Università degli Studi di Milano - Bicocca
Dipartimento di Fisica "G. Occhialini"
Corso di Laurea Triennale in Fisica

Quantum Simulation of the Ising Model

Supervisor:
Prof. Simone Alioli

Candidate:
Riccardo Colombo
Registration number:
896763

Academic Year 2024-2025

Contents

1 Introduction

The first idea of using quantum computers to simulate quantum systems is attributed to Richard Feynman, who proposed it in a paper named *Simulating Physics with Computers*[8] and published in 1982. He argued that reproducing the quantum behaviour of a microscopic system via a classical computer is intrinsically complicated and computationally expensive, while a quantum computer could overcome this difficulty, since it would have a quantum nature itself. This remained a conjecture for many years, but due to theoretical and technological advancements in the fields of *Quantum Information* and *Quantum Computation* it is nowadays a feasible task. This achievement was reached also thanks to the discovery of new applications of quantum computing, the most famous ones being Shor's algorithm for the factorization of numbers and Grover's research algorithm (published respectively by Peter Shor in 1994[5] and by Lov Grover in 1996[6]), which boosted the enthusiasm for this field by promising to solve other important practical problems of computer and information sciences.

The goal of this thesis is to present a simple simulation of a *1D Quantum Ising chain*, performed on a quantum computer of IBM's platform *IBM Quantum Platform*[4]. The 1D Ising Model is commonly reproduced on quantum computers: the reason behind this lies in the fact that it has an exact analytical solution, so it is particularly useful to evaluate the efficiency of a quantum computer.

The first chapter of this work covers the basic principles of quantum computing and the mathematical concepts behind it. It explains what is a qubit, its differences from classical bits and how it can be manipulated by operators called quantum gates, with a focus on simulation algorithms.

The second chapter contains an introduction to the Ising Model. It starts with a general description of the theory and then there is a famous technique used to deal with 1D Ising spin chains: the Jordan-Wigner transformation.

The third chapter is about the Fourier transform, an operator often used in quantum mechanics to transfer to the momentum space. In the case of this work, it is implemented in a step of the analytical solution of the problem described in the fourth part.

Finally, the fourth and final chapter covers the solution of finding the time evolution of 1D Ising chain. It describes first the analytical solution and the circuit used to implement it on a quantum computer, then the actual implementation of the simulation, with an analysis of the results.

2 Quantum Simulations

2.1 Qubits

The *quantum bit* (or in short *qubit*) is the fundamental part of a quantum computer. Like a classical bit, it can assume the two binary values 0 or 1 and represents the smallest piece of information which can be stored and processed. The data in a quantum computer, in fact, will result in sequences of qubits that will be appropriately manipulated.

Unlike classical ones, qubits can be subjected to a phenomenon called *superposition*, which consists in the said object being in more than one state at the same time with different probabilities. This deed influences the kind of algorithms it is possible to perform using a quantum computer.

2.1.1 Single qubit

Conceptually, a qubit is an element of a two-level Hilbert space, which can assume two possible values: 0 or 1.

Its represantation is:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.1}$$

Where α and β are two complex values which define the probability of each state based on the following relations:

$$\begin{cases} P(|\psi\rangle = |0\rangle) = |\alpha|^2 \\ P(|\psi\rangle = |1\rangle) = |\beta|^2 \\ |\alpha|^2 + |\beta|^2 = 1 \end{cases} \tag{2.2}$$

Despite this, when we perform a measurement, the state collapses on one of the two values, whose probability will then become 100%. For a qubit, unlike a classical bit, it is therefore impossible to determine its state perfectly.

$$\alpha\,|0\rangle + \beta\,|1\rangle \xrightarrow{measurement} |0\rangle$$

$$\alpha\,|0\rangle + \beta\,|1\rangle \xrightarrow{measurement} |1\rangle$$

It is important to keep in mind that, due to the superposition principle, a qubit contains more information than a classical one. The latter, indeed, is bound to assuming only one value, while a qubit can describe a probability distribution, so its state can be interpreted as whole spectrum of possibilities depending on the coefficients α and β. This information can be manipulated

and used to accomplish appropriate tasks, but cannot be directly accessed by the user.

It is usual to represent a qubit as a point on the Bloch sphere, a sphere of radius 1 and shown in Fig. 1.

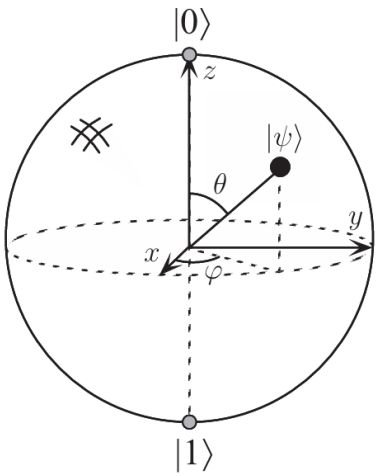


Figure 1: Graphical representation of the Bloch Sphere. Image taken from Ref.[1]

Indeed, since $|\alpha|^2 + |\beta|^2 = 1$, $|\psi\rangle$ can be expressed as:

$$|\psi\rangle = e^{i\gamma_1} \cos\frac{\theta}{2} |0\rangle + e^{i\gamma_2} \sin\frac{\theta}{2} |1\rangle \tag{2.3}$$

Defining $\gamma_2 - \gamma_1 = \varphi$ and neglecting the pure phase factor from equation 2.3, the state becomes:

$$|\psi\rangle = \cos\frac{\theta}{2} |0\rangle + e^{i\varphi} \sin\frac{\theta}{2} |1\rangle \tag{2.4}$$

Where θ and φ are the spherical coordinates on the Bloch sphere as shown in Fig. 1.

2.1.2 Multiple qubits

When we have more than one qubit, the total state is expressed as a combination of the states of N single qubits, which assumes the form:

$$|\psi\rangle = \sum_{j_1=0,1 \ldots j_N=0,1} \alpha_{j_1 \ldots j_N} |j_1 \ldots j_N\rangle \tag{2.5}$$

The terms $|j_1...j_N\rangle$ are tensor products of the elements of the computational basis $\{|0\rangle, |1\rangle\}$, while the $\alpha_{j_1...j_N}$ are normalized coefficients.

$$|j_1...j_N\rangle = \bigotimes_{n=1}^{N} |j_n\rangle \tag{2.6}$$

$$\sum_{j_1=0,1...j_N=0,1} |\alpha_{j_1...j_N}|^2 = 1 \tag{2.7}$$

For example, for two qubits the general representation will be:

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \tag{2.8}$$

An important phenomenon related to this kind of states is the *entanglement*. It occurs when the total state of two quantum systems cannot be expressed as a simple tensor product of two independent ones. An example is:

$$|\psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{2.9}$$

In fact, it is impossible to find two states built on the computational basis $\{|0\rangle, |1\rangle\}$ such that:

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}} = (\alpha_1|0\rangle + \beta_1|1\rangle) \bigotimes (\alpha_2|0\rangle + \beta_2|1\rangle)$$

In this case we cannot describe the two systems independently and the evolution of one will affect the other too.

2.2 Quantum logic gates

Theoretically, a *quantum logic gate* (or in short *quantum gate*) is a linear operator that acts on a 2^N-level system (with N being the number of qubits). The said operator turns a generic quantum state $|\psi\rangle$ into another one $|\psi'\rangle$. The new $|\psi'\rangle$ must be normalized too, this means that a quantum gate preserves the norm of the vector, in other words it is a *unitary operator*. When applying a quantum gate U to $|\psi\rangle$ we obtain:

$$|\psi'\rangle = U|\psi\rangle \tag{2.10}$$

$$\|\psi'\| = \langle\psi'|\psi'\rangle = \langle\psi|U^+U|\psi\rangle = \langle\psi|\psi\rangle = 1 \tag{2.11}$$

Where U^+ is the adjoint of U.

A quantum gate is the equivalent of a logic gate of classical computers, but the main difference is the reversibility property. Indeed, if we apply a series of unitary operators on a quantum state $|\psi_i\rangle$, it will always be possible (except when an error occurs) to transform the final state $|\psi_f\rangle$ in $|\psi_i\rangle$ again.

NOTE: This principle is the basis of quantum error correction algorithms, but in this thesis they are not discussed.

2.2.1 Single qubit gates

A *single qubit gate* is a quantum gate that acts on a single qubit, it therefore performs the task:

$$U(\alpha\,|0\rangle + \beta\,|1\rangle) \longrightarrow \alpha'\,|0\rangle + \beta'\,|1\rangle$$

As previously mentioned, the only constraint of this kind of operator is that it must be *unitary*, therefore the new coefficients α', β' will follow the normalization relation:

$$|\alpha'|^2 + |\beta'|^2 = 1 \tag{2.12}$$

A convenient way of representing a quantum gate is a matrix. Indeed, if we denote the computational basis with

$$|0\rangle \equiv \begin{bmatrix}1\\0\end{bmatrix};\ |1\rangle \equiv \begin{bmatrix}0\\1\end{bmatrix}$$

we can express a single quantum gate as a 2x2 unitary matrix:

$$U \equiv \begin{bmatrix}u_{11} & u_{12}\\ u_{21} & u_{22}\end{bmatrix} \tag{2.13}$$

Its effect on a generic vector will be:

$$U\begin{bmatrix}\alpha\\\beta\end{bmatrix} = \begin{bmatrix}u_{11}\alpha + u_{12}\beta\\ u_{21}\alpha + u_{22}\beta\end{bmatrix} \equiv (u_{11}\alpha + u_{12}\beta)\,|0\rangle + (u_{21}\alpha + u_{22}\beta)\,|1\rangle \tag{2.14}$$

An important gate is the *NOT*, which is represented by the matrix:

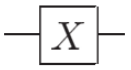
$$X \equiv \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} \tag{2.15}$$

When applied to a qubit it flips the state and generates the result:

$$X(\alpha\,|0\rangle + \beta\,|1\rangle) \equiv X\begin{bmatrix}\alpha\\\beta\end{bmatrix} = \begin{bmatrix}\beta\\\alpha\end{bmatrix} \equiv \beta\,|0\rangle + \alpha\,|1\rangle \tag{2.16}$$

The *NOT* gate is also called *X gate* and is one of a group of three operators named *Pauli matrices*. They are listed below, followed by their circuital representation:

- *X gate*: $X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$



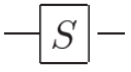
- *Y gate:* $Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$

- *Z gate:* $Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

Other important single qubit gates are:

- *Hadamard gate:* $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

- *S gate:* $S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$



- *T gate (a.k.a. $\pi/8$ gate):* $T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix} = e^{i\frac{\pi}{8}} \begin{bmatrix} e^{-i\frac{\pi}{8}} & 0 \\ 0 & e^{i\frac{\pi}{8}} \end{bmatrix}$
- *phase gate:* $P(\delta) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\delta} \end{bmatrix}$
- $R_x(\theta)$ *gate:* $R_x(\theta) = \begin{bmatrix} cos\frac{\theta}{2} & -isin\frac{\theta}{2} \\ -isin\frac{\theta}{2} & cos\frac{\theta}{2} \end{bmatrix}$
- $R_y(\theta)$ *gate:* $R_y(\theta) = \begin{bmatrix} cos\frac{\theta}{2} & -sin\frac{\theta}{2} \\ sin\frac{\theta}{2} & cos\frac{\theta}{2} \end{bmatrix}$
- $R_z(\theta)$ *gate:* $R_z(\theta) = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}$

NOTE: The last three of this list are called *rotation operators.*

2.2.2 Two qubit gates

For the majority of algorithms we have the necessity to act on more than one qubit at the same time or to act on a qubit depending from the state of a second one. In this case, we will use *multiple qubit gates.* The first class to be presented is called *two qubit gates* and, as the name suggests, they are gates that operate on two qubits simultaneously. The action of such an operator is therefore:

$$U(\alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle) \longrightarrow \alpha'_{00}|00\rangle + \alpha'_{01}|01\rangle + \alpha'_{10}|10\rangle + \alpha'_{11}|11\rangle$$

We can find a matrix representation defining a new computational basis:

$$|00\rangle \equiv \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} ; \ |01\rangle \equiv \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} ; \ |10\rangle \equiv \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} ; \ |11\rangle \equiv \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Thus, *two qubits gates* can then be represented as:

$$U \equiv \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ u_{21} & u_{22} & u_{23} & u_{24} \\ u_{31} & u_{32} & u_{33} & u_{34} \\ u_{41} & u_{42} & u_{43} & u_{44} \end{bmatrix}$$

Important gates of this kind are *controlled gates*. They make use of a secondary qubit named *control qubit*, which we represent as $|j_1\rangle$, and perform a certain operation on another $|j_2\rangle$ if and only if $|j_1\rangle$ is set to $|1\rangle$. Their function is schematized as follows:

$$U_C(\alpha_{00}\,|00\rangle+\alpha_{01}\,|01\rangle+\alpha_{10}\,|10\rangle+\alpha_{11}\,|11\rangle) \longrightarrow \alpha_{00}\,|00\rangle+\alpha_{01}\,|01\rangle+\alpha'_{10}\,|10\rangle+\alpha'_{11}\,|11\rangle$$

The circuital representation of a general controlled gate is shown in Fig.2.

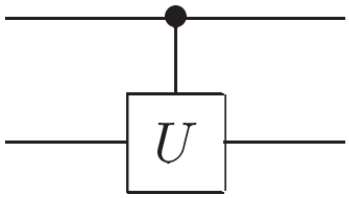


Figure 2: Circuital representation of a controlled gate. Image taken from Ref.[1]

The main gates of this type are:

- *controlled not (or X):* $CX = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
- *controlled Y:* $CY = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \end{bmatrix}$
- *controlled Z:* $CZ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
- *controlled Hadamard:* $CH = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/\sqrt{2} & 1/\sqrt{2} \\ 0 & 0 & 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$

- *controlled S:* $CS = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{bmatrix}$
- *controlled T:* $CT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\frac{\pi}{4}} \end{bmatrix}$

NOTE: Another gate that was used in this work is the *SWAP gate*, which is of the form

$$SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.2.3 Multiple qubit gates

For a generic algorithm we could be unsatisfied of using just two qubit gates. Generally, if we have a gate U that acts on N qubits, we consider it as an operator that acts on a 2^N-level space and it is represented as a $2^N \times 2^N$ matrix.

An important class of multiple qubit gates is composed by the quantum equivalents of two-bit logic gates. These are gates that take two bits as input and, depending on their values, return another bit as output. The said operations act as follows:

- *AND*: it returns 1 if both the bits of input are 1 and 0 otherwise
- *OR*: it returns 1 if at least one bit of input is 1 and 0 otherwise
- *XOR*: it returns 1 if the bits of input are different and 0 otherwise
- *NAND*: it returns 1 if at least a bit of input is 0 and 0 otherwise
- *NOR*: it returns 1 if both the bits of input are 0 and 0 otherwise

In quantum computing it is impossible to reproduce these operations, since they are not reversible, but they are usually emulated using *three qubit quantum gates*. Defining a new convenient computational basis, we can express the equivalent gates as operators that maintain the first two

qubits of a state intact and flips the third one if the appropriate condition is satisfied. For example, if we have the state

$$|\psi\rangle = \alpha\,|000\rangle + \beta\,|010\rangle + \gamma\,|100\rangle + \delta\,|110\rangle$$

when these gates are applied we get:

- $U_{AND}\,|\psi\rangle \to \alpha\,|000\rangle + \beta\,|010\rangle + \gamma\,|100\rangle + \delta\,|111\rangle$
- $U_{OR}\,|\psi\rangle \to \alpha\,|000\rangle + \beta\,|011\rangle + \gamma\,|101\rangle + \delta\,|111\rangle$
- $U_{XOR}\,|\psi\rangle \to \alpha\,|000\rangle + \beta\,|011\rangle + \gamma\,|101\rangle + \delta\,|110\rangle$
- $U_{NAND}\,|\psi\rangle \to \alpha\,|001\rangle + \beta\,|011\rangle + \gamma\,|101\rangle + \delta\,|110\rangle$
- $U_{NOR}\,|\psi\rangle \to \alpha\,|001\rangle + \beta\,|010\rangle + \gamma\,|100\rangle + \delta\,|110\rangle$

All these operations can be built using an appropriate sequence of a CNOT gate and single qubit gates. If we want to generalize this concept to N qubit gates, we can say that it is possible to build an operator of this kind with an arbitrary precision by constructing a sequence of CNOTs and two qubit gates. An example of a set of gates that let perform such a task is $\{CNOT, H, S, T\}$. A family of unitary operators like this is called *universal set of gates*.

2.3 Quantum simulation algorithms

As it has been said in the introduction, the idea of quantum computing was born in order to simulate physical systems more efficiently. Although for many phenomena it could be easier to use classical computers (since it could be necessary to compute only the classical behaviour), on a microscopic level or when we want to replicate quantum behaviour they become very inefficient, because the number of bits required to represent an acceptable approximation of a quantum system grows exponentially when the particles increase. To overcome this problem, quantum computers are, by their nature, optimal tools, but building good quantum algorithms to simulate physical phenomena might not be as straightforward as one may think.

When performing an algorithm, we essentially receive an input (our initial data) and finally produce an output (the information we want to retrieve). In our case, we have as input an initial quantum state (let us say $|\psi(0)\rangle$) and then receive as output the final state at a given time ($|\psi(t)\rangle$). After defining the premises, we have to think about a set of elementary actions that let us get from our beginning data to the completely processed ones, in other words a set of intermediate steps.

$$|\psi(0)\rangle \xrightarrow{1^{st}\text{ step}} |\psi_1\rangle \xrightarrow{2^{nd}\text{ step}} ... \xrightarrow{n^{th}\text{ step}} |\psi_n\rangle \xrightarrow{\text{final step}} |\psi(t)\rangle$$

In quantum mechanics it is possible to determine the final state via an operator called *evolution operator*. It is defined starting from Schrödinger's equation, which describes how a quantum system changes over time. It is of the form:

$$i\hbar\frac{\partial\psi}{\partial t} = H\psi \tag{2.17}$$

H is an operator called Hamiltonian and its eigenvalues represent the possible energies of the system. Given this, the state of the system at a time t is calculated:

$$|\psi(t)\rangle = e^{-\frac{i}{\hbar}Ht}\,|\psi(0)\rangle \tag{2.18}$$

Where $e^{-\frac{i}{\hbar}Ht}$ is the *evolution operator* (that from now on we will call $U_{ev}(t)$).

In quantum computing, the evolution operator can be replicated as a quantum gate and, theoretically, once we do this, the final state is computed as:

$$|\psi(t)\rangle = U_{ev}(t)\,|\psi(0)\rangle \tag{2.19}$$

The problem in doing this is that most of the times it is impossible to build a perfect U_{ev} with a universal set of quantum gates, but it is instead necessary to find an approximation. In this case, we could find one considering the Hamiltonian as a sum of n local and more simple Hamiltonians and then build the total U_{ev} starting from these newly-found local operators, that we suppose easier to calculate. The intuitive process that one might come up with is:

$$H = \sum_{j=1}^{n} H_j \tag{2.20}$$

$$U_{ev} = e^{-\frac{i}{\hbar}Ht} = \prod_{j=1}^{n} e^{-\frac{i}{\hbar}H_j t} \tag{2.21}$$

This last expression, although in some cases actually leads to the correct U_{ev}, is not generally true. In the next section it will be discussed when we can use Eq. 2.21 to compute the evolution operator and an alternative method when this is not possible.

2.3.1 Trotterization method

Let us suppose that we have a quantum system of N particles, whose state is described by the vector $|\psi\rangle$ that evolves based on the Hamiltonian H.

In many systems of the like, in nature we can identify two-body and local interactions, so H is written as:

$$H = \sum_{j=1}^{P(N)} H_j \tag{2.22}$$

In this expression $P(N)$ is a polynomial in N and the second term contains both two-body terms and one-body terms.

Let us now define the evolution operator of the system:

$$U_{ev}(t) = e^{-\frac{i}{\hbar}(\sum_{j=1}^{P(N)} H_j)t} \tag{2.23}$$

It is not always possible to express U_{ev} as a simple product of local operators of the form:

$$U_{j,local}(t) = e^{-\frac{i}{\hbar}H_j t} \tag{2.24}$$

The cases in which this is permitted are those in which the following relation is true for each $j, k \in \{1, ..., P(N)\}$:

$$[H_j, H_k] = H_j H_k - H_k H_j = 0 \tag{2.25}$$

Indeed, when this is true, for each value of t it holds the relation:

$$e^{-\frac{i}{\hbar}(H_j + H_k)t} = e^{-\frac{i}{\hbar}H_j t} \cdot e^{-\frac{i}{\hbar}H_k t} \tag{2.26}$$

It is possible to demonstrate by induction that, considering n as the number of local interactions in the total Hamiltonian, Eq. 2.26 and Eq. 2.25 lead to Eq. 2.21.

If the said condition 2.25 does not hold, then the total evolution operator cannot be calculated this way. Fortunately, though, there is a law called *Trotter Formula* that lets us approximate U_{ev} with an arbitrary precision. It had been proposed and demonstrated by H. F. Trotter in 1959[13], who found out that an operator $e^{(A+B)t}$ can be expressed as the limit:

$$e^{(A+B)t} = \lim_{m \to \infty} \left(e^{At/m} \cdot e^{Bt/m} \right)^m \tag{2.27}$$

This relation is true even when e^{At} and e^{Bt} do not commute for each value of t. Applying it to our specific case, we can obtain the evolution operator of a quantum system as:

$$U_{ev} = \lim_{m \to \infty} \left(\prod_{j=1}^{n} e^{-\frac{iH_j t}{\hbar m}} \right)^m \tag{2.28}$$

2.3.2 Quantum simulation scheme

In practical applications, when we want to build the evolution operator using the formula 2.28, we will choose a sufficiently large number m and repeat the single applications of all the local evolution operators for m times. The process is schematized below:

1. the initial state $|\psi(0)\rangle$ and the final time t are defined;
2. a precision to which approximate $|\psi(0)\rangle$ is chosen and an appropriate state $|\psi'_0\rangle$ is built;
3. a number m sufficiently large is chosen;
4. all the H_j that constitute H are identified;
5. all the appropriately approximated $U_j = e^{-\frac{iH_j t}{\hbar m}}$ are constructed;
6. the subsequent states $|\psi'_k\rangle$ are computed applying all the U_j operators for m times
$$|\psi'_0\rangle \rightarrow |\psi'_1\rangle \rightarrow ... \rightarrow |\psi'_m\rangle$$
7. $|\psi(t)\rangle = |\psi'_m\rangle$

NOTE: This method brings an error. This depend from the fact that in reality we never have an ideal application in which we can reach an arbitrary precision, but we are instead limited by the available computational resources and time.

NOTE: If all the H_j follow the relation 2.25 it is not necessary to undergo this process, since it would be possible to exactly build the evolution operator U_{ev} with the formula 2.21 and then apply it just once to $|\psi(0)\rangle$. In the simulation of the last chapter this property could be used in order to lower the error.

3 Ising Model

The *Ising model* is a theory proposed by the German physicist Wilhelm Lenz in 1920, in order to describe the phenomenon of *ferromagnetism*. In fact, in nature, some materials (referred to as *ferromagnetic materials*) exhibit the property of remaining magnetized even when an external magnetic field is not applied, and this had been an open problem for a long time in physics. Lenz hypothesized that the atoms in these materials can be treated as a linear lattice and then represent the magnetization with the alignment of some intrinsic angular momenta of these particles (years later Wolfgang Pauli hypothesized the concept of *spin*, exactly the kind of intrinsic angular momentum that appear in this theory). The Hamiltonian of a system of this type is built starting from two families of terms:

- *nearest neighbour interactions*: it is taken into account the coupling of the spins of the atoms in the lattice; moreover, it is supposed that the intensity of these interactions reduces exponentially when the distance between the particles increases, so terms referred to sites distant more than one place are neglected. The contribution of each one of these interactions is set by the relative coefficient κ_{ij};
- *couplings of the spins and the magnetic field*: following the same rules of a general angular momentum, the spin of a particle is associated to a potential by coupling with an external magnetic field. In the case of the Ising model, for the sake of simplicity, the magnetic field is supposed to be oriented along the z component of the spin, and the contributions of these terms are set by the coefficients λ_k.

NOTE: In 1920 Lenz did not have the knowledge of spin that we have today. In this work, this word is used because, after the discovery of this property of particles, the terms σ_j^i were exactly identified in the spin operators.

The Hamiltonian found by Lenz, therefore, assumes the form:

$$H = \sum_{j=1}^{N} \kappa_j \vec{\sigma}_j \cdot \vec{\sigma}_{j+1} + \sum_{k=1}^{N} \lambda_k \sigma_k^z = \sum_{j=1}^{N} \sum_{i=1}^{3} \kappa_{ij} \sigma_j^i \sigma_{j+1}^i + \sum_{k=1}^{N} \lambda_k \sigma_k^z \qquad (3.1)$$

Where N sets the number of sites that compose the system and κ_j is a matrix of the form:

$$\kappa_j = \begin{pmatrix} \kappa_{1j} & 0 & 0 \\ 0 & \kappa_{2j} & 0 \\ 0 & 0 & \kappa_{3j} \end{pmatrix}$$

In some cases this operator takes a more complicated form than that of Eq. 3.1. This can come from phenomena like the presence of a transverse magnetization or from the set of specific boundary conditions. This work is focused on simulating the 1D case with periodic boundary conditions and a transverse magnetic field, which will be better discussed in the next sections.

3.1 1D Ising chain

Although it had been proposed by Lenz, the model was solved by one of his doctoral students, *Ernst Ising*, who found a solution of the one-dimensional case in 1924. Years later the two-dimensional case was exactly solved by Lars Onsanger, while the three-dimensional one is still an open problem.

In the specific case of the 1D model, the Hamiltonian assumes the form:

$$H = \sum_{j=1}^{N} \kappa_j \sigma_j \sigma_{j+1} + \sum_{k=1}^{N} \lambda_k \sigma_k \tag{3.2}$$

A usual approximation is obtained by setting constant all the coefficients κ_j and so the λ_k, thus the Eq. 3.2 becomes:

$$H = \kappa \sum_{j=1}^{N} \sigma_j \sigma_{j+1} + \lambda \sum_{k=1}^{N} \sigma_k \tag{3.3}$$

This is the problem initially solved by Ising. Given the fact that this is not the case treated in this work, we will not focus on its exact solution, but we instead present some of its properties:

- this chain is ferromagnetic if $\kappa < 0$ and antiferromagnetic if $\kappa > 0$. In the first case, in fact, the spins of the particles tend to orient in the same direction in order to minimize the energy, thus obtaining a non-null magnetic moment. In the second case, instead, the spins orient in opposite directions, so the contribution to the mean magnetic moment of two neighbours is 0;
- the external magnetic field forces the spins to align in the same direction, while the thermal noise produces irregularities;
- when the external field is set to 0, no phase transition is observed for non-null finite temperatures.

The model simulated in this work considers the magnetic field to be transverse instead of aligned along the same axis of the spins. The next sections explain how to solve this problem using a technique called *Jordan-Wigner transformation*.

3.2 Jordan-Wigner transformation

The *Jordan-Wigner transformation* maps the spins of fermions of spin 1/2 onto fermionic creation and annihilation operators. It was published in 1928 by Pascual Jordan and Eugene Wigner[12] and it is particularly useful to solve one-dimensional and two-dimensional lattice models. In this thesis it is used to work on a 1D Ising chain with a transverse field, in which the Hamiltonian of the system has the form:

$$H = \kappa \sum_{j=1}^{N} \sigma_j^x \sigma_{j+1}^x + \lambda \sum_{j=1}^{N} \sigma_j^z \tag{3.4}$$

The term $\sigma_N^x \sigma_{N+1}^x$ is set by the periodic boundary condition, which will be explained later in this chapter.

To apply this kind of transformation, first it is important to understand the properties of the spin operator and Pauli matrices.

3.2.1 Single spin

In quantum mechanics, the spin operator of a fermion is described by the vector:

$$\text{Spin operator} = \frac{\hbar}{2}\vec{\sigma} = \frac{\hbar}{2}\begin{pmatrix}\sigma^x \\ \sigma^y \\ \sigma^z\end{pmatrix} \tag{3.5}$$

The components of $\vec{\sigma}$ are called *Pauli matrices* and are constructed as follows:

$$\sigma^x = \begin{pmatrix}0 & 1 \\ 1 & 0\end{pmatrix} \tag{3.6}$$

$$\sigma^y = \begin{pmatrix}0 & -i \\ i & 0\end{pmatrix} \tag{3.7}$$

$$\sigma^z = \begin{pmatrix}1 & 0 \\ 0 & -1\end{pmatrix} \tag{3.8}$$

In their matrix representation, the eigenstates of σ^z are:

$$|+\rangle = \begin{pmatrix}1 \\ 0\end{pmatrix};\ |-\rangle = \begin{pmatrix}0 \\ 1\end{pmatrix}$$

Thus leading to their eigenvalues being 1 and -1, in fact:

$$\sigma^z |+\rangle = |+\rangle;\ \sigma^z |-\rangle = -|-\rangle$$

These matrices are characterized by the following properties:

$$[\sigma^i, \sigma^j] = 2i\varepsilon_{ijk}\sigma^k \tag{3.9}$$

$$(\sigma^i)^2 = I \tag{3.10}$$

Where ε_{ijk} is the Levi-Civita symbol and I the identity operator. From these expressions we can see that applying twice the same Pauli matrix does not change the state of the system and that the order in which we apply the σ^i influences the final result (since these operators do not commute with each other).

Starting from Pauli matrices, it is also possible to define the *raising* and *lowering operators*, that act increasing or decreasing the spin of a particle:

$$\sigma^{\pm} = \frac{\sigma^x \pm i\sigma^y}{2} \tag{3.11}$$

$$\sigma^+ |-\rangle = |+\rangle\,;\ \sigma^- |+\rangle = |-\rangle\,;\ \sigma^{\pm} |\pm\rangle = 0$$

These operators follow the anti-commutation property:

$$\{\sigma^+, \sigma^-\} = \sigma^+\sigma^- + \sigma^-\sigma^+ = \{\sigma^-, \sigma^+\} = I \tag{3.12}$$

3.2.2 Multiple spins

In our case, we do not limit ourselves to a system with just one fermion, but there are numerous ones fixed on a lattice, each with its corresponding spin, thus leading to the presence of multiple spins that interact with each other. Since we have a chain of particles, we can define the spin operating on the j^{th} site as:

$$\frac{\hbar}{2}\vec{\sigma_l} = \frac{\hbar}{2}\begin{pmatrix}\sigma_l^x \\ \sigma_l^y \\ \sigma_l^z\end{pmatrix} \tag{3.13}$$

We can then define a *raising* and *lowering operator* for each of these sites, that, similarly to Eq. 3.11, is built as:

$$\sigma_l^{\pm} = \frac{\sigma_l^x \pm i\sigma_l^y}{2} \tag{3.14}$$

Each of these newly found operators act solely on their corresponding sites and their commutation and anti-commutation rules are slightly more complex:

$$[\sigma_l^i, \sigma_{l'}^j] = 2i\varepsilon_{ijk}\sigma_l^k\delta_{ll'} \tag{3.15}$$

$$\{\sigma_l^+, \sigma_{l'}^-\} = \delta_{ll'} \tag{3.16}$$

The term $\delta_{ll'}$ is the Dirac delta, an object that assumes the value 1 if $l = l'$ and 0 otherwise. This means that two arbitrary Pauli matrices acting on different particles commute with each other, even though they are not the same component of their respective $\vec{\sigma}_l$; in other words, it doesn't matter in which order we apply these operators on different particles. This is reasonable, since each Pauli matrix acts just on a single site, not affecting the rest of the lattice.

The eigenstates of the system are obtained as the tensor product of the eigenstates of the single sites, so for a chain of N particles there are 2^N eigenstates and they are calculated:

$$|\psi_n\rangle = \bigotimes_{l=1}^{N} |\psi_n\rangle_l = \bigotimes_{l=1}^{N} |\pm\rangle_l \tag{3.17}$$

When an operator σ_l^z is applied to $|\psi_n\rangle$, the returned result is:

$$\sigma_l^z |\psi_n\rangle = \begin{cases} |\psi_n\rangle & \text{if } |\psi_n\rangle_l = |+\rangle_l \\ -|\psi_n\rangle & \text{if } |\psi_n\rangle_l = |-\rangle_l \end{cases} \tag{3.18}$$

3.2.3 Fermionic formulation

A more convenient way to study a system of this kind is by using a *fermion formulation* of the spins of the sites. Let us consider the particles of the lattice, they can have spin up or down and our goal is to traduce this information into the presence or absence of a spinless fermion. Usually, a spin up is defined as the absence of such an object, or, to be more precise, a vacuum, and a spin down as the presence of a fermion of spin 0. Thus, we will redefine the eigenstates of the σ_l^z operators as:

$$|+\rangle_l = |0\rangle_l\,; \;\; |-\rangle_l = |1\rangle_l$$

NOTE: The word *fermion* has already been used in the previous sections to refer to the particles of the lattice. Anyway, in order to prevent confusion, from now on this term will be used to address the state $|1\rangle$.

Let us now suppose that the number of fermions on a site cannot surpass 1 (a hypothesis deriving from the exclusion principle) and consider the creation and annihilation operators $c_l^\dagger$ and c_l. To follow these rules, they

must respect the relations:

$$\begin{cases} c_l \left|0\right\rangle_l = 0 \\ c_l \left|1\right\rangle_l = \left|0\right\rangle_l \\ c_l^\dagger \left|0\right\rangle_l = \left|1\right\rangle_l \\ c_l^\dagger \left|1\right\rangle_l = 0 \end{cases} \tag{3.19}$$

Considering this, an intuitive choice is setting the creation of a fermion as the lowering operator and the destruction of one as the raising operator. Even though this would be the most direct way, we should also consider another rule. If we switch the places of two fermions, the total state must carry a minus sign, so the newly found operators must respect the anti-commutation properties:

$$\begin{cases} \{c_l, c_{l'}\} = 0 \\ \{c_l^\dagger, c_{l'}^\dagger\} = 0 \\ \{c_l, c_{l'}^\dagger\} = \delta_{ll'} \end{cases} \tag{3.20}$$

An appropriate way to obtain c_l and $c_l^\dagger$, therefore, is defining hardcore bosonic creation and annihilation operators (that are $b_l^\dagger = \sigma_l^-$ and $b_l = \sigma_l^+$) and then applying the *Jordan-Wigner transformation*, that takes the form:

$$\begin{cases} \sigma_l^+ = \left(\prod_{l'=1}^{l-1} e^{i\pi n_{l'}}\right) c_l \\ \sigma_l^- = \left(\prod_{l'=1}^{l-1} e^{i\pi n_{l'}}\right) c_l^\dagger \end{cases} \tag{3.21}$$

The operators $n_{l'}$ are the number operators, that count the number of particles on the $(l')^{th}$ site. In our case, this translates into the number of fermions, so $e^{i\pi n_{l'}}$ can assume the values:

$$e^{i\pi n_{l'}} = \begin{cases} 1 & \text{if } \left|n\right\rangle_l = 0 \\ -1 & \text{if } \left|n\right\rangle_l = 1 \end{cases} \tag{3.22}$$

At this point, for the sake of simplicity, let us call the full vacuum state:

$$\left|0\right\rangle = \bigotimes_{l=1}^{N} \left|0\right\rangle_l \tag{3.23}$$

The new eigenstates can now be obtained by applying the creation operators $c_l^\dagger$ on different sites. For example, if we have 6 sites, a possible eigenstate could be:

$$\left|--++-+\right\rangle = \left|110010\right\rangle = c_1^\dagger c_2^\dagger c_5^\dagger \left|0\right\rangle \tag{3.24}$$

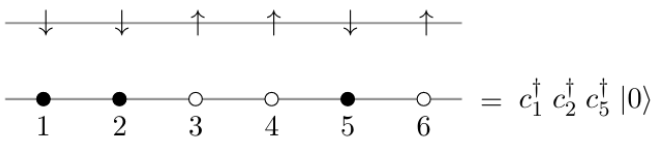


Figure 3: Example of a state with three fermions on 6 sites. Image taken from Ref.[11]

It is important to notice that this process do not alter the coefficients of the total state of the system, in fact after the reformulation we obtain:

$$|\psi\rangle = \sum_{n=1}^{2^N} \alpha_n |\psi_n\rangle = \sum_{n=1}^{2^N} \alpha_n \prod_{l=1}^{N} (c_l^\dagger)^{\beta_{ln}} |0\rangle \tag{3.25}$$

Where $\{\beta_{ln}\}$ is an appropriate set of coefficients that can assume the values 0 or 1 and let reproduce the state $|\psi_n\rangle$ via the operators $(c_l^\dagger)^{\beta_{ln}}$.

3.2.4 Properties of the fermionic formulation

Starting from c_l and $c_l^\dagger$, Pauli Matrices can be calculated as:

$$\begin{cases} \sigma_l^x = \left(\prod_{l'=1}^{l-1} e^{i\pi n_{l'}}\right)\left(c_l + c_l^\dagger\right) \\ \sigma_l^y = -i\left(\prod_{l'=1}^{l-1} e^{i\pi n_{l'}}\right)\left(c_l - c_l^\dagger\right) \\ \sigma_l^z = 1 - 2c_l^\dagger c_l \end{cases} \tag{3.26}$$

Moreover, in the next sections, we will use a kind of boundary conditions called *periodic boundary conditions* (or in short *PBC*). These let extend an Ising chain to infinity by setting a period L for which is valid:

$$\begin{cases} c_{l+L}^\dagger = c_l^\dagger \ \forall l \in \mathbf{Z} \\ c_{l+L} = c_l \ \forall l \in \mathbf{Z} \end{cases} \tag{3.27}$$

Usually, it is applied by adding another term after the sum of the $\sigma_l^i \sigma_{l+1}^i$ in the Hamiltonian, so that its form is equivalent to that of the hypothetical L^{th} term of the fermionic formulation.

NOTE: Usually the period is set to $L \equiv N$, with N being the number of particles initially considered.

3.2.5 Specific notation for this work

The previous sections have explained how to work with Ising spin chains following the usual notation, but the next chapters use a slightly different one. Indeed, the calculations to solve the problem that has been simulated in this work identify the presence of a fermion with a spin up, while the absence of one is represented with a spin down. The representation will therefore be:

$$|0\rangle_l = |-\rangle_l \,; \;\; |1\rangle_l = |+\rangle_l$$

This choice changes a little the form of the Pauli matrices and of the raising and lowering operators in terms of the fermionic ones:

$$\begin{cases} \sigma_l^x = \left(\prod_{l'=1}^{l-1} e^{i\pi n_{l'}}\right)\left(c_l + c_l^\dagger\right) \\ \sigma_l^y = i\left(\prod_{l'=1}^{l-1} e^{i\pi n_{l'}}\right)\left(c_l - c_l^\dagger\right) \\ \sigma_l^z = 2c_l^\dagger c_l - 1 \\ \sigma_l^+ = \left(\prod_{l'=1}^{l-1} e^{i\pi n_{l'}}\right) c_l^\dagger \\ \sigma_l^- = \left(\prod_{l'=1}^{l-1} e^{i\pi n_{l'}}\right) c_l \end{cases} \tag{3.28}$$

Other important properties used in this work are:

$$\begin{cases} \sigma_l^x \sigma_{l+1}^x = \left(c_l^\dagger - c_l\right)\left(c_{l+1} + c_{l+1}^\dagger\right) \\ \sigma_1^y \left(\prod_{l'=2}^{N-1} \sigma_{l'}^z\right) \sigma_N^y = \left(c_N^\dagger - c_N\right)\left(c_{N+1} + c_{N+1}^\dagger\right) \\ c_l = \left(\prod_{l'=1}^{l-1} -\sigma_l^z\right)\sigma_l^- \\ c_l^\dagger = \left(\prod_{l'=1}^{l-1} -\sigma_l^z\right)\sigma_l^+ \end{cases} \tag{3.29}$$

4 Fourier Transform

The *Fourier transform* (FT) is widely used in many fields (e.g. signal theory, quantum mechanics and quantum computing) and it is a linear operator on a functional space to another one which assumes the form:

$$f : \mathbb{R} \to \mathbb{R}, \quad F : \mathbb{R} \to \mathbb{R}, \quad f(x) \xrightarrow{FT} F(k) = \int_{\mathbb{R}} f(x) e^{-i2\pi kx} dx \tag{4.1}$$

Its inverse operator is called *anti-Fourier transform* and performs the operation:

$$F(k) \xrightarrow{FT^{-1}} f(x) = \int_{\mathbb{R}} F(k) e^{i2\pi kx} dk \tag{4.2}$$

It was first proposed by J. B. Joseph Fourier in its essay *The analytical theory of heat*[3] and he used it as a tool to express a generic function as a sum of periodic ones with different weights. Although in its work Fourier treated a less general case, this is the form used today in quantum physics.

Due to the need of discretizing values, in quantum computing this operator has a different form. In fact, in Eq. 4.1 we assumed to be in an infinite vector space composed of functions with real domains. When dealing with strings of qubits, instead, we have to consider them as finite dimensional vectors, a deed that drives to the so-called *discrete Fourier transform*. This will be explained in the next sections of the chapter, followed by the description of an important algorithm called *fast Fourier transform*. The last section, instead, will cover the *quantum FT*, which is the basis for many applications of quantum computing.

NOTE: For the sake of simplicity, from now on the term *Fourier transform* will refer to the discrete one.

4.1 Discrete Fourier transform

Let us consider a vector space V of dimension N, whose elements can then be expressed as:

$$\vec{x} \in V, \quad \vec{x} = \big(x_1,\ x_2,\ \ldots\ x_N\big)$$

with x_j being complex numbers.

The *Fourier transform* of $\vec{x}$ is another vector $\vec{y} \in V$ whose components y_k are calculated:

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{\frac{2\pi ikj}{N}} \tag{4.3}$$

Then, we can define an orthonormal basis:

$$\{\vec{e}_0, \vec{e}_1, \ldots \vec{e}_{N-1}\}$$

where the vectors $\vec{e}_n$ follow the relation:

$$(\vec{e}_n)_j = \delta_{nj} = \begin{cases} 1 \text{ if } n = j \\ 0 \text{ if } n \neq j \end{cases}$$

Given these premises and Eq. 4.3, we can express the Fourier transform of the vector $\vec{x}$ as:

$$\vec{y} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \sum_{j=0}^{N-1} x_j e^{\frac{2\pi i k j}{N}} \vec{e}_k = \sum_{k=0}^{N-1} y_k \vec{e}_k \tag{4.4}$$

This is called *classical Fourier transform* and it is the first step to define its quantum counterpart.

NOTE: This operator (to which we will refer as F) when applied to the vectors of the orthonormal basis gives the result:

$$F(\vec{e}_n) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \sum_{j=0}^{N-1} \delta_{nj} e^{\frac{2\pi i j k}{N}} \vec{e}_k = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i n k}{N}} \vec{e}_k$$

If we then calculate the Fourier transform of a vector $\vec{x} = \sum_{n=1}^{N} x_n \vec{e}_n$ we obtain the relation:

$$F(\vec{x}) = \sum_{n=1}^{N} x_n F(\vec{e}_n) \tag{4.5}$$

So we have verified that F is actually a linear operator.

The final part of this section is dedicated to the *fast Fourier transform* (FFT), today the fastest known algorithm able to exactly compute the classical FT of a vector.

4.1.1 Fast Fourier transform

The first algorithm able to calculate the *fast Fourier transform* was published in 1965 by James W. Cooley and John W. Tukey[15]. It is applied to vectors whose length is a power of 2 ($N = 2^M$) and consists of an iterative process.

If we take the components of the FT of the initial vector, we can parametrize it following the formula:

$$k = \frac{N}{2}k_1 + k_0 \tag{4.6}$$

with k_1 and k_0 in the ranges $k_1 = 0, 1$, $k_0 = 0, ...\frac{N}{2} - 1$.
The index of $\vec{x}$, instead, can be parametrized:

$$j = 2j_1 + j_0 \tag{4.7}$$

with j_1 and j_0 in the ranges $j_1 = 0, ...\frac{N}{2} - 1$ and $j_0 = 0, 1$.
Given these parametrizations, it is possible to express the Fourier transform of a vector $\vec{x}$ with the formula:

$$y_k = \sum_{j_0=0}^{1} \left(\sum_{j_1=0}^{N/2-1} x_{2j_1+j_0} e^{\frac{2\pi i(2j_1k_0)}{N}} \right) e^{\frac{2\pi i j_0 k}{N}} \tag{4.8}$$

Since the inner sum is performed over all the possible values of j_1, it depends solely on the indices j_0 and k_0. It is then a coefficient of the form:

$$\sum_{j_1=0}^{N/2-1} x_{2j_1+j_0} e^{\frac{2\pi i(2j_1k_0)}{N}} = X_{j_0,k_0} \tag{4.9}$$

The number of possible values for j_0 is 2 and for k_0 is $N/2$, so the different values for the coefficients X_{j_0,k_0} is N.
The components y_k now become:

$$y_k = y_{k_0,k_1} = \sum_{j_0=0}^{1} X_{j_0,k_0} e^{\frac{2\pi i j_0 k}{N}} \tag{4.10}$$

The number of operations needed to compute one of the coefficients X_{j_0,k_0} is $N/2$ and, after this, we need to define N components of $\vec{y}$, each calculated summing 2 terms. The number of total operations (defined as the amount of sums that we have to perform) is therefore $C = N\left(\frac{N}{2} + 2\right)$.

If we want to reduce the number of steps even more, we can consider the values X_{j_0,k_0} and divide them into two groups: $\{X_{0,j_0}\}$ and $\{X_{1,j_0}\}$.
Let us fix $j_0 = 0$, then Eq. 4.9 becomes:

$$X_{0,k_0} = \sum_{j_1=0}^{N/2-1} x_{2j_1} e^{\frac{2\pi i j_1 k_0}{N/2}} \tag{4.11}$$

We can see it as the FT of a vector of length $N/2$, so it can be computed in $\frac{N}{2}\left(\frac{N}{4}+2\right)$ steps. Repeating the procedure for $j_0 = 1$, the total number of operations becomes $C = N\left(\frac{N}{4}+4\right)$.

Iterating the process for m times we obtain:

$$C(m) = N\left(\frac{N}{2^m} + 2m\right) \tag{4.12}$$

If $N = 2^M$ the maximum amount of times we can apply this algorithm is $M-1$, for which we have:

$$C(M-1) = N \cdot 2M = 2N \log_2 N \tag{4.13}$$

In the end, the minimum number of steps that we have to implement in this scheme is the one shown in Eq. 4.13.

As we can see, this algorithm reduces the number of steps needed to compute the FT. For low values of N, the advantage is small, but when N is large the computational cost is significantly lower then that of a brute force method, for which it accounts to N^2 operations.

4.2 Quantum Fourier transform

When dealing with quantum computing, we make use of the *quantum Fourier transform* (QFT). Even though it has another name, the way it is computed is similar to its classic counterpart. We have to consider an orthonormal basis $\{|n\rangle\}$ with $n = 0, ...N-1$ and then we define the quantum Fourier transform of the vectors $|n\rangle$:

$$F|n\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i n k}{N}} |k\rangle \tag{4.14}$$

Finally, we can use the property of linearity to calculate the QFT of a generic state $|\psi\rangle = \sum_{n=1}^{N} x_n |n\rangle$ as follows:

$$F|\psi\rangle = \sum_{k=0}^{N-1} y_k |k\rangle \tag{4.15}$$

Where y_k is computed using the formula 4.3.

In the case of a string of M qubits, we can consider the computational basis, whose dimension will be $N = 2^M$. The state will then be $|j_1, ...j_M\rangle$ where j_n can assume the value 0 or 1. Moreover, it is helpful to implement

the notation $0.j_1 j_2 ... j_m$ to refer to the value $\sum_{n=1}^{N} \frac{j_n}{2^n}$. Thus, the QFT of an element of the computational basis can be written:

$$F\left|j_1, ...j_M\right\rangle = \frac{1}{2^{M/2}} \bigotimes_{n=0}^{M-1} \left(\left|0\right\rangle + e^{2\pi i 0.j_{M-n}...j_M}\left|1\right\rangle\right) \tag{4.16}$$

For example, for three qubits, the QFT of an element of the computational basis is:

$$F\left|j_1, j_2, j_3\right\rangle = \frac{\left(\left|0\right\rangle + e^{2\pi i 0.j_3}\left|1\right\rangle\right)\left(\left|0\right\rangle + e^{2\pi i 0.j_2 j_3}\left|1\right\rangle\right)\left(\left|0\right\rangle + e^{2\pi i 0.j_1 j_2 j_3}\left|1\right\rangle\right)}{2^{3/2}}$$

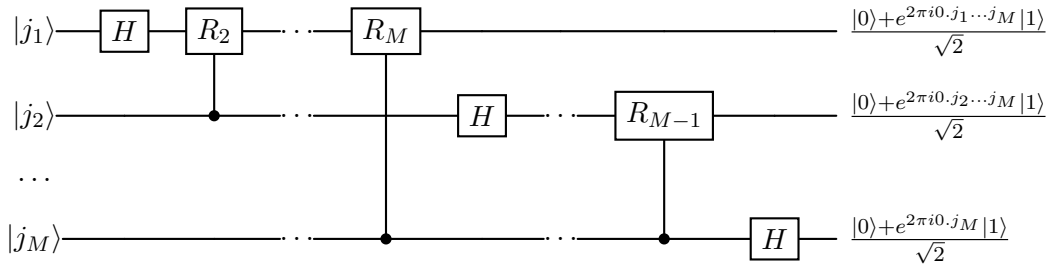
4.2.1 Computing the QFT on a quantum computer

In order to perform this transformation on a quantum computer, we have to understand how we can build it starting from basic gates. An appropriate circuit to do it is based on the gate:

$$R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{bmatrix}$$

Which can be constructed using a phase gate.

The circuit will then be as follows:



The first Hadamard gate on each row performs the operation:

$$H\left|j_n\right\rangle = \frac{\left|0\right\rangle + e^{2\pi i 0.j_n}\left|1\right\rangle}{\sqrt{2}} = \begin{cases} \frac{|0\rangle+|1\rangle}{\sqrt{2}} \text{ if } \left|j_n\right\rangle = \left|0\right\rangle \\ \frac{|0\rangle-|1\rangle}{\sqrt{2}} \text{ if } \left|j_n\right\rangle = \left|1\right\rangle \end{cases}$$

The following gates R_k, instead, act in a recursive way, following the formula:

$$R_k\left(\frac{\left|0\right\rangle + e^{2\pi i 0.j_n...j_{k-1}}\left|1\right\rangle}{\sqrt{2}}\right) = \frac{\left|0\right\rangle + e^{2\pi i 0.j_n...j_k}\left|1\right\rangle}{\sqrt{2}}$$

So, the circuit is equivalent to the operation:

$$|j_1, ...j_M\rangle \rightarrow \bigotimes_{n=1}^{M} \frac{|0\rangle + e^{2\pi i 0.j_n ... j_M} |1\rangle}{\sqrt{2}}$$

In the end, in order to get the desired Fourier transform, it is sufficient to swap the states of the qubits, thus obtaining the expression in Eq. 4.16.

With the method just described, it is possible to calculate the QFT with an $\mathcal{O}(M^2)$ number of gates. Such a task would be impossible to be completed by a classical computer, in fact, even the fast Fourier transform algorithm would require an $\mathcal{O}(M2^M)$ number of operations. This derives from the fact that a linear increase in the number of qubits coincides with an exponential growth in the dimension of their vector space.

5 Simulation of a 1D Quantum Ising chain

The simulation to perform on the quantum computer is that of a 1D Ising spin chain with transverse field and periodic boundary conditions (over a period of 4 sites) and it is based on the work in cit.[7]. The Hamiltonian of such a system has the form:

$$H = \sum_{j=1}^{N-1} \sigma_j^x \sigma_{j+1}^x + \sigma_1^y \left(\prod_{l=2}^{N-1} \sigma_l^z \right) \sigma_N^y + \lambda \sum_{j=1}^{N} \sigma_j^z \tag{5.1}$$

Since this problem possesses an exact analytical solution, the section is dedicated to its explanation. In particular, it will cover the diagonalization of the Hamiltonian, with also the quantum gates used to achieve it. Then it is followed by the description of the actual simulation and an explanation of the results (used to evaluate the efficiency of a quantum computer).

NOTE: The term in the center applies the periodic boundary conditions. It is equivalent to $\sigma_N^x \sigma_1^x$ for an even number of spins up and to $-\sigma_N^x \sigma_1^x$ for an odd number of them.

5.1 Analytical solution and building of the circuit

The analytical solution of this problem can be obtained in three macro-steps:

- Jordan-Wigner transformation;
- Fourier transform;
- Bogoliubov transformation.

The first transformation is achieved via the rules described in Chap. 3 and, as we will see, needs no quantum gate to be implemented. The second transfers the state to the momentum space and the third is a transformation commonly utilized to diagonalize certain Hamiltonians. A compendium of the calculations is shown in the next parts of the section.

5.1.1 Jordan-Wigner transformation

In order to implement the Jordan-Wigner transformation we can consider the relations that connect the operators σ_j^x and σ_j^z to the fermion annihilation and creation operators. In fact, we have:

$$\sigma_j^x \sigma_{j+1}^x = \left(c_j^\dagger - c_j \right) \left(c_{j+1} + c_{j+1}^\dagger \right) = c_j^\dagger c_{j+1} + c_{j+1} c_j + c_j^\dagger c_{j+1}^\dagger + c_{j+1}^\dagger c_j$$
$$\sigma_J^z = 2 c_j^\dagger c_j - 1$$

For what regards the periodic boundary conditions, instead, when we have an even number of sites and following the anticommutation properties of c_j and $c_j^\dagger$ we can transform the last term:

$$\sigma_1^y \left(\prod_{l=2}^{N-1} \sigma_l^z \right) \sigma_N^y = \sigma_1^x \left(\prod_{l=1}^{N-1} -\sigma_l^z \right) (-i\sigma_N^y) = \left(c_N^\dagger - c_N \right) \left(c_1 + c_1^\dagger \right)$$

So, the Hamiltonian after the Jordan-Wigner transformation becomes:

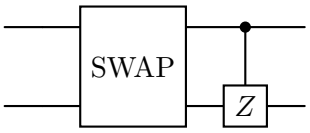
$$H \xrightarrow{JWT} H_{JW} = \sum_{j=0}^{N-1} \left(c_j^\dagger c_{j+1} + c_{j+1} c_j + c_j^\dagger c_{j+1}^\dagger + c_{j+1}^\dagger c_j \right) + \lambda \sum_{j=0}^{N-1} \left(2 c_j^\dagger c_j - 1 \right)$$

The indices have been switched of one place to adjust to the notation that will be used from now on. Moreover, it is important to note that these actions do not change the coefficients of the states expressed in the computational basis, so we do not need any gate to implement them.

NOTE: Usually, when we want to switch the state of two qubits we use a *SWAP gate*. For fermions, when we have two qubits set to $|1\rangle$ and we swap them, we have to multiply their state for a phase $e^{i\pi}$. In order to implement this functionality we need an *fSWAP gate*, which is represented as follows

$$\text{fSWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

An fSWAP is obtained by the concatenation of a SWAP and a CZ:



5.1.2 Fourier transform

In the next step we apply the FT in order to transfer to the momentum space. In particular, in this case we make use of the so-called *fermionic Fourier transform*, which modifies the creation and annihilation operators

as follows:

$$\begin{aligned} c_j^\dagger &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} b_k^\dagger e^{-\frac{2\pi i k j}{N}} = \frac{1}{\sqrt{N}} \sum_{k=-N/2+1}^{N/2} b_k^\dagger e^{-\frac{2\pi i k j}{N}} \\ c_j &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} b_k e^{\frac{2\pi i k j}{N}} = \frac{1}{\sqrt{N}} \sum_{k=-N/2+1}^{N/2} b_k^\dagger e^{-\frac{2\pi i k j}{N}} \end{aligned} \tag{5.2}$$

NOTE: The second equivalence is obtained due to the periodicity of the lattice and the complex exponential.

An important property used in this calculation is one concerning the sum of complex exponentials:

$$\sum_{j=1}^{N} e^{\frac{2\pi i (k-k') j}{N}} = N \delta_{k k'} \tag{5.3}$$

If we apply it to the first family of terms in the Hamiltonian we obtain:

$$\begin{aligned} \sum_{j=1}^{N} c_j^\dagger c_{j+1} &= \frac{1}{N} \sum_{j=1}^{N} \left(\sum_{k=-N/2+1}^{N/2} b_k^\dagger e^{-\frac{2\pi i k j}{N}} \right) \left(\sum_{k'=-N/2+1}^{N/2} b_{k'} e^{\frac{2\pi i k' (j+1)}{N}} \right) = \\ &= \frac{1}{N} \sum_{k} \sum_{k'} b_k^\dagger b_{k'} e^{\frac{2\pi i k'}{N}} \sum_{j} e^{\frac{2\pi i (k-k') j}{N}} = \sum_{k=-N/2+1}^{N/2} b_k^\dagger b_k e^{\frac{2\pi i k}{N}} \end{aligned} \tag{5.4}$$

By reapplying this logic to all of the Hamiltonian we retrieve:

$$H_{JW} \xrightarrow{FT} H_F = \sum_{k=-N/2+1}^{N/2} \left(2 \left(\cos\left(\frac{2\pi k}{N} \right) + \lambda \right) b_k^\dagger b_k + e^{\frac{2\pi i k}{N}} \left(b_k b_{-k} + b_k^\dagger b_{-k}^\dagger \right) \right)$$

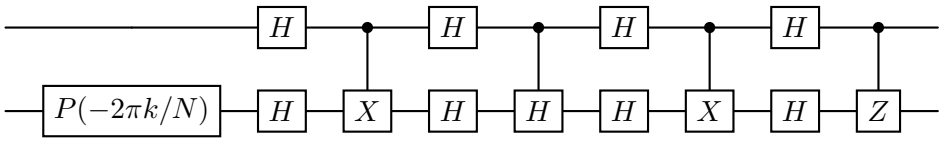
In this form the couplings that remain are those between opposite momenta, so H_F is not diagonal yet. In order to accomplish the diagonalization, there is one more transformation to apply.

NOTE: The constant term λN has been neglected. This is possible because it does not affect the dynamics of the system.

The circuit used to perform this transformation is based on the following two qubit gate:

$$F_k^N = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -\frac{e^{-\frac{2\pi i k}{N}}}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{e^{-\frac{2\pi i k}{N}}}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & -e^{-\frac{2\pi i k}{N}} \end{bmatrix}$$

In terms of basic gates, it is obtained as:



When $k = 0$, for any N we retrieve:

$$F_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

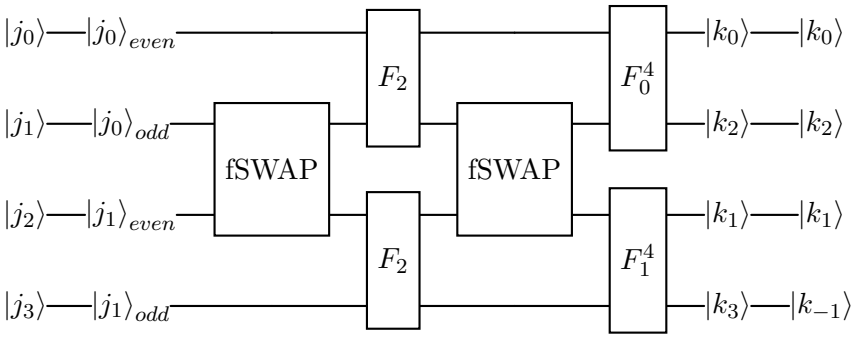
The circuit that applies F_2 is built in the same way as the one made for F_k^N, with the only difference that the first phase gate is omitted.

The complete circuit needed to perform the fermionic Fourier transform (fFT) for a generic N qubits system is based on the following algorithm:

- the qubits are reordered using the fSWAP gates, so that the first $N/2$ positions are occupied by the even qubits and the last $N/2$ positions by the odd ones;
- the reorganization is repeated for the two new sets. In the first group, we obtain first the $N/4$ even qubits and then the $N/4$ odd ones. The same logic is applied to the second set;
- the process is iterated until we obtain groups of just 2 qubits;
- the gate F_2 is applied to each group;
- the qubits are reorganized in the original order;

- the gates F_k^N is applied to each couple of qubits. In this step the k^{th} qubit of the even sites is sent to the k^{th} qubit of the final state, while k^{th} of the odd sites is sent to the $(k + N/2)^{\text{th}}$ qubit of the final state;
- finally, if a qubit has an index over $N/2$ this is decreased by N. This change is possible because of the periodicity of the lattice.

The circuit for the case of 4 qubits is schematized as follows:



After this, the implementation of the Fourier transform is complete and the state has finally been transported to the momentum space.

5.1.3 Bogoliubov transformation

The final step to diagonalize the Hamiltonian is the Bogoliubov transformation, a technique that comes in help in some problems of this kind.

First, it is better to express H_F in the form:

$$H'_F = \sum_{k=-N/2+1}^{N/2} \left(\epsilon_k \left(b_k^\dagger b_{-k}^\dagger - b_{-k} b_k \right) + i\delta_k \left(b_k^\dagger b_{-k}^\dagger - b_{-k} b_k \right) \right) \tag{5.5}$$

In this expression we have $\epsilon_k = \cos\left(\frac{2\pi k}{N}\right) + \lambda$ and $\delta_k = \sin\left(\frac{2\pi k}{N}\right)$ It has been possible to rewrite H_F in this form because the sum is over both positive and negative values of k and because the additional term $\sum_k \epsilon_k$ (which appears due to the anti-commutation rules of the creation and annihilation operators) is constant, so it can be neglected.

Then, the proper Bogoliubov transformation is applied as follows:

$$\begin{aligned} b_k &\xrightarrow{BT} a_k = u_k b_k + v_k b_{-k}^\dagger \\ b_k^\dagger &\xrightarrow{BT} a_k^\dagger = u_k^* b_k^\dagger + v_k^* b_{-k} \end{aligned}$$

In order to respect the anti-commutation rules for the new operators, the coefficients selected are:

$$\begin{aligned} u_k &= \cos\left(\tfrac{\theta_k}{2}\right) \\ v_k &= -i\sin\left(\tfrac{\theta_k}{2}\right) \\ \theta_k &= \arctan\left(\tfrac{\delta_k}{\epsilon_k}\right) = \arctan\left(\tfrac{\sin(2\pi k/N)}{cos(2\pi k/N)+\lambda}\right) \end{aligned}$$

Thus, the Hamiltonian has successfully been diagonalized and becomes:

$$\begin{aligned} H'_F \xrightarrow{BT} H'_B &= \sum_{k=-N/2+1}^{N/2} E_k\left(a_k^\dagger a_k - a_{-k}a_{-k}^\dagger - 1\right) = \\ &= \sum_{k=-N/2+1}^{N/2} 2E_k\left(a_k^\dagger a_k - \frac{1}{2}\right) \end{aligned} \tag{5.6}$$

Where $2E_k$ is the energy that a fermion has in the mode k and its value is $2E_k = 2\sqrt{\left(\lambda + \cos\left(\frac{2\pi k}{N}\right)\right)^2 + \sin^2\left(\frac{2\pi k}{N}\right)}$.

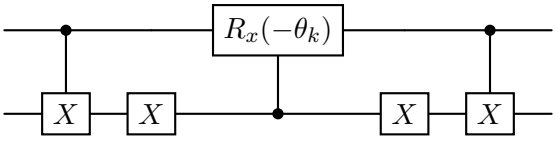
Finally, summing all of the constants neglected before, we can see that the diagonalized Hamiltonian is:

$$H_B = \sum_{k=-N/2+1}^{N/2} 2E_k\left(a_k^\dagger a_k + \epsilon_k\right) - N\left(\lambda + \frac{1}{2}\right) \tag{5.7}$$

The gate needed to implement this transformation is the following:

$$B_k^N = \begin{bmatrix} \cos\left(\frac{\theta_k}{2}\right) & 0 & 0 & i\sin\left(\frac{\theta_k}{2}\right) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ i\sin\left(\frac{\theta_k}{2}\right) & 0 & 0 & \cos\left(\frac{\theta_k}{2}\right) \end{bmatrix}$$

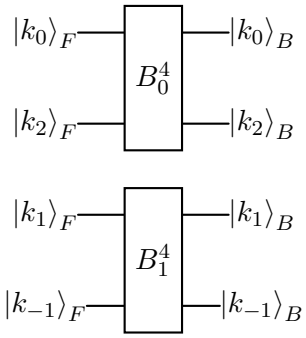
It acts on a couple of qubits of the form $|k_j, k_{-j}\rangle$ and is obtained from the circuit:



The algorithm applied in the process of decoupling of the k modes is divided into the steps:

- the fSWAPs are used to sort the k modes. After this process, there should be couples of qubits of the form $|k_l, k_{-l}\rangle$;
- the gate B_k^N is applied to each couple of qubits;
- the new Bogoliubov modes are reorganized in the original order using a sequence of fSWAPs.

For the case of 4 qubits there is no need to reorder the modes and the scheme of the circuit is the following:



5.1.4 Time evolution

An initial state $|\psi_0\rangle$ evolves over time following the rules of quantum mechanics, so if we denote with ε_j the j^{th} eigenvalue of the Hamiltonian, we can write the state at the time t as:

$$|\psi(t)\rangle = U_{ev}(t)\,|\psi_0\rangle = \sum_j e^{-i\varepsilon_j t}\,\langle j|\psi_0\rangle\,|j\rangle \tag{5.8}$$

Then, we need to implement the evolution operator too.

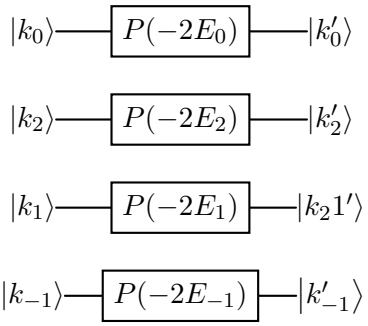
If we diagonalize the Hamiltonian via the described procedure, the eigenvectors of the said operator become elements of the computational basis and their energies are:

$$\varepsilon_j = \sum_{l=-N/2+1}^{N/2} 2E_l\delta_{1k_l} + \sum_{l=-N/2+1}^{N/2}\left(\epsilon_l - \lambda - \frac{1}{2}\right)$$

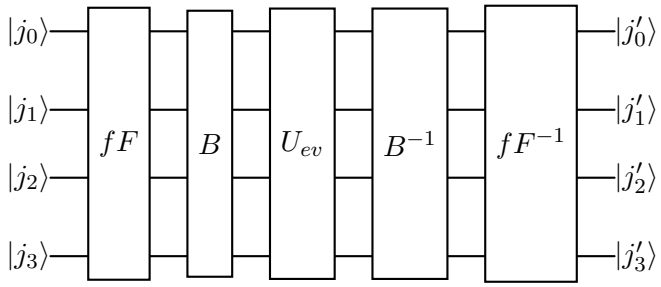
Since the second sum does not depend from the eigenvector considered, it changes the state of a pure phase. For this reason, we can neglect it without affecting the successive measures, so the energies can be rewritten:

$$\varepsilon'_j = \sum_{l=-N/2+1}^{N/2} 2\delta_{1k_l} \sqrt{\left(\lambda + \cos\left(\frac{2\pi l}{N}\right)\right)^2 + \sin^2\left(\frac{2\pi l}{N}\right)} \tag{5.9}$$

This operator can be implemented in a system of 4 qubits by the circuit:



In the end, the circuit that implements the simulation performs the diagonalization as previously described, successively applies the evolution operator and, finally, repeats the diagonalization process in the reverse way. The operations are summarized as follows:



5.2 Simulation on the quantum computer

An example of the many possible simulations to implement is the evolution of the magnetization for a chain with four spins up ($|++++\rangle$), which in the computational basis is represented as $|1111\rangle$. When we apply the diagonalization method as previously described, we get:

$$|\psi(0)\rangle = |1111\rangle = \sin\left(\frac{\phi}{2}\right)|1100\rangle_{diag} - i\cos\left(\frac{\phi}{2}\right)|1111\rangle_{diag} \tag{5.10}$$

With $\phi = \arctan(1/\lambda) = \theta_1$.
If we apply U_{ev} to $|\psi\rangle$, it turns into the state:

$$|\psi(t)\rangle = e^{-i2\lambda t}\left(\sin\left(\frac{\phi}{2}\right)|1100\rangle_{diag} - i\cos\left(\frac{\phi}{2}\right)e^{-i4t\sqrt{\lambda^2+1}}|1111\rangle_{diag}\right) \tag{5.11}$$

After this, we can apply the inverse of the diagonalizing procedure to decompose $|1100\rangle_{diag}$ and $|1111\rangle_{diag}$ into eigenstates of σ_j^z. The result is:

$$\begin{aligned} |\psi(t)\rangle &= e^{-i2\lambda t}\left(A(t)\left|\psi'\right\rangle + B(t)|1111\rangle\right) \\ A(t) &= -\frac{\sin(\phi)}{2}\left(1 - e^{-i4t\sqrt{\lambda^2+1}}\right) \\ B(t) &= \sin^2\left(\frac{\phi}{2}\right) + \cos^2\left(\frac{\phi}{2}\right)e^{-i4t\sqrt{\lambda^2+1}} \end{aligned} \tag{5.12}$$

With $|\psi'\rangle = \frac{1}{2}(|0011\rangle - |0110\rangle + |1001\rangle + |1100\rangle)$, which is an eigenstate of the average magnetization with eigenvalue 0.
The average magnetization Mz is defined:

$$Mz = \frac{1}{4}\sum_{j=0}^{3}\sigma_j^z \tag{5.13}$$

Its expected value over time will then be:

$$\langle Mz\rangle = \frac{2\lambda^2 + 1 + \cos\left(4t\sqrt{\lambda^2+1}\right)}{2\lambda^2+2} \tag{5.14}$$

5.2.1 Implementation on the simulator

The code that enabled the simulation described in the previous sections was written in *Qiskit*, a framework for python developed by IBM in order to write programs that can run on a quantum machine. Moreover, it was first tested on a local simulator and then run on a proper quantum computer. The simulator used was *AerSimulator* and came from the library *qiskit_aer*; it lets emulate the behaviour of a quantum computer on a classical one. Successively, the testing consisted of the evolution of the system for 20 different times, starting from 0.0 and arriving till 1.9, with jumps of 0.1. Since it was needed to calculate the average magnetization, for each time t the simulation ran for 1024 shots. Moreover, these tests were performed for different values of λ, which were 0.2, 0.5, 2 and 5.

The results obtained from the simulator for the first two cases are illustrated in the following graphs.

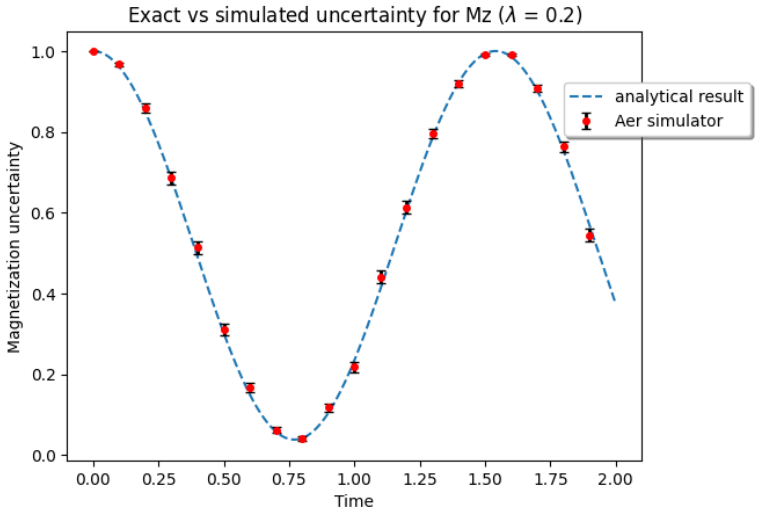


Figure 4: Graph for $\lambda = 0.2$

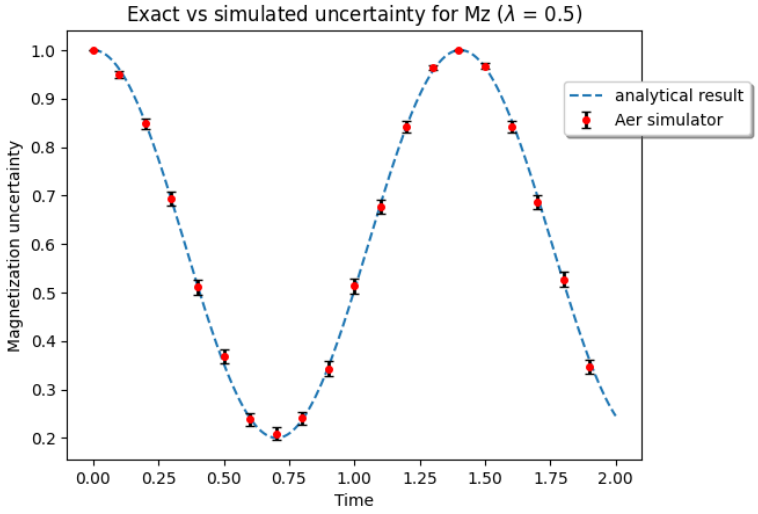


Figure 5: Graph for $\lambda = 0.5$

The error chosen was the standard deviation of the mean and, in this case, it turns out to be small.

For the cases with $\lambda = 2$ and $\lambda = 5$ the graphs are a little different.

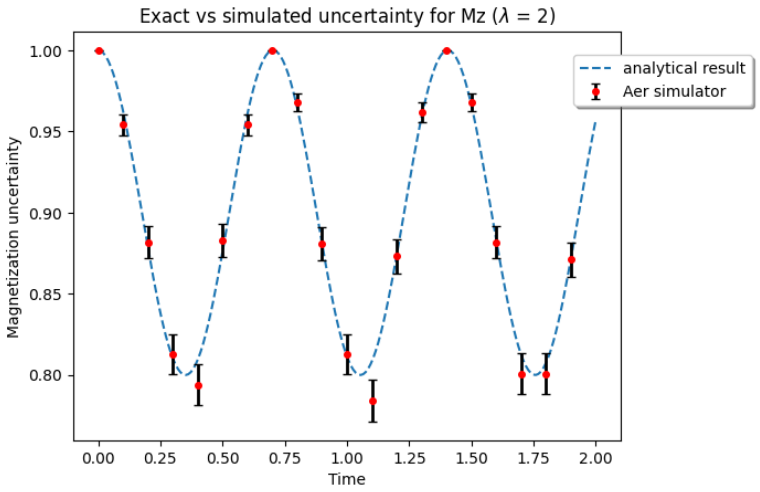


Figure 6: Graph for $\lambda = 2$

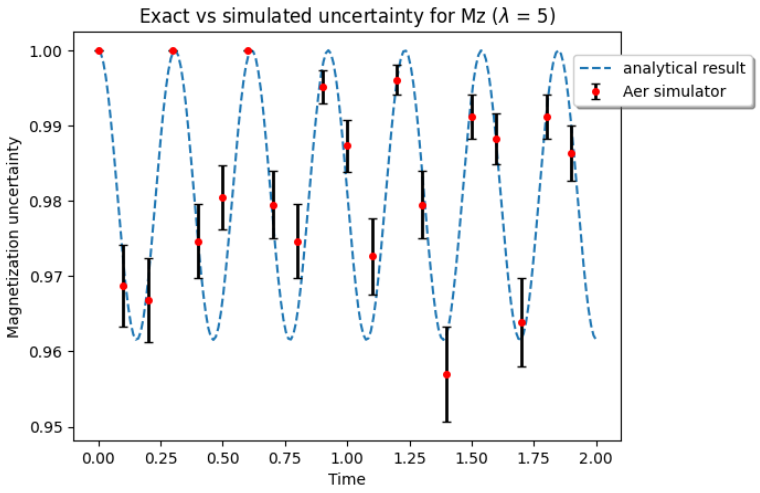


Figure 7: Graph for $\lambda = 5$

This time, the error is significant for the lower points of the graph and we can observe that it gradually decreases moving upwards. This can theoretically be explained by calculating the uncertainty associated to the observable

Mz. Indeed, it is:

$$\Delta Mz = \sqrt{\langle Mz^2 \rangle - \langle Mz \rangle^2} = \frac{\sqrt{(2\lambda^2 + 1 + s)(1 - s)}}{2\lambda^2 + 2} \tag{5.15}$$

Where $s = \cos\left(4t\sqrt{\lambda^2 + 1}\right)$ and can run in the range $[-1, 1]$.
It is clear that a minimum can be observed for $s = 1$, for which we have $\Delta Mz = 0$ and that corresponds to having $4t\sqrt{\lambda^2 + 1}$ equal to a multiple of 2π. The maximum, instead, corresponds to having $s = \frac{1-2\lambda^2}{3}$. For $\lambda = 0.2$ we have $s_{max} \simeq 0.30$ and, instead, for $\lambda = 0.5$ we have $s_{max} \simeq 0.17$. For the other two cases s_{max} actually lies outside the range $[-1, 1]$, so the real maximum coincides with $s_{max} = -1$.

In order to verify this hypothesis, some graphs that relate the standard deviation over time to its analytical value were produced. Below it is possible to see the graphs for $\lambda = 0.5$ and $\lambda = 2$.

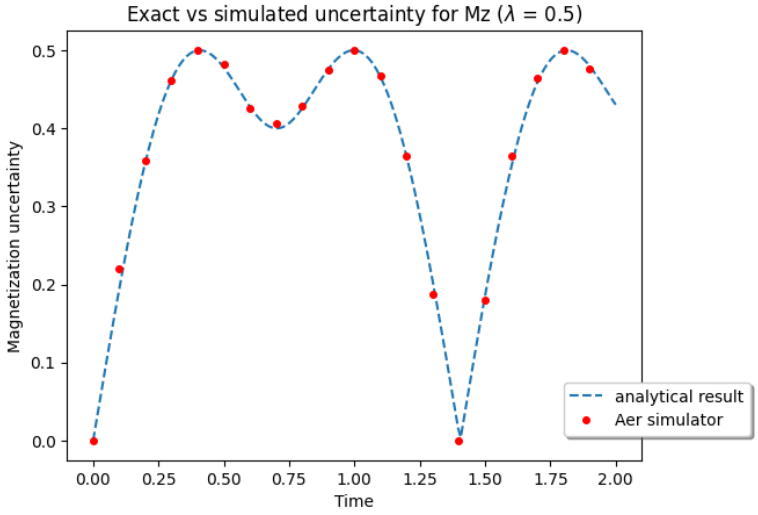


Figure 8: Standard deviations for $\lambda = 0.5$

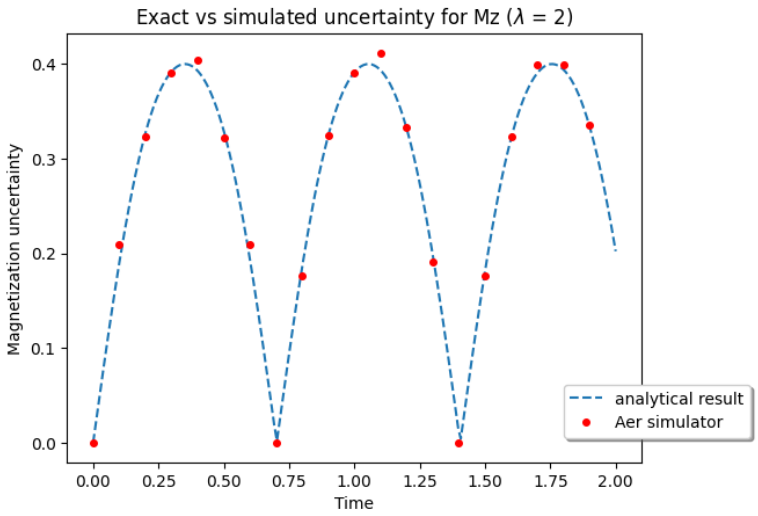


Figure 9: Standard deviations for $\lambda = 2$

The simulated standard deviations are compatible with the analytical uncertainties, in fact the simulated values have the form that we would expect from Eq. 5.15. So, this explains why they are so different for the various points that were considered.

NOTE: The study focused on the standard deviations, even though the errors in the first graphs were identified with the standard deviations of the mean. This is because the second can be obtained from the first by multiplicating them for a constant, so it was sufficient to study the standard deviations in order to get information about the standard deviations of the mean.

5.2.2 Implementation on the quantum computer

The proper simulation considered $\lambda = 0.5$ and was run on the quantum computer *ibm_sherbrooke* from the platform *IBM Quantum Platform* [4]. It is a machine equipped with 127 qubits and its set of basis gates is $\{\frac{1}{\sqrt{2}}(IX - XY), I, R_z, \sqrt{X}, X\}$, where I is the identity gate. Qiskit comprised functions to connect from remote to IBM's quantum computing resources using the aforementioned platform. The procedure to run the program involved the compilation of the circuit in terms of basis gates of the quantum machine, which is called transpilation. During this process, moreover, there is an

adding step called optimization, in which the compiler transforms the circuit into another equivalent one but with fewer gates.

After the tests were performed, the obtained results were confronted with the exact ones and they had a similar form but were not quantitatively compatible. The errors considered were again the standard deviations of the mean, but for the two data sets the reduced chi-squared amounted to $\tilde{\chi}^2 \simeq 61$. The comparison is shown in Fig. 10.

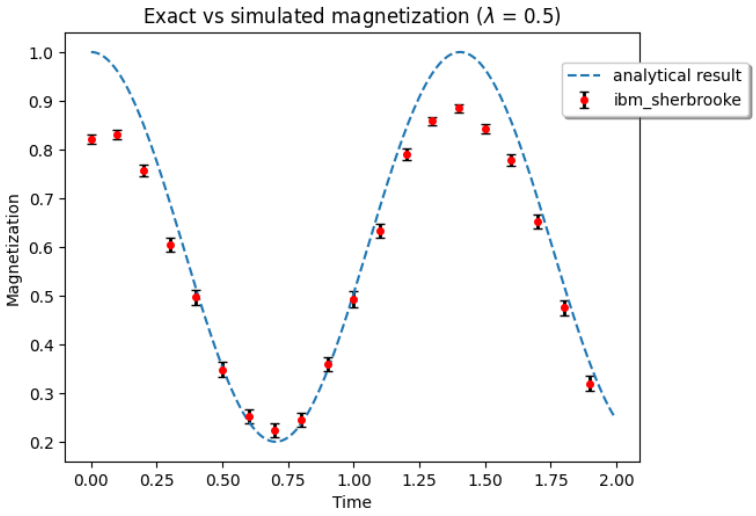


Figure 10: Estimated errors for $\lambda = 0.5$

Starting from this, an interpolation was implemented, so it was possible to determine an appropriate analytical function that fit the simulated data. The form of the said function is:

$$f(t) = \frac{2\lambda^2 + 1 + a\cos\left(4t\sqrt{\lambda^2+1}\right)}{2\lambda^2+1} + b \tag{5.16}$$

The errors that occurred during the runs made the magnetization shift below and reduced the amplitude of its oscillations. After this operation, the data were confronted with the fit and it was found they were still incompatible ($\tilde{\chi}^2 \simeq 3.9$). The final result for $\lambda = 0.5$ is summarized in Fig. 11. The values of a and b that were estimated from the fit are:

$$a = 0.81 \pm 0.01$$
$$b = -0.049 \pm 0.003$$

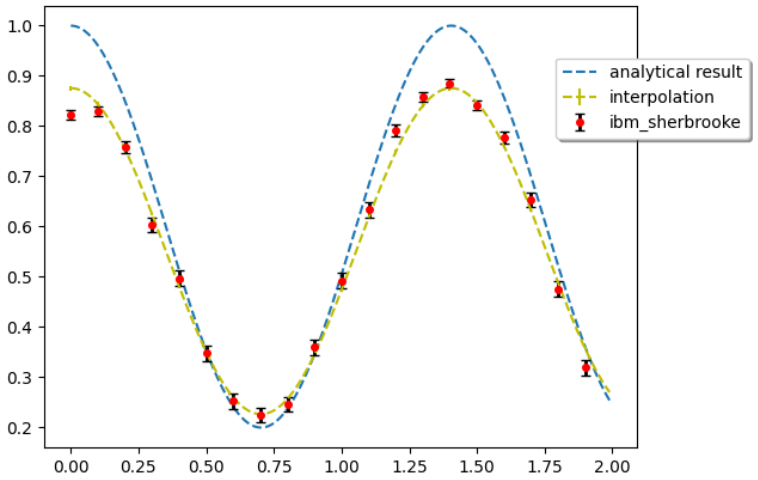


Figure 11: Interpolation for $\lambda = 0.5$

The reason behind the reduction in the amplitude can be explained using a statistical argument. The state $|\psi\rangle$ is composed of eigenstates of Mz with eigenvalues 0 and 1. If we suppose that the average variation of Mz ($\langle \delta Mz \rangle$) for the first is less than that of the second, it is direct to assume that the minimums of the magnetization will decrease less than the maximums. These premises are reasonable because the eigenstate $|1111\rangle$ is already in the maximum of Mz, so its value after an error can only decrease; for eigenstates with $Mz = 0$, instead, the magnetization could also increase, so $\langle \delta Mz \rangle$ must be smaller.

Another quantity that helps evaluate the quality of the simulation is the error rate (expressed as the probability to have an error by the end of the circuit). It was calculated as the sum of the differences between the theoretical probabilities of the eigenvalues 0 and 1 and their actual probabilities:

$$r = |P_{0,thrt} - P_{0,meas}| + |P_{1,thrt} - P_{1,meas}| \tag{5.17}$$

The result is:

$$r = 0.18 \pm 0.03$$

The uncertainty that was chosen is the standard deviation.

In conclusion, the results obtained are qualitatively compatible with the analytical ones, but not quantitatively. The probability of an error is not negligible and, indeed, affects significantly the final result.

6 Conclusions

This thesis had the goal of presenting the basic principles of quantum computation, an exotic field of applied physics and computer and information sciences. Its potential is intrinsically enormous, although it is focused on very specific tasks. One of them, that is quantum simulations, was directly tested on a real quantum computer by simulating the Ising model, a notorious theory which explains some magnetic phenomena and that can be used to assess the efficiency of a said machine.

Initially, this work explained the properties of qubits, the most important of which being superposition, and how to make use of them to build new kinds of algorithms. A specific focus was dedicated to quantum simulation algorithms, a class of tasks for which quantum computation is particularly powerful. Then, the following chapters covered everything that was needed to understand the final test, that is the technique of the Jordan-Wigner transformation and the Fourier transform. Finally, the test was performed by calculating the exact analytical solution of the problem and by building the circuit that applied it. The result was qualitatively compatible with the expected one, but quantitatively it was not satisfying.

The field of quantum simulations will be very important in the coming decades. At the current state, the difficulty to simulate quantum systems with classical computers is an unsurpassable obstacle, due to the exponential increase in the complexity of such programs. For this reason, quantum simulations can play a pivotal role in the future of fields like plasma physics, materials science and solid state physics, just to name a few. Despite their theoretical advantages, there are still some challenges to face in order to achieve a real quantum advantage in practical problems, like reducing the error rate of quantum circuits or increasing the coherence time of qubits and the rate of operations performed on a quantum chip. The hope is that research will gradually improve each of these parameters, so that one day it will be possible to obtain a new useful kind of computers, that can run algorithms still impossible to apply.

References

[1] Michael A. Nielsen, Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[2] Claude Cohen-Tannoudji, Bernard Diu, Franck Laloë. *Quantum Mechanics, Volume III: Fermions, Bosons, Photons, Correlations, and Entanglement*. Wiley-VCH, 1997.

[3] J. B. Joseph Fourier. *The Analytical Theory of Heat*. Cambridge University Press, 1878.

[4] IBM Quantum Platform. `https://quantum.ibm.com/`

[5] P. W. Shor. *Algorithms for quantum computation: discrete logarithms and factoring*. Proceedings 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, USA, pp. 124-134 (1994).

[6] Lov K. Grover. *A fast quantum mechanical algorithm for database search*. Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96. Philadelphia, Pennsylvania, USA: Association for Computing Machinery. pp. 212–219 (1996).

[7] Marc Ferras, Alba Cervera-Lierta. *Simulation of the 1d XY model on a quantum computer*. SciPost Phys. (2024).

[8] Richard P. Feynman. *Simulating Physics with Computers*. Int. J. Theor. Phys. 21, pp. 467-488 (1982).

[9] F. Verstraete, J. I. Cirac and J. I. Latorre. *Quantum circuits for strongly correlated quantum systems*. Phys. Rev. A 79, 032316 (2008).

[10] E. Ising. *Beitrag zur Theorie des Ferromagnetismus*. Zeitschrift für Physik 31, pp. 253-258 (1925).

[11] G. B. Mbeng, A. Russomanno, G. E. Santoro. *The quantum Ising chain for beginners*. SciPost Phys. Lect. Notes 82 (2024).

[12] P. Jordan, E. Wigner. *Über das Paulische Äquivalenzverbot*. Zeitschrift für Physik 47, No. 9. pp. 631-651 (1928).

[13] H. F. Trotter. *On the Product of Semi-Groups of Operators*. Proceedings of the American Mathematical Society, Vol. 10, N. 4 (1959).

[14] S. G. Brush. *History of the Lenz-Ising Model*. Rev. of Mod. Phys. 39, p. 883-893 (1967).

[15] J. W. Cooley, J. W. Tukey. *An alogrithm for the machine calculation of complex Fourier series*. Math. Comp. 19, pp. 297-301 (1965).

[16] N. N. Bogoliubov. *A new method in the theory of superconductivity*. Fortsch. Phys. 6, pp. 605-682 (1958).