower than the value provided by the noise model. In contrast, $p_{0,1}$ exceeds the value of the noise model. Despite the fact that the values for the experimentally obtained bit-flip probabilities deviate from the noise model, they only fluctuate moderately and we can extract a reasonable bit-flip probability by averaging over all repetitions. Comparing the different panels of Figure 3.11 closely, one can also observe that the values for the bit-flip probabilities provided by the noise model in panel (c) differ slightly from those in panel (a) and (b). The reason for that is that the data in the noise model is updated every day and our classical simulations as well as our simulations on real quantum hardware were not carried out the same day.

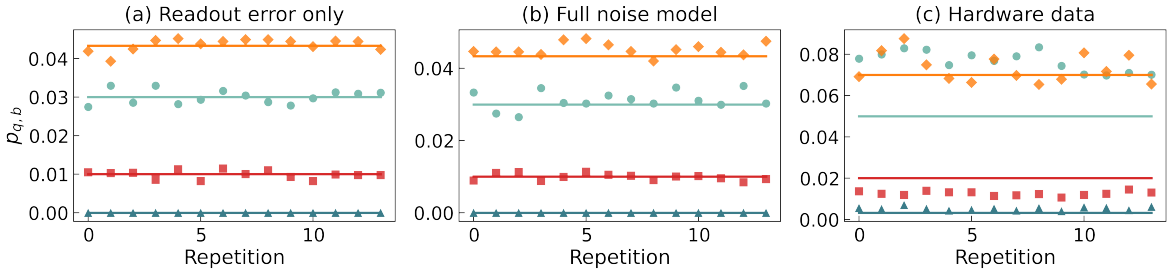


Figure 3.12: *Bit-flip probabilities $p_{0,0}$ (blue triangles), $p_{0,1}$ (orange dots), $p_{1,0}$ (green squares), $p_{1,1}$ (red diamonds) for the two-qubit case measured with the calibration procedure as a function of the repetition for (a) classically simulating ibmq_london with readout error only, (b) the full noise model, and (c) data obtained on the hardware. The solid lines correspond to the data provided by the noise model.*

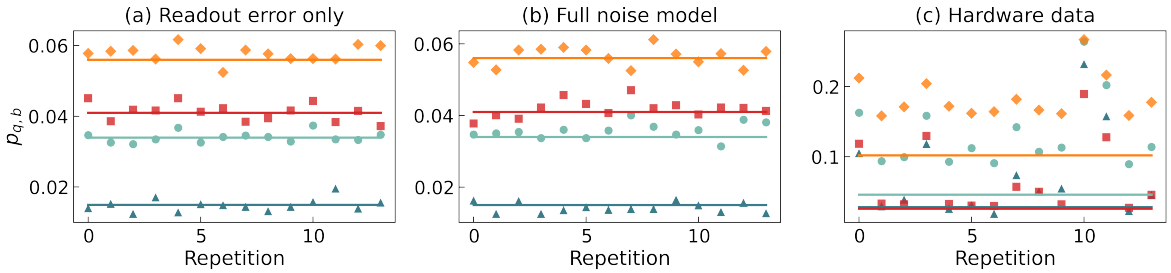


Figure 3.13: *Bit-flip probabilities $p_{0,0}$ (blue triangles), $p_{0,1}$ (orange dots), $p_{1,0}$ (green squares), $p_{1,1}$ (red diamonds) for the two-qubit case measured with the calibration procedure as a function of the repetition for (a) classically simulating ibmq_burlington with readout error only, (b) the full noise model, and (c) data obtained on the hardware. The solid lines correspond to the data provided by the noise model.*

We show the data for extracting the bit-flip probabilities for the imbq_burlington and ibmq_london device obtained in our two-qubit simulations in Figure 3.12 and Figure 3.13. The bit-flip probabilities obtained from classical simulating ibmq_burlington in in Fig-

chip name	first 4 points	full range
ibmq_london	0.460	0.298
ibmq_burlington	0.405	0.217

Table 3.1: *Exponents α obtained from fitting the function $Cn^{-\alpha}$ to our hardware data for the one qubit case for the mean absolute error in Figure 3.14 after applying the correction.*

ure 3.13(a) and Figure 3.13(b) show a similar picture than the previous cases. The data show a good agreement with the values provided in the noise model. On the contrary, the real quantum device's data does not agree very well with the noise model's values. Moreover, the values for $p_{0,0}$ and $p_{0,1}$ show large fluctuations. In this case, the theoretical values for the bit-flip probabilities differ between the simulator data and the hardware data. Most noticeably, the theoretical value for $p_{1,1}$ almost doubled during the time between the classical simulations and the experiments on quantum hardware. The results for the two-qubit case on ibmq_london show fairly similar behavior to the single-qubit case. The classical simulations in panels (a) and (b) yield as expected good agreement with the noise model's values. In contrast, the data obtained on the real quantum device (Figure 3.12(c)) do not agree with the data in the noise model, in particular for $p_{0,1}$ and $p_{1,0}$. Nevertheless, the experimental data are fairly consistent and allow us to determine the bit-flip probabilities for ibmq_london reliably. Again, we see that the theoretical values differ between the panels in the upper row and the lower row differ noticeably. This disagreement is once more due to the fact that the hardware data was taken on a different day than the simulator data, and the noise model has been updated in between.

Once we have extracted the bit-flip probabilities, we can mitigate the expectation value of quantum measurements. We begin with the one qubit case. We measure $\langle\psi| Z |\psi\rangle$ for a randomly chosen $|\psi\rangle$. Starting from the initial state $|0\rangle$, we can prepare any state on the Bloch sphere by first applying a rotation gate around the x-axis followed by a rotation around the z-axis. Hence, we choose the following circuit:

$$|0\rangle \;\boxed{R_x(\theta_0)}\;\boxed{R_z(\theta_1)}\;\boxed{\mathcal{M}} \tag{3.58}$$

We are interested in the scaling of the absolute error, defined as:

$$|\,\langle\psi|\,\tilde{Z}\,|\psi\rangle_{\text{measured}} - \langle\psi|\,Z\,|\psi\rangle_{\text{exact}}\,| \tag{3.59}$$

The absolute error and the standard deviation of our measurements are plotted as a function of the number of shot s. In Figure 3.14 the results for quantum simulation for the one qubit case are presented.

Correcting for the readout error yields a significant improvement and considerably reduces the mean and standard deviation of the absolute error. We can fit our data with a power law. While for a small number of shots around $n < 500$ we observe an exponent of about $1/2$, for a larger number of measurements, the curve for the corrected result starts to flatten out,

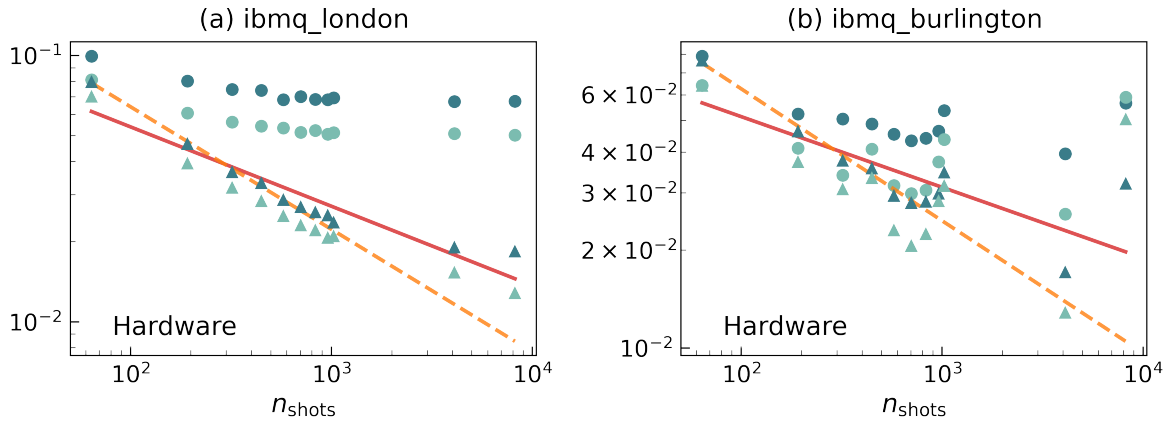
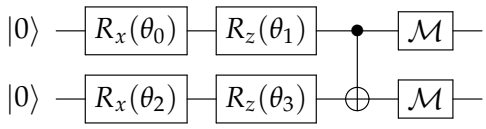


Figure 3.14: *Mean value (dark green) and standard deviation (light green) of the absolute error in Equation 3.59 after applying the correction procedure (triangles) and without it (dots) as a function of the number of shots n_{shots}. The solid red line corresponds to a power law fit to all our data points for the mean absolute error, the orange dashed line to fit including the lowest four number of shots. Different panels correspond to single-qubit data obtained on quantum hardware (a) ibmq_london and (b) ibmq_burlington.*

and the exponent obtained from the fit over the entire range is considerably smaller than $1/2$ (see Table 3.1 for details). Since increasing n should decrease the inherent statistical fluctuations of the projective measurements, and readout errors can be dealt with in our scheme, this might indicate that in addition to readout errors, other noise sources play a significant role. Their effects cannot be corrected with our procedure and thus dominate in the regime of a large number of shots.

We repeat the same procedure we did previously for the one qubit experiments but instead now we use a circuit of two qubits. Since we assume the bit-flip probabilities $p_{q,b}$ (with $q = 1, 2$, $b = 0, 1$) of the qubits to be independent of each other, we apply the same procedure that we used to obtain the bit-flip probabilities in the single-qubit case, but this time for each qubit individually.

After the calibration procedure, we prepare a two-qubit state using the following circuit:



where the angles $\theta_0, \theta_1, \theta_2, \theta_3$ are random numbers drawn uniformly from $[0, 2\pi]$, and the final CNOT gate allows for creating entanglement between the two qubits. As for the single-qubit case, we first simulate the quantum hardware classically before carrying out our experiments on an actual quantum device. In both cases we measure the noisy expectation value of $Z_2 \otimes Z_1$, $\mathbb{E}(\tilde{Z}_2 \otimes \tilde{Z}_1)$, and apply Equation 3.36 to correct for noise caused by readout errors. Again, we repeat the procedure for 1050 randomly chosen sets of angles and compute

chip name	first 4 points	full range
ibmq_london	0.478	0.390
ibmq_burlington	0.105	0.047

Table 3.2: *Exponents α obtained from fitting the power law $Cn^{-\alpha}$ to our hardware data for the two qubit case for the mean absolute error in Figure 3.15.*

the mean and the standard deviation of the absolute error defined as:

$$\text{Absolute error} = |\langle\psi|\tilde{Z}_2 \otimes \tilde{Z}_1|\psi\rangle_{\text{measured}} - \langle\psi|Z_2 \otimes Z_1|\psi\rangle_{\text{exact}}| \tag{3.60}$$

In Figure 3.15 we present the results for quantum simulations with two qubits. The absolute error and the standard deviation of the expectation value defined in Equation 3.60 are plotted as a function of the number of shot s.

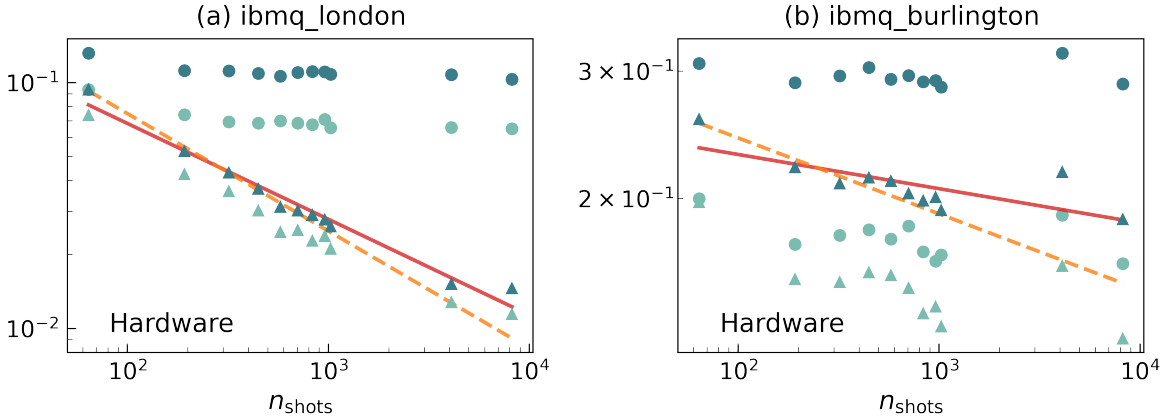


Figure 3.15: *Mean value (dark green) and standard deviation (light green) of the absolute error after applying the correction procedure (triangles) and without it (dots) as a function of the number of shots n_{shots}. The solid red line corresponds to a power law fit to all our data points for the mean absolute error, the orange dashed line to fit including the lowest four number of shots. Different panels correspond to two-qubit data obtained on quantum hardware (a) ibmq_london and (b) ibmq_burlington.*

We have been able to significantly reduce the mean of the absolute error and its standard deviation by correcting our data according to Equation 3.36. We gain a more significant improvement in the two-qubit experiments with respect to the one-qubit experiments. In particular, for our largest number of shots $s = 8192$, the mean and the standard deviation of the absolute error are reduced by approximately one order of magnitude. A power law again well describes the corrected data. By fitting the first 4 data points, we obtain an exponent of 0.48. Using the entire range of n shots for the fit, the exponent only decreases moderately to 0.39 (see also Tab. 3.2), thus showing that the readout error has still a significant contribution to the overall error. Our results for ibmq_london in Figure 3.15(a) show a qualitative agreement with the classical simulation. From the results for ibmq_burlington in Figure 3.15(b), we see that the data for this chip is significantly worse compared to ibmq_london. The mean value (standard deviation) of the absolute error

without applying any correction procedure is roughly a factor 3 larger than the one obtained on ibmq_london. Applying the correction procedure still yields an improvement, however, this time it is a lot smaller than for ibmq_london, as a comparison between Figure 3.15(a) and Figure 3.15(b) shows. While for a small number of shots, the absolute error's mean value after correction still shows a power-law decay, although with an exponent a lot smaller than 1/2. For a large number of shots, this trend stops, as fits of our data reveal (see also Tab. 3.2). This behavior is indicating that for ibmq_burlington the readout error is not the dominant one. Also in this case, other source of errors seem to have a significant contribution that we cannot correct for using our scheme.

3.2.6. Discussion and conclusions

This section presents a mitigation procedure that can mitigate bit-flip errors for any number of qubits, bit flip probabilities, and any operator. The computational overhead is moderate. Let us consider the transversal Ising Hamiltonian. The Hamiltonian, defined in Equation 3.57, has a number of operators that grow linearly with the number of qubits. The mitigation routine, for the term in the Hamiltonian in the Z computational basis $\propto Z_i \otimes Z_{i+1}$ and the term in the X computational basis $\propto X_i$, does not include any computational overhead compared to the naive measurement of the expectation value of the Hamiltonian. For n-local Hamiltonians, the individual Pauli terms do not act on all N qubits but on a given number $Q \leq n \leq N$ of qubits, independent of the total number N of qubits. For example, the Ising model and the Heisenberg model exhibit at most two-qubit interaction terms and thus have $Q \leq n = 2$. We now need to replace each term of the Hamiltonian of the Q non-identity Pauli matrices by up to 2^Q operators for our mitigation method. For each $Q \leq n$ we can estimate the total number using the upper bound $\sum_{Q \leq n} \binom{N}{Q} \leq (n+1)N^n$. Each of the matrices ω are triangular and can be inverted with a computational cost of $\mathcal{O}(4^n)$, which is constant and independent of the total number of qubits. We can put an upper bound on the overall computational complexity that is $\mathcal{O}(N^n)$. This computational complexity implies that the computational cost scales polynomially in the number of qubits N. Note that for n-local interactions between *adjacent* qubits, the computational complexity gets even further reduced, as for the transversal Ising model. Our mitigation routine also allows for pre-processing mitigation. We can evaluate the bit-flip corrected Hamiltonian and recover the bit-flip corrected expectation value of the original Hamiltonian instead of post-process the data. For example, we consider the Ising Hamiltonian $\mathcal{H}_{ZZ}$:

$$\mathcal{H}_{ZZ} = J \sum_{i=1}^{N} Z_i Z_{i+1} \tag{3.61}$$

Using the definition of the parameter $\gamma(O^i)$ in Equation 3.27, we can construct the bit-flip-corrected Hamiltonian $\mathcal{H}_{ZZ}^{\text{cor}}$ as:

$$\mathcal{H}_{ZZ}^{\text{cor}} = J \sum_{i=1}^{N} \frac{1}{\gamma(Z_i)\gamma(Z_{i+1})} \left[Z_i Z_{i+1} - \gamma(\mathbb{1}_i) Z_{i+1} - \gamma(\mathbb{1}_{i+1}) Z_i + \gamma(\mathbb{1}_{i+1})\gamma(\mathbb{1}_i) \right] \tag{3.62}$$

The noisy expectation value of $\mathcal{H}_{ZZ}^{\text{cor}}$ with respect to the bit-flip probability is equivalent to the expectation value of $\mathcal{H}_{ZZ}$ with respect to the quantum mechanical bit-flip probability. This Hamiltonian pre-processing routine can be of great advantage to VQS simulations and ease optimization problems.

3.3. Circuit expressivity

In variational quantum simulations and, more generally, quantum computing, it is essential to develop quantum circuits that are as expressive as possible. With the expressivity of a circuit, we mean the number of states it can generate. In this section, we develop a method to study and optimize the expressivity of quantum circuits. This method aims to identify and, in principle, remove superfluous parameters in a parametric quantum circuit. We will present the method and show a hardware-efficient way to recognize those parameters. We also present quantum simulations on IBMQ quantum computers that help to demonstrate the efficiency of our method. Another great tool for building an efficient quantum circuit is exploiting the symmetries of the system we want to study. In this section, we demonstrate how to include, remove or use those symmetries. Variational quantum circuit are at the center of variational quantum simulations [Peruzzo et al., 2014; McClean et al., 2016; Kandala et al., 2017; Hempel et al., 2018; Kokail et al., 2019].

The design of an efficient quantum ansatz is fundamental for the performance of variational quantum simulations. In NISQ devices, we are limited to very little circuit depth. We thus want to make the quantum simulations as efficient as possible regarding the number of gates. In this regard, we want to avoid as much as possible the presence of redundant parametric gates in our quantum circuit. The field of the design of parametric quantum circuits is deeply studied in the literature, see [Geller, 2018; Sim et al., 2019; Bataille, 2020; Sim et al., 2020; Rasmussen et al., 2020; Hubregtsen et al., 2020; Schuld et al., 2020; Fontana et al., 2020; Gard et al., 2020; Barron et al., 2020; Kim et al., 2020].

For example, in [Sim et al., 2019] the authors have proposed identifying the circuit expressivity by analyzing its ability to reach every point of the Hilbert space. The authors of [Kim et al., 2020] have demonstrated that, with a perfect quantum computer, one over parametrized quantum circuit can lead to an excellent approximation of ground states. This approach's applicability has two significant downsides in quantum computers nowadays. First, the proposed overparameterization of the quantum circuit is unfeasible due to quantum noise in the machines. Secondly, the number of quantum gates presented would annihilate any quantum advantage. In this work, we focus on the minimization of parametric quantum

gates. It is possible to develop an algebraic study of the entanglement effect of $CNOT$ and $SWAP$ gates [Sim et al., 2019] and minimize the number of nonparametric gates like in [Bataille, 2020]. We can see our work as complementary to [Bataille, 2020].

In this work we identify a quantum circuit as an operator that acts on a given initial fixed state in the Hilbert space to identify superfluous parameters of a quantum circuit. Thus, we can consider parametric quantum circuits as a map from the set of parameters to a subset of the quantum device's state space. In many cases, the set of reachable states by the quantum circuit forms a submanifold of the quantum device's state space. The (dimensional) expressivity of the quantum circuit can be understood as the dimension of this manifold of reachable states. We will also call this manifold the circuit manifold. In the ideal case scenario, we have that the submanifold's dimension generated by the quantum circuit is equal to the number of parameters of the circuit.

The goal of this section is to develop a method that allows for the identification of superfluous parameters in parametric quantum circuits. We organize this section as follows: first, we introduce the mathematical background for the dimensional expressivity analysis of quantum circuits. Then, we propose a hardware efficient implementation of the algorithm and provide proof of principle demonstration of the method's applicability with simulations on quantum hardware. We then study how to implement symmetries of the model we are investigating into quantum circuits.

3.3.1. Circuit manifold and dimension analysis

Given an initial state $|\psi_0\rangle$, a list of non-parametric gates, a list of parametric gates whose parameters can be grouped in a vector $\vec{\theta}$ and a certain ordering structure of those gates, we can define a parametric quantum circuit as the map $\mathcal{C}$:

$$\mathcal{C} := \vec{\theta} \rightarrow \left|\psi(\vec{\theta})\right\rangle \tag{3.63}$$

The map $\mathcal{C}$ is defined as a map from the parameter space $\vec{\theta}$ to the quantum device state space $\mathcal{S}$. We can assume that the parameter space $\vec{\theta}$ is a compact manifold. The compactness property is valid for the rotational gates where the parameter space is defined in:

$$\vec{\theta} \in (\mathcal{R}/2\pi\mathcal{Z})^N \tag{3.64}$$

Where N is the number of variational parameters. We can also assume that the state space $\mathcal{S}$ is a compact manifold. Since the parametric quantum states are elements of the Hilbert space $\mathcal{H}$, $\mathcal{S}$ is automatically a submanifold of $\mathcal{H}$. We also notice that the parametric quantum circuit is a continuously differentiable map if the basic parametric gates we use are the set of rotation gates $\{R_x, R_y, R_z\}$. The set of reachable states of $\mathcal{C}$ is a submanifold of $\mathcal{S}$. We define this manifold as the circuit manifold $\mathcal{M}$. $\mathcal{M}$ is the image of $\mathcal{C}$ with respect to the

domain $\vec{\theta}$ defined in Equation 3.64.

One of this study's goals is to identify and eliminate the dependent parameters in the parametric quantum circuit, i.e., which of the parameters $\{\theta_1, \ldots, \theta_N\}$ are not necessary to generate $\mathcal{M}$. We want to express the dependence (independence) of the parameters of the variational circuit locally. We can achieve this by looking at the tangent space of $\mathcal{M}$. The tangent space of $\mathcal{M}$ is locally spanned by the tangent vectors of the map. Those vectors are the partial derivative of the map $\mathcal{C}$ with respect to the parameters $\vec{\theta}$: $\{\partial_1\mathcal{C}(\vec{\theta}), \ldots, \partial_N\mathcal{C}(\vec{\theta})\}$. A parameter θ_k is redundant if $\partial_k\mathcal{C}(\vec{\theta})$ can be written as a linear combination of $\partial_j\mathcal{C}(\vec{\theta})$ with $j \neq k$. We can construct an inductive procedure to determine which parameters are redundant. Let us start with θ_1:

– θ_1 is always an independent parameter if the parametric gate on which θ_1 depends is non-trivial.

Given $\{\theta_1, \ldots, \theta_k\}$ independent parameters we can analyze θ_{k+1}:

– θ_{k+1} is a dependent parameter if $\partial_{k+1}\mathcal{C}(\vec{\theta})$ is a linear combination of $\{\partial_1\mathcal{C}(\vec{\theta}), \ldots, \partial_k\mathcal{C}(\vec{\theta})\}$

If $\partial_{k+1}\mathcal{C}(\vec{\theta})$ is not a linear combination of $\{\partial_1\mathcal{C}(\vec{\theta}), \ldots, \partial_k\mathcal{C}(\vec{\theta})\}$, θ_{k+1} is not a dependent parameter. Since the parameter space is a real space (see Equation 3.64), we need to check the linear independence with respect to real coefficients. This procedure allows us to identify the redundant parameters in our circuit. At this point, we can either remove the quantum gate associated with this parameter or keep the quantum gate and keep the parameter fixed to a constant during the optimization. If we remove a superfluous quantum gate, we gain efficiency in the simulation. On the other hand, non-parametric gates can help us to create and explore different circuit manifolds. For example, if we perform our quantum expressivity analysis during a variational quantum simulation, we rather keep the parametric gate to a fixed value and optimize the other directions rather than remove the gate and start from scratch again. We define the real partial Jacobian J_k for $\{\theta_1, \ldots, \theta_k\}$ as:

$$J_k = \begin{pmatrix} | & & | \\ \mathrm{Re}\,\partial_1 C & \cdots & \mathrm{Re}\,\partial_N C \\ | & & | \\ & & \\ | & & | \\ \mathrm{Im}\,\partial_1 C & \cdots & \mathrm{Im}\,\partial_N C \\ | & & | \end{pmatrix} \tag{3.65}$$

If J_k has full rank, all the vectors are linear independent. This means that all the parameters are linearly independent. The computational cost of classically simulating these vectors grows exponentially with the number of qubits. To mitigate this scaling problem, we take the square of the matrix:

$$S_k := J_k^* J_k \tag{3.66}$$

S_k is a $k \times k$ matrix. S_k has the same rank as J_k. It is sufficient to check the invertibility of S_k to check the vector's linear independence of J_k. S_k is invertible if the determinant is non zero: $\det S_k \neq 0$. In the following part of the section, we explicitly show few examples of how the dimensional analysis just introduced works.

One qubit and two independent parameters We can use these properties to check the dependence of the $k+1^{\text{th}}$ parameter with respect to the first k parameters in our inductive procedure. The computation of the determinant of S_k is not numerically advisable, as we will be demonstrated later in this section. This computation is challenging because we can run almost instantly into numerical instability.

We note that S_k is a positive matrix: $S_k \geq 0$. Every positive semidefinite matrix has only positive or zero eigenvalues. We assume the first $k-1$ parameters to be independent. Therefore, we need to compute only the smallest eigenvalue of S_k to check if the determinant of S_k is non-zero. In practical applications, we need to fix a threshold value ϵ from which we decide that $\lambda_k \geq \epsilon$ implies that the determinant is non zero. The value of ϵ depends on the accuracy of the hardware.
We start the practical presentation of the procedure by applying the dimensional expressivity analysis to a circuit composed of two parametric gates and one qubit.

$$\mathcal{C}(\vec{\theta}) = R_z(\theta_2)R_x(\theta_1)\,|0\rangle \tag{3.67}$$

In the coordinates of the Hilbert space $\mathbb{C}^2$, $C(\theta)$ can be represented as:

$$\mathcal{C}(\vec{\theta}) = \begin{pmatrix} \cos\frac{\theta_1}{2}\cos\frac{\theta_2}{2} - i\cos\frac{\theta_1}{2}\sin\frac{\theta_2}{2} \\ -i\sin\frac{\theta_1}{2}\cos\frac{\theta_2}{2} + \sin\frac{\theta_1}{2}\sin\frac{\theta_2}{2} \end{pmatrix} \tag{3.68}$$

We can compute $\partial_1 C$ and $\partial_2 C$ as:

$$\partial_1\mathcal{C}(\vec{\theta}) = \frac{1}{2}\begin{pmatrix} -\sin\frac{\theta_1}{2}\cos\frac{\theta_2}{2} + i\sin\frac{\theta_1}{2}\sin\frac{\theta_2}{2} \\ -i\cos\frac{\theta_1}{2}\cos\frac{\theta_2}{2} + \cos\frac{\theta_1}{2}\sin\frac{\theta_2}{2} \end{pmatrix} \tag{3.69}$$

$$\tag{3.70}$$

$$\partial_2\mathcal{C}(\vec{\theta}) = \frac{1}{2}\begin{pmatrix} -\cos\frac{\theta_1}{2}\sin\frac{\theta_2}{2} - i\cos\frac{\theta_1}{2}\cos\frac{\theta_2}{2} \\ i\sin\frac{\theta_1}{2}\sin\frac{\theta_2}{2} + \sin\frac{\theta_1}{2}\cos\frac{\theta_2}{2} \end{pmatrix} \tag{3.71}$$

We need to compute the Jacobian of the circuit $\mathcal{C}$. First, we notice that, for the Jacobian to have rank 1, there should exist two real number α and β such that:

$$\alpha\partial_1\mathcal{C}(\vec{\theta}) + \beta\partial_2\mathcal{C}(\vec{\theta}) = 0 \tag{3.72}$$

This equation has no solution for any value of $\vec{\theta}$. We thus already know that the Jacobian of $\mathcal{C}$ has rank two. We can still pursue the analysis to give an instructive example. Following

equation Equation 3.65, we can write the Jacobian of the circuit map $\mathcal{C}$ as:

$$J_2(\vec{\theta}) = \frac{1}{2}\begin{pmatrix} -\sin\frac{\theta_1}{2}\cos\frac{\theta_2}{2} & -\cos\frac{\theta_1}{2}\sin\frac{\theta_2}{2} \\ \cos\frac{\theta_1}{2}\sin\frac{\theta_2}{2} & \sin\frac{\theta_1}{2}\cos\frac{\theta_2}{2} \\ \sin\frac{\theta_1}{2}\sin\frac{\theta_2}{2} & -\cos\frac{\theta_1}{2}\cos\frac{\theta_2}{2} \\ -\cos\frac{\theta_1}{2}\cos\frac{\theta_2}{2} & \sin\frac{\theta_1}{2}\sin\frac{\theta_2}{2} \end{pmatrix} \tag{3.73}$$

In this formulation, we split the real and the imaginary parts as in Equation 3.65. We can study the form of J_2 for a set of values of $\vec{\theta} = \{0, 0\}$:

$$J_2(0,0) = \frac{1}{2}\begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & -1 \\ -1 & 0 \end{pmatrix} \tag{3.74}$$

Since it is clear that the rank of $J_2(0,0)$ is 2, we can conclude that in a neighborhood of $\vec{\theta} = \{0,0\}$ the two parameters are independent. If we construct the matrix $S_2(\vec{\theta}) = J_2^*(\vec{\theta})J_2(\vec{\theta})$, we can use the properties of sine and cosine and demonstrate, after a bit of algebra, that:

$$S_2(\vec{\theta}) = \frac{1}{4}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{3.75}$$

One qubit and two dependent parameters We can now study a trivial example to illustrate how the dimensional reduction works in a case where we can compute everything explicitly. In this case, the parameters of the circuit are dependent. The quantum circuit we want to study is:

$$\mathcal{C}(\vec{\theta}) = R_x(\theta_2)R_x(\theta_1)\,|0\rangle \tag{3.76}$$

The Jacobian of the circuit is:

$$J_2(\vec{\theta}) = \frac{1}{2}\begin{pmatrix} -\sin\frac{\theta_1}{2}\cos\frac{\theta_2}{2} - \cos\frac{\theta_1}{2}\sin\frac{\theta_2}{2} & -\cos\frac{\theta_1}{2}\sin\frac{\theta_2}{2} - \sin\frac{\theta_1}{2}\cos\frac{\theta_2}{2} \\ 0 & 0 \\ 0 & 0 \\ -\cos\frac{\theta_1}{2}\cos\frac{\theta_2}{2} + \sin\frac{\theta_1}{2}\sin\frac{\theta_2}{2} & \sin\frac{\theta_1}{2}\sin\frac{\theta_2}{2} - \cos\frac{\theta_1}{2}\cos\frac{\theta_2}{2} \end{pmatrix} \tag{3.77}$$

In this case it is again trivial to compute the S_1 and S_2 matrices. We note that:

$$S_1(\vec{\theta}) = \frac{1}{4} \tag{3.78}$$

Since the S_1 has one positive eigenvalue, as expected, the parameter θ_1 is non-trivial and independent. For θ_2, the situation is different. The explicit form of S_2 is:

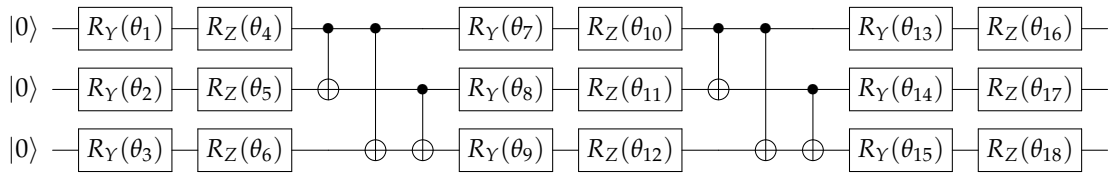


Figure 3.16: *QISKIT's* `EfficientSU2` *2-local circuit with* $N = 2$ *for three qubits.*

$$S_2(\vec{\theta}) = \frac{1}{4}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \tag{3.79}$$

S_2 has eigenvalues $\{0, \frac{1}{2}\}$. This means that S_2 is not invertible, and θ_2 does not increase the dimension of the manifold $\mathcal{M}$. We can fix the value of the parameter θ_2 to a constant. If we choose this constant to be 0, we can remove the gate associated with the parameter θ_2. This statement is true because $R_x(0) = \mathbb{1}$. We can now pass to a more instructive and practical example.

QISKIT's EfficientSU2 circuit analysis We analyze QISKIT's [Abraham et al., 2019] EfficientSU2 2-local circuit `EfficientSU2(3, reps=N)`. This circuit comprises $N + 1$ blocks of R_Y and R_Z gates applied to every qubit. These blocks are alternated with N blocks containing CNOT(q, q') gates for all $q < q'$. In the definition of the method `EfficientSU2(3, reps=N)` in [Qiskit Aer API documentation and source code] we can tune many parameters to the problem one is studying. We leave all the parameters at their default value except for the number of repetitions N. For example, setting $N = 2$, the method produces the circuit in Figure 3.16.

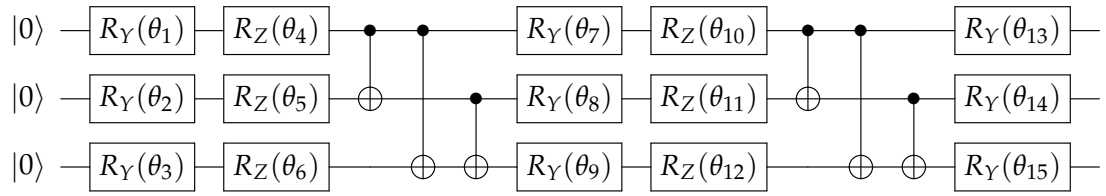


Figure 3.17: *Reduction of QISKIT's* `EfficientSU2` *2-local circuit with* $N \geq 2$ *to a maximally expressive circuit for three qubits using the minimal number of parameters.*

From a dimension analysis of the manifolds, it is clear that there are dependent parameters in the circuit in Figure 3.16. Let us call the number of qubits n. We need $2^{n+1} - 1 = 15$ real numbers to represent the Hilbert space generated by three qubits. Since we have eighteen free parameters, we already know that three parameters are redundant, but we do not know which ones. For an eighteen-dimensional space, a graphic representation of the matrices

S_k is no longer helpful for understanding the procedure. Here we give a summary of the results of the dimensional analysis. If we perform dimensional expressivity analysis, we get the reduced circuit in Figure 3.17. The difference between the two circuits is that we have just removed the last three gates. The first fifteen parameters are all independent, and the last three are dependent. Since this circuit is maximally expressive, it can generate any arbitrary product state. Let us suppose we want to run a Variational quantum simulation routine.

It is often good to start VQS simulations with a random product state. Given the circuit in Figure 3.17, it is non-trivial to generate an arbitrary product state. One would need to adjust the first 12 parameters to have a non-entangled state. This circuit is not the only one we can construct that is maximally expressive. We can choose any ordering of the parameters during the dimensional expressivity analysis. If we prioritize the last parameters, i.e., in the analysis, we first analyze the parameters $\{13, \ldots, 18\}$ and then the parameters $\{1, \ldots, 12\}$, we get a different quantum circuit. This different circuit is shown in Figure 3.18 and has useful features. The circuit in Figure 3.18 is still maximally expressive and allows for an easy generation of arbitrary product states. If we set the first nine parameters to zero, the effects of the *CNOT* gates are null. To generate an arbitrary product state, we have to fix the first nine parameters to zero and tune only the last six parameters. This circuit can generate any arbitrary product state.

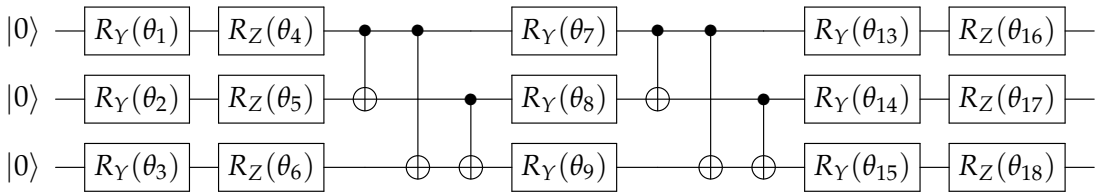


Figure 3.18: *Reduction of QISKIT's* `EfficientSU2` *2-local circuit with $N \geq 2$ to a maximally expressive circuit for three qubits using the minimal number of parameter but using a different ordering with respect to Figure 3.17.*

3.3.2. Efficient hardware implementation

The dimensional expressivity analysis (DEA) presented in the previous section works for any circuit, any number of parameters, and any qubit number. On the other hand, the practical application of this analysis is very challenging. These difficulties arise if we want to have a classical implementation of the DEA. The classical resources required for the application of the DEA are prohibitive. The storage of the matrix J_k requires the storage of a number of double-precision numbers $\propto 2^{n+1} \times k$, where n is the number of qubits [II]. This

[II] We do not need in principle to store the whole matrix. On the other hand, if we do not store the matrix, we would have to compute this matrix on the fly during the DEA and still have an exponential cost in the number of qubits.

approach is then doomed to fail in the region of about fifty qubits. We can overcome this problem with a hybrid quantum-classical implementation of the dimensional expressivity analysis. We propose to use the quantum computer to measure the matrix $S_k = J_k^* J_k$ as:

$$S_k = \frac{1}{4}\begin{pmatrix} S_{k-1} & A_k \\ A_k^* & c_k \end{pmatrix} \tag{3.80}$$

The matrix S_k is a $k \times k$ matrix. The largest value of k is equal to the number of parametric quantum gates in the quantum circuit. The magnitude of k is independent of the number of qubits N and, in all practical applications, does not have exponential scaling. One very common goal of quantum computing is to overcome exponential scaling of the computational cost in the simulation of many-body quantum systems. A circuit with an exponentially large number of parametric quantum gates still suffers from this problem and is not considered in this work. In this section, we have only considered local operators.

The storage of the matrix S_k requires the storage of $\mathcal{O}(k^2)$ double-precision numbers. The invertibility of S_k can be checked classically with $\mathcal{O}(k^3)$ CPU calls. The computations of the matrix elements of S_k require $\mathcal{O}(k^2\epsilon^{-2})$ QPU calls.

The value of ϵ depends on the acceptable noise level we require for the precision in evaluating the elements of $\{S_k\}_{i,j}$. Thus, the overall cost is polynomial in the number of parameters k and independent of the number of qubits. Of course, the number of parameters necessary for a variational quantum simulation is not independent of the qubit's number.

The more qubits we have implies that the Hilbert space we want to study is bigger, and thus we need more parametric quantum gates to study it. We do not address this problem deeply here. The dimension of the Hilbert space grows exponentially with the number of qubits. Thus, to fully represent all the possible states, we need an exponentially large number of parametric gates to cover the whole space. However, in most cases, the representation of the whole Hilbert space is not necessary. For example, if we want to study the ground-state properties of local Hamiltonians, we need to explore a small corner of the Hilbert space in which we know the entanglement is small [Eisert, 2013]. In subsection 4.2.2 we give an exhaustive demonstration of this concept in the context of tensor network simulations. This implies that, for this specific case, we do not need an exponentially large number of quantum gates to reach a precise enough solution to our problem. Moreover, the dimensional expressivity analysis computational cost is independent of the number of qubits.

The matrix element of S_k are given by real part of the scalar product of the derivative of the quantum circuit with respect to the parameters θ_m and θ_n as:

$$\{S_k\}_{m,n} = \langle \mathrm{Re}\,\partial_m C | \mathrm{Re}\,\partial_n C\rangle + \langle \mathrm{Im}\,\partial_m C | \mathrm{Im}\,\partial_n C\rangle = \mathrm{Re}\,\langle \partial_m C | \partial_n C\rangle \tag{3.81}$$

In this analysis we suppose that the parametric gates are generalized rotations gates:

$$R_G(\theta) = e^{-\frac{i}{2}\theta G} \tag{3.82}$$

Where G is an hermitian operator. We can have, for example, the usual rotation gates:

$$R_i(\theta) = e^{-\frac{i}{2}\theta\sigma_i} \tag{3.83}$$

or more complex two-qubit gates like the generic parameteric $XX^{i,j}$ gate:

$$R_{XX}^{i,j}(\theta) = e^{-\frac{i}{2}\theta\sigma_x^i\sigma_x^j}. \tag{3.84}$$

In this case, i and j do not need to be nearest neighbor qubits. The fundamental assumption that we make is that the derivative of the quantum circuit C with respect to those gate rotations can be written with just one gate insertion in C:

$$\frac{\partial C}{\partial\theta_i} = \gamma_i \tag{3.85}$$

γ_i has the same structure as C with just one more additional gate. Let us give a practical example. If we consider the one qubit circuit:

$$C(\vec{\theta}) = R_x(\theta_2)R_y(\theta_1)\left|0\right\rangle \tag{3.86}$$

We can derive the circuits γ_i as:

$$\begin{aligned}\gamma_1 &= R_x(\theta_2)\sigma_y R_y(\theta_1)\left|0\right\rangle \\ \gamma_2 &= \sigma_x R_x(\theta_2)R_y(\theta_1)\left|0\right\rangle\end{aligned} \tag{3.87}$$

The matrix element of S_k can thus be written as the real part of the scalar product of the quantum circuits:

$$\{S_k\}_{i,j} = \mathrm{Re}\left\langle\gamma_i|\gamma_j\right\rangle \tag{3.88}$$

Using the setup and circuit assumptions laid out above, we can measure the matrix elements $\{S_k\}_{i,j}$ with the same number of qubits plus an ancillary qubit and insertion of controlled gates. For example, if we consider the circuit $C(\theta_1,\theta_2)$ defined as:

$$C(\theta_1,\theta_2) = R_Z(\theta_2)R_X(\theta_1)\left|0\right\rangle. \tag{3.89}$$

We can derive γ_1 and γ_2 as:

$$\gamma_1 = R_Z(\theta_2)XR_X(\theta_1)\left|0\right\rangle \tag{3.90}$$

$$\gamma_2 = ZR_Z(\theta_2)R_X(\theta_1)\left|0\right\rangle \tag{3.91}$$

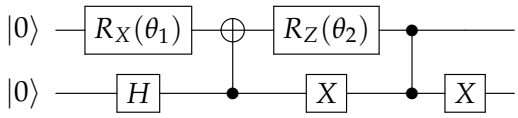
In particular, we note that for all such circuits C, the diagonal elements of S_k are given by

$$\mathrm{Re}\,\langle \partial_n C, \partial_n C\rangle = \frac{1}{4}. \tag{3.92}$$

In order to compute $\mathrm{Re}\,\langle \gamma_m, \gamma_n\rangle$ with $m \neq n$ on a quantum device, we need to construct, as explained in [Zhao et al., 2019], the state:

$$|\phi_{m,n}\rangle = \frac{|0\rangle \otimes |\gamma_m\rangle + |1\rangle \otimes |\gamma_n\rangle}{\sqrt{2}}. \tag{3.93}$$

The construction of this state requires an ancillary qubit and insertion of controlled gates CG_j after R_{G_j} where CG_m is controlled by the ancilla being $|0\rangle$ and CG_n is controlled by the ancilla being $|1\rangle$. The former can be achieved using a standard "control if $|1\rangle$" gate CG_j if it is conjugated with X gates on the ancilla, i.e., inserting $X_C G_j X$ after R_{G_j}. For example, the state $|\phi_{2,1}\rangle$ for $C(\theta_1, \theta_2)$ defined in Equation 3.89 is obtained using the following circuit:



Once the state $|\phi_{m,n}\rangle$ is prepared, we can apply a final Hadamard gate on the ancilla. This circuit produces the state

$$H_{\text{anc}}\,|\phi_{m,n}\rangle = \frac{|0\rangle \otimes (|\gamma_m\rangle + |\gamma_n\rangle) + |1\rangle \otimes (|\gamma_m\rangle - |\gamma_n\rangle)}{2} \tag{3.94}$$

and measuring the ancilla yields

$$p(\text{anc} = 0) = \frac{(\langle\gamma_m| + \langle\gamma_n|)\,(|\gamma_m\rangle + |\gamma_n\rangle)}{4} = \frac{1 + \mathrm{Re}\,\langle\gamma_m, \gamma_n\rangle}{2} \tag{3.95}$$

for the ancilla being measured in $|0\rangle$. Returning to the two gate circuit in Equation 3.89, we obtain

$$\mathrm{Re}\,\langle\gamma_2, \gamma_1\rangle = 2p(\text{anc} = 0) - 1, \tag{3.96}$$

retrieving $p(\text{anc} = 0)$ from the circuit shown in Figure 3.19.

In general, with this procedure, we can compute the matrix S_k using $\mathcal{O}(k^2\epsilon^{-2})$ QPU calls. The quantum algorithm part only requires an additional ancilla qubit to the starting qubits, two Hadamard gates, two X gates, and two controlled gates CG depending on the specific rotation gates R_G used in the quantum circuit. Now that we know how to compute the matrix elements of S_k, we can present the details on the implementation of the inductive procedure on the hardware. We can see that:

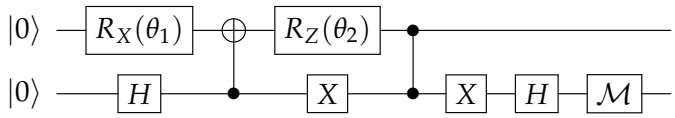


Figure 3.19: *Circuit to compute* $\mathrm{Re}\,\langle \partial_2 C, \partial_1 C\rangle$ *using an ancilla qubit (lower quantum wire) for the circuit in Equation 3.89. The gate with the* $\mathcal{M}$ *symbol indicates the measurement procedure.*

$$S_k = \begin{pmatrix} S_{k-1} & A_k \\ A_k^* & \frac{1}{4} \end{pmatrix} \tag{3.97}$$

with A_k defined as follow:

$$A_k := J_{k-1}^* \begin{pmatrix} \mathrm{Re}\,\partial_k C \\ \mathrm{Im}\,\partial_k C \end{pmatrix} = \begin{pmatrix} \frac{1}{4}\,\mathrm{Re}\langle \gamma_1, \gamma_k\rangle \\ \vdots \\ \frac{1}{4}\,\mathrm{Re}\langle \gamma_{k-1}, \gamma_k\rangle \end{pmatrix} \tag{3.98}$$

With this definition, we can state that S_k is invertible if and only if the following relation holds:

$$A_k^* S_{k-1}^{-1} A_k - \frac{1}{4} \neq 0 \tag{3.99}$$

This is strictly true if we assume $S_k - 1$ to be invertible. Noting that J_{k-1} has independent columns, we conclude that the Moore-Penrose pseudo-inverse[III] J_{k-1}^+ is a left-inverse of J_{k-1} and satisfies

$$J_{k-1}^+ = S_{k-1}^{-1} J_{k-1}^* \tag{3.100}$$

Furthermore, we obtain

$$A_k^* S_{k-1}^{-1} A_k = \begin{pmatrix} \mathrm{Re}\,\partial_k C \\ \mathrm{Im}\,\partial_k C \end{pmatrix}^* J_{k-1} J_{k-1}^+ \begin{pmatrix} \mathrm{Re}\,\partial_k C \\ \mathrm{Im}\,\partial_k C \end{pmatrix} \tag{3.101}$$

Since J_{k-1}^+ is only a left-inverse (unless in the case which will become the termination condition of the DEA), the central $J_{k-1}J_{k-1}^+$ will not reduce to the identity. However, if J_{k-1}^+ is a right-inverse as well, then we can write:

$$\frac{1}{4} - A_k^* S_{k-1}^{-1} A_k = \frac{1}{4} - \begin{pmatrix} \mathrm{Re}\,\partial_k C \\ \mathrm{Im}\,\partial_k C \end{pmatrix}^* \begin{pmatrix} \mathrm{Re}\,\partial_k C \\ \mathrm{Im}\,\partial_k C \end{pmatrix} = \frac{1}{4} - \frac{1}{4}\langle \gamma_k, \gamma_k\rangle = 0. \tag{3.102}$$

The condition proposed in Equation 3.102 holds until the dimension of the circuit, and the

[III] The Moore-Penrose pseudo-inverse of a matrix A is defined as a matrix A^+ that satisfies $AA^+A = A$, $A^+AA^+ = A^+$, $(AA^+)^* = AA^+$, and $(A^+A)^* = A^+A$.

dimension of the image $\mathcal{M}$ ($\mathcal{M}$ is the image of $\mathcal{C}$ with respect to the domain $\vec{\theta}$) are equal. If, for a given $\tilde{k}$, the condition in Equation 3.102 is not satisfied, we also know that all the parameters with $k \geq \tilde{k}$ are also dependent parameters and can be removed (or fixed to a constant). All operations are performed classically apart from the computation of the expectation of $\frac{1}{4}\,\mathrm{Re}\langle\gamma_l, \gamma_k\rangle$. Both independent and redundant parameters are generated for pedagogical reasons. We can assume that, in evaluating those matrix numbers, we make an error of order ϵ since the matrix elements $\mathrm{Re}\langle\gamma_i, \gamma_j\rangle$ come from a stochastic measurement. This means that the matrix S_k is estimated as $S_k + \delta_k$. Every matrix element of δ_k is smaller in absolute value than ϵ. If we choose to test S_k for invertibility by computing its eigenvalues, then we can use the eigenvalue stability theorem:

$$|\mathrm{error}(\lambda)| \leq \|\delta_k\|_F \leq k \leq N \tag{3.103}$$

for all eigenvalues λ of S_k, where

$$\|T\|_F := \sqrt{\sum_{i,j} |T_{i,j}|^2} \tag{3.104}$$

denotes the Frobenius norm[IV] of the matrix T. This implies that we can distinguish the smallest eigenvalue from zero. An eigenvalue $\lambda_{\min}$ can be considered zero inside machine precision if:

$$\lambda_{\min} \leq k\epsilon. \tag{3.105}$$

We do not need to worry about negative eigenvalues since the matrix S_k is semi-positive definite. However, we can still confuse an eigenvalue with being non-zero instead of zero if the tolerance ϵ is too big. We can solve this problem by increasing the precision of the evaluation of the matrix elements $\mathrm{Re}\langle\gamma_i, \gamma_j\rangle$. We remind the reader that the computational cost of the evaluation of the matrix elements at finite precision ϵ grows as $\mathcal{O}(\epsilon^{-2})$. This implies that the error scales as $\sim 1/\sqrt{m}$, where m is the number of QPU calls. Since we work on quantum computers, our method needs to be resistant to perturbations given by the quantum device's noise. We can have two different approaches to this study: study if a parameter depends on the others or if it is independent of the others. For the parameter dependence test, we need to test if $\det(S_k) = 0$. For the parameter independence test we need to verify if $\lambda_{\min} > k\epsilon$ will be satisfied. This is true if we can sufficiently reduce the noise level ϵ. Both of these tests ask whether some mathematical object is an element of a zero-measure set. Once we introduce the quantum device's perturbations, the probability of positively identifying a dependent parameter as being dependent is zero. We need noise-resistant criteria to discern if an eigenvalue is zero or not. This means that the identification

[IV] The Frobenius norm is also known as the Hilbert-Schmidt norm and is sometimes called Euclidean norm.

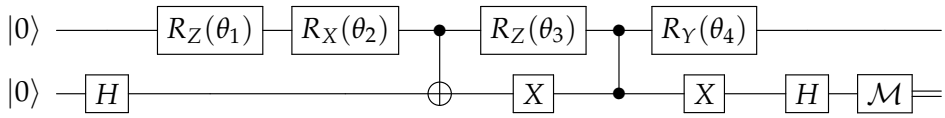


Figure 3.20: *Circuit for obtaining* $\mathrm{Re}\langle\gamma_2,\gamma_3\rangle$ *on quantum hardware for the ansatz in Equation 3.106. The ancillary qubit is initially put into superposition and acts as a control for the additional CNOT and CZ gates. After applying a Hadamard again on the ancilla, a final measurement reveals the probability for the qubit to be in state* $|0\rangle$*, which is related to* $\mathrm{Re}\langle\gamma_2,\gamma_3\rangle$.

criteria we deploy must discern if a value is zero or non-zero for a given level of noise. If we do not have this criterion, we could erroneously identify parameters as dependent instead of independent or vice-versa.

Hardware simulations To demonstrate our method's applicability, we have tested the algorithm proposed on different IBMQ quantum computers using QISKIT SDK [Qiskit Aer API documentation and source code]. Given a quantum circuit C, we will compute on the quantum hardware the matrix element $\{S_k\}_{i,j}$ inductively for any k using Equation 3.97. It is always important to remember that S_1 is non-zero for a non-trivial gate. As an instructive example, we consider the single-qubit circuit:

$$C(\theta_1,\theta_2,\theta_3,\theta_4) = R_y(\theta_4)R_z(\theta_3)R_x(\theta_2)R_z(\theta_1)\,|0\rangle\,. \tag{3.106}$$

The circuit in Equation 3.106 has four parameters for one qubit. The first R_z can generate any arbitrary global phase in the quantum state. The second and the third gates generate any rotation in the Block sphere. It is thus clear that the fourth parameter needs to be dependent. This implies that, for our analysis, the matrix S_4 must have three non-vanishing eigenvalues and one null eigenvalue.

To check if this is indeed the case, we choose various random parameter sets $(\theta_1,\theta_2,\theta_3,\theta_4)$, where each angle is drawn uniformly from the interval $[0,2\pi)$, and evaluate the spectra of the S_k matrices (note that S_1 is always $1/4$ and thus we do not need to measure it). To construct the matrices S_k, we measure the vector A_k, defined in Equation 3.98, on the quantum device. An example for a circuit allowing us to obtain one of the entries of A_k for our ansatz in Equation 3.106 is shown in Figure 3.20. The data for the low lying spectrum of the matrix S_k are shown in Table 3.3 and Figure 3.21. From the exact solutions, it is clear that S_4 has a vanishing eigenvalue. This implies that θ_4 is a dependent parameter. This is in agreement with the discussion above. Following the data for the exact solution in Table 3.3, we see clearly that for the set of parameters presented in Figure 3.21(a) Figure 3.21(b) and, Figure 3.21(c), S_3 has non-vanishing eigenvalues. This is not exactly true for Figure 3.21(d).

In the plots, it seems that the smallest eigenvalue of S_3 is compatible with zero. This is not true if we look at the exact solution. The smallest eigenvalue of S_3 is of order $\sim 10^{-3}$. This magnitude is big enough to distinguish it from zero. By construction, S_k has a single

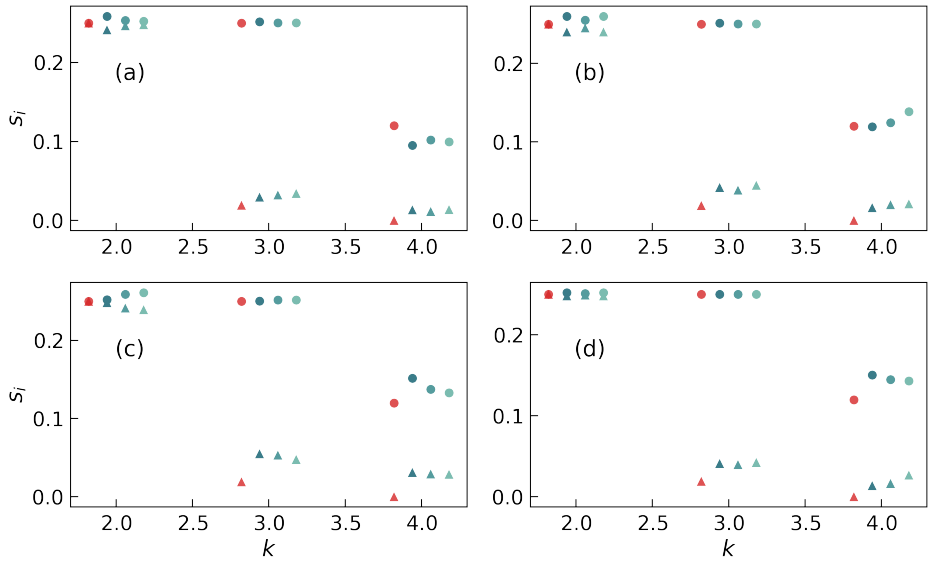


Figure 3.21: *Smallest (triangles) and second smallest (dots) eigenvalues of the matrices S_k, $k \geq 2$, for the circuit in Equation 3.106. The different panels each correspond to a different, randomly drawn parameter set $\theta_1, \ldots, \theta_4$. The red markers indicate the exact solution, the green colored markers indicate the results obtained from ibmq_vigo with 1000 measurements (darkest green), 4000 measurements (green) and 8000 measurements (lightest green). The data points for each k are slightly horizontally shifted for better visibility.*

vanishing eigenvalue if and only if θ_k is the first redundant parameter in the circuit. Our data also show that, apart from the parameter set close to a singular one, there is a clear separation between the smallest and the second smallest eigenvalue for S_k, where k is greater than 3. Hence, even if one can only resolve the spectra of the S_k matrices to a finite precision (e.g., due to a finite number of measurements or noise on a real device), we can still resolve the physical information we are interested in.

We can still find good agreement between the exact solutions and the simulations performed on the IBMQ devices looking at the hardware data. Due to the IBMQ quantum computer's noise, some eigenvalues are shifted with respect to the exact solutions. We also note that the eigenvalue's magnitude does not show a strong dependence on the number of shots. We can have consistent measurements already with 1000 shots. The effect of the

	(a)	(b)	(c)	(d)
s_2 exact	0.250	0.250	0.250	0.250
s_2 ibmq	0.247	0.243	0.248	0.242
s_3 exact	0.105	0.021	0.068	0.002
s_3 ibmq	0.115	0.034	0.082	0.022
s_4 exact	0	0	0	0
s_4 ibmq	0.022	0.018	0.012	0.007

Table 3.3: *Numerical values for the smallest eigenvalues S_k matrices for the exact case and the hardware results obtained with 8000 shots. The different columns correspond the different parameter sets shown in the panels of Figure 3.21.*

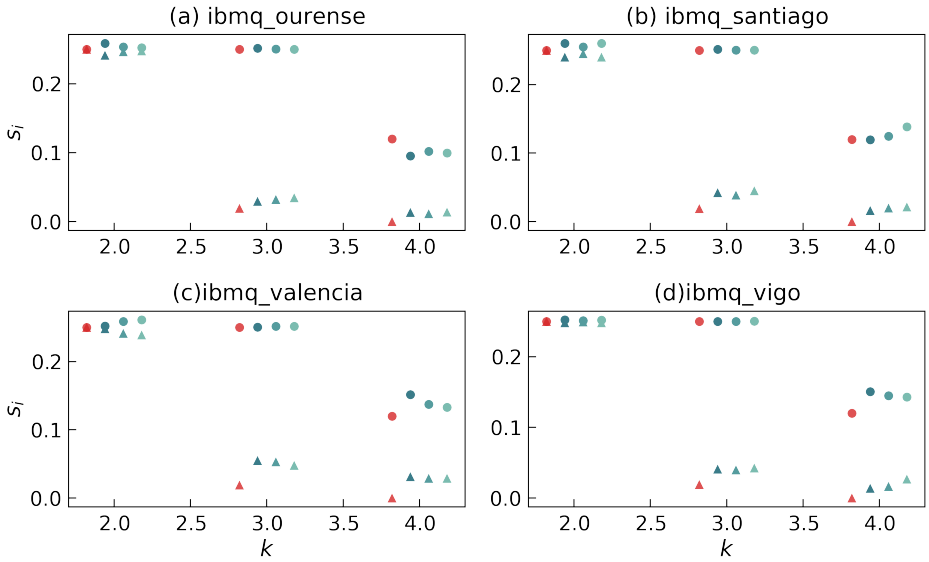


Figure 3.22: *Smallest (triangles) and second smallest (dots) eigenvalues of the matrices S_k, $k \geq 2$, for the circuit in Equation 3.106. The different panels correspond to a randomly drawn parameter set evaluated on the different hardware backends ibmq_ourense (a), ibmq_santiago (b), ibmq_valencia (c), and ibmq_vigo (d). The red markers indicate the exact solution, the green colored markers indicate the results obtained from ibmq_vigo with 1000 measurements (darkest green), 4000 measurements (green), and 8000 measurements (lightest green). The data points for each k are slightly horizontally shifted for better visibility.*

noise is to make the identification of the redundancy of the parameter θ_4 difficult. As we can see from Table 3.3, the measurement of the value of the smallest eigenvalue of S_4 is never exactly zero. Looking at Figure 3.21(b) and Figure 3.21(d), we see that these shifts in the spectrum might affect the determination of the redundant parameter. In Figure 3.21(d), the shifts due to noise lead to a clear separation between the smallest eigenvalues of S_3 and S_4, although the parameter set chosen is close to a singular one. On the contrary, in Figure 3.21(b), the effects of the noise cause the smallest eigenvalues of S_3 and S_4 to slightly deviate from zero and become more similar (see Table 3.3). As a result, the noise in current quantum devices might lead to misidentifying the superfluous parameter in some instances. We will discuss the effect of misidentification later in the section. Nevertheless, considering all our hardware results for the different parameter sets, the smallest eigenvalue of S_4 is consistently the closest to zero. Thus, despite the noise in ibmq_vigo, we can confidently identify θ_4 as a superfluous parameter.

Up to this point, we have explained the results extensively for ibmq_vigo. To assess the DEA's performance on different quantum hardware, we repeat the single-qubit experiment using the circuit in Equation 3.106 for randomly drawn parameter sets on different chips. Our results are shown in Figure 3.22 and Table 3.4. We can see only minor differences between the four hardwares analyzed here. The data from ibmq_ourense (Figure 3.22(a)), imbq_santiago (Figure 3.22(b)), ibmq_valencia (Figure 3.22(c)) and ibmq_vigo (Figure 3.22(d)) are in good agreement with the exact solution. Moreover, the shift of the values of the different eigenvalues is compatible between different devices. We also see a not strong

	(a)	(b)	(c)	(d)
s_2 exact	0.250	0.250	0.250	0.250
s_2 ibmq	0.245	0.238	0.238	0.242
s_3 exact	0.132	0.132	0.132	0.132
s_3 ibmq	0.123	0.132	0.120	0.131
s_4 exact	0	0	0	0
s_4 ibmq	0.024	0.022	0.017	0.014

Table 3.4: *Numerical values for the smallest eigenvalues of the S_k matrices for the exact case and the hardware results obtained with 8000 shots. The different columns correspond the different parameter sets shown in the panels of Figure 3.22.*

dependence on the number of shots. As in the previous case, It seems that we have enough precision with just 1000 measurements. From those data, we can identify the parameter θ_4 unambiguously as a redundant parameter, as we expect from the theoretical study.

EfficientSU2 quantum circuit study At this point, we can move to a more difficult (but not necessarily more instructive) example. We want to analyze the QISKIT's `EfficientSU2` circuit. We focus on the `EfficientSU2` circuit for two-qubits and one repetition, $N = 1$. This circuit has 8 parameters $\theta_1, \ldots, \theta_8$ originating from two blocks of R_Y and R_Z rotations, with a CNOT gate in between. The circuit `EfficientSU2` is illustrated in Figure 3.23. Applying the dimensional expressivity analysis, we find that the first 7 parameters are independent, and the final $R_Z(\theta_8)$ is superfluous.

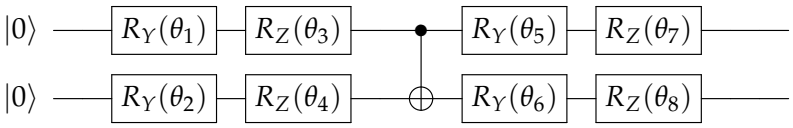


Figure 3.23: *`EfficientSU2` circuit for two-qubits and one repetition*

To check for that behavior with our algorithm, we proceed in the same way as the single-qubit examples. We draw various random parameter sets, measure the entries of the A_k vectors for those on quantum hardware and construct the S_k matrices from these data. An example of the circuit used to obtain the entries of the vector A_k is given in Figure 3.24. Afterward, we examine the spectra of the S_k matrices using a classical computer.

Our results obtained on ibmq_vigo for the low-lying spectrum of these matrices are shown in Figure 3.25 and Table 3.5.

Focusing first on the exact results, we see that it is harder, in this case, to identify the redundant parameter in the circuit unambiguously. While S_8 has as expected a vanishing eigenvalue for all the parameter combinations we study, Table 3.5 shows that the matrix S_7 has in general a very small, but non-vanishing, eigenvalue of the order of $10^{-4} - 10^{-5}$. We observe reasonable agreement in the results obtained from ibmq_vigo with the exact solution.

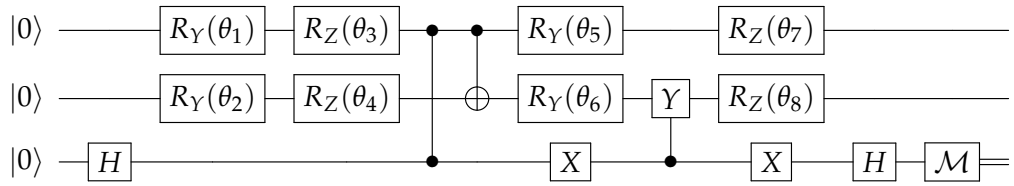


Figure 3.24: *Circuit for measuring* $\mathrm{Re}\langle\gamma_3, \gamma_6\rangle$ *on quantum hardware. The ancillary qubit is initially put into superposition and acts as a control for the additional CZ and CY gates. A final measurement on the ancilla reveals the probability for obtaining* 0 *which allows for obtaining* $\mathrm{Re}\langle\gamma_3, \gamma_6\rangle$ *according to Equation 3.95.*

Compared to the single-qubit case, the dependence on the number of measurements is slightly stronger. In general, the results deviate from the exact solution a little more (see Table 3.5). These more severe deviations are expected for the two-qubit case. The circuits necessary for computing the S_k matrices are deeper and contain an additional CNOT gate compared to the single-qubit case. Consequently, our results are more affected by noise. In particular, Figure 3.25(b) and Figure 3.25(c) reveal that noise can indeed obscure the identification of the superfluous parameter. In those cases, the smallest eigenvalues we obtain from the hardware data for S_7 and S_8 are quite small and very similar (see Table 3.5). This can create a situation where we could misinterpret S_7 as a dependent parameter. The results for the parameter sets shown in Figure 3.25(a) and Figure 3.25(d) hint that θ_8 is the superfluous parameter, however the difference between the smallest eigenvalue of S_7 and S_8 is enhanced by noise in those cases. The data obtained from ibmq_vigo still indicates that θ_8 is interdependent, but the results are a lot less precise compared to the single-qubit experiments. This picture also does not change using different hardware backends, as Figure 3.26 and Table 3.6 demonstrate.

Running the same randomly drawn parameter set on different quantum hardware, we observe somewhat similar performance. While the data from ibmq_valencia in Figure 3.26(c) seems slightly worse for the parameter set we study, none of the other backends can produce

	(a)	(b)	(c)	(d)
$\frac{s_5}{10^{-4}}$ exact	1267	1212	1955	1405
$\frac{s_5}{10^{-4}}$ ibmq	1525	1282	1544	1021
$\frac{s_6}{10^{-4}}$ exact	1.632	371.5	1496	120.9
$\frac{s_6}{10^{-4}}$ ibmq	414.0	465.3	1307	366.1
$\frac{s_7}{10^{-4}}$ exact	1.078	4.064	1.522	0.4057
$\frac{s_7}{10^{-4}}$ ibmq	256.1	242.1	263.7	257.2
$\frac{s_8}{10^{-4}}$ exact	0	0	0	0
$\frac{s_8}{10^{-4}}$ ibmq	171.9	240.7	218.5	193.2

Table 3.5: *Numerical values for the smallest eigenvalues of the* S_k *matrices for the exact case and the hardware results obtained with 8000 shots. The different columns correspond the different parameter sets shown in the panels of Figure 3.25.*

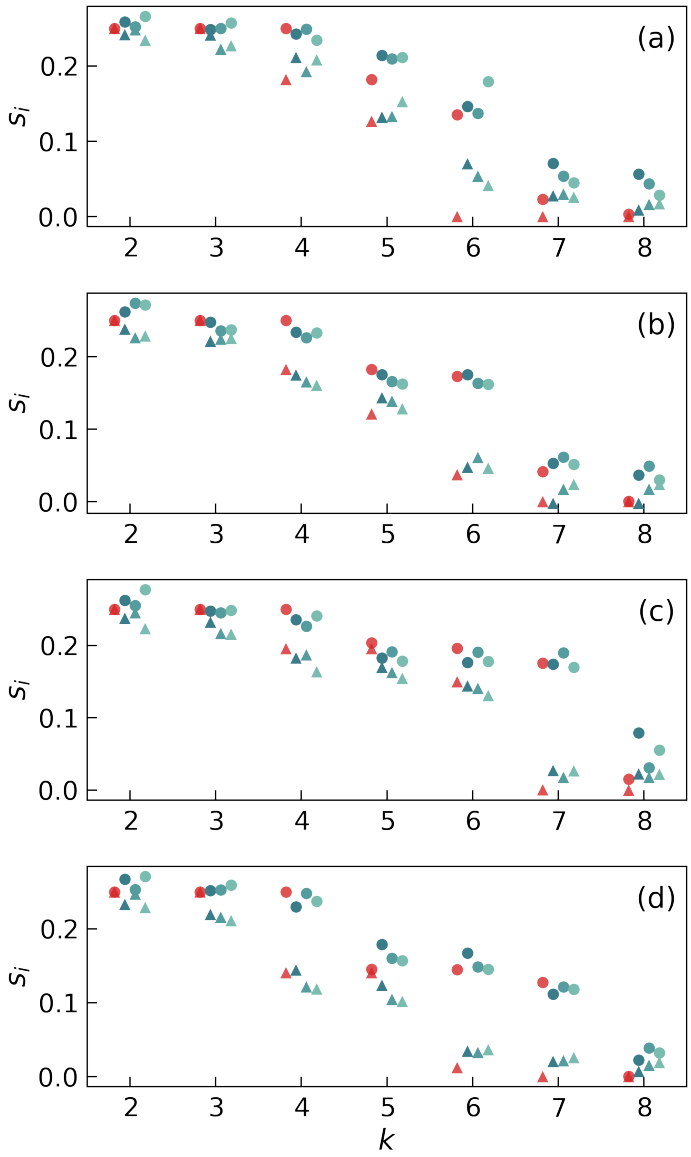


Figure 3.25: *Smallest (triangles) and second smallest (dots) eigenvalues of the matrices S_k, $k \geq 2$, for the* `EfficientSU2` *circuit for two qubits and a single repetition. The different panels each correspond to a randomly drawn parameter set $\theta_1, \ldots, \theta_8$. The red markers indicate the exact solution, the green colored markers indicate the results obtained from ibmq_vigo with 1000 measurements (darkest green), 4000 measurements (green) and 8000 measurements (lightest green). The data points for each k are slightly horizontally shifted for better visibility.*

the exact solution with notably better accuracy than the others. Again, our results give a hint that the rotation gate associated with θ_8 might be superfluous, but this is not unambiguously clear from the data, as Table 3.6 shows. Our results indicate that with the current noise levels in NISQ devices, it might not be possible to identify the circuit's independent parameters. Nevertheless, as we approach the interdependent parameters, some signals in the data might allow getting an idea of which parameters are redundant. Over-parametrization can sometimes be beneficial in optimization problems. In [Wiersema et al., 2020] the authors

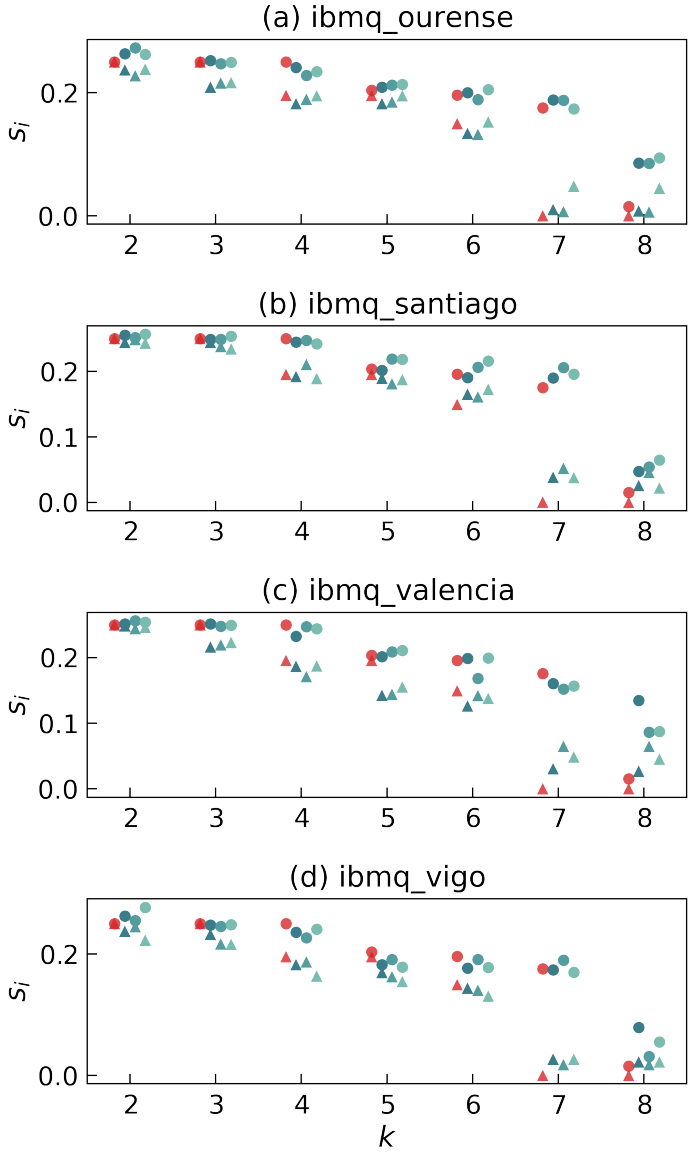


Figure 3.26: *Smallest (triangles) and second smallest (dots) eigenvalues of the matrices S_k for $k \geq 2$. The different panels correspond to a randomly drawn parameter set evaluated on the different chips (a) ibmq_ourense, (b) ibmq_santiago, (c) ibmq_valencia, and (d) ibmq_vigo. The red markers indicate the exact solution, the green colored markers indicate the results obtained from ibmq_vigo with 1000 measurements (darkest green), 4000 measurements (green) and 8000 measurements (lightest green). The data points for each k are slightly horizontally shifted for better visibility.*

have numerically observed that the optimization landscape of cost function built from over-parametrized quantum circuits and physical Hamiltonians becomes almost convex. However, in NISQ devices, this approach is still not feasible, and we need to use as few quantum gates as possible to avoid being overwhelmed by the device's noise.

	(a)	(b)	(c)	(d)
$\frac{s_5}{10^{-4}}$ exact	1955	1955	1955	1955
$\frac{s_5}{10^{-4}}$ ibmq	1949	1877	1553	1544
$\frac{s_6}{10^{-4}}$ exact	1496	1496	1496	1496
$\frac{s_6}{10^{-4}}$ ibmq	1524	1729	1381	1307
$\frac{s_7}{10^{-4}}$ exact	1.522	1.522	1.522	1.522
$\frac{s_7}{10^{-4}}$ ibmq	484.0	379.8	483.5	263.7
$\frac{s_8}{10^{-4}}$ exact	0	0	0	0
$\frac{s_8}{10^{-4}}$ ibmq	449.2	218.2	450.7	218.5

Table 3.6: *Numerical values for the smallest eigenvalues s_k of the S_k matrices for the exact case and the hardware results obtained with 8000 shots. The different columns correspond the different parameter sets shown in the panels of Figure 3.26.*

3.3.3. Implementation of physical symmetry

In this section, we study the implementation of the physical symmetries into a quantum circuit. As already discussed in section 3.1, when we talked about a translationally invariant quantum circuit ansatz, it is crucial to incorporate the model's physical symmetries into the quantum circuit we want to use. We will use the dimensional expressivity analysis to identify the quantum gates that generate certain symmetry and remove them. This procedure will leave us with a quantum circuit with fewer parameters but with the same expressivity.

In many applications of quantum computing (e.g., variational quantum simulations), the global phases of quantum states are irrelevant. Let us consider the global phase transformation of a state $|\psi\rangle$ for any real number θ:

$$|\psi\rangle \to e^{i\theta} |\psi\rangle \tag{3.107}$$

The quantum mechanical expectation value $\langle\psi| \mathcal{H} |\psi\rangle$ is invariant under this transformation [V]. A maximally expressive quantum circuit allows for the generation of any state $|\psi\rangle$ with a fixed global phase. This section wants to consider a maximally expressive quantum circuit and remove the redundant generation of any state with a fixed global phase. The removal of this redundant parameter is helpful, for example, for the variational quantum simulations classical optimization routine. It reduces the parameter space that we want to optimize classically and makes the quantum simulation more efficient.

Let us consider a generic quantum circuit $C(\theta)$. The circuit $\tilde{C}(\phi, \theta)$ is given by $C(\theta)$ with an additional $R_Z(\phi)$ gate as in Figure 3.27. The insertion of the $R_Z(\phi)$ rotation gate at the beginning of the circuit $\tilde{C}(\phi, \theta)$ adds a global phase to the state. $R_Z(\phi)$ is the generator of the symmetry we want to remove in $C(\theta)$. The quantum gate $R_Z(\phi)$ can generate any

[V] Global phases are mathematically relevant. For example, let us consider two different quantum states $|\phi\rangle$ and $|\rho\rangle$. The scalar product between these two states $\langle\phi|\rho\rangle$ depends on the global phases of the two states.

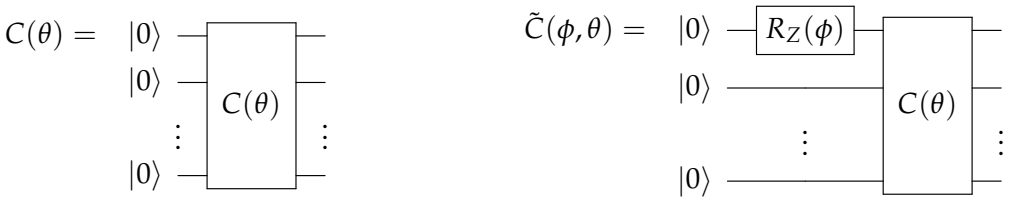


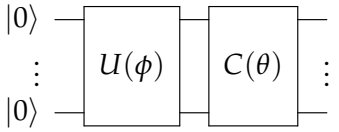
Figure 3.27: *On the left we have a schematic representation of a variational circuit $C(\theta)$. On the right we have a schematic representation of a variational circuit $\tilde{C}(\phi,\theta)$ that is equivalent to the circuit $C(\theta)$ with the insertion of a $R_Z(\phi)$ gate before the operator $C(\theta)$.*

transformation defined in Equation 3.107. The parameter ϕ is always labeled as independent in the dimensional expressivity analysis since it is the first parameter of the circuit. We suppose we have already performed the dimensional expressivity analysis on the circuit $C(\theta)$ and all the parameters in $C(\theta)$ are independent. When we analyze the parameters of $\tilde{C}(\phi,\theta)$, either all the parameters are independent, or one parameter is dependent on ϕ. Let us call this parameter θ_j. If adding $R_Z(\phi)$ to $C(\theta)$ makes θ_j a dependent parameter, the effect of θ_j is the generation of the global phase and can be removed.

We now formalize in a mathematically rigorous way how to identify these symmetries and incorporate or remove them. Let us suppose we have a quantum circuit $C(\theta)$ with $\theta = \{\theta_1, \ldots, \theta_n\}$ and the Jacobian is $C'(\theta)$. The reduced row echelon form[VI] for $C'(\theta)$ can be written as:

$$\begin{pmatrix} n\times n \\ 0_{(d-n)\times n} \end{pmatrix} \tag{3.108}$$

where d is the real dimension of the Hilbert space. We suppose that the unwanted symmetry we want to remove can be generated using a parametric gate $U(\phi)$ on some qubits to generate a global phase (e.g., for the global phase $U(\phi) = R_Z(\phi)$). Instead of analyzing C, we may analyze $\tilde{C}(\phi,\theta)$ given by the following circuit:



The dependence of U on ϕ should include at least one parameter ϕ_0 where $U(\phi_0) = \mathbb{1}$. This condition is necessary if the circuit U generates a symmetry. The dimension of ϕ depends on the dimension of the group of the symmetry it generates. For example, ϕ is of dimension one for the global symmetry generation ($U(1)$ group) or eight-dimensional for an $SU(3)$ symmetry. To proceed with this analysis, we need to consider the circuit manifold's local properties under a transformation of a compact group. A circuit is subject to a global

[VI] A matrix being in row echelon form means that Gaussian elimination has operated on the rows, and column echelon form means that Gaussian elimination has operated on the columns

symmetry if and only if the symmetry's action maps the circuit manifold into itself.

It is sufficient to check this property locally for a compact group. This makes the analysis much more straightforward. We can analyze the circuit in a neighborhood of ϕ_0 and extract the global properties of the circuit. Moreover, for this case, it sufficient to check the properties of independence on a single point to obtain independence on a neighborhood of the point. Since we plan to use the dimensional expressivity analysis on an extended circuit $\tilde{C}(\phi, \theta)$, this means that it is sufficient to check for parameter independence at ϕ_0 satisfying $U(\phi_0) = \mathbb{1}$ and sufficiently many θ. Therefore, we can identify the parameters generating a symmetry by looking at parameters that are shown as independent under dimensional expressivity analysis for the circuit C but dependent when analyzing $\tilde{C}$. For simplicity, we assume a one-dimensional symmetry again. Let us suppose that the symmetry is not generated in the circuit $U(\theta)$. One possible outcome for the reduced row echelon form of $\tilde{C}'(\phi_0, \theta)$ would be:

$$\begin{pmatrix} \phi_0 & \theta \\ 1 & 0 \\ 0 & \\ 0_{(d-n-1)\times 1} & 0_{(d-n-1)\times n} \end{pmatrix} \begin{matrix} \\ \\ n\times n \\ \\ \end{matrix}. \tag{3.109}$$

for almost all of the parameters θ. If the symmetry is generated by the quantum circuit $U(\theta)$, there exists one parameter θ_m such that the outcome for the reduced row echelon form of $\tilde{C}'(\phi_0, \theta)$ can be written as:

$$\begin{pmatrix} \phi & \theta_1, \ldots, \theta_{m-1} & \theta_m & \theta_{m+1}, \ldots, \theta_n \\ 1 & 0 & a & 0 \\ 0 & & b & 0 \\ 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{3.110}$$

where a, b, both $\mathbb{1}$, and each 0 are matrices/vectors of the appropriate dimension. a is non-zero, and the vector b may or may not be zero. If $b = 0$, then θ_m has locally the same effect as U. The gate corresponding to θ_m thus locally only contributes the unwanted symmetry. In general, b is expected to be non-zero. In this case, the gate corresponding to θ_m locally generates the unwanted symmetry. We can remove the unwanted symmetry by setting the parameter θ_m to a constant value. In general, it is not possible to remove directly the gate corresponding to the parameter θ_m. For example, if the dependence on θ_m in the quantum circuit U is given by $R_Y(\theta_m)$and $\theta_m = \pi$, we would remove a Y gate from the circuit U and modify the manyfold we are studying.

While it is not clear a priori whether or not the same θ_m is always responsible for the additional symmetry, it is at least locally always true. In other words, if we find that the m^{th} parameter (θ_m) generates an unwanted symmetry at a point θ, then there exists

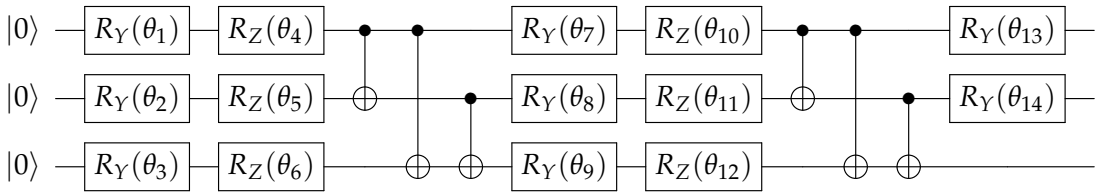


Figure 3.28: *Reduction of QISKIT's* `EfficientSU2` *2-local circuit similar to Figure 3.17, but with removing the option to change a global phase.*

a neighborhood of θ such that the unwanted symmetry is always generated by the m^{th} parameter.

Example We apply this procedure to remove the global phase symmetry to the circuit `EfficientSU2(3, reps=2)`. The application of this analysis to `EfficientSU2(3, reps=2)` at random values of θ shows that for all tested values, the parameter θ_{15} adds only a global phase. This implies that we can remove the gate on which θ_{15} depends or set $\theta_{15} = 0$. In this case, we still have maximally expressivity for variational quantum simulations. We can reduce the computational cost of the circuit while keeping the physical space dimension constant. The resultant circuit is presented in Figure 3.28.

3.3.4. Translationally invariant quantum circuits and sectors

As already introduced in section 3.1, it is crucial to construct a quantum circuit embedding the symmetries of the system that we are studying. We have shown that a translational invariant quantum circuit performs much better when studying a translational invariant Hamiltonian's ground state compared to a non-translational invariant circuit. In this subsection, we study translational invariant circuits more rigorously. The translation operator is also called in the literature as momentum operator. A quantum state is translationally invariant if it is an eigenvector of the momentum operator. We denote the corresponding eigenvalue as ω. ω must be a root of unity, i.e., for N qubits, we have $\omega^N = 1$. This sub-section aims to be a general overview of the topic. For more detailed derivation and demonstrations of the concepts, we refer to [Funcke et al., 2021]. The first concept we want to present here is that the momentum operator's eigenvalues divide the Hilbert space into different sectors with a defined eigenvalue. The Hilbert space subspace with definite eigenvalue is spanned by the eigenvector associated with the corresponding eigenvalue. The subspaces are highly degenerate and are spanned by different eigenvectors. Let us define the Hilbert space's subspace spanned by the eigenvector ω as $\text{Eig}(\omega)$. The Hilbert space H can be written as the direct sum of the subspace given by the momentum operator τ:

$$\mathrm{H} = \bigoplus_{\omega} \text{Eig}(\omega) \tag{3.111}$$

First, we note that, for a state $|\psi\rangle$ to be an eigenvector of the momentum operator τ it must hold:

$$\tau |\psi\rangle = \omega |\psi\rangle . \tag{3.112}$$

For a Hilbert space defined by N qubits, it must hold:

$$\tau^N = \mathbb{1}. \tag{3.113}$$

We can understand Equation 3.113 by thinking that if we have N qubit state and we translate it N times with τ, the state should not change. Given ω to be a root of the unity of order d, it holds that, if $d \mid N$ ("d divides N"), ω is an eigenvalue of the translation operator. To compute efficiently the dimension of the momentum subspace defined by the eigenvalue ω of order d, we define recursively $\#(d)$ as:

$$\#(d) := 2^d - \sum_{\substack{d' \mid d \\ d' < d}} \#(d') \tag{3.114}$$

with $\#(1) = 2$. We also define $[\#](d)$ as:

$$[\#](d) := \frac{\#(d)}{d} \tag{3.115}$$

With these definitions we can estimate the complex dimension $\dim_{\mathbb{C}} \mathrm{Eig}(\omega)$ of the eigenspace $\mathrm{Eig}(\omega)$:

$$\dim_{\mathbb{C}} \mathrm{Eig}(\omega) = \sum_{d \mid k \mid N} [\#](k). \tag{3.116}$$

The dimension on the real space $\mathbb{R}$ is given by the mapping of the degrees of freedom from $\mathbb{C}$ to $\mathbb{R}$. The different sectors are path-connected. This means that, given τ to be the momentum operator, λ an eigenvalue of τ, $|\phi\rangle$ and $|\psi\rangle$ two normalized eigenvectors of τ with respect to λ, there always exist a path γ, inside the sector, such that $\gamma(0) = |\phi\rangle$ and $\gamma(1) = |\psi\rangle$. This means that we can use our dimensional expressivity analysis to check whether we can reach any subsector's point given a subsector and its dimension. The path connectedness property of the different sectors ensures that, if we can generate a circuit manifold with the same dimension and eigenvalue ω as the sector related to ω, we are sure that we have full expressivity in this sector. All these concepts and equations are demonstrated in [Funcke et al., 2021].

In the following part of this section we show how to compute $\dim_{\mathbb{C}} \mathrm{Eig}(\omega)$. We can have a representation of the Hilbert space given by:

$$\mathrm{H} := \{ |j\rangle \, ; \, j \in \mathbb{N}_{0,<2^Q} \} \tag{3.117}$$

where $|j\rangle$ denotes the tensor product state $|b_N(j)_{N-1} \dots b_N(j)_0\rangle$ if and only if $b_N(j) := b_N(j)_{N-1} \dots b_N(j)_0$ is the binary representation of j. For example, $b_3(5) = 101$ with $b_3(5)_2 =$

$b_3(5)_0 = 1$ and $b_3(5)_1 = 0$. The translation operator τ_N is defined as the linear operator mapping H to itself and satisfying the following relation:

$$b_N(\tau_N(j)) := b_N(j)_{N-2} \dots b_N(j)_0 b_N(j)_{N-1} \tag{3.118}$$

The different ω are the eigenvalues of the operator τ. The momentum subspaces are defined by the degenerate eigenvectors space of τ. We need the following equivalence relation on $\mathbb{N}_{0,<2^N}$ to compute the dimension of each momentum sector:

$$j \sim_N k \ :\Longleftrightarrow \ \ t \in \mathbb{N} : \ j = \tau_N^t(k). \tag{3.119}$$

We denote the equivalence class of a element j with N qubits as $[j]_N$. For simplicity, we will order the elements by value, that is:

$$[j]_N = \{j_0, \dots, j_{\aleph_N(j)-1}\} \tag{3.120}$$

where $m < n$ implies $j_m < j_n$. With this ordering we can define:

$$\omega_N(j_n) := \exp\left(2\pi i \frac{n}{\aleph_N(j_n)}\right) \tag{3.121}$$

which is a bijection between the elements of $[j]_N$ and the roots of unity of order $\aleph_N(j)$. This implies that we can construct an eigenvector $e_N(j_n)$ of τ_N for each element of each $[j]_N$ by setting

$$\tilde{e}_N(j_n) := \sum_{k=0}^{\aleph_N(j)-1} \omega_N(j_n)^k \left|\tau_N^k(j_0)\right\rangle \tag{3.122}$$

and

$$e_N(j_n) := \frac{\tilde{e}_N(j_n)}{\|\tilde{e}_N(j_n)\|_{\ell_2(N)}}. \tag{3.123}$$

For example, $[1]_2 = \{1,2\}$ implies $e_2(1) = \frac{|01\rangle+|10\rangle}{\sqrt{2}}$ and $e_2(2) = \frac{|01\rangle-|10\rangle}{\sqrt{2}}$. By construction, the $e_N(j_n)$ are independent of all $e_N(k_m)$ if $[j]_N \neq [k]_N$ because they are linear combinations of linearly independent sets. Furthermore, the $e_N(j_n)$ are independent of the remaining $e_N(j_m)$ since they are eigenvectors of different eigenvalues of τ. In other words, we have found a basis transformation between H and a basis of eigenvectors of τ. In particular, we have found a complete set of eigenvectors for τ. In Appendix B we construct all the eigenvector subspaces for $N = 4$ qubits explicitly. Moreover, we check that the Hilbert space can be written as a direct sum of the different eigenspaces of the momentum operator.

3.3.5. Discussion and conclusions

In this section, we have presented a method to analyze the expressivity of a quantum circuit. We can use this method to design parametric quantum circuits to improve, for example, variational quantum simulations. Since a classical implementation of the dimensional expressivity analysis algorithm requires a prohibitive amount of resources, we have also developed a hybrid quantum-classical implementation of the dimensional expressivity analysis. This hardware-efficient implementation of the algorithm has a computational cost that is feasible in near-term computations. We have tested and benchmarked our dimensional expressivity analysis on IBMQ quantum hardware and demonstrated its applicability. We have shown how to generate an invariant quantum circuit under a certain symmetry. Finally, we have focused on the translational invariance symmetry. We have explicitly worked out all the details of the dimensional expressivity analysis and explicitly showed how the Hilbert space could be written as a direct sum of the different sectors defined by the eigenvalues of the momentum operator.

Chapter 4

Phases at finite winding number of an Abelian Lattice Gauge Theory

Quantum simulations are an up-and-coming tool to simulate quantum field theories. However, classical simulations still play a leading role in understanding lattice gauge theories in this moment of history. In particular, in recent years, there has been a boost in the development of tensor network methods to simulate lattice gauge theories. We refer to [Bañuls et al., 2020] for a recent review of the field.

There is no unique approach or method to simulate lattice field theories using tensor networks and not just one tensor network algorithm. For example, in [Bañuls et al., 2013a] and [Bañuls et al., 2013b], the authors have successfully mapped the Swinger model Hamiltonian into a spin Hamiltonian with long-range interaction. With the use of matrix product states, the authors have extracted the ground state's fundamental properties of the theory and conducted the extrapolation to the continuum limit. Other approaches are based on the formulation of gauge-invariant tensor networks in the quantum link formulation. For example, in [Tschirsich et al., 2019] the authors have used the quantum link model formulation introduced in [Chandrasekharan and Wiese, 1997] to perform quasi (2+1) dimensional simulations of a $U(1)$ quantum spin model. Many other works have been published in recent years that address these topics (e.g. see [Silvi et al., 2014a; Montangero, 2018; Bruckmann et al., 2019; Tagliacozzo et al., 2014; Buyens et al., 2014; Silvi et al., 2014b]).

In this chapter, we do not analyze tensor-network algorithms that go beyond the 1+1 dimensional case. Very successful tensor network algorithms and ansätze in higher dimensions are, for example, the multi-scale entanglement renormalization ansätz (MERA), projected entangled pair states (PEPS) [Evenbly and Vidal, 2011], and the tensor renormalization group algorithm (TRG) (see, e.g., [Levin and Nave, 2007; Evenbly and Vidal, 2015]).

In this chapter, after a brief introduction of the tensor network algorithm used for our numerical simulations, we study the condensation phenomena associated with the stringy excitations of an Abelian lattice gauge theory. It can be considered a generalization of the well-known Bose-Einstein condensation in the context of scalar field theories coupled with a

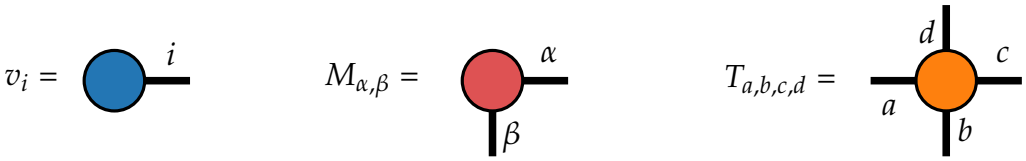


Figure 4.1: *Diagrammatic representation of different multidimentional tensors.*

chemical potential. For example, in scalar and fermionic theories, one can create ultra-local excitations. For a gauge theory, such excitations need to be closed loops that do not violate gauge invariance. The presence of string-like excitations that can spread over the entire lattice extent in a cylindrical geometry is also of great interest. These stringy excitations of a gauge theory have their own interesting dynamics, which can in principle be very different from the dynamics of point-particles in scalar theories or mesons in fermionic theories. The goal of this study is to demonstrate the presence of the string excitations in the ground state numerically. In scalar theories or fermionic theories, the expectation value of the particle number operator presents discrete jumps when the chemical potential is increased (e.g., [Bañuls et al., 2017]). When we raise the system's density, a higher particle state may have lower energy than the correspondent lower particle state. We expect to see this same behavior for string-like excitations in the here considered Abelian gauge theory. Tensor network algorithms have been chosen to simulate this model. This is motivated by the fact that, presently, there are no other numerical methods able to give a satisfying description of the properties of the ground state of the model at finite chemical potential.

This chapter is structured as follows: in section 4.1 we present the notation used for the diagrammatic representation of tensors and tensor contractions introduced by Penrose [Penrose, 1971]. In section 4.2, we present the basic concepts for understanding the matrix product state ansätz and the density matrix renormalization group algorithm. In section 4.3, we introduce the $U(1)$ quantum link model Hamiltonian. We then explore the different winding sectors of the ground state of the model in a ladder geometry at finite chemical potential.

4.1. Tensor notation

An n^{th}-rank tensor in m-dimensional space is a mathematical object that has n indices and m^n components. Tensors are generalizations of scalars (that have no indices), vectors (that have exactly one index), and matrices (that have exactly two indices) to an arbitrary number of indices. In the tensor network community, it is ubiquitous to represent tensor contraction in a diagrammatic form. In our diagrammatic form, we denote the tensors by solid shapes (e.g., square or circles) and tensor indices by lines emanating from these shapes. In Figure 4.1 we give an example of a vector v_i, a matrix $M_{\alpha,\beta}$ and a four-dimensional tensor $T_{a,b,c,d}$. We can represent tensor contraction with the following rule: connecting two index lines implies a contraction, or summation over the connected indices. In Figure 4.2 we present the

diagrammatic representation of the tensor contraction for a matrix-vector multiplication ($v_j = k_i m_{j,i}$) and the tensor contraction of two indeces between a four dimentional tensor and a three dimentional tensor ($G_{e,c,d} = J_{a,b,c,d} K_{e,a,b}$). We do not explicitly write all the

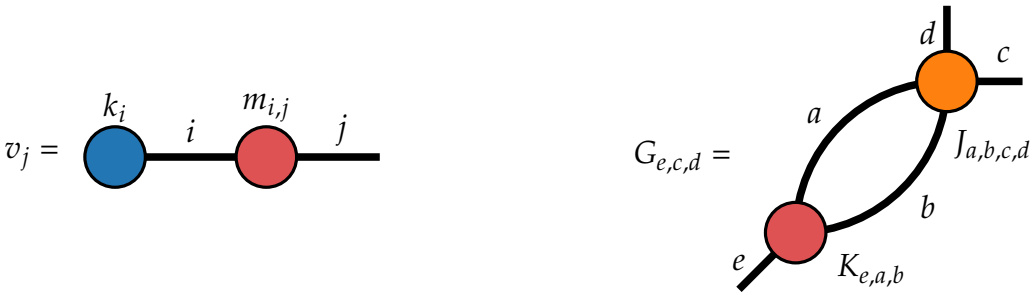


Figure 4.2: *Diagrammatic representation of two different tensor contractions.*

indexes in the diagrams and the tensors' names for simplicity when not necessary. We represent the identity (in the literature it is usually called an *isometry*) as a black line without a solid figure in the middle.

4.2. Matrix product states

The storage of a generic wave function of a quantum many-body system requires an exponentially large memory to be stored. For a spin chain of spin 1/2 and length n, we would need to store $\mathcal{O}(2^n)$ complex numbers on the computer's memory. Even the biggest supercomputer in the world cannot store a generic wave function of 60 spins. This exponential cost makes direct calculations of quantum mechanical systems unfeasible. We define the wave function of a many-body quantum state as [Schollwöck, 2011]:

$$|\psi\rangle = \sum_{s_1,\dots,s_n} c_{s_1,\dots,s_n} |s_1\rangle \otimes \dots \otimes |s_n\rangle \tag{4.1}$$

We can define $c_{s_1,\dots,s_n}$ as the contraction of tensors A^i as:

$$c_{s_1,\dots,s_n} = \sum_{\alpha,\beta,\dots,\gamma} A^1_{\alpha,\beta,s_1} A^2_{\beta,\delta,s_2} \dots A^n_{\gamma,\alpha,s_n} \tag{4.2}$$

The indexes $\alpha, \beta, \dots, \gamma$ can be contracted to recover the tensor $c_{s_1,\dots,s_n}$. This is an exact relation. We can reduce the dimension of the tensors by reducing the dimension of the greek indexes of the tensors A. This procedure and how to perform the truncation will be explained clearly later in this section. We can visualize the procedure of regularization of a wave function in Figure 4.3. On the left of the equation we have the full wavefunction and on the right the wavefunction represented as a matrix product state.
We can now compare the number of parameters in the quantum mechanical representation of the wave function and the matrix product state ansätz. We consider a many-body quantum system. We denote the local Hilbert space dimension (e.g., the spin dimension if we have

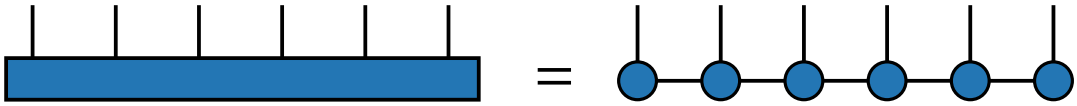


Figure 4.3: *Schematic form of the matrix product state ansätz.*

a spin chain) d. The number of lattice points is n. We then have $\mathcal{O}(d^n)$ parameters in the wave function's quantum mechanical representation. For the matrix product state we have another parameter. We set the maximal dimension of the greek indexes (i.e. α, β, ...) to be at most D (usually called *bond dimension* in the literature). The number of parameters in the matrix product state ansätz is $\mathcal{O}(dnD^2)$. The exponential scaling in the matrix product state ansätz is hidden in the bond dimension D. We need to have an exponentially large D to have an exact equivalence as in Figure 4.3. We can truncate this bond dimension by fixing a smaller maximal value of D. This truncation is at the core of the matrix product state ansätz and it can be done in an elegant and controlled fashion using the singular value decomposition (SVD) of the tensors A. The singular value decomposition decomposes a matrix as the product of three different matrices and highlights the matrix's singular values. Given a generic rectangular matrix M, it is always possible to find the matrices U, S, V such that:

$$M = U\,S\,V^{\dagger} \tag{4.3}$$

U is a matrix containing the left eigenvectors of M. Since U has orthonormal columns and thus it is also unitary $UU^{\dagger} = U^{\dagger}U = \mathbb{1}$. S is a diagonal matrix with non-negative entries. The diagonal elements of U are the singular values of M. The number of non-zero singular values is the rank of M. $V^{\dagger}$ is a matrix that contains the right eigenvectors of M. In the same way as U, $V^{\dagger}$ has orthonormal columns it is also unitary $VV^{\dagger} = V^{\dagger}V = \mathbb{1}$. Given a tensor of a matrix product state A^{i}_{α,β,s_i}, we can always define a rectangular matrix given by the fusion of two indexes. For example, we can fuse the index β and s_i in $\beta' \sim \beta * s_i$ such that the tensor A^i is a rectangular matrix. We can perform the SVD of this rectangular matrix and keep only the largest singular values. Moreover, we can remove the right and left eigenvectors of the matrix and reshape them back to the previous form. The bond dimension of the new tensor A^i will be reduced, but the biggest contribution to the wave function given by the biggest singular values will be kept. This is demonstrated rigorously in [White, 1992].

4.2.1. Canonical form

We define the matrix product state by the tensors A^i. On the other hand, the set of tensors A^i that define a generic wave function $|\psi\rangle$ are not unique. $|\psi\rangle$ is defined as the contraction of the A^i tensors. Given a element of $Gl_D(\mathbb{C})$[I] M, a matrix product state is invariant under

[I] $Gl_D(\mathbb{C})$ is called the general linear gruop of dimension D over $\mathbb{C}$ and it is the group of invertible matrices of dimention $D \times D$

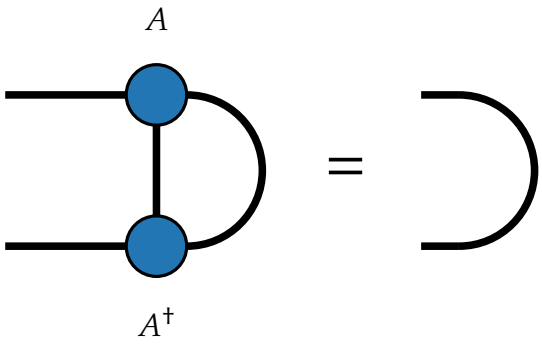


Figure 4.4: *Diagrammatic equivalence between a matrix product state contraction in the right canonical form and an isometry.*

the insertions:

$$A^{i}_{\alpha,\beta} A^{i+1}_{\beta,\gamma} = A^{i}_{\alpha,\beta} M_{\beta,\delta} M^{-1}_{\delta,\xi} A^{i+1}_{\xi,\gamma} \tag{4.4}$$

We say we are *fixing a gauge* when we fix the matrices M such that the tensors of the MPS satisfy certain relations (gauges) [Perez-Garcia et al., 2006]. Two particular gauges are the so-called *canonical forms*. They are particularly useful when computing expectation values of operators (e.g., the Hamiltonian H of a quantum system). These gauge conditions consist of choosing the matrices M such that the tensors A^i satisfy the following relations:

$$\sum_{\beta=1}^{D} \sum_{s=1}^{d} \left(A^{[s](i)}_{\alpha,\beta} \right)^{*} A^{[s](i)}_{\beta,\gamma} = \delta_{\alpha,\gamma} \tag{4.5}$$

and

$$\sum_{\beta=1}^{D} \sum_{s=1}^{d} A^{[s](i)}_{\alpha,\beta} \left(A^{[s](i)}_{\beta,\gamma} \right)^{*} = \delta_{\alpha,\gamma} \tag{4.6}$$

A matrix product state is in the *left canonical form* if it satisfies the relation in Equation 4.5. A matrix product state is in the *right canonical form* if it satisfies the relation in Equation 4.6. In Figure 4.4 we give a diagrammatic expression of Equation 4.6. We also present a very useful notation introduced by [Vidal, 2003] that highlights the singular values of a matrix product state's matrix. We can write the wave function $|\psi\rangle$ as:

$$|\psi\rangle = \sum_{s_1, \ldots, s_N} U^{s_1} S_1 U^{s_2} S_2 \ldots S_{N-1} U^{s_N} |s_1, \ldots, s_N\rangle \tag{4.7}$$

where we have at each site i a set of d matrices U^i if size D^2 and on each bond between two different sites a diagonal matrices S_i. The connection between the matrices A defined in Equation 4.2 and the matrices U and S in Equation 4.7 is given by the following relation:

$$A^{i}_{\alpha,\beta,s_i} = \left[U^{s_{i-1}}\right]_{\alpha,\alpha} \left[S_i\right]_{\alpha,\beta}. \tag{4.8}$$

Usually, during the SVD, we only keep the non-zero elements of U^{s_i-1}. This is an elegant way to regularize and reduce the dimension of the states in the wave function that we are studying.

4.2.2. Von-Neumann entropy in the MPS formalism

Entanglement plays a central role in the study of strongly correlated quantum systems since a highly entangled ground state is at the heart of a large variety of collective quantum phenomena [Osborne and Nielsen, 2002]. Quantum phase transitions occur at zero temperature and involve the appearance of long-range correlations. These correlations are not due to thermal fluctuations but to the intricate structure of a strongly entangled ground state of the system. For several 1D spin chain models both near and at the quantum critical regimes, the entanglement obeys universal scaling laws as dictated by the representations of the conformal group and its classification motivated by string theory [Calabrese and Cardy, 2009].

With a deep scaling analysis of the entanglement entropy, we can extract the central charge associated with the conformal theory that describes the universal properties of the quantum phase transition [Vidal et al., 2003]. Let us suppose we have a random state in a Hilbert space. We can cut the volume in two and compute their entanglement entropy. If we vary the system's volume, the entanglement entropy grows with the system's volume (volume law). There is one special class of state that does not follow this law. Ground states of gapped and local Hamiltonians follow an area law. In this case, the entanglement entropy increases proportionally with the area of the cut [Eisert et al., 2010]. A rigorous proof of the area law in 1+1 dimensions is given in [Hastings, 2007]. Therefore, the entanglement entropy of the ground state of a 1+1 dimensional gapped Hamiltonian does not depend on the volume and can be approximated by a constant.

We now describe how to compute the entanglement entropy in the matrix product state formalism where the von Neumann entropy is easily accessible. We also show how the bond dimension D of a matrix product state naturally induces an upper limit of the maximal entanglement entropy S of a matrix product state ($S = \mathcal{O}(\log_2 D)$).

Let us suppose we have the ground state of a gapped model on a lattice Λ described by the density matrix ρ. The von Neumann entropy is defined as:

$$S(\rho) = -Tr(\rho \log(\rho)) \tag{4.9}$$

At zero temperature, the entropy of the ground state is zero for any gapped Hamiltonian [Eisert et al., 2010]. We can not state this for a subset of the system. We define a subset A of Λ such that $B := \Lambda / A$. We define the reduced density matrix of the sub-system as:

$$\rho_A = Tr_B(\rho) \tag{4.10}$$

We define the von-Neumann entropy of the subsystem A as:

$$S(\rho_A) = -Tr(\rho_A \log(\rho_A)) \tag{4.11}$$

The entropy $S(\rho_A)$ is null if the subsystems A and B are product states. This is not true if there are quantum correlations between A and B. Quantum correlations can lead to non-vanishing values of $S(\rho_A)$. The von-Neumann entropy is also called entanglement entropy. For any partition A and B of the Hilbert space in which $|\psi\rangle$ is defined, it is always possible to write:

$$|\psi\rangle = \sum_{\alpha,\beta} c_{\alpha,\beta} |\alpha\rangle_A |\beta\rangle_B . \tag{4.12}$$

This decomposition is also called the Schmidt decomposition of the wave function $|\psi\rangle$. If we perform a singular value decomposition of the matrix c in Equation 4.12, we can write:

$$|\psi\rangle = \sum_{\alpha,\beta} \sum_{s_a} U_{\alpha,s_a} S_{s_a,s_a} V^{\dagger}_{s_a,\beta} |\alpha\rangle_A |\beta\rangle_B . \tag{4.13}$$

We can absorb U and V in A and B due to their orthonormality in those spaces and write:

$$|\psi\rangle = \sum_{s_a} s_a |\alpha'\rangle_A |\beta'\rangle_B \tag{4.14}$$

In this decomposition it is trivial to derive the reduced density matrix for the sub-system A in Equation 4.10:

$$\rho_A = \sum_{s_a} s_a^2 \left(|\alpha'\rangle \langle\alpha'| \right)_A \tag{4.15}$$

We consider a spin chain of local physical dimension d and N sites described by a MPS in the *left canonical form*. The density matrix ρ is defined as:

$$\rho = \sum_{s_1,\ldots,s_N}^{d} \sum_{s'_1,\ldots,s'_N}^{d} Tr\left[M^{1,s_1,s'_1} M^{2,s_2,s'_2} \ldots M^{N,s_N,s'_N} \right] |s_1, \ldots, s_N\rangle \langle s'_1, \ldots, s'_N| \tag{4.16}$$

where:

$$M^{i,s_i,s'_i} = A^{[s_i](i)} \otimes \left(A^{[s'_i](i)} \right)^{\dagger} . \tag{4.17}$$

Now let us partition the system into two subsystems A and B. A includes all sites up to l and B its complement. We can always rewrite a MPS in the form of Equation 4.7. From this form, it is trivial to derive the von-Neumann entanglement entropy. If we define:

$$|\alpha\rangle_A = \sum_{s_1,\ldots,s_l} U^{s_1} S_1 U^{s_2} S_2 \ldots S_{l-1} U^{s_l} |s_1, \ldots, s_l\rangle , \tag{4.18}$$

$$|\alpha\rangle_B = \sum_{s_1,\ldots,s_N} U^{s_{l+1}} S_{l+1} U^{s_{l+2}} S_{l+2} \ldots U^{s_N} S_N |s_{l+1}, \ldots, s_N\rangle . \tag{4.19}$$

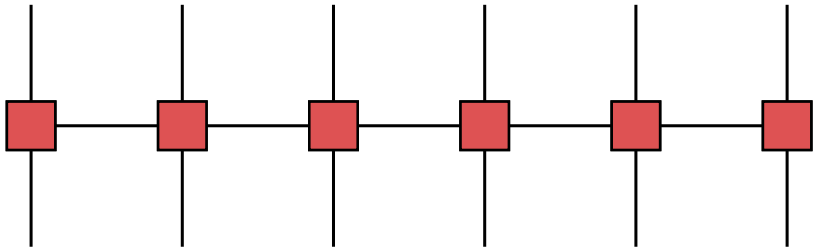


Figure 4.5: *Schematic form of matrix product operator*

We can derive the reduced density matrix ρ_A as:

$$\rho_A = \sum_{s_a} s_a^2 \left(|\alpha\rangle \langle\alpha|\right)_A = \sum_a \left(S_l\right)_{a,a}^2 \left(|\alpha\rangle \langle\alpha|\right)_A \tag{4.20}$$

where $\left(S_l\right)_{a,a}$ are the singular values of a matrix product state's matrices. Since we have the coefficients of the reduced density matrix ρ_A, we can compute the von-Neumann entropy as:

$$S(\rho_A) = -Tr(\rho_A \log_2(\rho_A)) = -\sum_a \left(S_l\right)_{a,a}^2 \log_2 \left(S_l\right)_{a,a}^2 . \tag{4.21}$$

From Equation 4.24 we can derive that, in a matrix product state, the maximal entanglement entropy of the ansätz state is $S(\rho_A) = \mathcal{O}(\log_2 D)$. We want to stress here that this quantity is trivially computable for a matrix product state.

4.2.3. Matrix product operator

Any operator $\hat{O}$ in a finite Hilbert space can be written, in the same fashion as the wave function in Equation 4.1, in the form:

$$\hat{O} = \sum_{s'_1,\dots,s'_N,s_1,\dots,s_N} c_{s'_1,\dots,s'_N,s_1,\dots,s_N} \left|s'_1,\dots,s'_N\right\rangle \left\langle s_1,\dots,s_N\right| \tag{4.22}$$

Given this definition, we can apply the same tools used to derive the matrix product state ansätz to derive the matrix product operator (MPO) associated with the operator $\hat{O}$. The matrix product operator formalism redefines the coefficients $c_{s'_1,\dots,s'_N,s_1,\dots,s_N}$ as the contraction of different matrices M^i, such that:

$$c_{s'_1,\dots,s'_N,s_1,\dots,s_N} = \sum_{\alpha,\beta,\dots,\gamma} M^1_{\alpha,\beta,s'_1,s_1} M^1_{\alpha,\beta,s'_2,s_2} \dots M^N_{\alpha,\beta,s'_N,s_N} \tag{4.23}$$

We show the diagrammatic form of a matrix product operator in Figure 4.5 From this definition we can define the matrix product operator correspondent to the operator $\hat{O}$ as:

$$\hat{O} = \sum_{s'_1,\dots,s'_N,s_1,\dots,s_N} \sum_{\alpha,\beta,\dots,\gamma} M^1_{\alpha,\beta,s'_1,s_1} M^1_{\alpha,\beta,s'_2,s_2} \dots M^N_{\alpha,\beta,s'_N,s_N} \left|s'_1,\dots,s'_N\right\rangle \left\langle s_1,\dots,s_N\right| \tag{4.24}$$

At this point, the definition of the MPO ansätz of $\hat{O}$ is equivalent to the quantum mechanical operator $\hat{O}$. This means that the computational cost of generating the matrices M^i is still exponential in the volume. We can apply the same tools to reduce the dimension of the greek indexes $\alpha, \beta, \ldots, \gamma$. We can apply a singular value decomposition to every tensor M^i and only keep the tensor's non-zero singular values. This procedure, although formally correct, is impossible to apply to practical cases due to the exponential cost in the storage of the tensor $c_{s'_1,\ldots,s'_N,s_1,\ldots,s_N}$. Usually, at least for local Hamiltonians, we can write the MPO of a Hamiltonian in a very concise way. We now give a practical example. We consider a spin 1/2 chain with the following Hamiltonian:

$$\mathcal{H} = \sum_{i=1}^{N-1} J_x \sigma_i^x \sigma_{i+1}^x + J_y \sigma_i^y \sigma_{i+1}^y + J_z \sigma_i^z \sigma_{i+1}^z + \sum_{i=1}^{N} h_z \sigma_i^z \tag{4.25}$$

where σ_i and σ_j are the Pauli matrices. The Hamiltonian in Equation 4.25 is an abbreviation. It is common to omit the tensor product of every term of the Hamiltonian with the identity for any site of the system. For example, $\sigma_i^x \sigma_{i+1}^x$ stands for $\mathbb{1}_1 \otimes \mathbb{1}_2 \cdots \otimes \mathbb{1}_{i-1} \otimes \sigma_i^x \otimes \sigma_{i+1}^x \otimes \mathbb{1}_{i+2} \cdots \otimes \mathbb{1}_N$. We need to include all the identity matrices in the matrix product operator. We can easly and explicitely derive all the matrix M for this Hamiltonian:

$$M^1 = \begin{bmatrix} h_z\sigma_z & J_x\sigma_x & J_y\sigma_y & J_z\sigma_z & \mathbb{1} \end{bmatrix} \tag{4.26}$$

$$M^N = \begin{bmatrix} \mathbb{1} \\ \sigma_x \\ \sigma_y \\ \sigma_z \\ h_z\sigma_z \end{bmatrix} \tag{4.27}$$

$$M^i = \begin{bmatrix} \mathbb{1} & 0 & 0 & 0 & 0 \\ \sigma_x & 0 & 0 & 0 & 0 \\ \sigma_y & 0 & 0 & 0 & 0 \\ \sigma_z & 0 & 0 & 0 & 0 \\ h_z\sigma_z & J_x\sigma_x & J_y\sigma_y & J_z\sigma_z & \mathbb{1} \end{bmatrix} \tag{4.28}$$

This Hamiltonian has a lot of physical applications, and we can use the same procedure for constructing the MPO of the local Hamiltonian.

Let us consider an operator $\hat{O}$ in the matrix product operator form and a many-body quantum state $|\psi\rangle$ in the matrix product state formalism. We want to compute the quantum mechanical expectation value of the operator $\hat{O}$ over the wave function $|\psi\rangle$. This expectation value can be computed very efficiently in the tensor network formalism. A diagrammatic

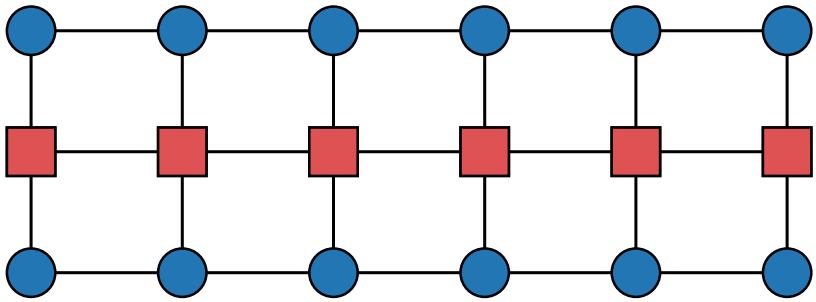


Figure 4.6: *Schematic form of the expectation value of an operator $\hat{O}$ in the matrix product operator form and a many body quantum state $|\psi\rangle$ in the matrix product state form.*

form of the expectation value is Figure 4.6. There is a very efficient way to contract all the network in such a way to have minimal computational cost. For the contraction, we start from the very first tensor of the MPS A^1, and we contract it with the first tensor of M^1 such that $A^1 M^1 = [AM]^1$. We then contract the tensor $[AM]^1$ with the tensor $\left[A^1\right]^\dagger$ and we get the tensor $\left[AMA^\dagger\right]^1$. We then contract the tensor $\left[AMA^\dagger\right]^1$ with A^2, then the resulting with M^2 and finally with $\left[A^2\right]^\dagger$. We continue with this pattern until site N. A schematic diagram of the efficient contraction is presented in Figure 4.7. To estimate the computational cost of this procedure, we again define the system's parameters. We consider the model with open boundary conditions. The matrix product state's bond dimension is at most D. The bond dimension of the matrix product operator is at most D'. The local dimension of the Hilbert space is d (two for a spin 1/2 chain). The number of lattice points is N. This type of contraction is very efficient, and the numerical cost is $\mathcal{O}(D^3 D'^2 d^2 N)$. Usually, D is much larger than the other parameters. If we consider periodic boundary conditions (which will no be studied in this work), the computational cost is $\mathcal{O}(D^5)$.

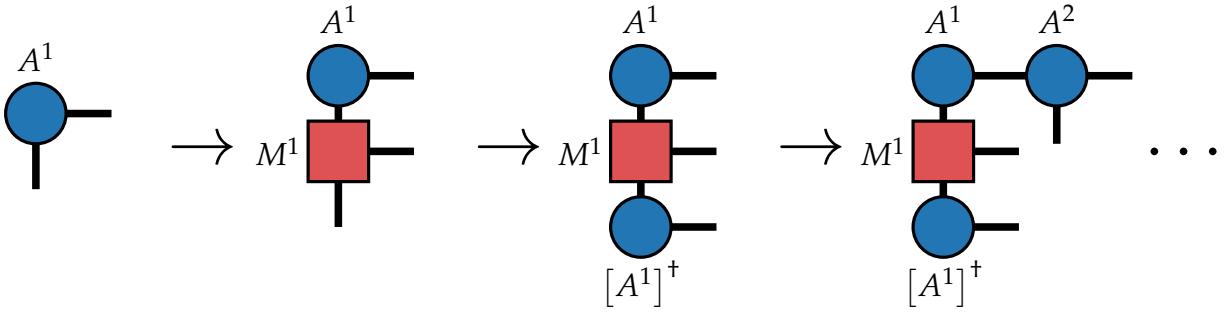


Figure 4.7: *Efficient tensor contraction between a matrix product states and a matrix product operator.*

4.2.4. Density matrix renormalization group

We can use the matrix product state as a variational ansätz for the computation of the ground state (lowest eigenvalue) of a given Hamiltonian $\mathcal{H}$. We can do this minimization at finite volume and zero temperature. In the literature, the most used variational minimization algorithm for the optimization of matrix product states is called the density matrix

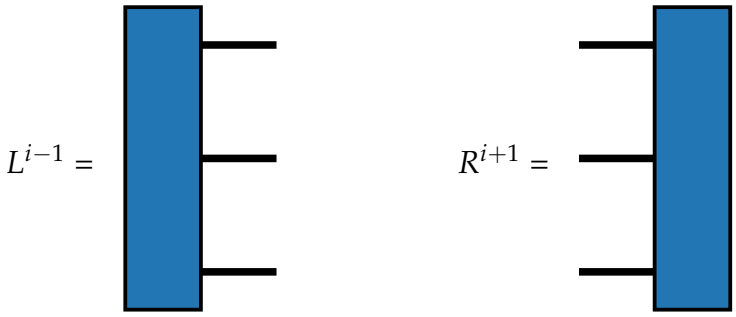


Figure 4.8: *Diagrammatic scheme of the left and right tensor environment.*

renormalization group (DMRG). Steven White firstly introduced it in [White, 1992]. At finite volume and for gapped Hamiltonians, the ground state is unique. This means that the minimum of the energy corresponds to the system's ground state. Having access to the ground state wave function (even if somehow truncated) is of great interest for physical (and, e.g., chemical) applications. From the ground state of a system, we can extract many fundamental physical properties of the system.

The problem we want to solve is a minimization problem. The tensor A^i contain the variational parameters of our minimization. The DMRG algorithm is based on local optimizations. In most cases, we observe exponential convergence of the energy as a function of the bond dimension. We consider the one-site DMRG, where we update the tensors A^i one by one. To illustrate the algorithm, we first consider the update of one tensor A^i at site i, for a generic i. If we consider the expectation value $\langle\psi|\,\mathcal{H}\,|\psi\rangle$ represented in Figure 4.6, we can fix all the tensors $A^{j\neq i}$ to a constant and minimize the expectation value of the energy with respect to the tensor A^i. The expectation value of the energy is given by:

$$E = \frac{\langle\psi|\,\mathcal{H}\,|\psi\rangle}{\langle\psi|\psi\rangle} \tag{4.29}$$

where E depends on the variational tensor A^i. The problem we want to solve is to find the minimum λ such that:

$$\langle\psi|\,\mathcal{H}\,|\psi\rangle = \lambda\,\langle\psi|\psi\rangle \tag{4.30}$$

For an efficient implementation of the algorithm, we need to build what are called in the tensor network language the left environment (L^{i-1}) and the right environment (R^{i+1}). The left environment is given by the contraction between all the tensor A^j up to site $i-1$ with the MPO tensors M^j and $\left[A^j\right]^\dagger$. Figure 4.7 explains how to perform the contraction starting from the left bond. We can perform the same kind of contraction starting from the right bond (i.e., from site N to site $i+1$). A diagrammatic form of the left and right environment is shown in Figure 4.8. We can represent the left part of Equation 4.30 as a contraction between the left environment, the tensor A^i, the tensor M^i, and the right environment. We impose the tensors A^j to be in the left gauge form for $j < i$ and in the right gauge from for $j > i$. It is always possible to impose these conditions when we have open boundary

conditions. We can map the optimization problem of the tensor A^i into a minimization problem of the map represented in Figure 4.9. This minimization problem can be solved

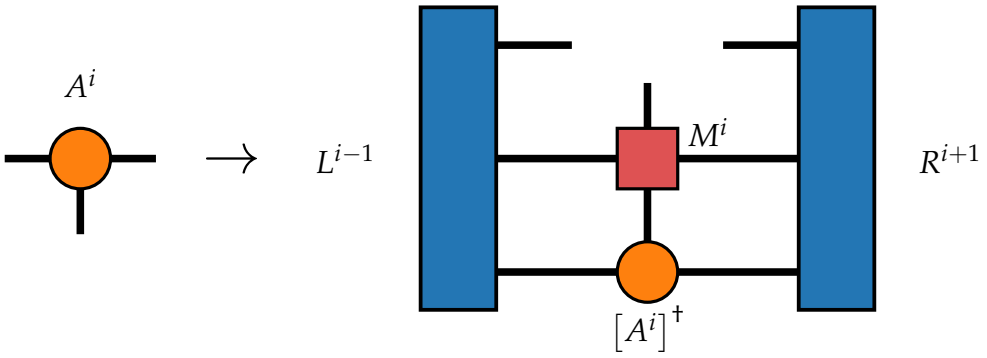


Figure 4.9: *Diagrammatic scheme of the Linear map induced by the minimization of the expectation value of the Hamiltonian. This is the most important step in the DMRG algorithm.*

using the Lanczos method [Lanczos, 1950]. Now that we have a way to update the tensor A^i, we can sweep through all tensors A until convergence. The algorithm works as follows:

- 1) Start from site 1 with a MPS in the right gauge form
- 2) Apply the update procedure
- 3) Transform A^1 in the left gauge form
- 4) Compute L^1 and R^3
- 5) proceed to update the A^2
- 6) Apply the same procedure until you reach the site N
- 7) Transform A^N into the right gauge and update the left environment.
- 8) Update A^{N-1} and continue until site 1.

When the energy, after one full update of all tensors from left to right (or right to left), do not change up to a small factor (in our numerical simulations, we usually fix the tolerance to 10^{-10}), we can say we have converged. When we reach convergence, you have the energy of the ground state and the ground state wave function. We can use the ground state wave function to compute any local observable expectation value we are interested in and extract physical quantities.

4.3. Winding number sectors analysis in the $U(1)$ quantum link model

The wave functions of gauge theories are members of continuous groups and have infinite-dimensional Hilbert space, even at finite volume. Quantum link models regulate these

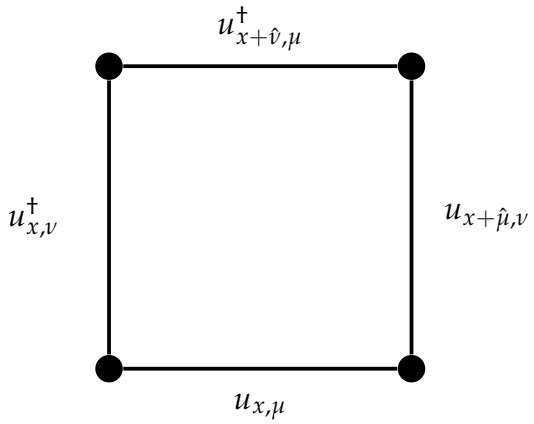


Figure 4.10: *Classical plaquette representation*

infinite-dimensional Hilbert spaces while maintaining exact gauge invariance [Horn, 1981; Orland and Rohrlich, 1990; Chandrasekharan and Wiese, 1997]. In [Brower et al., 1999] the authors have developed the quantum link model framework for an SU(N) gauge symmetry and provide a non-perturbative formulation of QCD using the domain wall formulation introduced in [Kaplan, 1992]. The U(1) quantum link model was proposed in [Rokhsar and Kivelson, 1988] on a two-dimensional square lattice in what is called a *"Quantum Dimer Model"*. This model aims at understanding quantum states or phases, which can give rise to high-temperature superconductivity. The authors of [Zheng and Sachdev, 1989; Read and Sachdev, 1990] showed that a dual theory to the U(1) quantum link model was essentially identical to models obtained from a semiclassical theory of the quantum fluctuations of the Neel order.

As we will show, quantum link models are different from their Wilsonian counterparts in terms of how the gauge-invariant Hilbert space is regulated. While a Wilson-type abelian gauge theory has an infinite-dimensional Hilbert space for every link, the quantum link models regulate this infinite-dimensional Hilbert space in a completely gauge-invariant fashion. The resulting finite-dimensional Hilbert space has size $(2S+1)$, where $S = 1/2, 1, 3/2, ...$, and the Wilsonian theory can be obtained when $S \to \infty$ [Schlittgen and Wiese, 2001]. The following sub-section defines the $U(1)$ quantum link model hamiltonian at zero temperature and finite volume. We then explore the winding number sectors structure of the ground state numerically. We also find a correlation between the flippability of the plaquettes and the transition point of different winding sectors.

4.3.1. $U(1)$ quantum link model

The model we choose to explore these phenomena is the U(1) quantum link model (QLM), which is the simplest U(1) lattice gauge theory in (2+1)-dimensions. To understand the claim of being the simplest, let us consider the standard Wilson-type lattice gauge theory. This theory uses a U(1) phase variable as a parallel transporter between two sites x and y,

associated with each link x, μ:

$$u_{x,\mu} = e^{i\varphi_{x,\mu}} = \cos(\varphi_{x,\mu}) + i\sin(\varphi_{x,\mu}) \tag{4.31}$$

We define the plaquette action as:

$$f_\square = u_{x,\mu} u_{x+\hat{\mu},\nu} u^\dagger_{x+\hat{\nu},\mu} u^\dagger_{x,\nu} \tag{4.32}$$

A schematic representation of the plaquette is presented in Figure 4.10. The plaquettes can be completely space-like, giving the magnetic field energy B. The plaquettes can be time-like (one space direction and one time direction), giving the electric field energy (in the continuum limit, the action is thus $E^2 + B^2$). The classical lattice action is given by:

$$\mathrm{S}[u] = -\frac{1}{2}\sum_\square (f_\square + f^\dagger_\square) + k(f_\square + f^\dagger_\square)^2 \tag{4.33}$$

where $\sum_\square$ denotes the sum over all plaquettes. The k-term is an adjoint term. Terms like this for the $U(1)$ lattice gauge theory were explored in the early days of lattice field theory in the Wilson formulation [Bhanot and Creutz, 1981]. These are typically higher representations of the fundamental representation. Such terms are useful to explore new phases, as well as to decrease cut-off effects when approaching the continuum limit. The action $\mathrm{S}[u]$ is invariant under the U(1) transformation:

$$u_{x,\mu} \to e^{i\alpha_x}\, u_{x,\mu}\, e^{-i\alpha_{x+\hat{\mu}}} \tag{4.34}$$

We now construct the quantum counterpart of the standard $U(1)$ gauge theory action. We want to replace the classical action with a quantum Hamilton operator.

$$\mathcal{H} = -J\sum_\square (U_\square + U^\dagger_\square) + \lambda(U_\square + U^\dagger_\square)^2 \tag{4.35}$$

where we have defined the plaquette operator $U_\square$ as:

$$U_\square = U_{x,\mu} U_{x+\hat{\mu},\nu} U^\dagger_{x+\hat{\nu},\mu} U^\dagger_{x,\nu} \tag{4.36}$$

U, $U^\dagger$ are operators that act on the Hilbert space. There are three operators associated with every link: U, $U^\dagger$, and E. These operators are not yet identified. The coupling λ in Equation 4.62 is also called in the literature the Rokhsar-Kivelson (RK) coupling. Here the operators U, $U^\dagger$, and E are operators acting on a Hilbert space and not a c-numbers, like $u_{x,\mu}$. We can decompose $U_{x,\mu}$ as:

$$U_{x,\mu} = C_{x,\mu} + iS_{x,\mu} \tag{4.37}$$

where C and S are Hermitian operators. $U^\dagger$ can be decomposed as:

$$U^\dagger_{x,\mu} = C_{x,\mu} - iS_{x,\mu} \tag{4.38}$$

We denote the infinitesimal Gauge transformation operator as G_x. The quantum link model's gauge symmetry requires that the Hamiltonian commutes with the generators G_x of infinitesimal gauge transformations at each lattice site. The gauge invariance requirements induce the following commutation relations, as demonstrated in [Chandrasekharan and Wiese, 1997]:

$$[E_{x,\mu}, U_{y,\nu}] = U_{x,\mu}\delta_{\mu,\nu}\delta_{x,y} \tag{4.39}$$
$$[E_{x,\mu}, U^\dagger_{y,\nu}] = U^\dagger_{x,\mu}\delta_{\mu,\nu}\delta_{x,y} \tag{4.40}$$
$$[U_{x,\mu}, U^\dagger_{y,\nu}] = 2E_{x,\mu}\delta_{\mu,\nu}\delta_{x,y} \tag{4.41}$$
$$\tag{4.42}$$

A representation of the operators C, S, E and G that satisfies these commutation relations is:

$$E_{x,\mu} = S^3_{x,\mu} \tag{4.43}$$

$$C_{x,\mu} = S^1_{x,\mu} \tag{4.44}$$

$$S_{x,\mu} = S^2_{x,\mu} \tag{4.45}$$

$$G_x = \sum_\mu \left(E_{x-\hat{\mu},\mu} - E_{x,\mu}\right) = \sum_\mu \left(S^3_{x-\hat{\mu},\mu} - S^3_{x,\mu}\right) \tag{4.46}$$

where the $S^i_{x,\mu}$ operators follow the angular momentum commutation relations:

$$\left[S^i_{x,\mu}\,,\, S^j_{y,\nu}\right] = i\delta_{x,y}\delta_{\mu,\nu}\epsilon_{i,j,k}S^x_{x,\mu} \tag{4.47}$$

We can realize these commutation relations with any representation of SU(2). The local Hilbert space formed by the link passes to be from infinite-dimensional to two-dimensional. We can recover the original theory with the limit of the local spin to infinity, as shown in [Schlittgen and Wiese, 2001]. For simplicity in the numerical simulations and having quantum simulations in our mind, we choose the spin 1/2 representation of the link and the Pauli matrices as a representation of those matrices. We can then identify $S^1 = \sigma^x$, $S^2 = \sigma^y$

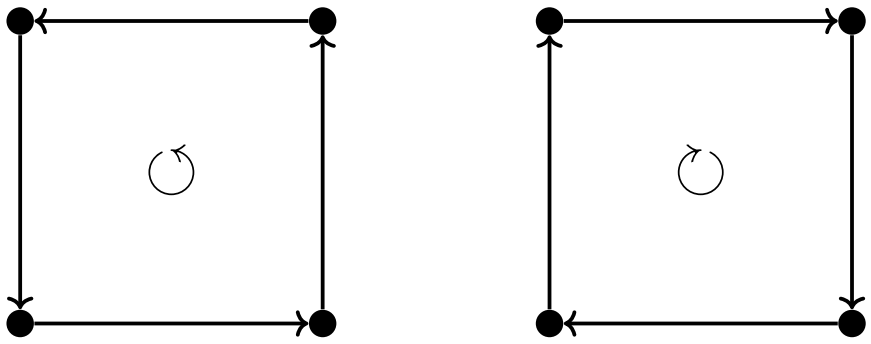


Figure 4.11: *Schematic representation of the Plaquette states $|\square_1\rangle$ and $|\square_2\rangle$.*

and $S^3 = \sigma^z$. The operators $U_{x,\mu}$ and $U^\dagger_{x,\mu}$ are given by:

$$U_{x,\mu} = S^x_{x,\mu} + iS^y_{x,\mu} = S^+_{x,\mu} \tag{4.48}$$

$$U^\dagger_{x,\mu} = S^x_{x,\mu} - iS^y_{x,\mu} = S^-_{x,\mu} \tag{4.49}$$

The system's Hamiltonian is invariant under any gauge transformation of the $U(1)$ group. The following commutation relation ensures this invariance:

$$[\mathcal{H}\,, G_x] = 0 \tag{4.50}$$

Since the two operators commute, the states in the Hilbert space can be characterized by the eigenstates of the operator G_x. This naturally divides the Hilbert space into different sectors. The gauge invariance restricts the sector of physical states to be in the eigenspace of states with 0 eigenvalues with respect to G. This means that any physical state $|\Psi\rangle$ must obey the relation:

$$G_x\,|\Psi\rangle = 0 \tag{4.51}$$

Thus, we can identify the eigenvalues of the operator $S^3_{x,\mu}$ as the electric fluxes associated with the links. We can make this identification because Gauss's law requires that the fluxes associated with links emanating from the same lattice point add up to zero. These electric fluxes, being the eigenvalues of S^3 can only be $\pm 1/2$. Thus, contrary to ordinary lattice gauge theories, the Hilbert space of a quantum link model is finite (on a finite lattice).

Flippability of the plaquettes We now consider all the possible states that one plaquette can have. In our quantum link model, since a plaquette is formed by four different spins $1/2$, the number of states is $n_{\text{states}} = 2^4 = 16$.

When we act on a generic plaquette state $|\psi\rangle$ with the plaquette operator $U_\square$, only two states are left. This is an effect of the regularization of the Hilbert space. In the quantum

link model we consider, the plaquette operator comprises raising and lowering operators in a finite Hilbert space (S^+ and S^-). To describe the action of the plaquette operator on a generic state, we start by focusing on a simple link. We have chosen a representation of the spin 1/2 link. The two basis states are $|\uparrow\rangle$ and $|\downarrow\rangle$. If the link is in the state $|\uparrow\rangle$, the state is annihilated by the raising operator S^+. The state $|\downarrow\rangle$ is transformed in the state $|\uparrow\rangle$ by the raising operator. The operator S^- acts accordingly.

The plaquette operators $U_\square$ and $U_\square^\dagger$ are composed by the combination of four different raising and lowering operators:

$$U_\square = S^+_{x,\mu} S^+_{x+\hat{\mu},\nu} S^-_{x+\hat{\nu},\mu} S^-_{x,\nu} \tag{4.52}$$

$$U_\square^\dagger = S^-_{x,\mu} S^-_{x+\hat{\mu},\nu} S^+_{x+\hat{\nu},\mu} S^+_{x,\nu} \tag{4.53}$$

In our representation, a plaquette can be any combination of the four different spins that compose the plaquette (e.g., $|\downarrow\downarrow\downarrow\uparrow\rangle$).

Between all the possible 16 states that a plaquette can have, only two are not annihilated by the action of the $U_\square$ and $U_\square^\dagger$ operators. The two possible states are: $|\square_1\rangle$ and $|\square_2\rangle$. To give a schematic representation of these two states, we consider a gauge link starting from the node x in the $\hat{\mu}$ direction. We represent the link as an arrow pointing in the $\hat{\mu}$ direction if the expectation value of σ^z on the state is +1 ($|\uparrow\rangle$). We represent the link as an arrow pointing in the $-\hat{\mu}$ direction if the expectation value of σ^z on the state is -1($|\downarrow\rangle$). We define $|\square_1\rangle$ as the classical state in which the plaquette state has a clockwise orientation($|\uparrow\uparrow\downarrow\downarrow\rangle$). We define $|\square_2\rangle$ as the classical state in which the plaquette state has a counter-clockwise orientation ($|\downarrow\downarrow\uparrow\uparrow\rangle$). The operator $U_\square$ acts on the two states as:

$$U_\square^\dagger |\square_1\rangle = |\square_2\rangle \tag{4.54}$$

$$U_\square |\square_2\rangle = |\square_1\rangle \tag{4.55}$$

Those two states are the only flippable states. We define the flippability operator O_{flipp}[II]

$$O_{\text{flipp}} = \sum_\square (U_\square + U_\square^\dagger)^2 = \sum_\square (U_\square U_\square^\dagger + U_\square^\dagger U_\square) \tag{4.56}$$

The operator O_{flipp} counts the flippable plaquette states in a given configuration. The operator O_{flipp} acts on the two flippable states as:

$$O_{\text{flipp}} |\square_1\rangle = |\square_1\rangle \tag{4.57}$$

$$O_{\text{flipp}} |\square_2\rangle = |\square_2\rangle \tag{4.58}$$

All the other possible states are annihilated by O_{flipp}.

[II]The operator O_{flipp} coincides with the quadratic term in the plaquette operator in the Hamiltonian $\mathcal{H}$.

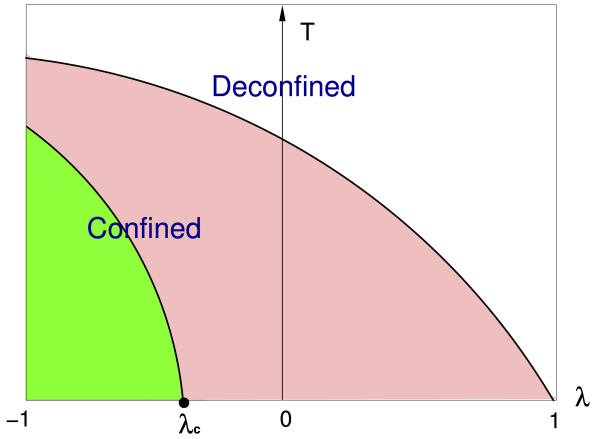


Figure 4.12: *Phase diagram of the $U(1)$ quantum link model at finite temperature adapted from [Banerjee et al., 2013].*

Chemical potential In 2+1 dimensions and on a finite lattice of spatial dimension $L_x \times L_y$, we define the point in the space x defined before as a couple (x, y) and the directions of the link to be $\mu \in \{x, y\}$. In addition to the U(1) gauge symmetry, the Hamiltonian also has the point group symmetries (translation, rotation, and parity). The Hamiltonian is invariant under the charge conjugation symmetry ($\mathbb{Z}^2$). The Hamiltonian also has a winding number symmetry ($\mathrm{U}(1) \otimes \mathrm{U}(1)$) corresponding to the two spatial directions. The generator of this symmetry are the operators:

$$W_x = \frac{1}{L_x} \sum_{x}^{L_x} \sum_{y}^{L_y} S^3_{(x,y),\hat{x}} \tag{4.59}$$

for the x direction and

$$W_y = \frac{1}{L_y} \sum_{x}^{L_x} \sum_{y}^{L_y} S^3_{(x,y),\hat{y}} \tag{4.60}$$

for the y direction. The generators of this symmetry commute with the Hamiltonian:

$$[\mathcal{H}, W_x] = [\mathcal{H}, W_y] = 0 \tag{4.61}$$

As for Gauss's law, we can simultaneously diagonalize $\mathcal{H}$, W_y and W_x. We can add these terms to the Hamiltonian and explore the different phases of the phase space as a function of the couplings μ_x and μ_y:

$$\mathcal{H}' = -J \sum_{\square} (U_\square + U_\square^\dagger) + \lambda (U_\square + U_\square^\dagger)^2 + \mu_x \sum_x W_x + \mu_y \sum_y W_y \tag{4.62}$$

Since W_x and W_y are good quantum numbers, the energy eigenstate is also a winding number eigenstate. We can call the couplings μ_x and μ_y chemical potentials in the sense

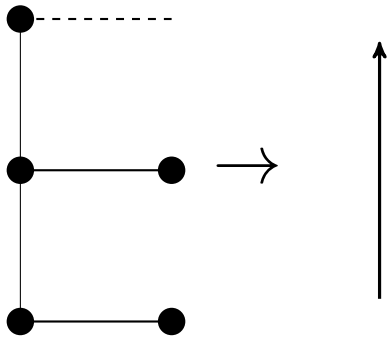


Figure 4.13: *Mapping the local degrees of freedom(four spin 1/2 of dimension 2) into a single spin (of dimension 16).*

that they are conjugate to a conserved quantity, in analogy with the chemical potential in fermionic and scalar systems. The operators W_x and W_y allow for the formation of stringy excitations that spread over the lattice. Moreover, in analogy with the particle number operators in fermionic and scalar theories, the ground state energy is given by:

$$E_{GS} = E_{\mathcal{H}} - \mu_x N^x - \mu_y N^y \tag{4.63}$$

where we have introduce N^x and N^y as the expectation value of W_x and W_y on the ground state respectively. We call N^x and N^y the winding numbers of the state.

Phase diagram At zero temperature and zero chemical potential, the model undergoes a phase transition for $J = 1$ and λ between -1 and 1. An overview of the phase diagram is presented in Figure 4.12 adapted from [Banerjee et al., 2013]. We can identify three different phases of the model in the phase diagram. At zero temperature, the model is confining for $\lambda < 1$. At finite temperature T, it has a deconfinement phase transition. The deconfinement phase transition point reaches zero temperature at $\lambda = 1$. At λ_c, a quantum phase transition separates two phases with spontaneously broken translational symmetry. The phase at $\lambda < \lambda_c$ has a spontaneously broken charge conjugation symmetry. The exact values of the phase transition points and an extended analysis of the phase diagram are presented in [Banerjee et al., 2013].

4.3.2. Numerical results

Our numerical simulation studies the $U(1)$ quantum link model in 2+1 dimensions with open boundary conditions in the y direction and periodic boundary conditions in the x direction. We study this system using the density matrix renormalization group algorithm with the matrix product state ansätz. We study a system in a ladder geometry in which we keep the L_y dimension fixed to 2, and the L_x dimension can vary. This geometry is motivated by the numerical computational cost.

The density matrix renormalization group algorithm and the ladder geometry of the

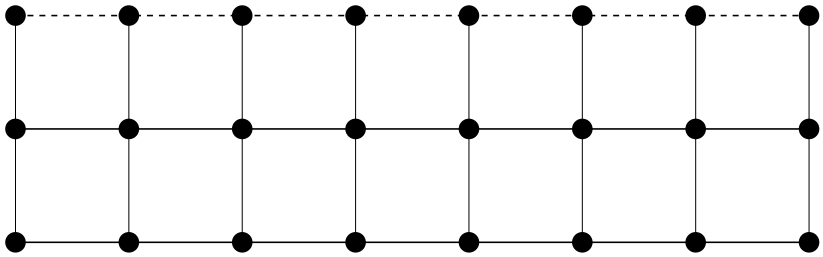


Figure 4.14: *Lattice configuration*

lattice allow us to simulate the model as a $1+1$ dimensional model. We map all the spins in the y direction into a unique index with dimension $d = 2^{L_y} = 16$, as showed in Figure 4.13. This computational base is repeated over all the lattice.

We have been able to perform simulations only with $L_y = 2$ because the numerical cost grows exponentially with L_y[III]. A scheme of the lattice geometry is presented in Figure 4.14. The dashed line means that periodic boundary conditions are imposed. In the numerical studies, we have imposed $\mu_x = 0$. A large value of μ_x would fix all the horizontal links and destroy all the dynamics. The open boundary conditions affect the commutation relation between the Hamiltonian and the operator W_y. The commutator $[\mathcal{H}, W_y]$ is no longer zero when open boundary conditions are imposed. In our numerical simulation we explore the effect of the chemical potential (μ_y) on the ground state properties of the theory.

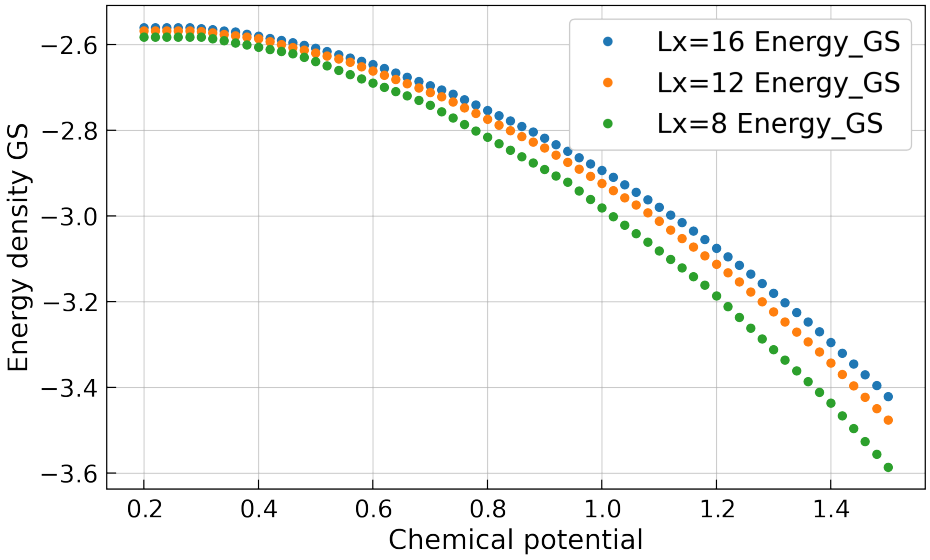


Figure 4.15: *Energy density (Energy / Volume) as a function of the chemical potential for different lattice sites*

We fix $J = 1$ and $\lambda = -1$ to focus on the chemical potential effects on the ground state. This choice of parameters ensures the system to be far away from the phase transition point.

[III]The exponential scaling is given by the local Hilbert space dimension and the representation of the matrix product operator of the system mapped into a one-dimensional spin-chain. When we increase L^y, much more terms need to be included in the matrix product operator making the simulations unfeasible for $L^y > 10$. If we want to study a full $2+1$ dimensional model, we would need to use a different algorithm.

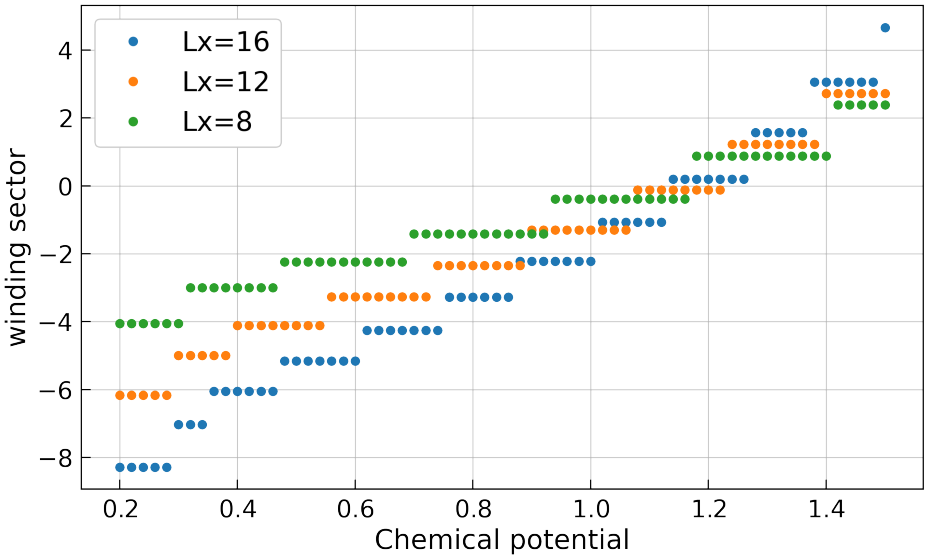


Figure 4.16: *Expectation value of the winding number as a function of the chemical potential for different lattice sites*

Tensor network algorithms are the only known methods to analyze the phase diagram of the model at a finite chemical potential, at least when generic values are considered. In this study, we do not aim to perform a continuum limit and a thermodynamic limit. We study three different volumes with $L_x = 8$, $L_x = 12$, and $L_x = 16$. In Figure 4.15, we plot the ground state's energy density as a function of the chemical potential.

In Figure 4.16 we plot the expectation value of the winding number operator W_y as a function of the chemical potential μ_y for the three different volumes. The expectation value of W_x is always precisely zero for every state we checked. We have studied this numerically. We used this information to crosscheck convergence at the end of the simulations. We measured N_y on the ground state for every value of the chemical potential. In Figure 4.16 we see that N_y acquires discrete values depending on the chemical potential. This means that when we increase the chemical potential, the ground state is in a different winding number sector. The magnitude of the jumps is not constant as a function of the chemical potential. In Figure 4.17 we plot the difference in the expectation value of the winding number operator divided by the chemical potential as a function of the chemical potential. We choose this particular form to highlight that the magnitude of these differences Δ winding grows almost linearly with the chemical potential. Asymptotically the ratio $\Delta_{\text{winding}}/\mu$ is constant. We can thus express the difference in winding number as:

$$\Delta_{\text{winding}} = a\mu_y + b \tag{4.64}$$

The offset value seems to converge to 1. More precise and extensive numerical simulations on larger volumes are needed to confirm this value. In Figure 4.18 the expectation values of the flippability operator and the expectation values of the winding number operator are plotted

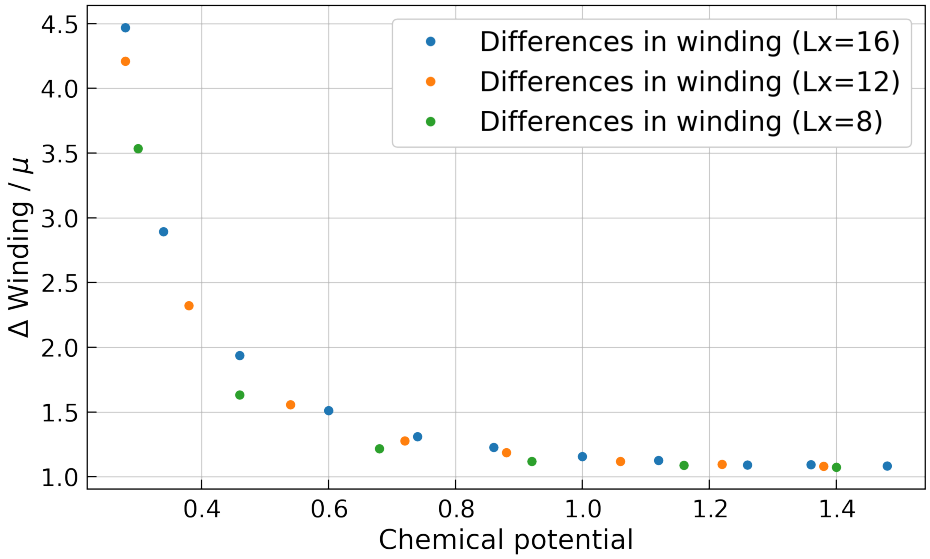


Figure 4.17: *Differences in winding numbers divided by the chemical potential as a function of the chemical potential for different lattice sites*

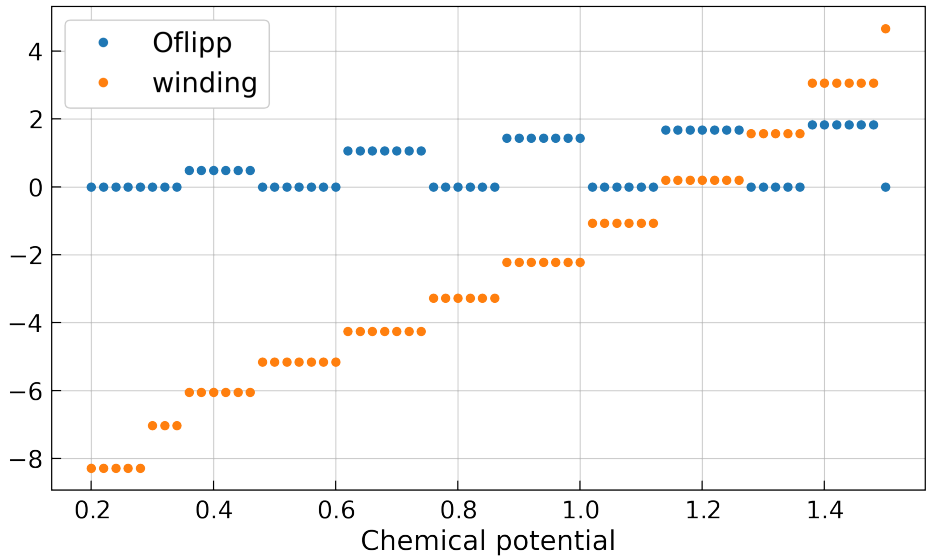


Figure 4.18: *Average flippability of plaquettes (expectation value of O_{flipp} and expectation value of the operator W_x as a function of the chemical potential for $L_x = 16$ (32 plaquettes)*

as a function of the chemical potential μ_y. When we focus on the expectation value of the O_{flipp} operator, we notice that there are discrete jumps from zero to non-zero values. At the same chemical potential at which the O_{flipp} operator jumps, we also observe a corresponding discontinuity in the winding number. This points out an apparent correlation between O_{flipp} and the winding number sectors. While the winding jumps are expected in analogy with the discrete jumps in the particle number happening in scalar and fermionic theories (e.g., see [Bañuls et al., 2017]), we did not have similar expectations for the flippability of the ground states. Moreover, we cannot explain physically why the winding number and the flippability

jumps happen at the same chemical potential values. This unexpected correlation needs further investigation. Nevertheless, this is a new effect that was never observed nor studied before.

4.4. Conclusions and outlooks

In this chapter, we have introduced the matrix product state as a variational ansätz. We have described the matrix product operator formalism for generic operators and made an explicit example for a local Hamiltonian. We have applied this machinery to a U(1) quantum link model in 2+1 dimensions in a ladder geometry. We have studied the properties of the model's ground state at finite volume and finite chemical potential. We have explored the properties of the different winding sectors of the ground state and studied the different winding sectors. We have also seen a correlation between the winding number sectors and the average flippability of the plaquettes. We did not expect this correlation before this study, and a deeper numerical analysis of this phenomenon is presently being performed. This model is also a good candidate for cold atom simulations [Lewenstein et al., 2007]. It is also possible to extend the U(1) quantum link model Hamiltonian and include an external magnetic field that passes through the lattice. The action of this external magnetic field is to modify the kinetic term of the Hamiltonian, adding a phase $\alpha_\square$ to the plaquette operation:

$$U_\square \to U_\square e^{i\alpha_\square} \tag{4.65}$$

$$U_\square^\dagger \to e^{-i\alpha_\square} U_\square^\dagger \tag{4.66}$$

This modification can alter the ground state properties and give rise to interesting phenomena. Moreover, we can also simulate fermionic fields in $1+1$ dimensions. The basic building block for tensor network simulations of fermion fields is the Jordan-Wiegner transformation [Jordan and Wigner, 1928]. It allows the mapping of fermionic degrees of freedom into a spin system. In Appendix A we derive the spin formulation for the lattice Hamiltonian of the Gross-Neveu model [Gross and Neveu, 1974]. The Gross-Neveu model is a fermionic model with a four-Fermi interaction term. The model can be seen as a toy model for quantum chromodynamics. In the large N (number of flavors) limit, we can demonstrate that the model is asymptotically free. The model also presents a dynamical symmetry breaking of chiral symmetry. Numerical studies of the Gross-Neveu model are being performed presently.

Chapter 5

Conclusions

Quantum simulations of lattice gauge theories play a fundamental role in first principle calculations in the context of high energy physics. The goal of this thesis was to improve the state-of-the-art simulation methods for first-principle calculations and apply those methods to relevant physical models. We have addressed this problem using three approaches: machine learning, quantum computing, and tensor networks.

In the context of machine learning, we have developed a method to estimate thermodynamic observables in lattice field theories based on normalizing flows. The method we have developed has two main advantages compared to Markov Chain Monte Carlo. When we want to compute the free energy of the system in MCMC, we have to generate overlapping distributions at various, subsequent parameter values until the target parameter is reached. This procedure is increasingly complex when we have to cross a phase transition in the parameter space and can lead to the accumulation of systematic errors. This integration is not needed in our machine learning-based approach. Moreover, in the MCMC integration, one requires an integration constant to evaluate the free energy. This is again not required with our approach. We have demonstrated the applicability of our method by studying the ϕ^4 theory in two dimensions. Our approach can compute the system's free energy with much smaller errors compared to Monte Carlo simulations. Future work will focus on scaling this approach to larger volumes and different models, with the final goal of simulating non-Abelian gauge theories and, ultimately, quantum chromodynamics.

In the context of quantum computing, we have introduced the variational quantum simulation algorithm and presented two works that can improve the applicability of this algorithm. One of the most significant limitations of modern quantum computers is the noise of the simulations. Read-out noise is one of the more relevant noises in most quantum computers. We have developed a classical bit-flip correction method to mitigate measurement errors on noisy quantum computers in this context. This method is based on the replacement of the operator we want to measure with a different operator in such a way that the noisy expectation value of the new operator corresponds to the quantum mechanical expectation value of the original operator. Our method, which scales only

polynomially in the number of qubits, only requires the knowledge of the different bit-flip probabilities during read-out for each qubit. We can easily compute these quantities with a suitable calibration procedure. The computational overhead is moderate for local Hamiltonians. We have tested our method numerically, and we have provided a proof of principle demonstration of this procedure on IBM's quantum hardware. Our approach can be applied to any platform and lends itself to superconducting qubits and other architectures such as trapped ions. We aim at extending this mitigation procedure to different errors in future and ongoing works, starting with the depolarising errors. As already demonstrated in this work, scalable error mitigation can drastically improve quantum simulations.

The design of efficient variational quantum circuits for quantum simulation is also crucial in practical simulations. In this thesis, we have developed a dimensional expressivity analysis of parametric quantum circuits that can help to design these circuits. This analysis relies on comparing the tangent spaces of this map and the manifold of reachable states by a quantum circuit. This analysis also allows for the identification and implementation of physical symmetries in variational quantum circuits. We have shown how to identify parameters that correspond to an unwanted symmetry. We have applied this approach to remove a global phase in parametric quantum circuits for variational quantum simulations. Moreover, we have also shown how the Hilbert space of translationally invariant Hamiltonians is divided into different sectors, and how to compute those dimensions. We have applied the dimensional expressivity analysis to interesting examples and developed a hardware-efficient quantum algorithm to perform this analysis. The algorithm scales polynomially in the number of parameters of the parametric quantum circuit. This algorithm is helpful for any quantum simulation that we will perform in the future and is highly recommended to anyone who wants to perform quantum simulations.

Tensor network simulations are an up-and-coming tool to simulate quantum field theories. In particular, in recent years, there has been a boost in the development of tensor network methods to simulate lattice gauge theories. This thesis used the density matrix renormalization group with the matrix product state ansatz to study the 2+1 dimensional $U(1)$ quantum link model in a ladder geometry. The study's goal was to explore the stringy excitations present in the ground state when chemical potential is added to the system. The use of tensor network algorithms is motivated by the fact that no other algorithm is presently known to simulate the model at a finite chemical potential. We have demonstrated the presence of these stringy excitations numerically and we have found a correlation between the different winding number sectors and the flippability of the ground state. We did not expect this correlation before this study, and a deeper numerical analysis of this phenomenon is currently ongoing. Other work focuses on a better understanding of the physical phenomena we have seen in our numerical simulations. The long-term goal would be to simulate a quantum link model Abelian gauge theory on a quantum simulator.

Appendix A

Gross-Neveu Hamiltonian mapped into a spin Hamiltonian

In this appendix, we derive the (flavored) Gross-Neveu model [Gross and Neveu, 1974] Hamiltonian 1+1 dimensions in a formulation that we can use for matrix product state calculations. Such calculations are being performed presently. In [Lenz et al., 2020] the authors have identified inhomogeneous phases of the model at finite number of flavors, finite chemical potential, and finite temperature. The authors use Monte Carlo simulations to analyze the model. Monte Carlo simulations are bounded to an even number of flavors due to a sign problem. We can overcome this problem using the density matrix renormalization group algorithm. Our goal is to check whether the behaviors observed by [Lenz et al., 2020] still persist with tensor network simulations.

The Lagrangian density of the Gross-Neveu (GN) model in 1+1 dimension is:

$$\mathcal{L} = \sum_{a=1}^{N} \bar{\psi}^a (i\partial\!\!\!/ - m)\psi^a + \frac{g^2}{2N}\left(\sum_{a=1}^{N} \bar{\psi}^a \psi^a\right)^2 \tag{A.1}$$

where a runs from 1 to N. The hamiltonian is:

$$H_{GN} = \int dx \left(\sum_{a=1}^{N} \bar{\psi}^a (\gamma_1 \partial_1 - m)\psi^a + \frac{g^2}{2N}\left(\sum_{a=1}^{N} \bar{\psi}^a \psi^a\right)^2\right) \tag{A.2}$$

This Hamiltonian can be rewritten using a dummy bosonic field Φ that does not have a kinetic term. This corresponds to a Yukawa theory with interaction mediated by an infinitely massive field that reproduce the GN effective four fermion interaction:

$$H_{\Phi} = \int dx \left(\sum_{a=1}^{N} \bar{\psi}^a (\gamma_1 \partial_1)\psi^a + gm\Phi\left(\sum_{a=1}^{N} \bar{\psi}^a \psi^a\right) + \frac{g^2}{2N}\left(\sum_{a=1}^{N} \bar{\psi}^a \psi^a\right)\right) \tag{A.3}$$

It is trivial to show that H_{GN} and H_{Φ} describe the same dynamics when you compute

the equation of motion for the field Φ. The H_{GN} Hamiltonian causes the system to create bosonic pairs. This phenomenon can lead to the condensation of the bosonic pairs and lead to a Φ expectation value on the vacuum non zero. Thus, the rise of a mass term in the dynamic theory leads to a dynamichal symmetry breaking.
We have to define the representation of the Clifford algebra we will use in 1+1 dimension. We define:

$$\gamma_0 = \sigma^z \tag{A.4}$$

$$\gamma_0\gamma_1 = \sigma^x \tag{A.5}$$

We introduce the discretization of the space in d points divided by lattice spacing j and the staggered formulation of the fermion mass (that introduces $2d$ lattice sites dividing add and even sites). Following the usual convention, we represent the spinor $\psi_I^{a\dagger}$ as:

$$\psi_i^{a\dagger} = \left(C_{2I-1}^{a\dagger}\ C_{2I}^{a\dagger}\right) \tag{A.6}$$

where we have put the positive (negative) eigenvalue of γ_0 in the odd (even) sites (thus, we will have $2d$ sites instead of the original lattice with d sites($i = 2I$)). We want to express the therm:

$$H_{kin} = \sum_{i=1}^{i=2d} \bar{\psi}_i^a (i\gamma_1\partial_1)\psi_i^a \tag{A.7}$$

as a function of the operators C_i. H_{kin} is equivalent to:

$$\sum_{I=1}^{I=d} \left(C_{2I-1}^{a\dagger}\ C_{2I}^{a\dagger}\right) \gamma_0\gamma_1 * \partial_1 \begin{pmatrix} C_{2I-1}^a \\ C_{2I}^a \end{pmatrix} \tag{A.8}$$

Introducing the symmteric discrete derivative:

$$\partial_1 f(x) = \frac{1}{2a}(f(x+1) - f(x-1)) \tag{A.9}$$

we obtain:

$$\sum_{I=1}^{I=d} \left(C_{2I-1}^{a\dagger}, C_{2I}^{a\dagger}\right) \sigma^x \begin{pmatrix} C_{2I+1}^a - C_{2I-3}^a \\ C_{2I+2}^a - C_{2I-2}^a \end{pmatrix} \tag{A.10}$$

that leads to the following formulation of H_{kin}:

$$H_{kin} = \sum_{I=1}^{I=d} C_{2I}^{a\dagger} C_{2I+1}^a - C_{2I}^{a\dagger} C_{2I-3}^a + C_{2I-1}^{a\dagger} C_{2I+2}^a - C_{2I-1}^{a\dagger} C_{2I-2}^a \tag{A.11}$$

This equation can be simplified and leads to:

$$H_{kin} = \sum_{i=1,a}^{i=2d} C_{i+1}^{a\dagger} * C_i^a + H.C \tag{A.12}$$

Let us now define the number operator n_i^a :

$$n_i^a = C_i^{a\dagger} C_i^a \tag{A.13}$$

and

$$N_i = \sum_{a=1}^{a=N} n_i^a \tag{A.14}$$

with the property:

$$n_i^a n_i^a = n_i^a \tag{A.15}$$

The mass term of the hamiltonian can be written as:

$$H_{mass} = -m \sum_{i=1}^{i=d} \bar{\psi}_i^a \psi_i^a \tag{A.16}$$

If we write explicitly $\bar{\psi}_i^a \psi_i^a$, using the staggered formulation, we obtain:

$$H_{mass} = -m \sum_{I=1}^{I=2d} (n_{2I-1}^a - n_{2I}^a) = -m \sum_{i=1}^{i=d} (-1)^{i+1} n_i^a \tag{A.17}$$

The interaction term can be written as:

$$H_{int} = \frac{g^2}{2N} \sum_{i=1}^{i=2d} \left(\sum_{a=1}^{N} \bar{\psi}_i^a \psi_i^a \right) \left(\sum_{b=1}^{N} \bar{\psi}_i^b \psi_i^b \right). \tag{A.18}$$

We have already evaluated the therm $\bar{\psi}_i^b \psi_i^b$ when we evaluated the mass term:

$$\bar{\psi}_I^b \psi_I^b = n_{2I-1}^b - n_{2I}^b \tag{A.19}$$

so we can write H_{int} as:

$$H_{int} = \frac{g^2}{2N} \left(\sum_{i=1}^{i=2d} N_i^2 - 2 \sum_{I=1}^{I=d} N_{2I-1} N_{2I} \right) \tag{A.20}$$

For our purpose it is better to rewrite this term as a function of n_i^a:

$$H_{int} = \frac{g^2}{2N} \sum_{i=1}^{i=2d} \left(\sum_{a=1}^{N} n_i^a n_i^a + 2 \sum_{a<b=1}^{N} n_i^a n_i^b \right) - \sum_{I=1}^{I=d} \sum_{a,b=1}^{N} n_{2I-1}^a n_{2I}^b \tag{A.21}$$

We have three terms in the hamiltonian on the lattice: H_{kin} , H_{mas} and H_{int}. In the staggered

formulation, they can be written as:

$$H_{kin} = \sum_{a=1}^{N} \sum_{i=1}^{i=2d} C_{i+1}^{a\dagger} C_i^a + H.C \tag{A.22}$$

$$H_{mass} = -m \sum_{i=1}^{i=2d} (-1)^{i+1} n_i^a \tag{A.23}$$

$$H_{int} = \frac{g^2}{2N} \left[\sum_{i=1}^{i=2d} \left(\sum_{a=1}^{N} n_i^a + 2 \sum_{a<b=1}^{N} n_i^a n_i^b \right) - \sum_{I=1}^{I=d} \sum_{a,b=1}^{N} n_{2I-1}^a n_{2I}^b \right] \tag{A.24}$$

A map between the operators C_i^a and the spin opertors $\sigma^x, \sigma^y, \sigma^z$ (Pauli operators) that act on a spin $1/2$ system has been introduced by [Jordan and Wigner, 1928]. The map is:

$$C_j^{a,\pm} = \prod_{i<j} \left[\sigma_{i,a}^z\right] \sigma_{j,a}^{\pm} \tag{A.25}$$

$$n_j^a = \frac{1 + \sigma_{j,a}^z}{2} \tag{A.26}$$

We want to map our fermionic system into a spin system. We will do this process for every term of the Hamiltonian separately. Let us consider the kinetic term:

$$H_{kin} = \sum_{a=1}^{N} \sum_{i=1}^{i=2d} C_{i+1}^{a\dagger} C_i^a + H.C. = \tag{A.27}$$

$$= \sum_{I=1}^{I=d} C_{2I+1}^{a\dagger} C_{2I}^a + \sum_{I=1}^{I=d} C_{2I}^{a\dagger} C_{2I-1}^a + HC \tag{A.28}$$

Let us focus on the first term of this sum:

$$C_{2I+1}^{a\dagger} C_{2I}^a = \left[\prod_{j<2I+1} \left[\sigma_{j,a}^z\right] \sigma_{2I+1,a}^{+} \right]^{\dagger} \prod_{l<2I} \left[\sigma_{l,a}^z\right] \sigma_{2I,a}^{-} \tag{A.29}$$

Since the sigma operators commute at different sites, $(\sigma^+)^\dagger = \sigma^-$, $\sigma^z\sigma^z = 1$ and $\sigma^z\sigma^- = -\sigma^-$ we can rewrite this term as:

$$C_{2I+1}^{a\dagger} C_{2I}^a = -\sigma_{a,2I+1}^{-} \sigma_{a,2I}^{-} \tag{A.30}$$

Following the same procedure for the other terms we can derive:

$$H_{kin} = \sum_{a=1}^{N} \sum_{I=1}^{i=d} \left(\sigma_{a,2I}^{+} \sigma_{a,2I-1}^{+} - \sigma_{a,2I+1}^{-} \sigma_{a,2I}^{-} \right) + H.C. \tag{A.31}$$

The H_{mass} transformation is trivial:

$$H_{mass} = \frac{-m}{2} \sum_{a=1}^{i=N} \sum_{i=1}^{i=2d} (-1)^{i+1} \left(1 + \sigma_{i,a}^z\right) \tag{A.32}$$

The interaction term is definitely the most complicate (for semplicity of the notation the sum over I always goes from 1 to $2d$, the sum over I always goes from 1 to d, and the sum over a or b goes from 1 to n):

$$H_{int} = \frac{g^2}{2N}\left[\sum_{i,a}\frac{1+\sigma^z_{i,a}}{2} + \sum_{i,a<b}\frac{1+\sigma^z_{i,a}}{2}\frac{1+\sigma^z_{i,b}}{2} - \frac{1}{2}\sum_{I,a,b}\frac{1+\sigma^z_{2I-1,a}}{2}\frac{1+\sigma^z_{2I,b}}{2}\right] \tag{A.33}$$

This can be rewritten, negletting the constant term, as:

$$H_{int} = \frac{g^2}{2N}\left[\frac{3}{2}\sum_{i,a}\sigma^z_{i,a} + \sum_{i,a<b}\sigma^z_{i,a}\sigma^z_{i,b} - \frac{1}{2}\sum_{i,a,b}\sigma^z_{2I-1,a}\sigma^z_{2I,b}\right] \tag{A.34}$$

If we fin $N = 1$, we have the simple following Hamiltonian:

$$\mathcal{H}_{N=1} = -\frac{1}{2}\sum_{n=0}^{L-2}\left(\sigma^+_n\sigma^-_{n+1} + \sigma^-_n\sigma^+_{n+1}\right) + \sum_{n=1}^{L} m_0(-1)^n\left(\sigma^z_n\right) + \tag{A.35}$$

$$+ \frac{g^2}{2}\sum_{n=0}^{L/2-1}\left(1-\sigma^z_{2n}\sigma^z_{2n+1}\right) \tag{A.36}$$

Appendix B

Translational invariant Hilbert space for four qubits

In this appendix we construct all the eigenvector subspaces for $N = 4$ qubits. The equivalence classes $[j]_4$ are:

- $[0]_4 = \{0 = 0000\}$,
- $[1]_4 = \{1 = 0001, 2 = 0010, 4 = 0100, 8 = 1000\}$,
- $[3]_4 = \{3 = 0011, 6 = 0110, 12 = 1100, 9 = 1001\}$,
- $[5]_4 = \{5 = 0101, 10 = 1010\}$,
- $[7]_4 = \{7 = 0111, 14 = 1110, 13 = 1101, 11 = 1011\}$,
- $[15]_4 = \{15 = 1111\}$,

For four qubits we have four different eigenvalue of τ: 1, -1, i, and $-i$. The corresponding dimension of the Hilbert space is given by:

- $\dim \mathrm{Eig}(1) = \#\{[0]_4, [1]_4, [3]_4, [5]_4, [7]_4, [15]_4\} = 6$ (orders divisible by 1),
- $\dim \mathrm{Eig}(-1) = \#\{[1]_4, [3]_4, [5]_4, [7]_4\} = 4$ (orders divisible by 2),
- $\dim \mathrm{Eig}(i) = \#\{[1]_4, [3]_4, [7]_4\} = 3$ (orders divisible by 4),
- $\dim \mathrm{Eig}(-i) = \#\{[1]_4, [3]_4, [7]_4\} = 3$ (orders divisible by 4).

And the different eigenspaces are spanned by:

- $\mathrm{Eig}(1)$:
  1. $e_4(0_0) = |0000\rangle$,
  2. $e_4(1_0) = \frac{|0001\rangle + |0010\rangle + |0100\rangle + |1000\rangle}{\sqrt{4}}$,

3. $e_4(3_0) = \frac{|0011\rangle+|0110\rangle+|1100\rangle+|1001\rangle}{\sqrt{4}}$,
4. $e_4(5_0) = \frac{|0101\rangle+|1010\rangle}{\sqrt{2}}$,
5. $e_4(7_0) = \frac{|0111\rangle+|1110\rangle+|1101\rangle+|1011\rangle}{\sqrt{4}}$,
6. $e_4(15_0) = |1111\rangle$.

– Eig(-1):

1. $e_4(1_1) = \frac{|0001\rangle-|0010\rangle+|0100\rangle-|1000\rangle}{\sqrt{4}}$,
2. $e_4(3_1) = \frac{|0011\rangle-|0110\rangle+|1100\rangle-|1001\rangle}{\sqrt{4}}$,
3. $e_4(5_1) = \frac{|0101\rangle-|1010\rangle}{\sqrt{2}}$,
4. $e_4(7_1) = \frac{|0111\rangle-|1110\rangle+|1101\rangle-|1011\rangle}{\sqrt{4}}$.

– Eig(i):

1. $e_4(1_2) = \frac{|0001\rangle+i|0010\rangle-|0100\rangle-i|1000\rangle}{\sqrt{4}}$,
2. $e_4(3_2) = \frac{|0011\rangle+i|0110\rangle-|1100\rangle-i|1001\rangle}{\sqrt{4}}$,
3. $e_4(7_2) = \frac{|0111\rangle+i|1110\rangle-|1101\rangle-i|1011\rangle}{\sqrt{4}}$.

– Eig$(-i)$:

1. $e_4(1_3) = \frac{|0001\rangle-i|0010\rangle-|0100\rangle+i|1000\rangle}{\sqrt{4}}$,
2. $e_4(3_3) = \frac{|0011\rangle-i|0110\rangle-|1100\rangle+i|1001\rangle}{\sqrt{4}}$,
3. $e_4(7_3) = \frac{|0111\rangle-i|1110\rangle-|1101\rangle+i|1011\rangle}{\sqrt{4}}$.

Finally we can compute the dimension of the different sectors as:

– $\dim_{\mathbb{C}} \text{Eig}(1) = [\#](1) + [\#](2) + [\#](4) = 2 + 1 + 3 = 6$,

– $\dim_{\mathbb{C}} \text{Eig}(-1) = [\#](2) + [\#](4) = 1 + 3 = 4$,

– $\dim_{\mathbb{C}} \text{Eig}(i) = [\#](4) = 3$,

– $\dim_{\mathbb{C}} \text{Eig}(-i) = [\#](4) = 3$.

The sum of the dimension of the different sectors is 16. This is the same dimension as the Hilbert space H. In this case, as expected, the H can be written as a direct sum of the different momentum sectors:

$$\mathrm{H} = \bigoplus_{\omega=1,-1,i,-i} \text{Eig}(\omega) \tag{B.1}$$

It is important to stress that we can apply our dimensional expressivity analysis in any possible momentum eigenspace. The knowledge of the dimension of the space allows us to estimate the maximal dimension needed for the dimensional expressivity analysis of any sector.

Bibliography

Tanabashi et al. Review of particle physics. *Phys. Rev. D*, 98:030001, Aug 2018. doi: 10.1103/PhysRevD.98.030001. URL https://link.aps.org/doi/10.1103/PhysRevD.98.030001.

Kenneth G. Wilson. Confinement of quarks. *Phys. Rev. D*, 10:2445–2459, Oct 1974. doi: 10.1103/PhysRevD.10.2445. URL https://link.aps.org/doi/10.1103/PhysRevD.10.2445.

Stefan Schaefer, Rainer Sommer, and Francesco Virotta. Critical slowing down and error analysis in lattice QCD simulations. *Nucl. Phys. B*, 845:93–119, 2011. doi: 10.1016/j.nuclphysb.2010.11.020.

Gurtej Kanwar, Michael S. Albergo, Denis Boyda, Kyle Cranmer, Daniel C. Hackett, Sébastien Racanière, Danilo Jimenez Rezende, and Phiala E. Shanahan. Equivariant flow-based sampling for lattice gauge theory. *Phys. Rev. Lett.*, 125:121601, Sep 2020. doi: 10.1103/PhysRevLett.125.121601. URL https://link.aps.org/doi/10.1103/PhysRevLett.125.121601.

Danilo Jimenez Rezende, George Papamakarios, Sébastien Racanière, Michael S. Albergo, Gurtej Kanwar, Phiala E. Shanahan, and Kyle Cranmer. Normalizing flows on tori and spheres, 2020.

Denis Boyda, Gurtej Kanwar, Sébastien Racanière, Danilo Jimenez Rezende, Michael S. Albergo, Kyle Cranmer, Daniel C. Hackett, and Phiala E. Shanahan. Sampling using $su(n)$ gauge equivariant flows, 2020.

Phiala E. Shanahan, Daniel Trewartha, and William Detmold. Machine learning action parameters in lattice quantum chromodynamics. *Phys. Rev. D*, 97:094506, May 2018. doi: 10.1103/PhysRevD.97.094506. URL https://link.aps.org/doi/10.1103/PhysRevD.97.094506.

Kim A. Nicoli, Christopher J. Anders, Lena Funcke, Tobias Hartung, Karl Jansen, Pan Kessel, Shinichi Nakajima, and Paolo Stornati. Estimation of thermodynamic observables in lattice field theories with deep generative models. *Phys. Rev. Lett.*, 126:032001, Jan 2021. doi: 10.1103/PhysRevLett.126.032001. URL https://link.aps.org/doi/10.1103/PhysRevLett.126.032001.

Boram Yoon, Tanmoy Bhattacharya, and Rajan Gupta. Machine learning estimators for lattice qcd observables. *Phys. Rev. D*, 100:014504, Jul 2019. doi: 10.1103/PhysRevD.100.014504. URL https://link.aps.org/doi/10.1103/PhysRevD.100.014504.

Rui Zhang, Zhouyou Fan, Ruizi Li, Huey-Wen Lin, and Boram Yoon. Machine-learning prediction for quasiparton distribution function matrix elements. *Phys. Rev. D*, 101:034516, Feb 2020. doi: 10.1103/PhysRevD.101.034516. URL https://link.aps.org/doi/10.1103/PhysRevD.101.034516.

William Detmold, Gurtej Kanwar, Michael L. Wagman, and Neill C. Warrington. Path integral contour deformations for noisy observables. *Physical Review D*, 102(1), Jul 2020. ISSN 2470-0029. doi: 10.1103/physrevd.102.014514. URL http://dx.doi.org/10.1103/PhysRevD.102.014514.

Christof Gattringer and Christian Lang. *Quantum chromodynamics on the lattice: an introductory presentation*, volume 788. Springer Science & Business Media, 2009.

David J. E. Callaway and Aneesur Rahman. Lattice gauge theory in the microcanonical ensemble. *Phys. Rev. D*, 28:1506–1514, Sep 1983. doi: 10.1103/PhysRevD.28.1506. URL https://link.aps.org/doi/10.1103/PhysRevD.28.1506.

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, jun 1953. doi: 10.1063/1.1699114.

Philippe de Forcrand, Massimo D'Elia, and Michele Pepe. 't hooft loop in su(2) yang-mills theory. *Phys. Rev. Lett.*, 86:1438–1441, Feb 2001. doi: 10.1103/PhysRevLett.86.1438. URL https://link.aps.org/doi/10.1103/PhysRevLett.86.1438.

Owe Philipsen. The qcd equation of state from the lattice. *Progress in Particle and Nuclear Physics*, 70:55–107, May 2013. ISSN 0146-6410. doi: 10.1016/j.ppnp.2012.09.003. URL http://dx.doi.org/10.1016/j.ppnp.2012.09.003.

Kieran Holland, Michele Pepe, and Uwe-Jens Wiese. Revisiting the deconfinement phase transition in su(4) yang-mills theory in 2+1 dimensions. *Journal of High Energy Physics*, 2008(02):041–041, Feb 2008. ISSN 1029-8479. doi: 10.1088/1126-6708/2008/02/041. URL http://dx.doi.org/10.1088/1126-6708/2008/02/041.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová. Machine learning and the physical sciences. *Reviews of Modern Physics*, 91(4), Dec 2019. ISSN 1539-0756. doi: 10.1103/revmodphys.91.045002. URL http://dx.doi.org/10.1103/RevModPhys.91.045002.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

S B Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica*, 31(3):249–268, 2007. URL http://www.informatica.si/PDF/31-3/11_Kotsiantis%20-%20Supervised%20Machine%20Learning%20-%20A%20Review%20of...pdf.

Baran Yildiz, Jose Bilbao, and Alistair Sproul. A review and analysis of regression and machine learning models on commercial building electricity load forecasting. *Renewable and Sustainable Energy Reviews*, 73:1104–1122, 06 2017. doi: 10.1016/j.rser.2017.02.023.

İlhan Uysal and Halil Altay Güvenir. An overview of regression techniques for knowledge discovery. *Knowledge Engineering Review*, 14:319–340, 12 1999. doi: 10.1017/S026988899900404X.

This person does not exist. https://thispersondoesnotexist.com. Accessed: 2021-01-28.

Zhengwei Wang, Qi She, and Tomas E. Ward. Generative adversarial networks: A survey and taxonomy. *CoRR*, abs/1906.01529, 2019. URL http://arxiv.org/abs/1906.01529.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference, 2019.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows, 2016.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation, 2015.

Michael S. Albergo, Denis Boyda, Daniel C. Hackett, Gurtej Kanwar, Kyle Cranmer, Sébastien Racanière, Danilo Jimenez Rezende, and Phiala E. Shanahan. Introduction to normalizing flows for lattice field theory, 2021.

David JC MacKay and David JC Mac Kay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Stephen L. Adler. Over-relaxation method for the monte carlo evaluation of the partition function for multiquadratic actions. *Phys. Rev. D*, 23:2901–2904, Jun 1981. doi: 10.1103/PhysRevD.23.2901. URL https://link.aps.org/doi/10.1103/PhysRevD.23.2901.

Sebastian Nowozin. Debiasing evidence approximations: On importance-weighted autoencoders and jackknife variational inference. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HyZoi-WRb.

P.J. Bickel and Kjell A. Doksum. *Mathematical Statistics: Basic Ideas and Selected Topics*. Holden-Day Series in Probability and Statistics. Prentice Hall, 1977. ISBN 9780135641477. URL https://books.google.com.vc/books?id=I_AuswEACAAJ.

Kim A. Nicoli, Shinichi Nakajima, Nils Strodthoff, Wojciech Samek, Klaus-Robert Müller, and Pan Kessel. Asymptotically unbiased estimation of physical observables with neural samplers. *Phys. Rev. E*, 101:023304, Feb 2020. doi: 10.1103/PhysRevE.101.023304. URL https://link.aps.org/doi/10.1103/PhysRevE.101.023304.

John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018. ISSN 2521-327X. doi: 10.22331/q-2018-08-06-79. URL https://doi.org/10.22331/q-2018-08-06-79.

F. Arute et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505, 2019. doi: 10.1038/s41586-019-1666-5.

Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, Oct 1997. ISSN 1095-7111. doi: 10.1137/s0097539795293172. URL http://dx.doi.org/10.1137/S0097539795293172.

Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15), Oct 2009. ISSN 1079-7114. doi: 10.1103/physrevlett.103.150502. URL http://dx.doi.org/10.1103/PhysRevLett.103.150502.

S. P. Jordan. Quantum algorithm zoo. http://math.nist.gov/quantum/zoo/. Accessed: 2021-01-28.

Ashley Montanaro. Quantum algorithms: an overview. *npj Quantum Information*, 2(1), Jan 2016. ISSN 2056-6387. doi: 10.1038/npjqi.2015.23. URL http://dx.doi.org/10.1038/npjqi.2015.23.

Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023, Feb 2016. ISSN 1367-2630. doi: 10.1088/1367-2630/18/2/023023. URL http://dx.doi.org/10.1088/1367-2630/18/2/023023.

P. J. J. O'Malley et al. Scalable quantum simulation of molecular energies. *Phys. Rev. X*, 6: 031007, 2016. doi: 10.1103/PhysRevX.6.031007.

A. Kandala, A. Mezzacapo, K. Temme, et al. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nat.*, 549:242, 2017. doi: 10.1038/nature23879.

Y. Shen, X. Zhang, S. Zhang, J.-N. Zhang, M.-H. Yung, and K. Kim. Quantum implementation of the unitary coupled cluster for simulating molecular electronic structure. *Phys. Rev. A*, 95:020501, 2017. doi: 10.1103/PhysRevA.95.020501.

J. I. Colless, V. V. Ramasesh, D. Dahlen, M. S. Blok, M. E. Kimchi-Schwartz, J. R. McClean, J. Carter, W. A. de Jong, and I. Siddiqi. Computation of molecular spectra on a quantum processor with an error-resilient algorithm. *Phys. Rev. X*, 8:011021, 2018. doi: 10.1103/PhysRevX.8.011021.

E. F. Dumitrescu, A. J. McCaskey, G. Hagen, G. R. Jansen, T. D. Morris, T. Papenbrock, R. C. Pooser, D. J. Dean, and P. Lougovski. Cloud quantum computing of an atomic nucleus. *Phys. Rev. Lett.*, 120:210501, 2018. doi: 10.1103/PhysRevLett.120.210501.

C. Hempel et al. Quantum chemistry calculations on a trapped-ion quantum simulator. *Phys. Rev. X*, 8:031022, Jul 2018. doi: 10.1103/PhysRevX.8.031022.

M. Ganzhorn, D.J. Egger, P. Barkoutsos, P. Ollitrault, G. Salis, N. Moll, M. Roth, A. Fuhrer, P. Mueller, S. Woerner, I. Tavernelli, and S. Filipp. Gate-efficient simulation of molecular eigenstates on a quantum computer. *Phys. Rev. Applied*, 11:044092, 2019. doi: 10.1103/PhysRevApplied.11.044092.

C. Kokail, C. Maier, R. van Bijnen, et al. Self-verifying variational quantum simulation of lattice models. *Nat.*, 569:355, 2019. doi: 10.1038/s41586-019-1177-4.

T. Hartung and K. Jansen. Zeta-regularized vacuum expectation values. *J. Math. Phys.*, 60(9): 093504, 2019. doi: 10.1063/1.5085866.

K. Jansen and T. Hartung. Zeta-regularized vacuum expectation values fromquantum computing simulations. *Proc. Sci. LATTICE2019*, 363:153, 2020. doi: 10.22323/1.363.0153.

Y. Li and S. C. Benjamin. Efficient variational quantum simulator incorporating active error minimization. *Phys. Rev. X*, 7:021050, 2017. doi: 10.1103/PhysRevX.7.021050.

K. Temme, S. Bravyi, and J. M. Gambetta. Error mitigation for short-depth quantum circuits. *Phys. Rev. Lett.*, 119:180509, 2017. doi: 10.1103/PhysRevLett.119.180509.

J. R. McClean, M. E. Kimchi-Schwartz, J. Carter, and W. A. de Jong. Hybrid quantum-classical hierarchy for mitigation of decoherence and determination of excited states. *Phys. Rev. A*, 95:042308, 2017. doi: 10.1103/PhysRevA.95.042308.

X. Bonet-Monroig, R. Sagastizabal, M. Singh, and T. E. O'Brien. Low-cost error mitigation by symmetry verification. *Phys. Rev. A*, 98:062339, 2018. doi: 10.1103/PhysRevA.98.062339.

S. Endo, S. C. Benjamin, and Y. Li. Practical quantum error mitigation for near-future applications. *Phys. Rev. X*, 8:031027, 2018. doi: 10.1103/PhysRevX.8.03102.

S. McArdle, X. Yuan, and S. Benjamin. Error-mitigated digital quantum simulation. *Phys. Rev. Lett.*, 122:180501, 2019. doi: 10.1103/PhysRevLett.122.180501.

S. Endo, Q. Zhao, Y. Li, S. Benjamin, and X. Yuan. Mitigating algorithmic errors in a hamiltonian simulation. *Phys. Rev. A*, 99:012334, Jan 2019. doi: 10.1103/PhysRevA.99.012334.

A. Kandala, K. Temme, A. D. Córcoles, et al. Error mitigation extends the computational reach of a noisy quantum processor. *Nat.*, 567:491, 2019. doi: 10.1038/s41586-019-1040-7.

J. R. McClean, Z. Jiang, N. C. Rubin, R. Babbush, and H. Neven. Decoding quantum errors with subspace expansions. *Nat. Comm.*, 11(1), 2020. doi: 10.1038/s41467-020-14341-w.

M. Otten and S. K. Gray. Recovering noise-free quantum observables. *Phys. Rev. A*, 99: 012338, 2019a. doi: 10.1103/PhysRevA.99.012338.

M. Otten and S. K. Gray. Accounting for errors in quantum algorithms via individual error reduction. *npj Quantum Inf.*, 5:11, 2019b. doi: 10.1038/s41534-019-0125-3.

R. Sagastizabal et al. Experimental error mitigation via symmetry verification in a variational quantum eigensolver. *Phys. Rev. A*, 100:010302, 2019. doi: 10.1103/PhysRevA.100.010302.

M. Urbanek, B. Nachman, and W. A. de Jong. Quantum error detection improves accuracy of chemical calculations on a quantum computer. *arXiv:1910.00129*, 2019. URL https://arxiv.org/abs/1910.00129.

O. Crawford, B. van Straaten, D. Wang, T. Parks, E. Campbell, and S. Brierley. Efficient quantum measurement of pauli operators in the presence of finite sampling error. *arXiv:1908.06942*, 2019. URL https://arxiv.org/abs/1908.06942.

B. Chungheon, O. Tomohiro, T. Seigo, and C. Byung-Soo. Density matrix simulation of quantum error correction codes for near-term quantum devices. *Quantum Sci. Technol.*, 5 (1):015002, 2019. doi: 10.1088/2058-9565/ab5887.

A. D. Córcoles, E. Magesan, S. J. Srinivasan, A. W. Cross, M. Steffen, J. M. Gambetta, and J. M. Chow. Demonstration of a quantum error detection code using a square lattice of four superconducting qubits. *Nat. Commun.*, 6(1):6979, 2015. doi: 10.1038/ncomms7979.

S. Sheldon, E. Magesan, J. M. Chow, and J. M. Gambetta. Procedure for systematically tuning up cross-talk in the cross-resonance gate. *Phys. Rev. A*, 93(6):060302, 2016. doi: 10.1103/PhysRevA.93.060302.

S. S. Tannu and M. K. Qureshi. Mitigating measurement errors in quantum computers by exploiting state-dependent bias. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '52, page 279, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450369381. doi: 10.1145/3352460.3358265.

K. Yeter-Aydeniz, E. F. Dumitrescu, A. J. McCaskey, R. S. Bennink, R. C. Pooser, and G. Siopsis. Scalar quantum field theories as a benchmark for near-term quantum computers. *Phys. Rev. A*, 99:032306, Mar 2019. doi: 10.1103/PhysRevA.99.032306.

K. Yeter-Aydeniz, R. C. Pooser, and G. Siopsis. Practical quantum computation of chemical and nuclear energy levels using quantum imaginary time evolution and lanczos algorithms. *npj Quantum Information*, 6(1), jul 2020. doi: 10.1038/s41534-020-00290-1.

Lena Funcke, Tobias Hartung, Karl Jansen, Stefan Kühn, Paolo Stornati, and Xiaoyang Wang. Measurement error mitigation in quantum computers through classical bit-flip correction. 2020. arXiv:2007.03663.

Lena Funcke, Tobias Hartung, Karl Jansen, Stefan Kühn, and Paolo Stornati. Dimensional Expressivity Analysis of Parametric Quantum Circuits. *Quantum*, 5:422, March 2021. ISSN 2521-327X. doi: 10.22331/q-2021-03-29-422. URL https://doi.org/10.22331/q-2021-03-29-422.

J. I. Cirac and P. Zoller. Quantum computations with cold trapped ions. *Phys. Rev. Lett.*, 74: 4091–4094, May 1995. doi: 10.1103/PhysRevLett.74.4091. URL https://link.aps.org/doi/10.1103/PhysRevLett.74.4091.

Ibm quantum. https://quantum-computing.ibm.com. Accessed: 2021-04-14.

Qiskit Aer API documentation and source code. Accessed on 15/02/2021.

Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. doi: 10.1017/CBO9780511976667.

Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5(1), Jul 2014. ISSN 2041-1723. doi: 10.1038/ncomms5213. URL http://dx.doi.org/10.1038/ncomms5213.

Ken M. Nakanishi, Keisuke Fujii, and Synge Todo. Sequential minimal optimization for quantum-classical hybrid algorithms. *Physical Review Research*, 2(4), Oct 2020. ISSN 2643-1564. doi: 10.1103/physrevresearch.2.043158. URL http://dx.doi.org/10.1103/PhysRevResearch.2.043158.

C. Alexandrou, L. Funcke, T. Hartung, K. Jansen, S. Kuehn, G. Polykratis, P. Stornati, and X. Wang. Using classical bit-flip correction for error mitigation including 2-qubit correlations, 2021.

M. R. Geller. Sampling and scrambling on a chain of superconducting qubits. *Physical Review Applied*, 10(2), 2018. ISSN 2331-7019. doi: 10.1103/physrevapplied.10.024052.

S. Sim, P. D. Johnson, and A. Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, 2(12):1900070, 2019. ISSN 2511-9044. doi: 10.1002/qute.201900070.

M. Bataille. Quantum circuits of cnot gates. *arXiv:2009.13247*, 2020. URL https://arxiv.org/abs/2009.13247.

S. Sim, J. Romero, J. F. Gonthier, and A. A. Kunitsa. Adaptive pruning-based optimization of parameterized quantum circuits. *arXiv:2010.00629*, 2020. URL https://arxiv.org/abs/2010.00629.

S. E. Rasmussen, N. J. S. Loft, T. Bækkegaard, M. Kues, and N. T. Zinner. Reducing the amount of single-qubit rotations in vqe and related algorithms. *Advanced Quantum Technologies*, 3:2000063, 2020. doi: 10.1002/qute.202000063.

T. Hubregtsen, J. Pichlmeier, P. Stecher, and K. Bertels. Evaluation of parameterized quantum circuits: on the relation between classification accuracy, expressibility and entangling capability. *arXiv:2003.09887*, 2020. URL https://arxiv.org/abs/2003.09887.

M. Schuld, R. Sweke, and J. J. Meyer. The effect of data encoding on the expressive power of variational quantum machine learning models. *arXiv:2008.08605*, 2020. URL https://arxiv.org/abs/2008.08605.

E. Fontana, N. Fitzpatrick, D. Muños Ramo, R. Duncan, and I. Rungger. Evaluating the noise resilience of variational quantum algorithms. *arXiv:2011.01125*, 2020. URL https://arxiv.org/abs/2011.01125.

B. T. Gard, L. Zhu, G. S. Barron, N. J. Mayhall, S. E. Economou, and E. Barnes. Efficient symmetry-preserving state preparation circuits for the variational quantum eigensolver algorithm. *npj Quantum Information*, 6(1), 2020. ISSN 2056-6387. doi: 10.1038/s41534-019-0240-1.

G. S. Barron, B. T. Gard, O. J. Altman, N. J. Mayhall, E. Barnes, and S. E. Economou. Preserving symmetries for variational quantum eigensolvers in the presence of noise. *arXiv:2003.00171*, 2020. URL https://arxiv.org/abs/2003.00171.

Joonho Kim, Jaedeok Kim, and Dario Rosa. Universal effectiveness of high-depth circuits in variational eigenproblems. *arXiv:2010.00157*, 2020. URL https://arxiv.org/abs/2010.00157.

H. Abraham et al. Qiskit: An Open-source Framework for Quantum Computing. *Zenodo*, 2019. doi: 10.5281/zenodo.2562111.

J. Eisert. Entanglement and tensor network states, 2013.

L. Zhao, Z. Zhao, P. Rebentrost, and J. Fitzsimons. Compiling basic linear algebra subroutines for quantum computers. *1902.10394*, 2019. URL https://arxiv.org/abs/1902.10394.

Roeland Wiersema, Cunlu Zhou, Yvette de Sereville, Juan Felipe Carrasquilla, Yong Baek Kim, and Henry Yuen. Exploring entanglement and optimization within the hamiltonian variational ansatz. *PRX Quantum*, 1:020319, Dec 2020. doi: 10.1103/PRXQuantum.1.020319. URL https://link.aps.org/doi/10.1103/PRXQuantum.1.020319.

Mari Carmen Bañuls, Rainer Blatt, Jacopo Catani, Alessio Celi, Juan Ignacio Cirac, Marcello Dalmonte, Leonardo Fallani, Karl Jansen, Maciej Lewenstein, Simone Montangero, and et al. Simulating lattice gauge theories within quantum technologies. *The European Physical Journal D*, 74(8), Aug 2020. ISSN 1434-6079. doi: 10.1140/epjd/e2020-100571-8. URL http://dx.doi.org/10.1140/epjd/e2020-100571-8.

Mari Carmen Bañuls, Krzysztof Cichy, J. Ignacio Cirac, Karl Jansen, and Hana Saito. Matrix product states for lattice field theories, 2013a.

M.C. Bañuls, K. Cichy, J.I. Cirac, and K. Jansen. The mass spectrum of the schwinger model with matrix product states. *Journal of High Energy Physics*, 2013(11), Nov 2013b. ISSN 1029-8479. doi: 10.1007/jhep11(2013)158. URL http://dx.doi.org/10.1007/JHEP11(2013)158.

Ferdinand Tschirsich, Simone Montangero, and Marcello Dalmonte. Phase diagram and conformal string excitations of square ice using gauge invariant matrix product states. *SciPost Physics*, 6(3), Mar 2019. ISSN 2542-4653. doi: 10.21468/scipostphys.6.3.028. URL http://dx.doi.org/10.21468/SciPostPhys.6.3.028.

S Chandrasekharan and U.-J Wiese. Quantum link models: A discrete approach to gauge theories. *Nuclear Physics B*, 492(1-2):455–471, May 1997. ISSN 0550-3213. doi: 10.1016/s0550-3213(97)80041-7. URL http://dx.doi.org/10.1016/S0550-3213(97)80041-7.

Pietro Silvi, Enrique Rico, Tommaso Calarco, and Simone Montangero. Lattice gauge tensor networks. *New Journal of Physics*, 16(10):103015, Oct 2014a. ISSN 1367-2630. doi: 10.1088/1367-2630/16/10/103015. URL http://dx.doi.org/10.1088/1367-2630/16/10/103015.

Simone Montangero. *Introduction to Tensor Network Methods: Numerical simulations of low-dimensional many-body quantum systems*. 01 2018. ISBN 978-3-030-01408-7. doi: 10.1007/978-3-030-01409-4.

Falk Bruckmann, Karl Jansen, and Stefan Kühn. O(3) nonlinear sigma model in $1+1$ dimensions with matrix product states. *Phys. Rev. D*, 99:074501, Apr 2019. doi: 10.1103/PhysRevD.99.074501. URL https://link.aps.org/doi/10.1103/PhysRevD.99.074501.

L. Tagliacozzo, A. Celi, and M. Lewenstein. Tensor networks for lattice gauge theories with continuous groups. *Phys. Rev. X*, 4:041024, Nov 2014. doi: 10.1103/PhysRevX.4.041024. URL https://link.aps.org/doi/10.1103/PhysRevX.4.041024.

Boye Buyens, Jutho Haegeman, Karel Van Acoleyen, Henri Verschelde, and Frank Verstraete. Matrix product states for gauge field theories. *Physical Review Letters*, 113(9), Aug 2014. ISSN 1079-7114. doi: 10.1103/physrevlett.113.091601. URL http://dx.doi.org/10.1103/PhysRevLett.113.091601.

Pietro Silvi, Enrique Rico, Tommaso Calarco, and Simone Montangero. Lattice Gauge Tensor Networks. *New J. Phys.*, 16(10):103015, 2014b. doi: 10.1088/1367-2630/16/10/103015.

G. Evenbly and G. Vidal. Tensor network states and geometry. *Journal of Statistical Physics*, 145(4):891–918, Jun 2011. ISSN 1572-9613. doi: 10.1007/s10955-011-0237-4. URL http://dx.doi.org/10.1007/s10955-011-0237-4.

Michael Levin and Cody P. Nave. Tensor renormalization group approach to two-dimensional classical lattice models. *Phys. Rev. Lett.*, 99:120601, Sep 2007. doi: 10.1103/PhysRevLett.99.120601. URL https://link.aps.org/doi/10.1103/PhysRevLett.99.120601.

G. Evenbly and G. Vidal. Tensor network renormalization. *Phys. Rev. Lett.*, 115:180405, Oct 2015. doi: 10.1103/PhysRevLett.115.180405. URL https://link.aps.org/doi/10.1103/PhysRevLett.115.180405.

Mari Carmen Bañuls, Krzysztof Cichy, J. Ignacio Cirac, Karl Jansen, and Stefan Kühn. Density induced phase transitions in the schwinger model: A study with matrix product states. *Phys. Rev. Lett.*, 118:071601, Feb 2017. doi: 10.1103/PhysRevLett.118.071601. URL https://link.aps.org/doi/10.1103/PhysRevLett.118.071601.

Roger Penrose. Applications of negative dimensional tensors. *Combinatorial mathematics and its applications*, 1:221–244, 1971.

Ulrich Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326(1):96–192, Jan 2011. ISSN 0003-4916. doi: 10.1016/j.aop.2010.09.012. URL http://dx.doi.org/10.1016/j.aop.2010.09.012.

Steven R. White. Density matrix formulation for quantum renormalization groups. *Phys. Rev. Lett.*, 69:2863–2866, Nov 1992. doi: 10.1103/PhysRevLett.69.2863. URL https://link.aps.org/doi/10.1103/PhysRevLett.69.2863.

D. Perez-Garcia, F. Verstraete, M. M. Wolf, and J. I. Cirac. Matrix Product State Representations. *arXiv e-prints*, art. quant-ph/0608197, August 2006.

Guifré Vidal. Efficient classical simulation of slightly entangled quantum computations. *Physical Review Letters*, 91(14), Oct 2003. ISSN 1079-7114. doi: 10.1103/physrevlett.91.147902. URL http://dx.doi.org/10.1103/PhysRevLett.91.147902.

Tobias J. Osborne and Michael A. Nielsen. *Quantum Information Processing*, 1(1/2):45–53, 2002. ISSN 1570-0755. doi: 10.1023/a:1019601218492. URL http://dx.doi.org/10.1023/A:1019601218492.

Pasquale Calabrese and John Cardy. Entanglement entropy and conformal field theory. *Journal of Physics A: Mathematical and Theoretical*, 42(50):504005, Dec 2009. ISSN 1751-8121. doi: 10.1088/1751-8113/42/50/504005. URL http://dx.doi.org/10.1088/1751-8113/42/50/504005.

G. Vidal, J. I. Latorre, E. Rico, and A. Kitaev. Entanglement in quantum critical phenomena. *Physical Review Letters*, 90(22), Jun 2003. ISSN 1079-7114. doi: 10.1103/physrevlett.90.227902. URL http://dx.doi.org/10.1103/PhysRevLett.90.227902.

J. Eisert, M. Cramer, and M. B. Plenio. Colloquium: Area laws for the entanglement entropy. *Rev. Mod. Phys.*, 82:277–306, Feb 2010. doi: 10.1103/RevModPhys.82.277. URL https://link.aps.org/doi/10.1103/RevModPhys.82.277.

M B Hastings. An area law for one-dimensional quantum systems. *Journal of Statistical Mechanics: Theory and Experiment*, 2007(08):P08024–P08024, aug 2007. doi: 10.1088/1742-5468/2007/08/p08024. URL https://doi.org/10.1088/1742-5468/2007/08/p08024.

Cornelius Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Natl. Bur. Stand. B*, 45:255–282, 1950. doi: 10.6028/jres.045.026.

D. Horn. Finite matrix models with continuous local gauge invariance. *Physics Letters B*, 100(2):149–151, 1981. ISSN 0370-2693. doi: https://doi.org/10.1016/0370-2693(81)90763-2. URL https://www.sciencedirect.com/science/article/pii/0370269381907632.

Peter Orland and Daniel Rohrlich. Lattice gauge magnets: Local isospin from spin. *Nuclear Physics B*, 338(3):647–672, 1990. ISSN 0550-3213. doi: https://doi.org/10.1016/0550-3213(90)90646-U. URL https://www.sciencedirect.com/science/article/pii/055032139090646U.

R. Brower, S. Chandrasekharan, and U.-J. Wiese. Qcd as a quantum link model. *Physical Review D*, 60(9), Sep 1999. ISSN 1089-4918. doi: 10.1103/physrevd.60.094502. URL http://dx.doi.org/10.1103/PhysRevD.60.094502.

David B. Kaplan. A method for simulating chiral fermions on the lattice. *Physics Letters B*, 288(3-4):342–347, Aug 1992. ISSN 0370-2693. doi: 10.1016/0370-2693(92)91112-m. URL http://dx.doi.org/10.1016/0370-2693(92)91112-M.

Daniel S. Rokhsar and Steven A. Kivelson. Superconductivity and the quantum hard-core dimer gas. *Phys. Rev. Lett.*, 61:2376–2379, Nov 1988. doi: 10.1103/PhysRevLett.61.2376. URL https://link.aps.org/doi/10.1103/PhysRevLett.61.2376.

Wei Zheng and Subir Sachdev. Sine-gordon theory of the non-néel phase of two-dimensional quantum antiferromagnets. *Phys. Rev. B*, 40:2704–2707, Aug 1989. doi: 10.1103/PhysRevB.40.2704. URL https://link.aps.org/doi/10.1103/PhysRevB.40.2704.

N. Read and Subir Sachdev. Spin-peierls, valence-bond solid, and néel ground states of low-dimensional quantum antiferromagnets. *Phys. Rev. B*, 42:4568–4589, Sep 1990. doi: 10.1103/PhysRevB.42.4568. URL https://link.aps.org/doi/10.1103/PhysRevB.42.4568.

B. Schlittgen and U.-J. Wiese. Low-energy effective theories of quantum spin and quantum link models. *Phys. Rev. D*, 63:085007, Mar 2001. doi: 10.1103/PhysRevD.63.085007. URL https://link.aps.org/doi/10.1103/PhysRevD.63.085007.

Gyan Bhanot and Michael Creutz. Variant actions and phase structure in lattice gauge theory. *Phys. Rev. D*, 24:3212–3217, Dec 1981. doi: 10.1103/PhysRevD.24.3212. URL https://link.aps.org/doi/10.1103/PhysRevD.24.3212.

D. Banerjee, F. J. Jiang, P. Widmer, and U. J. Wiese. The (2 + 1)-d U(1) quantum link model masquerading as deconfined criticality. *J. Stat. Mech.*, 1312:P12010, 2013. doi: 10.1088/1742-5468/2013/12/P12010.

Maciej Lewenstein, Anna Sanpera, Veronica Ahufinger, Bogdan Damski, Aditi Sen(De), and Ujjwal Sen. Ultracold atomic gases in optical lattices: mimicking condensed matter physics and beyond. *Advances in Physics*, 56(2):243–379, Mar 2007. ISSN 1460-6976. doi: 10.1080/00018730701223200. URL http://dx.doi.org/10.1080/00018730701223200.

Pascual Jordan and Eugene P. Wigner. About the Pauli exclusion principle. *Z. Phys.*, 47: 631–651, 1928. doi: 10.1007/BF01331938.

David J. Gross and Andre Neveu. Dynamical Symmetry Breaking in Asymptotically Free Field Theories. *Phys. Rev. D*, 10:3235, 1974. doi: 10.1103/PhysRevD.10.3235.

Julian Lenz, Laurin Pannullo, Marc Wagner, Björn Wellegehausen, and Andreas Wipf. Inhomogeneous phases in the gross-neveu model in 1 + 1 dimensions at finite number of flavors. *Physical Review D*, 101, 05 2020. doi: 10.1103/PhysRevD.101.094512.

List of Figures

List of Tables

Acknowledgments

There is a long list of people I want to thank for this work. I think that writing their names here would be superfluous. My biggest gratitude goes to Karl Jansen for giving me the opportunity and the privilege to work with him and for his knowledgeable guidance. I am also very grateful to Agostino Patella for being my tutor at Humboldt University, for his support and his advice in these three years. Thanks to all my collaborators and colleagues for all the valuable discussions and the always friendly interactions.

Thanks to all my friends in Berlin that always made me feel at home in this city. For the next part, I will switch to Italian. Voglio ringraziare tutta la mia famiglia, senza la quale, tutto questo non sarebbe mai stato possibile. Da essa ho sempre ricevuto un incondizionato supporto, senza il quale non sarei mai riuscito a raggiungere tutti questo risultati. A tutti voi dico semplicemente: Grazie!

Erklärung

Ich erkläre, dass ich die vorliegende Dissertation selbständig und nur unter Verwendung der von mir gemäß § 7 Abs. 3 der Promotionsordnung der Mathematisch-Naturwissenschaftlichen Fakultät, veröffentlicht im Amtlichen Mitteilungsblatt der Humboldt-Universität zu Berlin Nr. 42/2018 am 11/07/2018, angegebenen Hilfsmittel angefertigt habe.

Berlin, March 21, 2022

Paolo Stornati