## Università degli studi di Padova

Department of Physics and Astronomy

Master thesis in Physics of Data

# Quantum Computer simulation via Tensor Networks

*Supervisor*
Simone Montangero
Università di Padova

*Master Candidate*
Marco Ballarin

# Abstract

In recent years lots of efforts have been spent in the realization of quantum computers able to reproduce quantum circuits involving increasing number of qubits with the greatest possible accuracy. The final goal is to reach the limit, the so-called quantum supremacy, where a classical computer is no longer able to reproduce the results of a quantum machine. Indeed, simulating quantum many-body systems is very computationally demanding due to the exponential scaling of the Hilbert space with the number of qubits. In order to perform a classical simulation of a quantum circuit acting on a qubits register, one must choose between two possible approaches: the first is an exact description of the qubits' state, possible up to a maximum reachable number of qubits. The second, instead, consists of representing the state approximately. But, even quantum processors are not able to reproduce exactly a given quantum circuit: their coupling with the environment, which is minimized but not removed by the experimental implementation, induces errors through quantum channels like decoherence or bit-flip. Therefore an approximate representation of the qubits' state is acceptable as long as its errors are comparable with the experimental ones. The tensor network methods allow one to approximate a quantum state by efficiently compressing its information, introducing a controllable error. In this thesis, these methods will be used to simulate a quantum computer on a large computational cluster, to push as far as possible the classical simulation framework.

To all those who sustained me during my education.

# Contents

# List of Figures

# List of Tables

# Listings

# Introduction

Quantum mechanics has given truthful results since its introduction in 1925 [1]. Important results have been achieved via analytical techniques, such as the tunneling effect [2]. However, it is not possible to attack analytically every problem. For this reason, simulations are one of the most used methods to explore a natural phenomenon. Simulations are usually performed on a classical computer. However, this approach is not efficient, if we are describing classically something that is quantum: in 1982 Richard Feynman said, *"Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical"* [3]. Thus, we expect that quantum simulations will bring great improvements to the research effort: in the last decade, there has been an outstanding effort to develop quantum computers, i.e. machines able to exploit quantum features to study phenomena otherwise inaccessible. In quantum computers, the classical bit, which can encode the values $\{0, 1\}$, is replaced by the quantum bit (qubit), which can also encode a superposition of those two states. Furthermore, we can employ the entanglement, a unique quantum resource. Using quantum computers would enable us to perform efficient simulations of quantum systems.

Building quantum computers is a challenge. The interaction of the quantum system with the environment introduces errors, to the extend that the system may easily lose its quantum behavior, by the means of a process called decoherence. Quantum computers available today are noisy and reduced in size, thus commonly referred to as Noisy Intermediate Scale Quantum computers (NISQ). For example, IBM, using superconducting qubits [4], promised to build a quantum computer composed of 1000-qubits by 2023, even though their largest current quantum computer is made of 65 qubits. There are other physical platforms which are currently used to implement quantum computers, such as trapped ions [5], Rydberg atoms [6] or photonic hardware [7]. It is essential to benchmark all these quantum computers to attest their performances. Classical simulations are one of the possible techniques to benchmark quantum computers, even though these simulations are exponentially demanding due to the exponential scaling of the Hilbert space in which the quantum state is defined, as a function of the system size. For example, to exactly describe the 65-qubits machine one would need more than $10^{11}$ GB of memory. Nonetheless, we are often interested in particular states, that belong to peculiar subspaces of the full Hilbert space. Many different techniques were introduced over the years by the physics community to represent and evolve such states, like the Real Space Renormalization Group, the Density Matrix Renormalization Group [8] or the Tensor Network Methods [9]. Tensor Network Methods can be used for the simulation of quantum computers. We focus on Tensor Networks in Chapter 2: they enable us to run simulations of many qubits even on a personal computer.

In this thesis, we present a quantum computer simulator based on a tensor network ansatz called Matrix Product States [10, 11]. The developed program runs on the Cineca cluster, and specifically the Marconi 100 supercomputer. We test this simulator, understanding its capabilities and use cases. Furthermore, we also tackle systems more computationally demanding than quantum circuits composed of qubits. In particular, we simulate photonic circuits, where the dimension of the single degree of freedom is not two, as in the qubit's case, but is theoretically infinite. Since it is not possible to simulate unbounded quantities, we impose a maximum dimension. We apply such tool to address an instance of quantum supremacy, which was achieved using the Gaussian boson sampling protocol [12].

This thesis is organized as follows:

- In the Introduction we present the reasons for this work, briefly introducing its structure;
- In Chapter 1 we introduce the quantum computing framework, focusing on the limits of the exact classical simulation, both in the qubit's and photonic case;
- In Chapter 2 we explain in detail the Matrix Product State formalism, highlighting its capabilities and the main steps to develop it to the quantum computing framework;
- In Chapter 3 we focus on the implementation on the Cineca cluster, going through the developed code and presenting the obtained results;

- In the Conclusion we conclude this thesis, suggesting possible future works that employ the framework developed.

# 1
# Quantum Computing

In this chapter, we introduce the basics of the quantum computing framework. First, we address how the classical computation techniques can be extended in the quantum case. In Section 1.3.1 we survey the methods for the evolution of quantum states, focusing on the application of quantum gates [13]. We explore one of the error sources in the current quantum processor hardware. Then, in Section 1.4 we overview different simulation methods for quantum circuits [14, 15]. Finally, in Section 1.5, we address one particular implementation of the framework, namely photonic quantum computers [16, 17, 18]. We focus on the criticisms arising when simulating Gaussian boson sampling hardware [19].

## 1.1 Classical computation in a nutshell

Classical computation is based on boolean algebra. Its fundamental unit is the *bit*, a binary variable with values $\{0, 1\}$. Each task performed by a computer is translated, at the lowest level, into operations applied to a set of bits. These operations, called gates, can be modeled as binary functions $f : \{0, 1\}^n \to \{0, 1\}$, where $n$ is the number of input bits. Only a few gates, forming the so-called *universal* set, are implemented in the hardware because it has been proven that every binary function is modular in a set of elementary logic gates [20].

We now present the truth tables of a universal gate set as an example. A truth table is a table where on the left column we list all the possible configurations of the input bits, while on the right the gate output is reported. The chosen universal gate set is composed by:

- NOT, which negates the input bit;

- AND, which outputs 1 if and only if both inputs are 1;
- OR, which outputs 1 if either of the inputs is 1;
- COPY, which copies the input bit state to another one.

The corresponding truth tables are reported in Table 1.1. Having defined the ba-

| $a$ | COPY | NOT |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

**(a)** Truth table of one-bit gates.

| $a$ | $b$ | AND | OR |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 |

**(b)** The truth table of two-bits gates.

**Table 1.1:** Gates truth tables. In the first row, we have the bit and gate name. On the following rows, we have the states. On the left, we report the input states, while on the right the correspondent gate output.

sics states and operations in the classical regime we can now proceed, introducing the quantum bit.

## 1.2 The Qubit

The fundamental unit of quantum computation is the qubit. It is the quantum version of the classical bit. It is a two-level system, whose general state $|\psi\rangle$ can be represented as follows:

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \quad \text{with} \quad |\alpha_0|^2 + |\alpha_1|^2 = 1, \qquad \alpha_0, \alpha_1 \in \mathbb{C}, \tag{1.1}$$

where on the right we have the normalization condition.

There is another way of defining the state of a single qubit, namely as a unit vector on the Bloch Sphere (Figure 1.1 ). This sphere is useful for having a visual representation of the state. We can uniquely identify a quantum state through the angles $\theta \in [0, \pi], \phi \in [0, 2\pi]$ as follows:

$$|\psi\rangle = \cos\frac{\theta}{2} |0\rangle + e^{i\phi} \sin\frac{\theta}{2} |1\rangle = \begin{pmatrix} \cos\frac{\theta}{2} \\ e^{i\phi} \sin\frac{\theta}{2} \end{pmatrix}. \tag{1.2}$$

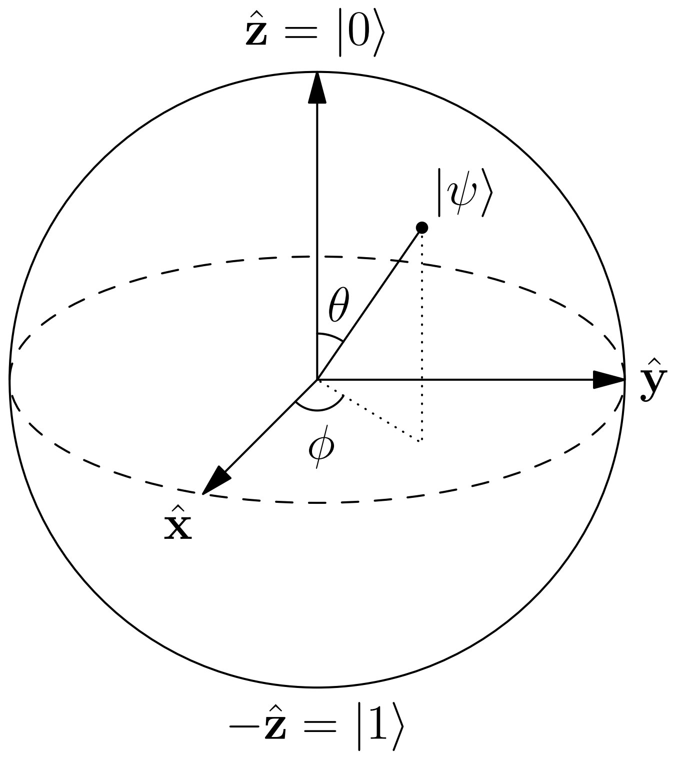


**Fig. 1.1:** The Bloch sphere is a sphere with unitary radius. Through the angles $\theta$, $\phi$ we can uniquely identify the pure state of a qubit, as shown in Equation (1.2). When the unit vector is aligned with the $z$ axis, it represents the $|0\rangle$ state if it points upward, or the $|1\rangle$ state if it points downward. Image is taken from Wikipedia.

We presented the state of a single qubit. The more important features of quantum computing, such as entanglement, arise when we have a quantum many-body state, composed in our case by $n$ qubits.

### 1.2.1 Composite systems and entanglement

We are now interested in describing the state of many qubits. We introduce it for a general $d$-level system, since we need this more general framework in Section 1.5. The state of a single degree of freedom $|\phi\rangle_i$ lives in a Hilbert space $\mathcal{H}_i$ with dimension $\dim(\mathcal{H}_i) = d$, and we can write it as:

$$|\phi\rangle_i = \sum_{j=1}^{d} c_j |j\rangle_i, \quad \sum_{j=1}^{d} |c_j|^2 = 1, \tag{1.3}$$

where $\{|j\rangle_i\}_{j=1,\dots,d}$ is an orthonormal basis of $\mathcal{H}_i$.

If we now consider $n$ degrees of freedom their state is defined in the tensor product of the local Hilbert spaces $\mathcal{H}_i$:

$$\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \cdots \otimes \mathcal{H}_n = \bigotimes_{i=1}^{n} \mathcal{H}_i. \tag{1.4}$$

The most general state $|\psi\rangle \in \mathcal{H}$ can be expressed as a linear combination of the tensor product of the orthonormal basis:

$$|\psi\rangle = \sum_{\vec{j}} c_{\vec{j}} |j\rangle_1 |j\rangle_2 \dots |j\rangle_n, \quad \sum_{\vec{j}} |c_{\vec{j}}|^2 = 1. \tag{1.5}$$

We notice that $c_{\vec{j}}$ is a $n$-dimensional tensor, with local dimension $d$, and thus have $d^n$ elements. We can so state that the Hilbert state of a composite system *scales exponentially* with the number of degrees of freedom.

We proceed now and define the *entanglement*, a purely quantum resource [21, 22]. We consider the so-called *bipartite entanglement* relative to the bipartitions $(A, B)$ of a pure state $|\psi\rangle$, each with $A$ $(B)$ degrees of freedom of local dimension $d$. By introducing bases $\{|\phi_i^A\rangle\}$ and $\{|\phi_j^B\rangle\}$ for the two subsystems, which are respectively of dimension $d^A$ and $d^B$, we can write any state of the whole system in a product basis as follows:

$$|\psi\rangle = \sum_{ij} \alpha_{ij} |\phi_i^A\rangle |\phi_j^B\rangle. \tag{1.6}$$

However, using the Schmidt decomposition [23], we can turn the double sum into a single one:

$$|\psi\rangle = \sum_{\alpha}^{r} \lambda_\alpha \left|\phi_\alpha^A\right\rangle \left|\phi_\alpha^B\right\rangle, \quad \sum_\alpha \lambda_\alpha^2 = 1, \tag{1.7}$$

where $r \in \left[1, \min(d^A, d^B)\right]$ is the Schmidt rank. The Schmidt rank is equal to 1 only for a *product state*, which by definition is not entangled, whereas a Schmidt rank $r > 1$ indicates non-zero entanglement between the two parts. From the Schmidt coefficients $\{\lambda_\alpha\}$, which are real, non-negative, and unique (for a given state), we can obtain the *entanglement entropy*. This is simply the Von Neumann entropy evaluated on a subsystem:

$$S_V = -\sum_\alpha \lambda_\alpha^2 \ln \lambda_\alpha^2. \tag{1.8}$$

## 1.3 State evolution

The evolution of a quantum state $|\psi\rangle$ is obtained solving the Schrödinger differential equation:

$$i\hbar \frac{\partial |\psi\rangle}{\partial t} = H |\psi\rangle, \tag{1.9}$$

where $H$ is the Hamiltonian of the system. We denote with $|n\rangle$ the eigenvector of $H$ with eigenvalue $\epsilon_n$. We can then solve Equation (1.9) by writing $|\psi\rangle$ in the eigenvector basis, obtaining:

$$|\psi(t)\rangle = U(t) |\psi(0)\rangle = \sum_n e^{-\frac{i}{\hbar}\epsilon_n t} c_n |n\rangle, \tag{1.10}$$

where $U(t)$ is a *unitary* operator.

Using an appropriate Hamiltonian we can evolve the system arbitrarily. This means that we can manipulate a system by applying different Hamiltonians for a given amount of time. Indeed, we can abstract our reasoning even further: instead of Hamiltonians we take into account only its effect, the evolution operator $U(\bar{t}) = U$, where $\bar{t}$ is the exact time needed for the transformation we are interested in. We now focus on the qubits case.

### 1.3.1 Quantum gates

At the detail level of this thesis, we are not interested in the microscopic Hamiltonians that generate the dynamics of the qubit. Instead, we take into account only their unitary representations $U$. These operators are called *quantum gates.* We so focus on the *quantum circuit* model, where the qubits are represented as lines, called qubit wires, where the gates are applied, with time flowing from left to right. Examples of simple quantum circuits are shown in Figure 1.2, 1.3.

Quantum gates are analogous to classical gates, with the important difference being that they are unitary operators. This means that quantum circuits are always *reversible* since we know that for a unitary operator $U$ holds $UU^\dagger = \mathbb{1}$, wherewith $(\cdot)^\dagger$ we denote the adjoint operation. We list now some of them as an example.

#### One-qubit gates

One-qubit gates acts only on a single qubit, and can be modeled as a $2 \times 2$ unitary matrix. Some examples, which graphical representation is in Figure 1.2, are:

- The Hadamard gate. It acts on the computational basis as follows: $\{\left|0\right\rangle, \left|1\right\rangle\} \xrightarrow{H} \{\frac{\left|0\right\rangle+\left|1\right\rangle}{\sqrt{2}}, \frac{\left|0\right\rangle+\left|1\right\rangle}{\sqrt{2}}\}$. Its matrix representation is:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{1.11}$$

- The phase shift. It adds a relative phase to the qubit's state: $\frac{\left|0\right\rangle+\left|1\right\rangle}{\sqrt{2}} \xrightarrow{P(\phi)} \frac{\left|0\right\rangle+e^{i\phi}\left|1\right\rangle}{\sqrt{2}}$.

$$P(\phi) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}. \tag{1.12}$$

- The NOT or X gate. It acts on the computational basis as follows, flipping the state: $\{\left|0\right\rangle, \left|1\right\rangle\} \xrightarrow{X} \{\left|1\right\rangle, \left|0\right\rangle\}$.

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{1.13}$$

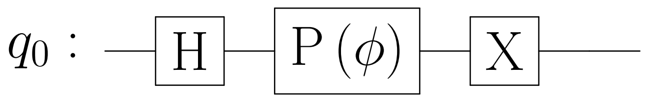


**Fig. 1.2:** One-qubit gates applied to the qubit 0. Starting from the left, we have the Hadamard gate (Equation (1.11)), the Phase gate (Equation (1.12) and the NOT gate (Equation (1.13)).

### Two-qubits gates

Quantum circuits act on qubit registers modifying their state. One common type of two-qubits gates is called controlled-gate, in which the first qubits acts as a *control* on the application of a one-qubit gate on the second qubit, called *target*. Two-qubits gates are represented by unitary $4 \times 4$ matrices. They are particularly important in quantum circuits since they enable us to create entanglement. Some examples, shown in Figure 1.3, are:

- The Controlled Not, also named as CNOT or CX. It applies a NOT gate on the target qubit if and only if the control qubit is in the $|1\rangle$ state.

$$CX = \begin{pmatrix} \mathbb{1}_{2\times 2} & \mathbb{O}_{2\times 2} \\ \mathbb{O}_{2\times 2} & X \end{pmatrix}, \tag{1.14}$$

  where $\mathbb{1}_{2\times 2}$ is the $2 \times 2$ identity matrix and $\mathbb{O}_{2\times 2}$ is the $2 \times 2$ matrix with only 0 elements.

- The Control Phase Shift, it applies a phase shift represented by $P(\theta)$ to the target qubit if and only if the control qubit is in the $|1\rangle$ state.

$$CPHASE(\theta) = \begin{pmatrix} \mathbb{1}_{2\times 2} & \mathbb{O}_{2\times 2} \\ \mathbb{O}_{2\times 2} & P(\theta) \end{pmatrix}. \tag{1.15}$$

- The Swap, it swaps the states of the qubits.

$$SWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{1.16}$$

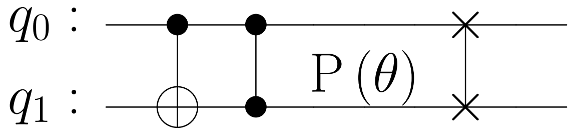


**Fig. 1.3:** Two-qubits gates applied to the qubit 0 and 1. Starting from the left, we have the CNOT gate (Equation (1.14)), the CPhase gate (Equation (1.15) and the SWAP gate (Equation (1.16) ). Notice that while the CNOT gate has a control qubit (black dot) and a target qubit (crossed circle), the CPhase and SWAP are symmetric across control/target.

It has been shown that a universal set of qubits gates is formed by the Hadamard gate $H$, the $\pi/4$ phase shift $T = P(\pi/4)$ and the control NOT $CX$. This is particularly important since it means that in order to have a universal quantum computer we only need to implement one and two-qubits gates. For this reason, we do not spend any effort in defining gates involving more qubits.

We now define an important quantity of quantum circuits: the *circuit depth*. The circuit depth is the length of the longest path from the input (or from a preparation) to the output (or a measurement gate), moving forward in time along each qubit wire. The stopping points on the path are the gates, the allowed paths that must be considered can enter and exit those gates on any input or output, and the length is the number of jumps from each gate to the next gates along the path. We suggest the following approach to calculate the depth: (a) consider each gate takes the same time to be applied, which we call time step. It is an approximation, which does not hold in general for 1- vs 2-qubit gates in experiments or simulations. (b) Multiple gates can be implemented within the same time step if no qubit appears in more than one gate and (c) the original order of gates does not change from the viewpoint of each qubit, i.e., a sequence of gates $G_{1,2}$ $G_3$ can be swapped, but $G_{1,2}$ $G_1$ cannot be swapped. Then, the depth is the number of time steps needed to simulate the quantum circuit. For example, both the circuits on Figure 1.3 and 1.2 has depth 3, while the circuit on Figure 1.4 has depth 7, since the gates enclosed in the dashed rectangle can be performed at the same time.

As an example of a more complex quantum circuit that contains some of the gates defined above, we present now the Quantum Fourier Transform algorithm.

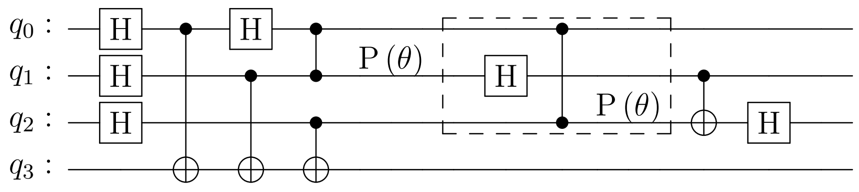


**Fig. 1.4:** Example of quantum circuit to better understand the definition of circuit depth. Gates on the same column or enclosed in the dashed rectangle can be executed at the same time, meaning that they count as one unit in the computation of the depth. Thus, this circuit has depth 7.

### 1.3.2 Quantum Fourier Transform

The Quantum Fourier Transform (QFT) [24] is the quantum analogue of the inverse discrete Fourier transform for qubits. It is the main ingredient of many quantum algorithm [25], and we employ it in Chapter 3 to test the performances of the developed code.

Given a sequence of $N$ complex terms $\{f_k\}_{k=0,\dots,N-1}$, with $f_k \in \mathbb{C}$, the **inverse Discrete Fourier Transform** is a linear transformation $\mathcal{F}^{-1}\colon \mathbb{C}^N \to \mathbb{C}^N$ mapping each $f_k \mapsto \tilde{f}_k \in \mathbb{C}$ to:

$$\tilde{f}_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} f_j \exp\left(\frac{2\pi i}{N} j\, k\right). \tag{1.17}$$

The quantum analogue, i.e. the **Quantum Fourier Transform** (QFT), is a linear transformation performed on $n$ qubits, which acts on the states of the computational basis $\{|j\rangle\}_{j=0,\dots,N-1}$, with $N = 2^n$, according to:

$$|QFT\rangle\langle QFT|\,(|j\rangle) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \exp\left(\frac{2\pi i}{N} j\, k\right) |k\rangle \quad \forall\, |j\rangle \in \mathbb{C}^N. \tag{1.18}$$

The QFT can be written as a sequence (circuit) of quantum gates, which can be implemented in a quantum computer.

To do so, we start by representing $k \in \{0, \dots, N-1\}$ as a binary number:

$$k = k_{n-1}2^{n-1} + \cdots + k_0 2^0 = \sum_{t=0}^{n-1} k_t 2^t,$$

We consider $k_i$ as the $i-$th digit, with $k_0$ being the least significant one (i.e., the Least Significant Bit, LSB), while the Most Significant Bit (MSB) is $k_{n-1}$. This allows us to rewrite the elements of the computational basis as the tensor product of the local bases at each qubit:

$$|k\rangle = \bigotimes_{l=1}^{n} |k_{n-l}\rangle \equiv |k_{n-1} \cdots k_0\rangle \, . \tag{1.19}$$

With this notation, (1.18) becomes a nested sequence of summations:

$$|QFT\rangle\!\langle QFT| \, (|j\rangle) = \frac{1}{\sqrt{N}} \sum_{k_{n-1}=0}^{1} \cdots \sum_{k_0=0}^{1} \exp\left( \frac{2\pi i j}{2^n} \sum_{t=0}^{n-1} 2^t k_t \right) |k_{n-1} \cdots k_0\rangle \, . \tag{1.20}$$

By defining $l = n - t$, the summation highlighted in red can be rewritten as:

$$\sum_{t=0}^{n-1} \frac{2^t k_t}{2^n} = \sum_{t=0}^{n-1} \frac{k_t}{2^{n-t}} = \sum_{l=1}^{n} \frac{k_{n-l}}{2^l} .$$

Then, by converting the exponential of the summation into a product of exponentials, Equation (1.20) becomes:

$$|QFT\rangle\!\langle QFT| \, (|j\rangle) = \frac{1}{\sqrt{N}} \sum_{k_{n-1}=0}^{1} \cdots \sum_{k_0=0}^{1} \left[ \prod_{l=1}^{n} \exp\left( 2\pi i j \frac{k_{n-l}}{2^l} \right) \right] |k_{n-1} \cdots k_0\rangle \, .$$

We can now use Equation (1.19) to separate the qubits:

$$\begin{aligned}
|QFT\rangle\!\langle QFT| \, (|j\rangle) &= \frac{1}{\sqrt{N}} \sum_{k_{n-1}=0}^{1} \cdots \sum_{k_0=0}^{1} \bigotimes_{l=1}^{n} \exp\left( 2\pi i j \frac{k_{n-l}}{2^l} \right) |k_{n-l}\rangle = \\
&= \frac{1}{\sqrt{N}} \bigotimes_{l=1}^{n} \left[ \sum_{k_{n-l}=0}^{1} \exp\left( 2\pi i j \frac{k_{n-l}}{2^l} \right) |k_{n-l}\rangle \right] \\
&= \frac{1}{\sqrt{N}} \bigotimes_{l=1}^{n} \left[ |0\rangle + \exp\left( 2\pi i j \frac{1}{2^l} \right) |1\rangle \right] .
\end{aligned}$$

Finally, we convert $j$ in binary notation too:

$$j = j_{n-1} j_{n-2} \ldots j_0 = \sum_{t=0}^{n-1} j_t 2^t ,$$

and we also introduce the fractional binary notation:

$$0.j_l j_{l+1} \dots j_m = \frac{1}{2} j_l + \frac{1}{4} j_{l+1} + \dots + \frac{1}{2^{m-l+1}} j_m .$$

In this way, the term $j/2^l$ can be rewritten as:

$$\frac{j}{2^l} = j_{n-1} j_{n-2} \dots j_{l+1} . j_l \dots j_0 .$$

Thanks to the properties of the exponential, the integer, and the fractional part can be factorized. Note that $\exp(2\pi i j_{n-1} \dots j_{l+1}) = 1$, and so it can be removed.

This allows to further expand the tensor product:

$$\begin{aligned} |QFT\rangle\langle QFT| (|j\rangle) = \frac{1}{\sqrt{N}} & \Big[ |0\rangle + \exp(2\pi i 0.j_0) |1\rangle \Big]_{n-1} \otimes \\ & \otimes \Big[ |0\rangle + \exp(2\pi i 0.j_1 j_0) |1\rangle \Big]_{n-2} \otimes \dots \otimes \\ & \otimes \Big[ |0\rangle + \exp(2\pi i 0.j_{n-1} j_{n-2} \dots j_0) |1\rangle \Big]_0 . \end{aligned} \tag{1.21}$$

This final expression can be used now to express the QFT operation via elementary gates, leading to the quantum circuit implementation of the algorithm.

We start by noticing that the *last* qubit after the transformation (the $[\dots]_{n-1}$ term) only depends on the first one ($|j_0\rangle$), as it emerges by applying a Hadamard gate:

$$|QFT\rangle\langle QFT| (|j\rangle)_{n-1} \equiv \left|\tilde{j}\right\rangle_{n-1} = \frac{1}{\sqrt{2}} \Big[ |0\rangle + \exp(2\pi i 0.j_0) |1\rangle \Big] = \mathrm{H} |j_0\rangle . \tag{1.22}$$

In fact, there are only two possible cases, since $j_0$ can be either 1 or 0:

$$j_0 = 0 : \qquad \left|\tilde{j}\right\rangle_{n-1} = \frac{1}{\sqrt{2}} \Big[ |0\rangle + |1\rangle \Big] = \mathrm{H} |0\rangle \tag{1.23}$$

$$j_0 = 1 : \qquad \left|\tilde{j}\right\rangle_{n-1} = \frac{1}{\sqrt{2}} \Big[ |0\rangle + \exp(\pi i) |1\rangle \Big] = \frac{1}{\sqrt{2}} \Big[ |0\rangle - |1\rangle \Big] = \mathrm{H} |1\rangle . \tag{1.24}$$

The qubit immediately before ($n-2$-th) is indeed more complex to derive, especially due to its phase. The $0.j_1$ term can be computed, similarly to before, via a Hadamard gate applied to $|j_1\rangle$, however, the $0.0j_0$ expression requires a *controlled phase* (CPHASE gate) dependent on $|k_0\rangle$.

We define a $k$-order CPHASE gate as follows:

$$\mathrm{P}_k = \begin{pmatrix} \mathbb{1}_2 & \mathbb{O} \\ \mathbb{O} & \mathrm{P}(2\pi i/2^k) \end{pmatrix} . \tag{1.25}$$

Thus:

$$\left|\tilde{j}\right\rangle_{n-2} = \mathrm{P}_2^{|j_0\rangle} \mathrm{H} \left|j_1\right\rangle . \tag{1.26}$$

The same argument can be repeated for all the other qubits. For $m \geq 1$, we get:

$$\left|\tilde{j}\right\rangle_m = \mathrm{P}_{m+1}^{|j_0\rangle} \cdots \mathrm{P}_3^{|j_{m-2}\rangle} \mathrm{P}_2^{|j_{m-1}\rangle} \mathrm{H} \left|j_m\right\rangle = \left( \prod_{l=0}^{m-1} \mathrm{P}_{m+1-l}^{|j_l\rangle} \right) \mathrm{H} \left|j_m\right\rangle .$$

Summarising all the previous deductions and computations, one can obtain the final quantum circuit, displayed in Figure 1.5. It is worth observing that a SWAP operation of order $O(n)$ — or at least a *renaming* of qubits — must be implemented to maintain the original qubit order since it is inverted by the QFT.

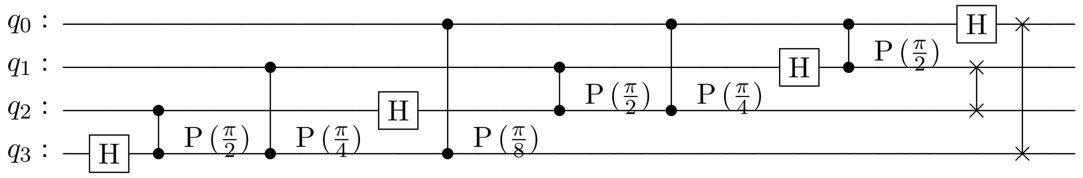


**Fig. 1.5:** Circuit implementation for the QFT on $n = 4$ qubits.

In conclusion, considering the number of quantum gates adopted in this circuit ($n^2$), one can at first infer that the order of the algorithm is $O(n^2)$, which compared to its classical counterpart (Fast Fourier Transform order: $O(nN)$) is exponentially more efficient.

We now present another configuration of the QFT circuit, that makes use only of *local* gates. We classify a gate as *non-local* if it is applied to non-adjacent qubits. In Chapter 2, we will see the importance of a local circuit.

To understand the new structure, let us recall that the QFT reverses the order of qubits. This reversal can be removed by iteratively swapping neighboring qubits. For instance, for $n = 4$, if we name the qubits with their index after the QFT, we get the reversed order 3210. The first-place qubit ("3") can be brought to its correct position (the rightmost one) by applying 3 SWAPs: 3210 $\rightarrow$ 2310 $\rightarrow$ 1230 $\rightarrow$ 2103. A second "pass" of SWAPs can be used to move also the "2" to its right position (2103 $\rightarrow$ 1203 $\rightarrow$ 1023), and a last SWAP brings "0" and "1" to

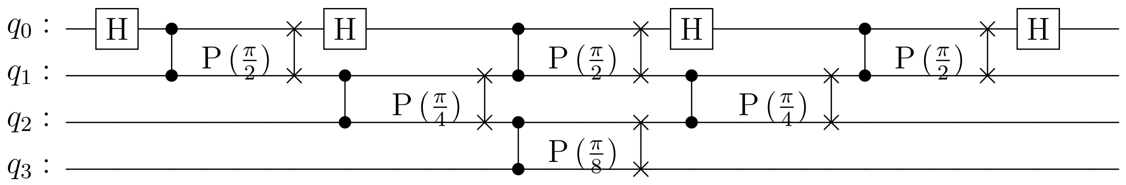


**Fig. 1.6:** Local circuit implementation for the QFT with $n = 4$ qubits.

their correct places: 1023 → 0123. Now, note that in the original circuit (Figure 1.5), the qubit at the $i$-th place interacts with a CPHASE with all the qubits "to its right". For instance, "3" in 3210 interacts with "2", "1" and "0" in succession. But during the first pass of SWAPs, "3" is moved iteratively "to the right", and becomes neighbor of qubits "2", "1" and "0" in sequence. This property holds for the other qubits too, meaning that the entire circuit can be rewritten by placing a SWAP after each CPHASE, as shown in the figure. In particular, all CPHASEs with the same phase are applied on the same "layer", and that a "pass" of SWAPs is achieved from the top-left to the bottom-right as shown in Figure 1.6. This wiring makes all gates local and preserves the initial ordering of qubits. We show the new circuit configuration in Figure 1.6.

Up to now, we have discussed the *unitary* evolution of a quantum state. This has been done because the simulation methods that are presented in Chapter 2 are aimed at the simulation of *pure* states undergoing a *unitary* evolution. However, these methods can introduce an error on the state. We justify this error, stating that if it is comparable with the device's error, then the simulation is meaningful. We then briefly overview a way to characterize this error in the following section.

### 1.3.3 Quantum channels

In real experiments, it is impossible to perfectly insulate a quantum system, as the system interacts with the environment. Even though the evolution of the system+environment is still unitary, we can observe only the system, and thus observe a non-unitary evolution. In order to describe a non-unitary evolution of a quantum system we introduce a more general tool to represent quantum states, the *density matrix* $\rho \in \mathcal{H}$. Using it we can represent *mixed states*, i.e. states where we do not have complete knowledge. It is defined as:

$$\rho = \sum_i p_i \left|\psi_i\right\rangle \left\langle\psi_i\right| , \tag{1.27}$$

where $p_i$ is the probability of being in the quantum state $|\psi_i\rangle \in \mathcal{H}$. The density matrix has the following properties:

1. $\rho$ is hermitian;
2. $\rho$ is a non-negative operator, i.e. $\langle\psi|\,\rho\,|\psi\rangle \geq 0 \quad \forall\,|\psi\rangle$;
3. We define the trace of an operator A as $\mathrm{Tr(A)} = \sum_k \langle k|\,\mathrm{A}\,|k\rangle$ where $\{|k\rangle\}_{k=1,\dots,n}$ is a basis of the Hilbert space $\mathcal{H}$. Then, $\mathrm{Tr}(\rho) = 1$.

We can describe a *pure state* using a density matrix. In that case we have $p_j = 1$, $p_{i\neq j} = 0$. Furthermore, a density matrix is pure if and only if $\rho^2 = \rho$ and $\mathrm{Tr}(\rho^2) = 1$. Instead, if $\mathrm{Tr}(\rho^2) < 1$ then the state is mixed.

If we have a composite system we can focus only on a subsystem, tracing away the other. We consider a bipartite system divided into two parts $(A, B)$, each with $A(B)$ degrees of freedom of local dimension d. By introducing bases $\{\left|\phi_i^A\right\rangle\}$ and $\{\left|\phi_j^B\right\rangle\}$ for the two subsystems, which are respectively of dimension $d_A$ and $d_B$. If a state $\rho$ is defined in $A \otimes B$ then we can focus on the state $\rho^A$ defined only on $A$ as:

$$\rho^A = \mathrm{Tr}_B(\rho) = \sum_j \left\langle\phi_j^B\right|\rho\left|\phi_j^B\right\rangle . \tag{1.28}$$

Indeed, when we trace away a subsystem we can pass from a pure to a mixed state. Let us consider, as an example, a Bell state $|\psi\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right)$. We can write the related density matrix as:

$$\rho = |\psi\rangle\langle\psi| = \frac{1}{2}\left(|00\rangle\langle 00| + |11\rangle\langle 11|\right) = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} .$$

We notice that $\rho^2 = \rho$, and so we confirm that the state is pure. If we trace away the second qubit we obtain:

$$\rho_1 = \mathrm{Tr}_2(\rho) = \sum_j \langle j|_2\,\rho\,|j\rangle_2 = \frac{1}{2}(|0\rangle_1\langle 0|_1 + |1\rangle_1\langle 1|_1).$$

We notice that $\rho_1^2 \neq \rho_1$. This means that $\rho_1$ is mixed. We so passed from a pure to a mixed state by focusing on a subsystem.

We now have all the ingredients to present the motivation behind the introduction of quantum channels. If $A$ is the quantum system and $B$ is the environment, then the total system $A+B$ undergoes a unitary evolution. However, since we can observe only $A$ we are effectively *tracing away* $B$, passing from a pure state to a mixed one. We can model the evolution of $A$ through a non-unitary evolution using the *Kraus representation* [26]:

$$\rho_A(t) = \mathrm{Tr}_B\big(\rho(t)\big) = \sum_k E_k \rho_A(t=0) E_k^\dagger, \qquad \text{with } \sum_k E_k^\dagger E_k = \mathbb{1}. \tag{1.29}$$

The $E_k$ are called Kraus operators.

We list here some examples of quantum channels, highlighting their effect on the qubit's state:

- *Amplitude damping.* It is the process that makes the excited state $|1\rangle$ decaying into the ground state $|0\rangle$;
- *Phase damping.* It is the process that eliminates the coherence: it transforms the state $\frac{1}{2}\big(|0\rangle + |1\rangle\big)\big(\langle 0| + \langle 1|\big)$ into $\frac{1}{2}\big(|0\rangle\langle 0| + |1\rangle\langle 1|\big)$.
- *Depolarizing channel.* It is the process that depolarizes the qubit, diminishing the projections of the state over the axis $x$, $y$, and $z$. Over the time the arbitrary state decays in $\frac{1}{2}\big(|0\rangle\langle 0| + |1\rangle\langle 1|\big)$.

It is not the aim of this dissertation of going into details about quantum channels and non-unitary evolution. The important message from this subsection is that there are errors in the states of quantum computers, due to the interaction with the environment. We can so exploit this fact, and state that symmetrically we can tolerate errors in the simulations, as long as they are small enough. We can set the threshold of the acceptability at an error of $10^{-4}$, which is the *fault-tolerant* threshold.

## 1.4 Simulation methods

We now review some methods for the exact simulation of quantum circuits. It is indeed a difficult task since the dimension of the Hilbert space scales exponentially with the number of qubits $n$.

### 1.4.1 Linear algebra

The most straightforward simulation method involves the exact simulation of the state. We represent the state of a $n$-qubits system $|\psi\rangle$ as a $2^n$ dimensional vector, and then apply the gates. Indeed, we can not simply use the expression for the gates presented in Section 1.3.1, but extend it on the entire Hilbert space. A one-qubit gate $\mathrm{G}_i = \begin{pmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{pmatrix}$ acting on the $i$-th qubit can be extended on the full space as follows:

$$\mathrm{U} = \mathbb{1}_1 \otimes \mathbb{1}_2 \otimes \cdots \otimes \mathbb{1}_{i-1} \otimes \mathrm{G}_i \otimes \mathbb{1}_{i+1} \otimes \cdots \otimes \mathbb{1}_n. \tag{1.30}$$

The problem is that now U is a $2^{2n}$ matrix, which explodes even faster than the state. However, we do not need to construct the full U, and we can perform apply the gate G directly on the state-vector. We denote the amplitudes of the state vector $|\psi\rangle$ by their binary index. For example, if we are worknig with $n = 2$ qubits then $|\psi\rangle = (\alpha_{00}, (\alpha_{01}, (\alpha_{10}, (\alpha_{11})$. Then, the gate induces a transformation to pairs of amplitudes whose indices differ in $i$-th bits of their binary index:

$$\alpha'_{*\cdots*0_i*\cdots*} = g_{11} \cdot \alpha_{*\cdots*0_i*\cdots*} + g_{12} \cdot \alpha_{*\cdots*1_i*\cdots*}, \tag{1.31}$$

$$\alpha'_{*\cdots*1_i*\cdots*} = g_{21} \cdot \alpha_{*\cdots*0_i*\cdots*} + g_{22} \cdot \alpha_{*\cdots*1_i*\cdots*}, \tag{1.32}$$

where with $*$ we denote all the possible configuration of the other binary indices. The number of operation needed to perform an update for single-qubit gates is so $2^n$. The operations are analogous for two-qubits gates. This means that, for updating the state vector we perform $O(2^n)$ operations. We recall that a set of universal gate is formed by only one-qubits gates and two-qubits gates [13], and so it is sufficient to be able to simulate those.

There are many techniques to speed up this process, such as running the simulation on GPUs, or on multiple threads [27, 28]. However, as we will see in Chapter 3, even running the simulation on 128 threads on the m100 cluster enable us to run the QFT only on up to 32 qubits.

### 1.4.2 Stabilizers

We have seen in the previous section that the exact simulation of an arbitrary quantum circuit is exponentially difficult. However, this simulation becomes much more feasible if we put some constraints on the quantum circuit. The Gottesman-Knill theorem [29] states that, if we only apply gates from the Clifford group

(CNOT, Hadamard, $\bar{P} = P(\frac{\pi}{2})$), then the circuit can be efficiently simulated on a classical computer. In particular, using the graph state formalism [30], it is possible to simulate circuits in $O(n \log n)$ time. We now give a brief overview of the method.

The key idea of the stabilizer formalism is to represent a quantum state $|\psi\rangle$, not by a vector of amplitudes, but by a stabilizers group, consisting of unitary matrices that stabilize $|\psi\rangle$. A unitary matrix U *stabilizes* a quantum state $|\psi\rangle$ if $|\psi\rangle$ is an eigenvector of U with eigenvalue 1, i.e. $\mathrm{U}\,|\psi\rangle = |\psi\rangle$. We do not neglect a global phase. Notice that if U and V both stabilize $|\psi\rangle$ then so do UV and $\mathrm{U}^{-1}$, and thus the set $\mathrm{Stab}(|\psi\rangle)$ of stabilizers of $|\psi\rangle$ is a group. Also, it is not hard to show that if $|\psi\rangle \neq |\phi\rangle$ then $\mathrm{Stab}(|\psi\rangle) \neq \mathrm{Stab}(|\phi\rangle)$. Remarkably, though, a large and interesting class of quantum states can be specified uniquely by much smaller stabilizer groups—specifically, the intersection of $\mathrm{Stab}(|\psi\rangle)$ with the Pauli group. A well-known fact from group theory says that any finite group G has a generating set of size at most $\log_2 \mathrm{G}$. So if $|\psi\rangle$ is a stabilizer state on $n$ qubits, then the group $S(|\psi\rangle)$ of Pauli operators that stabilize $|\psi\rangle$ has a generating set of size $n = \log_2 2^n$. Each generator takes $2n + 1$ bits to specify: 2 bits for each of the $n$ Pauli matrices, and 1 bit for the phase. So the total number of bits needed to specify $|\psi\rangle$ is $n(2n + 1)$.

What Gottesman and Knill showed, furthermore, is that these bits can be updated in polynomial time after a CNOT, Hadamard, phase, or measurement gate is applied to $|\psi\rangle$. The updates corresponding to unitary gates are very efficient, requiring only $O(n)$ time for each, while the measurements are more demanding, requiring $O(n^3)$.

However, there have been subsequent works that further optimize the algorithm [30, 14]. Furthermore, the Clifford gate set is a non-universal gate set, even if there has been some effort to extend the formalism to include also the $T$ gate, which promotes the set to universal [15].

## 1.5 Linear optics model

We define a *continuous-variable (CV) model* as a model where the quantum operators underlying the model have continuous spectra. Many physical systems, such as light, are continuous. These systems are defined in an infinite-dimensional Hilbert space, and so offer a different implementation of quantum computing with respect to qubits. The CV model is a natural fit for simulating bosonic systems.

In particular, we focus on the Gaussian Boson Sampling protocol.

The simplest CV system is the bosonic harmonic oscillator, defined via the canonical mode operators $\hat{a}$ and $\hat{a}^\dagger$. These satisfy the commutation relation $[\hat{a}, \hat{a}^\dagger] = \mathbb{1}$. It is also common to work with the quadrature operators:

$$\hat{x} = \sqrt{\frac{\hbar}{2}}\left(\hat{a} + \hat{a}^\dagger\right), \quad \hat{p} = -i\sqrt{\frac{\hbar}{2}}\left(\hat{a} - \hat{a}^\dagger\right), \tag{1.33}$$

where $\hat{x}$ is the position and $\hat{p}$ the momentum. We can picture a fixed harmonic oscillator mode as a single wire in the quantum circuits. These *qumodes* are the fundamental information-carrying units of CV quantum computers. We can implement a quantum computer model by combining several qumodes, each with their own operators $\{\hat{a}_i \hat{a}_i^\dagger\}_{i=1,\dots,m}$, and evolving them through suitable quantum gates. Furthermore, qubit-based computations can be reproduced with this model, by using the Gottesman-Knill-Preskill (GKP) embedding [31].

### 1.5.1 Qumodes

Qumodes are the CV counterpart of qubits. Even though we are in a CV model, we mainly represent the qumodes using a discrete basis, namely the *Fock basis* $|n\rangle, n \in \mathbb{N}$. They are the eigenstates of the number operator $\hat{n} = \hat{a}^\dagger \hat{a}$. They form a discrete countable basis for the states of a single qumode. We can identify with $n$ the number of photons in a qumode. We recall, for completeness, the effect of the operators $\hat{a}$, $\hat{a}^\dagger$ on a general Fock state $|n\rangle$:

$$\hat{a}\,|n\rangle = \sqrt{n}\,|n-1\rangle, \quad \hat{a}^\dagger\,|n\rangle = \sqrt{n+1}\,|n+1\rangle. \tag{1.34}$$

It is important to notice that, since the Fock states form a basis, it is possible to write any qumode state as a linear combination of the basis. For example, we present the Fock decomposition of a coherent state $|\alpha\rangle$, which is an eigenstate of the operator $\hat{a}$:

$$|\alpha\rangle = e^{-\frac{|\alpha|^2}{2}} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}}\,|n\rangle. \tag{1.35}$$

By recalling that our focus is on the classical simulation of a quantum computer, it is clear that we can not take into account an infinite basis set. For this reason, we introduce a cutoff in the Fock basis $f_c$, called *Fock space cutoff*. This means that the basis is now $|n\rangle, n \in [0, f_c]$. It is true that this is an approximation on the state, but as long as $f_c$ is chosen high enough this approximation is meaningful. This is because we always start from the *vacuum state*, namely $|0\rangle$, and we increment the number of photons in a mode through quantum gates.

There are other interesting representations of the CV model states. We briefly address the Gaussian states since they are very useful for understanding intuitively the effect of Gaussian gates.

#### Gaussian states

As we have seen in Section 1.3, the state $|0\rangle$, called *vacuum* in photonics, can be evolved according to:

$$|\psi\rangle = e^{-itH}\,|0\rangle\,,$$

where $H$ is a bosonic Hamiltonian and $t$ the evolution time. States where the Hamiltonian is at most quadratic in the operators $\hat{x}$ and $\hat{p}$ are called Gaussian. For a single qumode, Gaussian states are parameterized by two continuous complex variables: a displacement parameter $\alpha \in \mathbb{C}$ and a squeezing parameter $z \in \mathbb{C}$. Gaussian states are so-named because we can identify each Gaussian state, through its displacement and squeezing parameters, with a corresponding Gaussian distribution. The displacement gives the center of the Gaussian, while the squeezing determines the variance and rotation of the distribution.

### 1.5.2 Linear optics gates

Gates in quantum optics are particularly difficult to represent on the Fock basis. They are usually defined in terms of the operators $\hat{a}, \hat{a}^\dagger$. In this section, we address some of the quantum optics gates that we need for the Gaussian boson sampling protocol.

We stress that, since we need to simulate the gate application, we are not interested in the operator representation, but in the matrix representation in the Fock basis. So, if we call $\mathrm{U}(\hat{a}, \hat{a}^\dagger, \theta)$ a generic gate, then we need:

$$\mathrm{U}_{nm} = \langle n|\, U(\hat{a}, \hat{a}^\dagger, \theta)\, |m\rangle\,, \quad n, m \leq f_c, \tag{1.36}$$

where $f_c$ is the Fock space cutoff.

In order to present the effect of the single-mode gates, it is interesting to look at their effect in the position and momentum quadrature. We show these results in Figure 1.8.

Indeed, the same representation used for qubits-based quantum circuits holds for qumodes-based circuits. We present in Figure 1.7 the diagrammatic example of the gates.

### Rotation gate

The phase space rotation gate of an angle $\theta \in [0, 2\pi]$ is a single-qumode gate. It is defined as:

$$R(\theta) = e^{i\theta\hat{a}^\dagger\hat{a}}. \tag{1.37}$$

This form is particularly interesting, because in the exponent we can notice the number operator $\hat{n} = \hat{a}^\dagger\hat{a}$. We recall that the Fock states are eigenstates of $\hat{n}$. We can easily find the matrix elements of $R(\theta)$:

$$\begin{aligned} R(\theta)_{mn} &= \langle m| R(\theta) |n\rangle = \langle m| e^{i\theta n} |n\rangle = e^{i\theta n} \langle m|n\rangle \\ &= \begin{cases} 0 & \text{if } n \neq m \\ e^{i\theta n} & \text{if } n = m \end{cases}. \end{aligned} \tag{1.38}$$

The rotation gate is so a *diagonal matrix*. Furthermore, notice that $R(\theta)$ preserves the number of photons in a mode.

In the quadrature representation we can write the effect of the rotation gate as:

$$\begin{aligned} R^\dagger(\theta)\hat{x}R(\theta) &= \hat{x}\cos\theta - \hat{p}\sin\theta \\ R^\dagger(\theta)\hat{p}R(\theta) &= \hat{p}\cos\theta + \hat{x}\sin\theta. \end{aligned}$$

We notice that it rotates the position and momentum to each other.

In this particular case, the computation of the matrix elements was easy. However, it is not the same in the following. For this reason, we simply present the results.

### Displacement gate

The displacement gate $D(r, \phi)$, $r \in [0, \infty)$ $\phi \in [0, 2\pi)$, is a single-qumode gate defined as follows:

$$D(\alpha) = e^{\alpha\hat{a}^\dagger - \alpha^*\hat{a}}, \quad \alpha = re^{i\phi}. \tag{1.39}$$

The matrix elements of this gate were derived by Cahill and Glauber[32]:

$$D(\alpha)_{mn} = \sqrt{\frac{n!}{m!}} \alpha^{m-n} e^{-\frac{|\alpha|^2}{2}} L_n^{m-n}(|\alpha|^2), \tag{1.40}$$

where $L_n^m(x)$ is the generalized Laguerre polynomial.

In the quadrature representation we can write the effect of the displacement gate as:

$$\begin{aligned} D^\dagger(\alpha)\hat{x}D(\alpha) &= \hat{x} + \sqrt{2\hbar}\,\mathrm{Re}\{\alpha\}\hat{\mathbb{1}} \\ D^\dagger(\alpha)\hat{p}D(\alpha) &= \hat{p} + \sqrt{2\hbar}\,\mathrm{Im}\{\alpha\}\hat{\mathbb{1}}. \end{aligned}$$

We observe that it shifts the position and the momentum operator of a quantity which is proportional to respectively the real and imaginary part of $\alpha$.

## Squeezing gate

The squeezing gate $S(r, \phi)$, $r \in [0, \infty)$ $\phi \in [0, 2\pi)$, is a single-qumode gate defined as follows:

$$S(z) = e^{\frac{1}{2}\left[z{*}\hat{a}^2 - z(\hat{a}^\dagger)^2\right]}, \quad z = re^{i\phi}. \tag{1.41}$$

A Fock decomposition related to this gate was obtained by Krall [33]:

$$\begin{aligned} f_{n,m}(r,\phi,\beta) = \langle n|\, S(z^*)D(\beta)\,|m\rangle = \sqrt{\frac{n!}{\mu n!}} e^{\frac{\beta^2\nu^*}{2\mu} - \frac{|\beta|^2}{2}} \cdot \\ \cdot \sum_{j=0}^{\min(m,n)} \frac{\binom{m}{j}\left(\frac{1}{\mu\nu}\right)^{j/2} 2^{\frac{j-m}{2}+\frac{j}{2}-\frac{n}{2}} \left(\frac{\nu}{\mu}\right)^{n/2} \left(-\frac{\nu^*}{\mu}\right)^{\frac{m-j}{2}} \mathrm{H}_{n-j}\left(\frac{\beta}{\sqrt{2\mu\nu}}\right) \mathrm{H}_{m-j}\left(-\frac{\alpha^*}{\sqrt{-2\mu\nu^*}}\right)}{(n-j)!}, \end{aligned} \tag{1.42}$$

where $\nu = e^{-i\phi}\sinh(r)$, $\mu = \cosh(r)$, $\alpha = \beta\mu - \beta^*\nu$ and $\mathrm{H}_n(x)$ are the Hermite polynomials. In particular, to retrieve the squeezing gate, we are interested in the case $\beta \to 0$, and we have:

$$\mathrm{H}_n(0) = \begin{cases} 0 & \text{if } n \text{ is odd} \\ (-1)^{\frac{n}{2}} 2^{\frac{n}{2}} (n-1)!! & \text{if } n \text{ is even} \end{cases}. \tag{1.43}$$

We can so deduce that $f_{n,m}(r,\phi,0)$ is zero if $n$ is even and $m$ is odd or vice versa. So we obtain:

$$f_{n,m}(r,\phi,0) = D(r,\phi)_{nm} =$$
$$= \begin{cases} 0 & \text{if } n \text{ is odd and } m \text{ even or vice versa} \\ \sqrt{\frac{n!}{\mu n!}} \sum_{j=0}^{\min(m,n)} (-1)^{\gamma} \frac{\binom{m}{j}\left(\frac{1}{\mu\nu}\right)^{j/2}\left(\frac{\nu}{\mu}\right)^{n/2}\left(-\frac{\nu^*}{\mu}\right)^{\frac{m-j}{2}}(n-j-1)!!(m-j-1)!!}{(n-j)!} & \text{otherwise} \end{cases} \tag{1.44}$$

In the quadrature representation we can write the effect of the displacement gate as:

$$S^\dagger(z)\hat{x}S(z) = e^{-r}\hat{x}$$
$$S^\dagger(z)\hat{p}S(z) = e^{r}\hat{p}.$$

We notice that it shrinks the position while enlarging the momentum operator.

### Beamsplitter gate

The squeezing gate $BS(\theta,\phi)$, $\theta \in [0,2\pi)$ $\phi \in [0,2\pi)$, is a two-qumode gate defined as follows:

$$BS(\theta,\phi) = e^{\theta\left(e^{i\phi}\hat{a}_1\hat{a}_2^\dagger - e^{-i\phi}\hat{a}_1^\dagger\hat{a}_2\right)}. \tag{1.45}$$

The parameter $\theta$ is called transmittivity angle. The transmission amplitude of the beamsplitter is $t = \cos(\theta)$. By setting $\theta = \pi/4$ the 50-50 beamsplitter got implemented. Instead, $\phi$ is the phase angle. The reflection amplitude of the beamsplitter is $r = e^{i\phi}\sin(\theta)$. The value $\phi = \pi/2$ corresponds the symmetric beamsplitter.

The beamsplitter can be written in the Fock basis as [34]:

$$\begin{aligned} BS(\theta,\phi)_{n_1n_2}^{m_1m_2} &= \langle m_1m_2|\, BS(\theta,\phi)\, |n_1n_2\rangle = \\ &= e^{-i\phi(n_1-m_1)} \sum_{k=0}^{n_1}\sum_{l=0}^{n_2} (-1)^{n_1-k} r^{n_1+n_2-k-l} t^{k+l} \cdot \\ &\quad \frac{\sqrt{n_1!n_2!m_1!m_2!}}{k!(n_1-k)!l!(n_2-l)!}\delta_{m_1,n_2+k-l}\delta_{m_2,n_1-k+l}, \end{aligned} \tag{1.46}$$

where $\delta_{i,j}$ is the Kronecker delta. Using beamsplitters and phase gates is possible to build a multi-mode interferometer [35].

We now proceed by illustrating a protocol that uses the CV model, namely the Gaussian boson sampling.

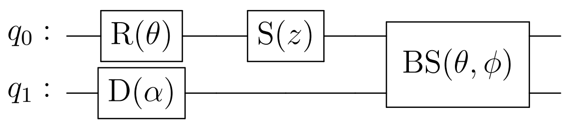


**Fig. 1.7:** Circuital representation of a bosonic circuit with its gates. On the qumode $q_0$ we have the Rotation phase gate (1.37) and the squeezing gate (1.41). On the qumode $q_1$ instead a displacement gate (1.39) is applied. Finally, on the right we show a beamsplitter gate (1.45) applied to both qumodes.

### 1.5.3 Gaussian boson sampling

Gaussian Boson Sampling (GBS) is a prototype model of photonic quantum computation [19]. It consists of preparing a multi-mode Gaussian state and measuring it in the Fock basis. Theoretically, the output distribution of a GBS device cannot be simulated in polynomial time with classical computers. It has been recently used in an experiment that claimes quantum supremacy [12].

GBS is computationally equivalent to sampling from the Hafnian function of a matrix. Given a graph $G$ with adjacency matrix $E$; the Hafnian of $E$ is the number of perfect matchings of the graph $G$. A matching of a graph $G$ is a subset of edges $M$ such that no two edges in $M$ have a vertex in common. A matching $M$ is perfect if every vertex is incident to exactly one edge in $M$. The Hafnian can be seen as a generalization of the Permanent of a matrix, which gives the number of perfect matchings for a bipartite graph. Given the adjacency matrix $E$, the relation between Hafnian and Permanent is:

$$\text{Haf}\begin{pmatrix} 0 & E \\ E^T & 0 \end{pmatrix} = \text{Per}(E). \tag{1.47}$$

We recall that the permanent of a $n \times n$ matrix $A$ of elements $a_{ij}$ is defined also as:

$$\text{Per}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i,\sigma(i)}, \tag{1.48}$$

where the $S_n$ is the symmetric group which contains all permutations of the numbers $1, 2, \dots, n$.

A general pure Gaussian state can be prepared from a vacuum state by a sequence of single-mode squeezing, multi-mode linear interferometry, and single-mode displacements. We show an example of the circuit for $n = 4$ modes in Figure

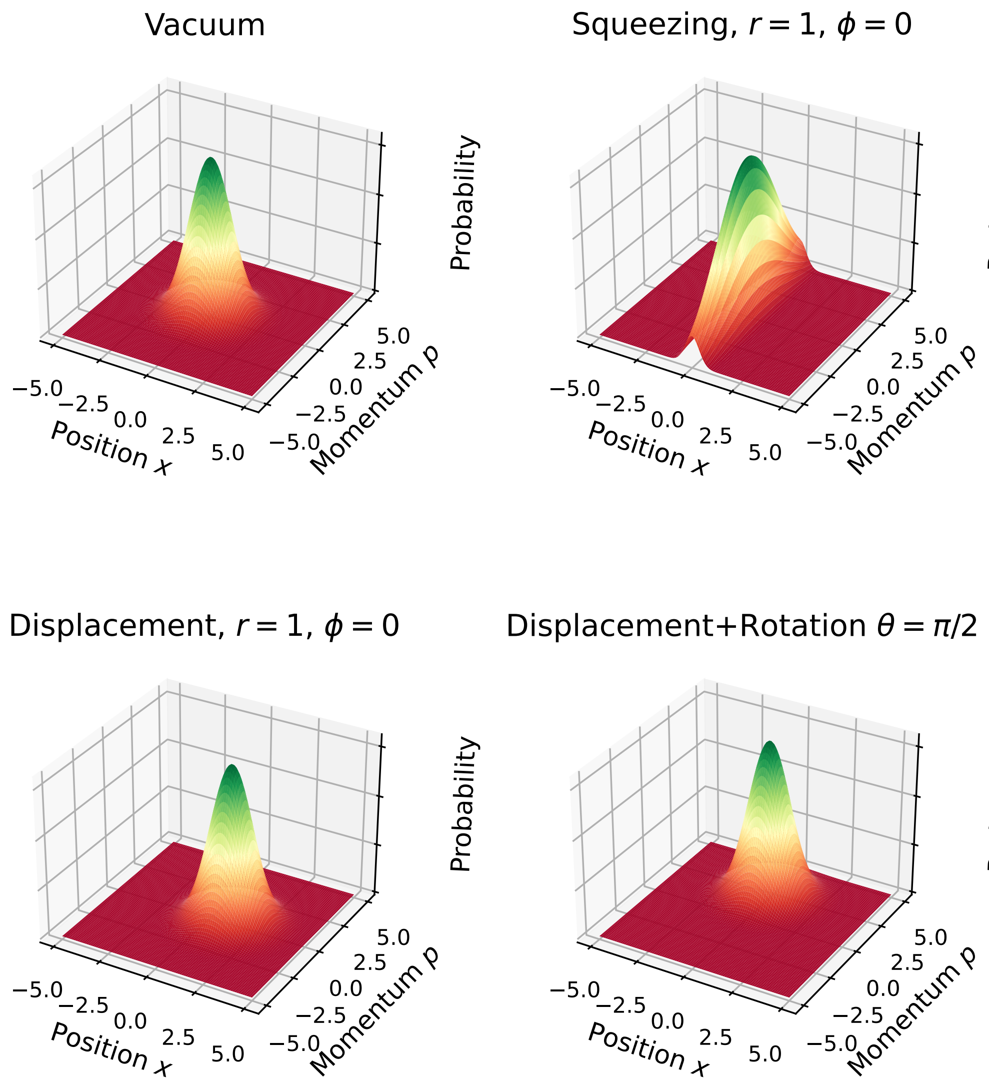


**Fig. 1.8:** Effect of linear optics gaussian gates on the vacuum state. Starting from the upper left, we see the Gaussian probability centered in 0. In the upper right, we apply the squeezing operator (1.41). We notice that the probability is reduced on the position axis, while it increments on the momentum one. On the lower left, we observe the effect of the displacement operator (1.39), which shifts the gaussian state. Finally, in the lower right, we see the effect of the rotation gate (1.37), starting from the state in the lower left.

1.9. Let us define the output state of a GBS as $|s\rangle = |s_1, s_2, \ldots, s_n\rangle$, where $s_i$ is the number of photons in the $i$-th mode. It was shown in [19] that for a Gaussian state with zero mean, which can be prepared only using squeezing followed by linear interferometry, the probability $p(s)$ of observing an output state $s$ is:

$$p(s) = \frac{1}{\sqrt{\det(\mathbf{Q})}} \frac{\mathrm{Haf}(\mathcal{A})}{\prod_{i=1}^{n} s_i!}, \tag{1.49}$$

where:

$$\mathbf{Q} = \Sigma + \mathbb{1}/2, \quad \mathcal{A} = \mathbf{X}\left(\mathbb{1} - \mathbf{Q}^{-1}\right), \quad \mathbf{X} = \begin{pmatrix} \mathbb{O} & \mathbb{1} \\ \mathbb{1} & \mathbb{O} \end{pmatrix},$$

and $\Sigma$ is the covariance matrix of the Gaussian state. When the state is pure, the matrix $\mathcal{A}$ can be written as $\mathcal{A} = \mathbf{A} \oplus \mathbf{A}^*$ and the probability distribution becomes:

$$p(s) = \frac{1}{\sqrt{\det(\mathbf{Q})}} \frac{|\mathrm{Haf}(\mathbf{A})|^2}{\prod_{i=1}^{n} s_i!}.$$

$\mathbf{A}$ is an arbitrary symmetric matrix with eigenvalues bonded between $-1$ and 1. Therefore, we can use a GBS device to encode symmetric matrices.

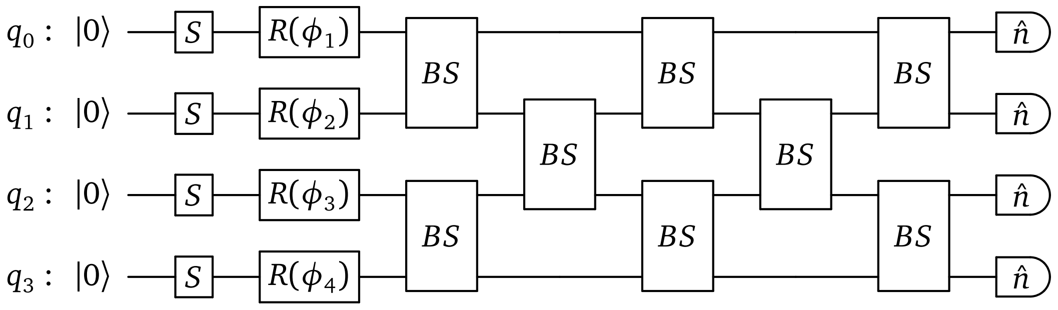


**Fig. 1.9:** Gaussian boson sampling circuit, in the case where we have no displacement. First, we put the system in a non-trivial Gaussian state using the squeezing operators and apply a linear interferometer which is decomposed in rotation and beamsplitter gates. Finally, we perform measurements of the photon number in each mode, performing a projective measurement on the Fock basis. Image from strawberry fields documentation.

### Encoding a matrix in a GBS device

In GBS without displacements, we can specify the symmetric matrix $\mathbf{A}$ by choosing the correct gate parameters. Employing the Takagi-Autonne decomposition, we can write:

$$\mathbf{A} = \mathrm{U}\ \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)\mathrm{U}^T, \tag{1.50}$$

where U is the unitary matrix that specifies the linear interferometer. The values $0 \leq \lambda_i < 1$ uniquely determine the sqeezing parameters $r_i$ through the relation $\tanh r_i = \lambda_i$. They also determine the mean photon number $\bar{n}$ of the distribution as follows:

$$\bar{n} = \sum_{i=1}^{m} \frac{\lambda_i^2}{1 - \lambda_i^2}. \tag{1.51}$$

It is possible to encode an arbitrary matrix A, by re-scaling the matrix with a parameter $c > 0$ such that $c$A satisfies the condition on the $\lambda_i$ in the above decomposition.

In the next chapter, we address the tensor network formalism, and how we can employ it to efficiently simulate quantum circuits.

# 2
# Matrix Product States

The Matrix Product States (MPS) are an efficient way of representing a quantum state of 1-dimensional systems. We motivate their use in Section 2.1, where we also present their definition, along with an intuitive way to physically construct them. Then, Section 2.2 introduces the graphical notation of tensor networks, of which MPS are a specific case. In Section 2.2.1 we discuss the choice of gauge for a tensor network, which can be used to simplify computations and improve the stability of algorithms. Thereafter, Section 2.2.2 contains a procedure to convert states from a full representation to an MPS. Section 2.3 introduces the operations that we can apply on an MPS. In Section 2.3.1 we finally present the Time Evolving Block Decimation: a standard technique to evolve a quantum many-body state under the effect of a local Hamiltonian, which is really important in the MPS framework, even though it is not used in this thesis.

## 2.1 Motivation and Construction

A generic wave function of a many-body quantum system can be expressed by listing its coefficients in a chosen base:

$$|\psi\rangle = \sum_{k=1}^{N} \mathbf{c}_k \, |k\rangle \, . \tag{2.1}$$

As we have already seen in the previous chapter, the number $N$ of needed coefficients scales exponentially with the system's size $n$. For example, if we have $n$ degrees of freedom (sites) with local dimension $d$, the most general wave function has $d^n$ coefficients:

$$|\psi\rangle = \sum_{s_1,\ldots,s_n=1}^{d} \mathbf{C}_{s_1\ldots s_n} |s_1 s_2 \ldots s_n\rangle .$$

However, we are usually interested in some specific states, e.g. the low energy ones. Their coefficients are not completely random, and so we may seek a more compressed representation.

In fact, we can argue that Equation (2.1) is an extremely inefficient way to define a state because most of the states we are interested in belong to a tiny subset of the whole system's Hilbert space.

A first intuition comes from the fact that most Hamiltonians are *local*, i.e. only sites that are *close* to each other interact significantly. For instance, consider a 1D spin chain with a finite correlation length $\xi$ [36, p. 9]. Two sites $A$ and $B$ which lie at a distance $l_{AB} \gg \xi$ are effectively independent, and their state $\psi_{AB}$ can be well approximated by a product state, thus requiring fewer coefficients to be fully specified.

Moreover, most of the Hilbert space cannot be quickly reached by time-evolution under a *local* Hamiltonian [37, sec. 3.4]. So, every state that can be "reasonably" prepared (either in an experiment or by nature) belongs to a tiny corner of the whole space of possible states.

But specifying a $|\psi\rangle$ by listing $d^N$ coefficients is highly inefficient: it would be better to have some representation that is "specialized" to the corner of the Hilbert space we are most interested in. As we will now see, MPS offer one such representation.

Consider an $n$-body system, with local dimension $d$ and open boundary conditions. A pure state $|\psi\rangle$ can be written as a Matrix Product State as follows [38]:

$$|\psi\rangle = \sum_{s_1,\ldots,s_n=1}^{d} \sum_{\alpha_1,\ldots,\alpha_n=1}^{\chi} \mathbf{M}^{[1],s_1}_{1\alpha_1} \mathbf{M}^{[2],s_2}_{\alpha_2\alpha_3} \cdots \mathbf{M}^{[n-1],s_{n-1}}_{\alpha_{n-2}\alpha_{n-1}} \mathbf{M}^{[n],s_n}_{\alpha_{n-1}1} |s_1 s_2 \ldots s_n\rangle . \quad (2.2)$$

The core idea is that each tensor $\mathbf{M}^{[i],s_i}_{\alpha_i\alpha_{i+1}}$ is a local description for the $[i]$-th site, which allows one to apply a *local* operator to a certain site without the need to change all the other coefficients.

For a fixed $s_i$, $\mathbf{M}^{[i],s_i}_{\alpha_i\alpha_{i+1}}$ is a $\chi \times \chi$ complex matrix, meaning that (2.2) is the sum of basis elements weighted by matrix products — which is why it is called a

Matrix Product State. The integer $\chi$ is called the MPS **bond dimension**, and a sufficiently high $\chi$ is needed if we want to express a truly *general* $|\psi\rangle$ in such form. However, the idea is that MPS with a *lower* $\chi$ can still encode all the meaningful states, albeit clearly not *all* possible states. In particular, to correctly describe any quantum state, assuming that all the sites has the same bond dimension, the local dimension needed is $\chi = d^{\lfloor \frac{n}{2} \rfloor}$.

More precisely, MPS are suitable to describe states with "low entanglement", as defined in Section 1.2.1. Usually, various many-body ground states have "low entanglement", and this statement can be made rigorous [39] for 1-dimensional systems that have a gap between the ground state and the first excited state.

Consider now the entanglement entropy as a function of the size $N$ of the left half of the bipartition. It is possible to prove that $S(N)$ obeys an area law for ground states of gapped 1-dimensional systems [40], according to which $S(N)$ is proportional to the size of the boundary that is left after the bipartition.

In 1-dimensional systems, the boundary consists of 2 points and hence the entanglement entropy does not scale with $N$[1]. Thus, such states are said to have "low entanglement".

Now, we show a way to construct states with a controllable amount of entanglement, which in turn naturally leads to the MPS representation in Equation (2.2) [38, sec. 2.2.5].

Consider a system of $n$ sites, each with local dimension $d$. We want to construct a state for this system such that it has a "limited" amount of entanglement. One way to do so is offered by the Valence Bond Picture [41, sec. 2.2] [42, sec. 1.2.1].

First, we associate to each site a pair of $\chi$-dimensional auxiliary sites, each representing one bond (Figure 2.1). In this way, we can independently set the entanglement contained in each bond. This is done by preparing any two auxiliary sites $i$ and $i+1$, which correspond to the same bond of two neighboring sites (e.g. auxiliary sites 2 and 3 in Figure 2.1), in a maximally entangled state:

$$|\omega_\chi\rangle = \frac{1}{\sqrt{\chi}} \sum_{k=1}^{\chi} |k\rangle_i \, |k\rangle_{i+1} \, . \tag{2.3}$$

In fact, the von Neumann entropy for $|\omega_\chi\rangle$ is maximal, and equal to $S = \ln \chi$. The number $\chi$ is called the **bond dimension**, and fixes the amount of entanglement present in each bond.

[1]Near the boundary of a system there may be a dependence on $N$ due to boundary effects.

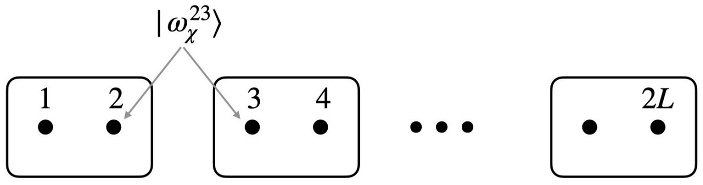


**Fig. 2.1:** Construction of maximally entangled bonds $|\omega_\chi\rangle$ between physical sites. Each site consists of two auxiliary sites (with states belonging to $\mathbb{C}^\chi \times \mathbb{C}^\chi$) so that a chain with $L$ sites consists of $2L$ auxiliary sites.

If we assume open boundary conditions, the first and last auxiliary sites do not participate in any bond, and they are respectively set to $|\alpha\rangle$ and $|\beta\rangle$ (boundary states).

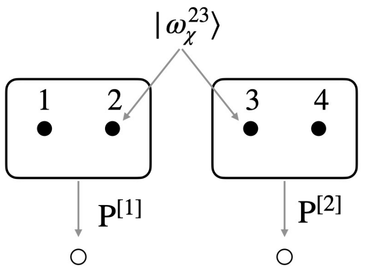


**Fig. 2.2:** We construct maps $\mathbf{P}^{[i]} : \mathbb{C}^\chi \times \mathbb{C}^\chi \to \mathbb{C}^d$, mapping states on the auxiliary sites to states on the physical sites. These maps do not increase the system's entanglement.

Now, we project the states of each pair of auxiliary sites to states of a single physical site (Figure 2.2). This is done by applying to each pair $[i]$ a local linear map $\mathbb{C}^\chi \times \mathbb{C}^\chi \to \mathbb{C}^d$ given by:

$$\mathbf{P}^{[i]} = \sum_{s=1}^{d} \sum_{\alpha,\beta=1}^{\chi} \mathbf{M}_{\alpha\beta}^{[i],s} |s\rangle^{[i]} \langle\alpha\beta|_{\text{aux}}^{[i]} . \tag{2.4}$$

In other words, each element $|\alpha\beta\rangle_{i,\text{aux}}$ of the basis of the $i$-th pair of auxiliary sites is mapped to a physical state given by:

$$|\alpha\beta\rangle_{\text{aux}}^{[i]} \mapsto \sum_{s=1}^{d} \mathbf{M}_{\alpha\beta}^{[i],s} |s\rangle^{[i]} .$$

Consider, for example, a chain of 2 physical sites, and thus 2 pairs of auxiliary sites and exactly one bond between them. The initial state of the extended system is:

$$\left|\tilde{\psi}\right\rangle = |\alpha\rangle\, |\omega_\chi\rangle\, |\beta\rangle = \frac{1}{\sqrt{\chi}} |\alpha\rangle_1^{[1]} \sum_{k=1}^{\chi} |k\rangle_2^{[1]} |k\rangle_3^{[2]} |\beta\rangle_4^{[2]} .$$

By applying Equation (2.4) to both sites (and ignoring the normalisation for brevity) we get:

$$\begin{aligned} |\psi\rangle &= (\mathbf{P}^{[1]} \otimes \mathbf{P}^{[2]}) \Big( \sum_{k=1}^{\chi} |\alpha k\rangle^{[1]} |k\beta\rangle^{[2]} \Big) = \\ &= \sum_{k=1}^{\chi} \sum_{s_1=1}^{d} \sum_{s_2=1}^{d} \mathbf{M}_{\alpha k}^{[1],s_1} \mathbf{M}_{k\beta}^{[2],s_2} |s_1 s_2\rangle , \end{aligned}$$

which is the MPS representation for a 2-body system, with bond dimension $\chi$. The same reasoning can be extended to $n$ sites (with $n-1$ bonds), which leads back to Equation (2.2).

Note that applying $\mathbf{P}^{[i]}$ to each physical site $[i]$ is a LOCC transformation [41, sec. 2.2], i.e. it can be executed by means of only Local Operations and Classical Communication. Thus, it cannot add more "quantum entanglement" between different sites. More precisely, the entanglement entropy of $|\psi\rangle$ is bounded by that of $\left|\tilde{\psi}\right\rangle$, which is fixed by $\chi$. Consider a bipartition splitting sites $i$ and $i+1$, and let $\rho$ be the left (or right) reduced density matrix of $|\psi\rangle$. Then:

$$\mathcal{S}_{\mathrm{VN}}(\rho) \leq \log \chi .$$

So, we can conclude that any MPS with bond dimension $\chi$ has a "low entanglement", i.e. a bipartition entanglement not greater than $\log \chi$ along any bipartition.

Therefore, since an MPS can encode *any*[2] state (see [sec. 2.3][41] for a proof), we can say that MPS are a good representation for any "low entanglement" state.

In fact, the MPS representation is particularly useful when $\chi$ is small. In particular, we can truncate the bond dimension $\chi$ to reduce the computational size of a quantum state, while still retaining most (if not all) of the information.

For example, a chain of $n$ qubits ($d = 2$) in a GHZ state can be exactly encoded by an MPS with bond dimension $\chi = 2$. Explicitly, it is realised by taking, in the

[2]Note, however, that the MPS decomposition for a particular state is not unique. So, different MPS may not correspond to different states.

above construction, $|\omega_2\rangle = |00\rangle + |11\rangle$ and the mapping $\mathbf{P} = |0\rangle\langle 00|_{\text{aux}} + |1\rangle\langle 11|_{\text{aux}}$ at each site (we ignore the normalisation for brevity). The MPS tensors $\mathbf{M}^{[i]}_{\text{ghz}}$ are all equal to:

$$\mathrm{M}^{[i],0}_{\text{ghz}} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \qquad \mathrm{M}^{[i],1}_{\text{ghz}} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}.$$

The whole tensor $\mathbf{M}$ can be written "all at once" as a vector-valued matrix:

$$\mathbf{M}^{[i]}_{\text{ghz}} = \begin{pmatrix} |0\rangle & \mathbf{0} \\ \mathbf{0} & |1\rangle \end{pmatrix}.$$

The main advantage of this representation comes from the fact that the number of coefficients in an MPS scales as $O(nd\chi^2)$, i.e. linearly with $n$ for a fixed $\chi$, while a full representation needs $O(d^n)$ coefficients, i.e. a number of coefficients that is exponential in $n$.

For instance, to represent an $n$-qubit GHZ state as a full vector we would need $2^n$ coefficients, but only $8n$ if the MPS representation is used.

Thus, MPS offer a way to accurately represent low entanglement states (i.e. all the "interesting" ones) which is extremely compressed if compared to the usual full representation in Equation (2.1).

## 2.2 Tensor networks

A notation such as the one used in Equation (2.2) can be particularly heavy. For this reason, we make use of graphical representations, first introduced by R. Penrose [43].

Consider one of the terms appearing in Equation (2.2), i.e. $\mathbf{M}^{[i],s_i}_{\alpha\beta}$. This is an object with three indices, that is an order-3 tensor. We can graphically represent it as a coloured rectangle with 3 "legs", each representing a different index (Figure 2.3).

Auxiliary between tensors are represented by joining with a line the two indices being contracted. Our convention for representing Matrix Product States is shown in Figure 2.4.

### 2.2.1 Gauge Freedom

A fundamental operation to do on the Matrix Product States is the application of operators. If we want to measure the expectation value of a local operator on

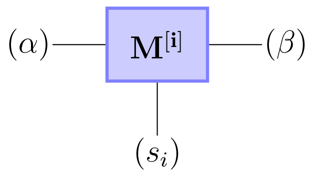


**Fig. 2.3:** MPS diagram for the $[i]$-th site MPS tensor $\mathbf{M}^{[i],s_i}_{\alpha\beta}$. The number of "legs" is the number of indices of the tensor, which is 3 for a single $\mathbf{M}$-tensor. The downward-pointing leg is conventionally chosen to be the one representing the "physical" index, while the other two represent the "virtual" indices that are contracted in the MPS representation.

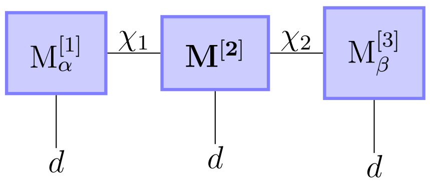


**Fig. 2.4:** MPS representation for a 3-body system, encoding the following tensor contraction: $\sum_{\gamma=1}^{\chi_1}\sum_{\delta=1}^{\chi_2}\mathbf{M}^{[1],s_1}_{\alpha\gamma}\mathbf{M}^{[2],s_2}_{\gamma\delta}\mathbf{M}^{[3],s_3}_{\delta\beta}$. Note that the boundary conditions $|\alpha\rangle$ and $|\beta\rangle$ fix the very first and last indices of the $\mathbf{M}$ tensor chain, reducing the first and last $\mathbf{M}$ tensor to just matrices $\mathrm{M}^{[1]}_{\alpha}$ and $\mathrm{M}^{[3]}_{\beta}$ (i.e. objects with 2 indices). However, we can treat them as 3-indexes tensors, by keeping the dimension of the indexes $\alpha, \beta$ restricted to 1. Moreover, in general the bond dimensions ($\chi_1$ and $\chi_2$) may be different for each bond, and they are denoted above the links between the contracted indices. All the physical sites have dimension $d$, which is shown below the downward links, representing the physical indices.

a system of $n$ sites, we would have to perform the full contraction of all the $n$ tensors (Figure 2.5).

However, the MPS representation is not unique. In fact, we may insert in a bond any two matrices X and $\mathrm{X}^{-1}$ whose product equates an identity (Figure 2.6). Then, each matrix is contracted with the nearest tensor, changing the numerical representation of the MPS, but not the overall contraction of the chain, i.e. the physical state it is representing. This is the so-called **gauge freedom** of tensor networks.

By choosing the right kind of transformations, we can pick the particular MPS representation which is most suited to our needs. For example, consider the computation in Figure 2.5. To simplify the contraction, we could choose $\mathrm{M}_1$ and $\mathrm{M}_2$ so that $\mathrm{M}_1\mathrm{M}_1^\dagger = \mathrm{id}_{\chi_1}$ and $\mathrm{M}_2\mathrm{M}_2^\dagger = \mathrm{id}_{\chi_2}$. In this way, to compute the

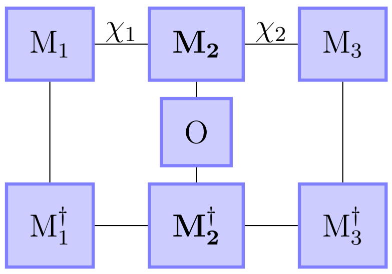


**Fig. 2.5:** Expected value of a single-site operator O on a 3-body MPS.

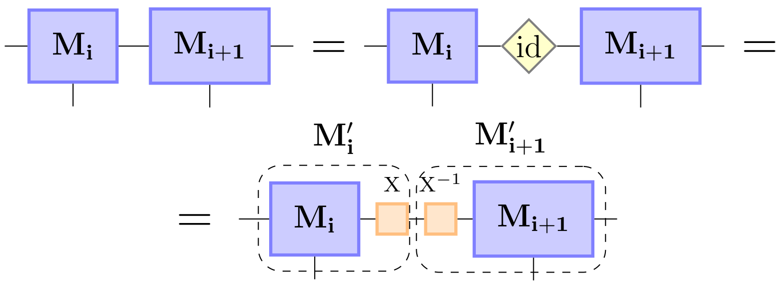


**Fig. 2.6:** Any bond can be rewritten as the contraction with an identity matrix, which can then be decomposed into the matrix product of some generic matrix X and its inverse $X^{-1}$. These can be in turn be contracted into the neighboring tensors. In the end, the tensor coefficients are changed, but the tensor network remains the same, in the sense that the result of any contraction with external tensors is as before.

expected value we would only need to consider the tensor $\mathbf{M_2}$ on which the gate O is acting.

In general, in a tensor network, if all branches connected to a tensor **A** form an isometry between their open indices and their indices connected to **A** (as it happens for $\mathbf{M_2}$ in the above example), then **A** is said to be a **center of orthogonality** [44, def. 3.3].

Setting a center of orthogonality is useful also if one wants to compress **A**, for instance by reducing its dimension, or by decomposing it into smaller tensors. In fact, suppose that $\mathbf{A'}$ is some (smaller) tensor used to locally approximate **A**. Let **H** be the tensor obtained by contracting the whole original network, and similarly let $\mathbf{H'}$ be the result of contracting the whole network with $\mathbf{A'}$ in place of **A**. In our picture, **H** and $\mathbf{H'}$ would be physical states in a full representation.

Then, if **A** is a centre of orthogonality, the local approximation error $\|\mathbf{A} - \mathbf{A'}\|$ is the same as the global approximation error on the whole network $\|\mathbf{H} - \mathbf{H'}\|$, where $\|\cdots\|$ denotes the Frobenius norm [44, Theorem 3.4]:

$$\mathbf{A} \text{ centre of orthogonality} \Rightarrow \|\mathbf{A}-\mathbf{A}'\| = \|\mathbf{H}-\mathbf{H}'\|, \tag{2.5}$$

$$\|\mathbf{T}\| = \sqrt{\mathrm{Tr}(\mathbf{T}^\dagger \mathbf{T})} = \sqrt{\sum_{\alpha_1,\alpha_2,\ldots,\alpha_k} |\mathbf{T}_{\alpha_1\alpha_2\ldots\alpha_k}|^2}. \tag{2.6}$$

It is then clear that setting the center of orthogonality guarantees trackable and minimal errors.

A simple way to fix a tensor **A** as the center of orthogonality is to iterate QR decompositions in each branch connected to **A** [44, 45, 46], as shown in Figure 2.7 for a 3-body MPS. The resulting network is said to be in the **unitary gauge**. We recall the the QR decomposition of a $(\chi \times \chi)$ matrix requires $O(\chi^3)$ operations.

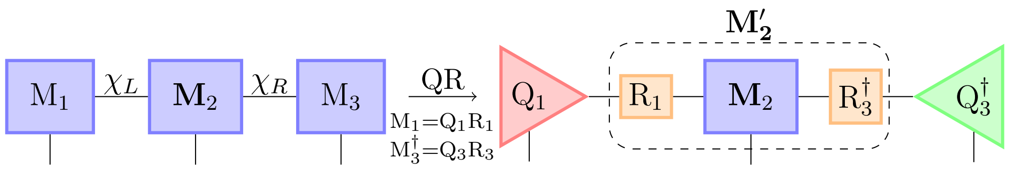


**Fig. 2.7:** Procedure for setting a site (e.g. the second) as the center of orthogonality in an MPS (in the figure, a 3-body MPS) by repeated QR decomposition. We conventionally draw left (right)-orthogonal tensors (e.g. $\mathrm{Q}_1$ and $\mathrm{Q}_3$) as red (green) triangles. They are oriented such that, if they are contracted with their hermitian conjugates along the indices they are "pointing" to, they form a projector. Instead, if they are contracted along all the other indices, they form an identity. For instance, $\mathrm{Q}_1$ is shown "pointing to" its second index. So $\sum_\beta (\mathrm{Q}_1)_{i\beta}(\mathrm{Q}_1^\dagger)_{\beta j}$ is a projector, while $\sum_\alpha (\mathrm{Q}_1^\dagger)_{i\alpha}(\mathrm{Q}_1)_{\alpha j} = \delta_{ij}$ is an identity. The opposite holds for $\mathrm{Q}_3$, since it is "pointing to" its first index.

If by following this procedure one sets the rightmost (leftmost) site as the center of orthogonality, the MPS is said to be in left-canonical (right-canonical) form. Usually, MPS are initialized in one of these two forms. There are however cases in which one could prefer random initialization, where the orthogonality has to be enforced, like in-ground state searches [47].

Note that setting a center of orthogonality through QR decompositions does not completely fix the gauge of the network. In fact, each bond can still be modified by adding a **unitary** matrix and its inverse $\mathrm{UU}^\dagger = \mathrm{id}$, without changing the physical state nor moving the center of orthogonality.

### 2.2.2 Passing from a full to a matrix product state

Consider a state of an $n$-body system with local dimension $d$, written in full representation as a set of $d^n$ indices $\mathbf{C}_{\alpha_1 \dots \alpha_n}$. This can be interpreted as an order-$n$ tensor, which can be rewritten as an MPS (i.e. a 1d tensor network) through repeated tensor decomposition.

To do so, we first gather all indices except the first into a unique index, effectively *reshaping* the order-$n$ tensor into a matrix:

$$\mathbf{C}_{\alpha_1 \dots \alpha_n} \quad (\alpha_i = 1, \dots, d) \xrightarrow{\text{Reshape}} C_{\alpha_1 \beta} \quad (\alpha_1 = 1, \dots, d;\ \beta = 1, \dots, d^{n-1}). \tag{2.7}$$

This matrix can be now decomposed using the Singular Value Decomposition (SVD) as shown in Equation (2.8). We recall that the SVD of a $(\chi \times \chi)$ matrix requires $O(\chi^3)$ operations. Using the SVD we can ensure the same isometry that we obtain with a QR, but on top of that, we can apply a useful truncation in the state, which we explore more in detail in Section 2.3.

$$\mathrm{C}_{\alpha_1 \beta} = \sum_{\gamma=1}^{r_1} \mathrm{U}_{\alpha_1 \gamma} \mathrm{S}_{\gamma\gamma} \mathrm{V}^{\dagger}_{\gamma\beta}, \qquad 1 \leq r_1 \leq \min(d, d^{n-1}) = d. \tag{2.8}$$

Note that the first physical index $\alpha_1$ appears only in the matrix $\mathrm{U}_{\alpha_1 \gamma}$, which we now rewrite as a tensor $\mathbf{M}^{[1],\alpha_1}_{1\gamma}$, following the MPS notation. Then we absorb S into V by writing $\mathrm{S}_{\gamma\gamma} \mathrm{V}^{\dagger}_{\gamma\beta} \equiv \mathrm{F}_{\gamma\beta}$:

$$C_{\alpha_1 \beta} = \sum_{\gamma_1=1}^{r_1} \mathbf{M}^{[1],\alpha_1}_{1\gamma_1} \mathrm{F}_{\gamma\beta}. \tag{2.9}$$

By splitting the index $\beta$ into the physical indices $\alpha_2 \dots \alpha_n$, we can reshape $\mathrm{F}_{\gamma\beta}$ to an order-$n$ tensor:

$$C_{\alpha_1 \beta} = \sum_{\gamma=1}^{r_1} \mathbf{M}^{[1],\alpha_1}_{1\gamma_1} \mathbf{F}_{\gamma_1 \alpha_2 \dots \alpha_n}. \tag{2.10}$$

These steps are shown in Figure 2.8 for an order-3 tensor.

Now we repeat the SVD to extract the second physical index $(\alpha_2)$ from the $\mathbf{F}$ tensor. This is done by regrouping the indices as $\delta = (\gamma, \alpha_2)$ (size $d^2$ if there was no truncation) and $\epsilon = (\alpha_3, \dots, \alpha_n)$ (size $d^{n-2}$), i.e. reshaping $\mathbf{F}$ into a matrix $\mathrm{F}_{\delta\epsilon}$, which can then be decomposed as follows:

$$\mathrm{F}_{\delta\epsilon} = \sum_{\gamma_2=1}^{r_2} \mathrm{U}_{\delta\gamma_2} \mathrm{S}_{\gamma_2\gamma_2} \mathrm{V}^{\dagger}_{\gamma_2\epsilon}, \qquad 1 \leq r_2 \leq \min(d^2, d^{n-2}).$$

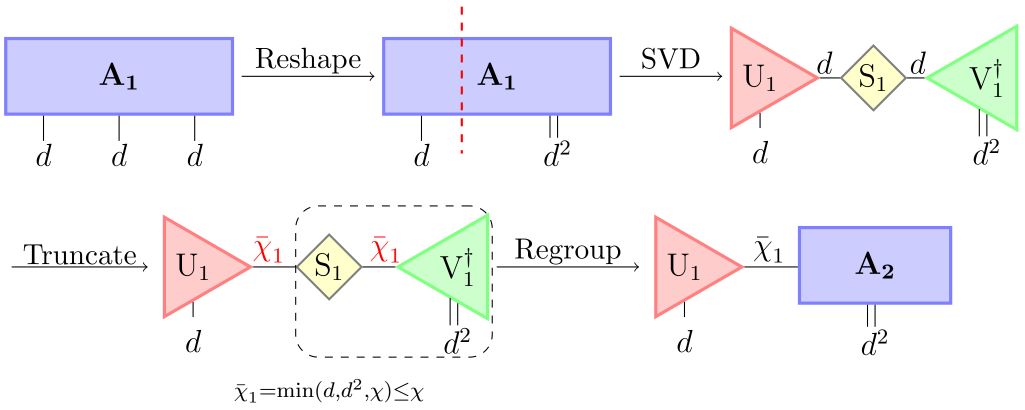


**Fig. 2.8:** First iteration of the process to map an order-3 tensor $\mathbf{A_1}$ on an MPS. First, the rightmost tensor, which is $\mathbf{A_1}$, is *reshaped* to a matrix $d \times d^2$. Then a compact *SVD* is performed, resulting in a bond dimension which is at most $\min(d, d^2) = d$. The bond dimension is now *truncated* to $\bar{\chi}_1 = \min(d, d^2, \chi)$, so that it is not greater than $\chi$. Then, calling $\lambda_1 > \lambda_2 > \dots \lambda_i > \dots > \lambda_\chi$ the eigenvalues of the singular matrix $\mathrm{S}_1$, we neglect all the eigenvalues $\lambda_i$ such that $\frac{\lambda_i}{\lambda_1} < \epsilon$, where $\epsilon$ is an arbitrary threshold called cut ratio. Finally, the singular values are *regrouped* into the rightmost tensor.

Now we split again the $\delta = (\gamma, \alpha_2)$ index, and rename the tensor $\mathbf{U}_{\gamma_1\alpha_2\gamma_2} \equiv \mathbf{M}^{[2],\alpha_2}_{\gamma_1\gamma_2}$, following the MPS notation:

$$\mathbf{C}_{\alpha_1\alpha_2\epsilon} = \sum_{\gamma_1=1}^{r_1}\sum_{\gamma_2=1}^{r_2} \mathbf{M}^{[1],\alpha_1}_{1\gamma_1}\mathbf{M}^{[2],\alpha_2}_{\gamma_1\gamma_2}[\mathrm{S}_{\gamma_2\gamma_2}\mathrm{V}^\dagger_{\gamma_2\epsilon}].$$

Note that, if the system contains $n \geq 4$ sites, the number of singular values $r_2$ in the second decomposition can be up to $d^2$, which is a factor $d$ higher than the maximum range of singular values $r_1$ at the previous decomposition. This means that, if no approximation is added, the bond dimension increases exponentially, up to $d^{\lfloor n/2 \rfloor}$.

Then, after grouping again S and $\mathrm{V}^\dagger$, the whole procedure can be applied once more. By repeating it until all physical indices are split into separate tensors we arrive at the MPS representation in Equation (2.2). Again, in Figure 2.9 we show the application of the second step of this algorithm.

## 2.3 Operations on the MPS

After defining the structure of Matrix Product States we focus on the possible operations applicable. We first introduce the operation needed to perform the

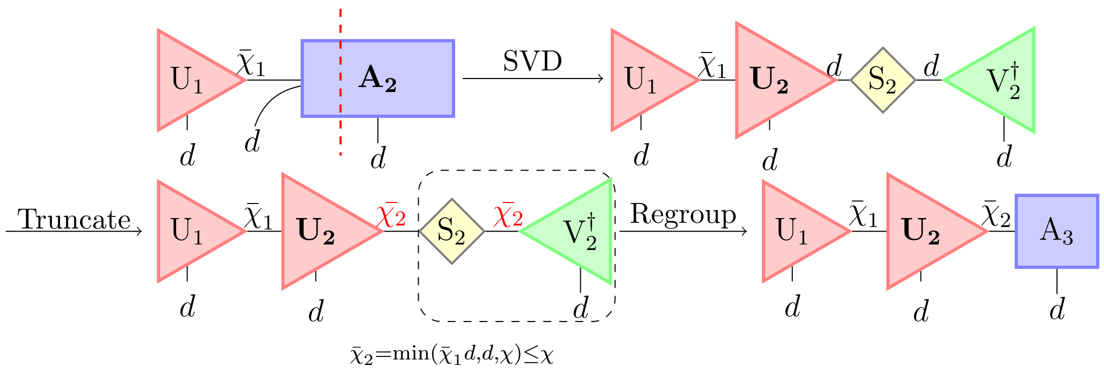


**Fig. 2.9:** Second iteration of the mapping of an order-3 tensor to an MPS. At the start, we have `last_svd_dim` $= \bar{\chi}_1$. The rightmost tensor, $\mathbf{A_2}$, is *reshaped* to a matrix $\bar{\chi}_1 d \times d$, and then a compact *SVD* is performed, leading to a bond dimension which is at most $\min(\bar{\chi}_1 d, d) = d$. This is now *truncated* to $\bar{\chi}_2 = \min(\bar{\chi}_1 d, d, \chi)$. We then neglect all the singular values $\lambda_i$ of S$_i$ such that $\frac{\lambda_i}{\lambda_1} < \epsilon$. The remaning singular values are then *regrouped* into the rightmost tensor, and the algorithm ends.

time evolution of the quantum state, namely the application of quantum gates, as seen in Section 1.3. In particular, we focus on the importance of a suitable approximation in the application of two-qubit gates. Then, we explain how to efficiently perform projective measurements, which are a fundamental step of all quantum algorithms, since they are the only possibility of accessing the information of the quantum state in the experimental implementation. Finally, we show how to measure the entanglement along any bipartition of the system, underlying the easiness in this representation, and a protocol to obtain the exact probability of any state in a qubit system, employing a binary tree.

### One-site operation

The operation with the most favorable computational scaling that we can apply on MPS is the one-site operation. Given a order-2 tensor $\mathbf{G}$ its application on the $i$-th site of the MPS is defined as:

$$\mathbf{M'}^{[\sigma_i]}_{\alpha_i,\alpha_i+1} = \sum_{s_i} \mathbf{M}^{[s_i]}_{\alpha_i,\alpha_i+1} \mathbf{G}_{s_i\sigma_i}. \tag{2.11}$$

It is a local operation, and thus modifies only the application site, as presented in Figure 2.10. In the quantum circuit framework, it is translated into a one-qubit gate. It is important to notice that single-qubit gates are unitary, and thus do not change the gauge of the MPS.

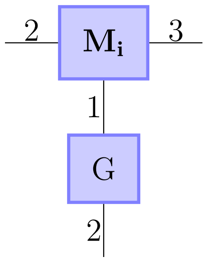


**Fig. 2.10:** Tensor contraction schemas for the application of a one-site operator **G**. Following our convention, the physical index is numbered as 1, and then the rest is numbered left to right. For the operato **G**, instead, we start from the upper left link, numbering it with 1, and then continue clockwise.

## Two-site operation

A two-site operation involves two adjacent sites in the MPS. We restrict ourselves to neighboring sites. Given an order-4 tensor **G**, applied to the $i$-th, $(i+1)$-th sites, we can decompose the computations in three different steps.

1. The $i$-th or $(i+1)$-th site is set as the center of orthogonality of the tensor network, using the QR decompositions. In the algorithm, the site is chosen to minimize the number of QR decompositions to perform. For simplicity, in the following, we assume the chosen site is the $i$-th.

2. We contract the sites $i$ and $i+1$ and the tensor **G**:

$$\mathbf{U} = \sum_{s_i, s_{i+1}} \sum_{\alpha_{i+1}} \mathbf{U}^{s_i}_{\alpha_i \alpha_{i+1}} \mathbf{U}^{s_{i+1}}_{\alpha_{i+1} \alpha_{i+2}} \mathbf{G}^{s_i s_{i+1}}_{\sigma_i \sigma_{i+1}}. \tag{2.12}$$

   This first step is presented in Figure 2.11. However, the algorithm does not present a three-tensor contraction option. For this reason, computationally, we first contract the two sites and then contract the resulting tensor with the operator **G**.

3. We apply an SVD to come back to the MPS structure. However, a trivial application of the SVD would make the bond dimension increase at each application. For this reason, we cut the singular value diagonal matrix S, such that the maximum number of non-zero eigenvalues is fixed at the maximum bond dimension $\chi_{max}$. Furthermore, we neglect all the eigenvalues $\lambda_1 > \lambda_2 > \dots \lambda_i > \dots \lambda_{\chi_{max}}$ of S such that $\frac{\lambda_i}{\lambda_1} < \epsilon$, where $\epsilon$ is called cut ratio. We call $\hat{\mathbf{S}}$ the singular value matrix after the application of the

approximations.

$$\mathbf{U} = \mathbf{U}_i\mathbf{S}\mathbf{V}_{i+1}^\dagger \simeq \hat{\mathbf{U}}_i\hat{\mathbf{S}}\hat{\mathbf{V}}_{i+1}^\dagger = \hat{\mathbf{U}}_i\mathbf{U}'_{\mathbf{i+1}}, \tag{2.13}$$

This second step is presented in Figure 2.12. We stress that, since $\mathbf{U}_i$ was set as center of orthogonality, we are minimizing the error in the truncation of $\mathbf{S}$.

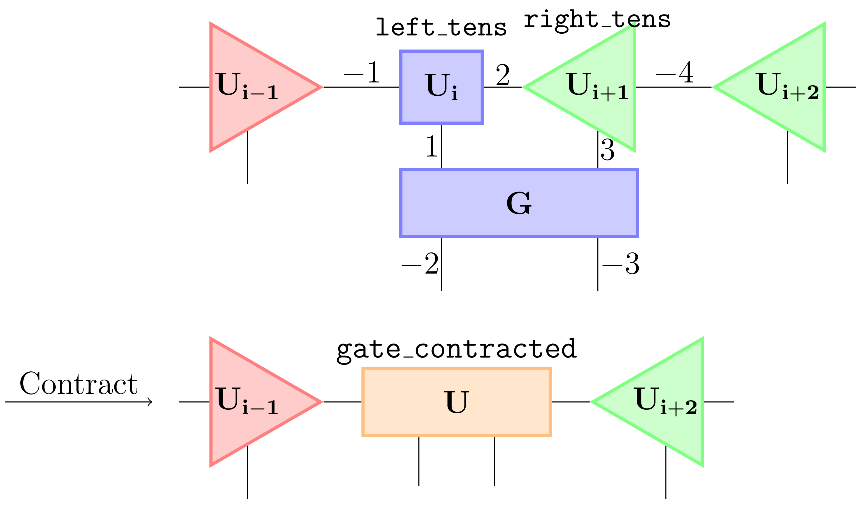


**Fig. 2.11:** When a 2-site operator is applied, the site at `pos` ($\mathbf{U_i}$) is set to the centre of orthogonality. Then the quantum gate $\mathbf{G}$ is contracted with the sites at `pos` and `pos` $+ 1$ ($\mathbf{U_i}$ and $\mathbf{U_{i+1}}$ in the figure).

## Projective measurement

It is very important to measure the matrix product state. In particular, we focus on projective measurements on the computational basis. It is important to measure each site one at a time, since each single-site measurement further projects the state in a smaller Hilbert space, resulting in a final product state. For this reason, we can not implement parallel techniques, like the use of the OpenMP library [48], to speed up this process. We now discuss the procedure, showing that the algorithm scales as $O(n\chi^2d^2)$ [49].

We suppose that we want to measure a state $|\psi_A\rangle$ represented as an MPS. First, we set as orthogonality center the first site, i.e. we work in a right-canonical condition. First, we compute the expectation values of a projector $P_1(m)$, which

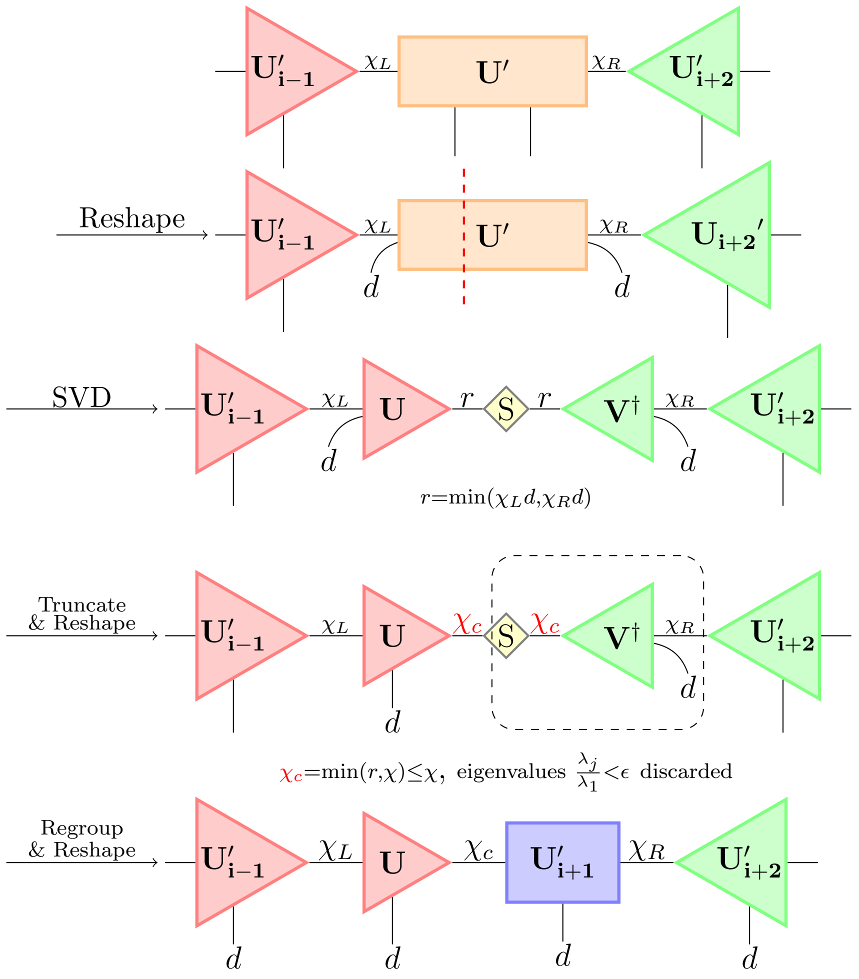


**Fig. 2.12:** Final steps of the application of a 2-site operator. First, $\mathbf{U}$ (i.e. the gate contracted with the MPS), which is now the centre of orthogonality, is reshaped to a matrix $\chi_L d \times \chi_R d$. Then a compact SVD is performed, leading to a bond dimension which is at most $r = \min(\chi_L d, \chi_R d)$. Columns (rows) of $\mathbf{U}$ ($\mathbf{V}$) are removed to truncate the bond dimension from $r$ to $\chi_c = \min(r, \chi) \leq \chi$. Then, the first $\chi_c$ elements of S $\{\lambda_i\}_{i=1,\ldots,\chi_c}$ with ratio $\frac{\lambda_j}{\lambda_1} > \epsilon$ are *regrouped* into $\mathbf{V}^\dagger$ to complete the splitting procedure. The result is a new MPS state, incorporating the action of the quantum operator.

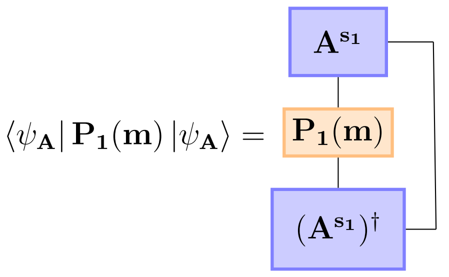
project the site 1 on one of its possible states $m \in \{m_i\}_{i=1,\ldots,M}$. As discussed in Section 2.2.1 setting the center of orthogonality let us compute expectation values by only considering the site where the operator is applied. Explicitly:

$$\langle\psi_A| P_1(m) |\psi_A\rangle = \sum_{\alpha_1, s_1', s_1} (\mathbf{A}^{s_1'})^\dagger_{\alpha_1} \mathbf{P}_1^{s_1 s_1'}(m) \mathbf{A}^{s_1}_{\alpha_1}. \tag{2.14}$$

We then collapse the state of site 1 in $|s_1\rangle = |m_i\rangle$ with probability $p_{m_i} = \langle\psi_A| P_1(m_i) |\psi_A\rangle$. The action of a generic single-site operator would be followed by an SVD to orthogonalize site 1 and turn site 2 into the center. However, the fact that we are using projectors allows for a more efficient algorithms. Indeed, the repeated action of single-site projectors on any state produces a product state, which can be represented as an MPS with $\chi = 1$. If the above step of applying the projector $P_1(m_i)$ to site 1 and left orthogonalizing $\mathbf{A}^{s_1}$ was performed, the new bond index connecting $\mathbf{A}^{s_1}$ to $\mathbf{A}^{s_2}$ could be truncated such that it only takes on one value. For this reason, one can therefore directly replace $\mathbf{A}^{s_1}$ with the $1 \times 1$ matrix $\mathbf{A}^{s_1} = \langle s_1|m_i\rangle$. Finally, to ensure that we are still describing the same state one should propagate the measure on the second site:

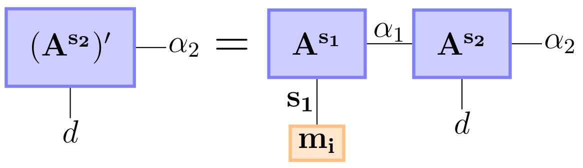
$$\mathbf{A}^{s_2}_{\alpha_1\alpha_2} \to \mathbf{A}^{s_2}_{m_i\alpha_2} = p_{m_i}^{-1/2} \sum_{s_1\alpha_1} \langle m_i|s_1\rangle\, \mathbf{A}^{s_1}_{\alpha_1} \mathbf{A}^{s_2}_{\alpha_1\alpha_2}. \tag{2.15}$$

The measurement procedure can be further propagated until the end of the MPS.



$\langle\psi_\mathbf{A}| \mathbf{P_1(m)} |\psi_\mathbf{A}\rangle =$
$\mathbf{A^{s_1}}$
$\mathbf{P_1(m)}$
$(\mathbf{A^{s_1}})^\dagger$

**(a)** Tensor diagram of the expectation value of the projector $\mathbf{P_1(m)}$. Since the MPS is in the left-canonical gauge all the right tensors contract to the identity, producing the easy diagram presented above.



**(b)** Propagation of the measurement from the first tensor to the second one. The procedure can be then reproduced to propagate the measurement from the $i$-th tensor to the $(i+1)$-th tensor. We contract the measured state-vector $\mathbf{m_1}$ with the physical index of $\mathbf{A^{s_1}}$, and then contract $\mathbf{A^{s_1}}$ and $\mathbf{A^{s_2}}$. Notice that, even though in Equation (2.15) there is also the index $m_1$ it is not reported in the diagram since it can only assume one value.

**Fig. 2.13:** Projective Measurement steps for an MPS. We put the state in a left-canonical form, and then perform iteratively the two steps above. It is important to notice that they can be reproduced for all sites, since in step (b) we obtain a tensor with 2 indexes, as $\mathbf{A^{s_1}}$

## Entanglement measure

Since the MPS are better suited to describe low-entanglement states it is important to have a way of measure it. Indeed, the structure of an MPS makes this measurement very easy. Defining the entanglement of two partitions $[1,\dots,i]$, $[i+1,\dots,n]$ as the von Neumann entanglement entropy the procedure is the following:

1. Set the site $i$ as center of orthogonality;
2. We recall that the legs of the tensor are numbered, starting from left to right, as $2,1,3$. We reshape the tensor $\mathbf{A^{s_i}}$ into a matrix, by merging the the indexes $2,1$. Then, we perform an SVD decomposition of the matrix:
$$\mathbf{A^{s_i}} = (\mathbf{A^{s_i}})'\mathbf{S}\mathbf{V_i^\dagger}. \tag{2.16}$$
3. Compute the von Neumann entropy using the eigenvalues $\{\lambda_j\}_{j=1,\dots,\chi}$ of $\mathbf{S}$:
$$S_V = -\sum_j^\chi \lambda_j^2 \log\lambda_j^2. \tag{2.17}$$
4. Contract the tensors $\mathbf{S}\mathbf{V_i^\dagger}$ to the tensor $\mathbf{A^{s_{i+1}}}$

Again, we can notice how the entanglement of the state can be controlled through the maximum bond dimension $\chi$.

### Probability measure

We have seen how to perform projective measurements. However, using MPS let us analyze more precisely the probabilities related to the quantum state.

We define with $\epsilon_p \in [0,1]$ the threshold probability, i.e. the minimum probability that the algorithm is able to observe. Then, we simply analyze the state in a binary tree, as shown in Figure 2.14. The algorithm on a quantum state $|\psi\rangle$ is composed as follows:

1. Compute the expectation value of the Pauli matrix $\sigma_z$ on the $i$-th qubit. Then, the probability of measuring respectively 0 and 1 are:
$$p(q_i^0|q_0^{s_0}q_1^{s_1}\dots q_{i-1}^{s_{i-1}}) = \frac{\langle\psi|\,\sigma_z\,|\psi\rangle+1}{2}, \tag{2.18}$$
$$p(q_i^1|q_0^{s_0}q_1^{s_1}\dots q_{i-1}^{s_{i-1}}) = 1 - p(q_i^0|q_0^{s_0}q_1^{s_1}\dots q_{i-1}^{s_{i-1}}), \tag{2.19}$$
where $s_j$ are the configuration of the qubit state relative to the specific branch of the binary tree we are investigating;

2. Check if the probability of the state up to the $i$-th qubit is less than the probability threshold $\epsilon_p$. In that case, discard that branch of the binary tree. If

$$\prod_{j=0}^{j=i} p(q_j^0|q_0^{s_0}q_1^{s_1}\dots q_{i-1}^{s_{i-1}}) < \epsilon_p, \tag{2.20}$$

then discard the branch.

3. Start from point 1. with the qubit $i+1$.

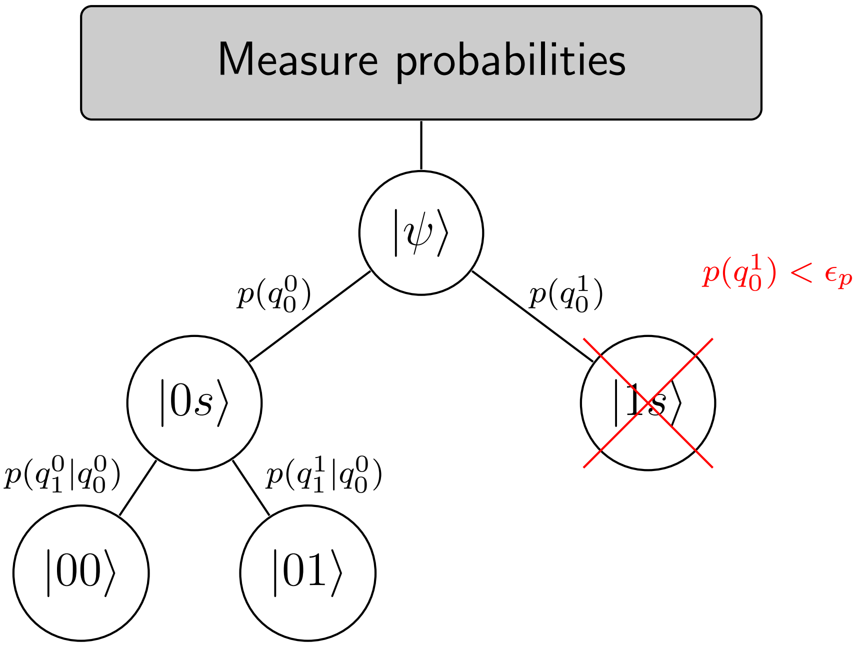


**Fig. 2.14:** Binary tree which represents the computation of the probability of the measurements of a quantum state $|\psi\rangle$. We denote with $q_i^s$ the $i$-th qubit with value $s \in \{0,1\}$. If the probability of a branch is less than $\epsilon_p$ then that branch is cut and no longer analyzed. At the end of the procedure, we obtain the probability of measuring all states $|\phi\rangle$ with $p(|\phi\rangle) > \epsilon_p$.

### 2.3.1 Time evolving block decimation

Even though we use only the operations described above in this thesis, it is important to stress the original techniques used with MPS. The Time Evolving Block

Decimation [50] algorithm is a technique to time-evolve a quantum state represented with an MPS under the effect of a local Hamiltonian. The development of such a technique was already motivated in Section 2.1, stressing the importance of the locality of the interactions.

We define the Hamiltonian $\mathrm{H}_n$ in the following way:

$$\mathrm{H}_n = \sum_{i=1}^{n} K_1^i + \sum_{i=1}^{n} K_2^{[i,i+1]}, \tag{2.21}$$

where we have highlighted the division between the one-body and the two-body (the interaction) term. To exploit the TEBD it is useful to decompose the Hamiltonian as a sum of two possibly non-commuting terms, where the first is acting on the *even* sites, while the second on the *odd* sites:

$$\mathrm{H}_n = \sum_{\text{even } i} K_1^i + K_2^{[i,i+1]} + \sum_{\text{odd } i} K_1^i + K_2^{[i,i+1]} \tag{2.22}$$

$$= \sum_{\text{even } i} \mathrm{E}_i + \sum_{\text{odd } i} \mathrm{O}_i = \mathrm{E} + \mathrm{O}. \tag{2.23}$$

Indeed, there is a freedom in the arrangement of the one-body terms, and the structure presented in Equation (2.22) is only a possibility. At this point, recalling that the evolution operator under an Hamiltonian is $U(t) = e^{-i\mathrm{H}_n t}$ we apply a *Suzuki-Trotter Decomposition* [51]:

$$U(t) = e^{-i\mathrm{H}_n t} = \left[e^{-i\mathrm{H}_n \delta}\right]^{t/\delta} = \left[e^{\frac{\delta}{2}\mathrm{E}} e^{\delta \mathrm{O}} e^{\frac{\delta}{2}\mathrm{E}}\right]^N + O(\delta^2), \tag{2.24}$$

where $N = \frac{t}{\delta}$ is called the Trotter number.

At this point, the simulation becomes trivial. Indeed, all the $\mathrm{E}_i$ or $\mathrm{O}_i$ commute and can so be applied in parallel, as two-qubits gates. We so apply in sweeps all the gates on even sites, then on odd sites, and then again on the even ones, as shown in Figure 2.15.

For time-independent Hamiltonians there are two possible error sources in the TEBD algorithm:

1. the Suzuki-Trotter expansion. In the case of a $p$-th order approximation, the error is of order $\delta^{p+1}$. Taking into account that, to evolve the system in $N$ steps up to time $T$ the error becomes:

$$\epsilon_{ST} = N\delta^{p+1} = \frac{T}{\delta}\delta^{p+1} = T\delta^p. \tag{2.25}$$

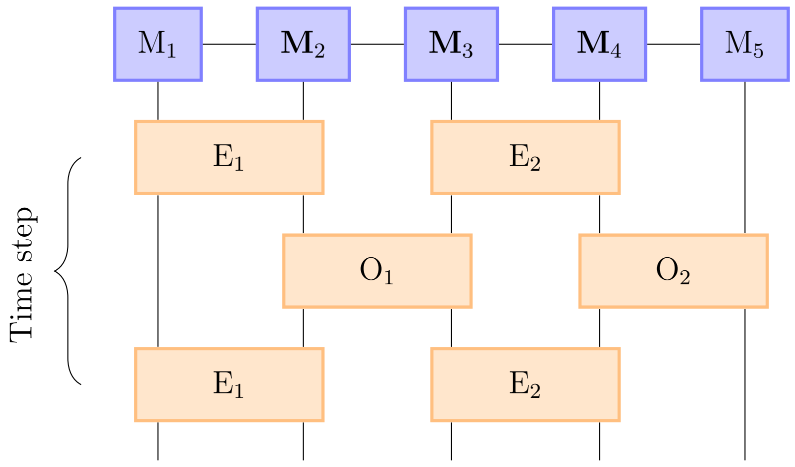


**Fig. 2.15:** Time Evolving Block Decimation algorithm. A local Hamiltonian H$_n$ is decomposed through a Suzuki-Trotter decomposition into a series of unitary operators acting respectively on even and odd sites. These operators are 2-sites unitary gates and can be applied to the MPS using the procedures explained in Section 2.3. The procedure represented above is then repeated $N$ times to obtain the evolution from time 0 to time $t$. We show the second-order decomposition as an example.

We can write the unapproximated state as $\left|\tilde{\psi}\right\rangle = \sqrt{1-\epsilon^2}\left|\psi\right\rangle + \epsilon\left|\psi^{\perp}\right\rangle$, where $\left|\psi^{\perp}\right\rangle$ is the part neglected by the approximation. Then, if we define the final error as:

$$\epsilon(T) = 1 - \mathcal{F}(\tilde{\psi}, \psi) = 1 - 1 + \epsilon^2 = \epsilon^2. \tag{2.26}$$

It is important to notice that the Suzuki-Trotter error is *independent* of the dimension of the system.

2. the Hilbert space truncation. First, the smallest contributions of the Schmidt spectrum are left away. We can so write the error as:

$$\epsilon(\chi) = 1 - \prod_{i=1}^{n-1}(1 - \sum_{\alpha=\chi}^{d^n}(\lambda_\alpha^{[i]})^2, \tag{2.27}$$

where with the superscript $i$ we denote the $i$-th site. The second error source is coming from the normalization of the state, since when we construct the reduced density matrix of the state, its trace is multiplied by the factor:

$$|\left\langle\psi_\chi|\psi_\chi\right\rangle|^2 = 1 - \frac{\epsilon_i}{1-\epsilon_i} = \frac{1-2\epsilon_i}{1-\epsilon_i}, \tag{2.28}$$

where $\epsilon_i = \sum_{\alpha=\chi}^{d^n} |\lambda_\alpha^{[i]}|^2$.

Both errors can be controlled: the first increasing the order of the Suzuki-Trotter decomposition, the second increasing the bond dimension $\chi$.

# 3 QC-MPS: MPS simulator for Quantum Computers

The aim of this project is the development of an efficient Matrix Product State simulator for quantum computers: QC-MPS. In Section 3.1, we overview the design choices done for the implementation of the program, highlighting the importance of a high-level interface. In Section 3.2 we instead analyze the performances of the simulator, in particular focusing on the Quantum Fourier Transform algorithm, introduced in Section 1.3.2. This step is beneficial because the QFT is one of the building blocks of many important quantum algorithms, such as the Shor algorithm [25]. Then, we study the Gaussian Boson Sampling protocol [19], concentrating on minimizing the computational resources. Finally, in Section 3.3 we explore possible ways of improving the simulator, by manipulating the quantum circuit and by parallelizing the algorithm.

## 3.1 Implementation

The programming language chosen for the simulator is Fortran, due to its efficiency in numerical computations. The simulator is compiled using the GNU compiler `gfortran`, even though it is also compatible with the Intel compiler `ifort`. Even though the Intel compiler is faster for many tasks, the choice of `gfortran` been done to run the simulator on Marconi 100 supercomputer, which does not support `ifort`.

The simulator presents all the methods explained in Chapter 2, that we list here for completeness:

- Application of single-site operations;
- Application of two-site operations;
- Perform projective measurements;
- Perform entanglement measurements;
- Perform probability measurements;
- Compute inner product between different MPS;
- Computation of the expectation value of single-site operators;
- Computation of the expectation value of separable operators.

As previously stated, the emulator is written in Fortran for efficiency. However, a low-level programming language is in particular difficult to approach for tackling many different problems. It would be demanding to prepare every quantum circuit in Fortran language, and then compile the program. For this reason, we developed a python interface, which combines the full potential of the Fortran code with the easiness of use of an interpreted language. Furthermore, the interface is compatible with packages widely used by the community, as we see in the following sections.

### 3.1.1 Qiskit interface

Qiskit [52] is an open source python library, developed by IBM, for programming quantum computers. The python interface of QC-MPS lets the user build his circuit using qiskit, and then run it on the MPS Fortran simulator. The interface already takes into account the constraints of using MPS, i.e. the **linear topology** and the possibility of using only one and two-qubits gates. These procedures are accomplished trough the qiskit function `transpile`, which stochastically searches for the equivalent circuit minimizing the number of gates.

We briefly explain here the main commands needed in qiskit to work with a quantum circuit, following the example in Listing 3.1. First, in line 3, we define a quantum circuit with 4 qubits. Then, we can apply an available quantum gate with name `g_name` as `QuantumCircuit.g_name`. This syntax means that the gates are *methods* of the quantum circuit *class*, which takes as input the gate parameters and the target qubits. Finally, we apply a linear mapping in lines 10 and 11, which enforces a *linear topology* in the circuit.

```python
from qiskit import QuantumCircuit, transpile, transpiler
# The quantum circuit definition
qc = QuantumCircuit(4)
# We apply the Hadamard gates to the first qubit
qc.h(0)
# And the CNOT gates to the others
for i in range(1, n_qub):
  qc.cx(0, i)
# Apply linear mapping
linear_map = transpiler.CouplingMap.from_line(qc.num_qubits)
linear_qc = transpile(qc, coupling_map=linear_map)
```

**Listing 3.1:** Example of a qiskit quantum circuit in python. We initialize a GHZ circuit, linearizing its topology.

The available gates in qiskit, which are also the ones available in QC-MPS, are reported in Table 3.1. We point towards the qiskit-terra GitHub page [53] for a formal definition of each gate.

| **Gates available in the QC simulator** | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Constant one-qubit** | x | y | z | h | id | s | sdg | sx | sxdg | t | tdg |
| **Parametric one-qubit** | p | r | rx | ry | rz | u | u1 | u2 | | | |
| **Constant two-qubit** | swap | dcx | ecr | iswap | ch | cx | cy | cz | | | |
| **Parametric two-qubit** | rxx | ryy | rzx | rzz | cp | cr | cry | crz | cu | cu1 | |

**Table 3.1:** Name of the gates available in the MPS simulator, following the definition of qiskit. A more detailed description of these gates, and in particular their matrix form, can be found in the qiskit-terra GitHub page [53]

We show now the main function of the interface, `run_qiskit`. It presents the following inputs parameters:

- `qc`: the qiskit quantum circuit;
- `chi`: the maximum bond dimension of the simulation. We recall that this is the maximum number of eigenvalues of the singular matrix kept after an SVD;

- `cut_ratio`: the cut ratio of the simulation. We recall that, if $\frac{\lambda_i}{\lambda_1} \leq$`cut_ratio`, we discard the eigenvalue $\lambda_i$, where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ are the eigenvalues of the singular matrix after an SVD;
- `do_statevector`: a boolean flag. If it is evaluated `True`, then the function returns the final MPS state-vector. Since the state-vector is described by $2^n$ coefficients, with $n$ the number of qubits, it is not advised to enable the flag for $n > 30$.
- `nshots`: the number of measurements of the system to be performed at the end of the simulation;
- `linearize`: a boolean flag. If set to `True`, the function applies the linearization procedure to the system. It is possible to set the flag to `False`, since the user may want to linearize the circuit only once, and then pass that circuit multiple times to the function;
- `basis_gates`: a list of strings, containing the names of the available gates, chosen from those in Table 3.1. If an empty list is passed, then the function assumes that all the gates used in the circuit are present in the simulator;
- `optimization`: an integer number that controls the level of optimization of qiskit transpiler. The higher the number, the more optimized is the circuit. However, it also increases the transpilation time. For further information refer to the qiskit documentation;
- `save_unformatted_mps`: a boolean flag. If set to true, the function saves the MPS final state in an unformatted format, for further uses;
- `input_nml`: the name of the namelist file, which contains all the parameters of the simulation, and is read by the Fortran backend.

As an output, the function returns a dictionary, which is a python structure that connects data to labels. The results are:

- `meas`, it is a dictionary containing the number of occurrences of the states after the measurements, in the form {`State: n_occurrences`}. For example, if the state $|000\rangle$ has been measured 15 times and the state $|111\rangle$ 12 times, then the dictionary will be {`'000':15, '111':12`}. Only the measured states are set as keys in the dictionary, thus avoiding the exponential explosion of the Hilbert space;

- `statevector`, if the `do_statevect` is `True` it contains the statevector in the form of a numpy array, otherwise it is `None`;
- `time`, the CPU time for the simulation, computed in Fortran;
- `cutted_sing_vals`, which is an array where every entry is the sum of the discarded singular values at a given SVD;
- `entanglement`, which is the array of the bond entanglement along the MPS chain at the end of the simulation.

Even though there are other functions in the interface, we skip them all for brevity. The user can further adapt the simulations and measurements with these functions.

### 3.1.2 Strawberry fields interface

Strawberry fields [54] is an open source python library, developed by XANADU, for programming photonic devices. We report an example of code to generate a linear optic circuit, shown in Listing 3.2. First, in line 4 we initialize a circuit with 4 qumodes. Then, the application of the gates is obtained using the `with` construct on line 6. To apply a gate, we first call the gate with its parameters, add the vertical slash `|` and then write the target qumodes. This procedure is exemplified from lines 7 to 25 for both single and two-qumodes gates. It is important to notice that strawberry fields, inside the `with` context overwrites the logical `or` operator, usually identified with the vertical slash `|`.

```
1  import strawberryfields as sf
2  from strawberryfields.ops import Sgate, Rgate, BSgate
3  # initialize a 4 mode program
4  gbs = sf.Program(4)
5  
6  with gbs.context as q:
7      # squeezing gates
8      Sgate(1) | q[0]
9      Sgate(1) | q[1]
10     Sgate(1) | q[2]
11     Sgate(1) | q[3]
12 
13     # rotation gates
14     Rgate(0.5719)  | q[0]
15     Rgate(-1.9782) | q[1]
```

```
    Rgate(2.0603)   | q[2]
    Rgate(0.0644)   | q[3]

    # beamsplitter array
    BSgate(0.7804, 0.8578)  | (q[0], q[1])
    BSgate(0.06406, 0.5165) | (q[2], q[3])
    BSgate(0.473, 0.1176)   | (q[1], q[2])
    BSgate(0.563, 0.1517)   | (q[0], q[1])
    BSgate(0.1323, 0.9946)  | (q[2], q[3])
    BSgate(0.311, 0.3231)   | (q[1], q[2])
```

**Listing 3.2:** An example of a linear optic circuit initialization using the strawberry fields python library. In particular, this example shows a Gaussian Boson sampling circuit for 4 qumodes.

We do not repeat the considerations on the python interface, since it is analogous to the qiskit's one. The only differences are that the name of the function is `run_sf`, and that there is the additional parameter of the `fock_cutoff`, introduced in Section 1.5.

## 3.2 Results

In this section, we analyze the simulator and study random circuits and important cases, such as the QFT presented in Section 1.3.2. First, in Section 3.2.1, we check if the simulator presents correct results, focusing on small-scale simulations that can be reproduced exactly. Then, in Section 3.2.2, we review its time scaling, confronting it to the exact simulation performed on the CINECA cluster. Next, we perform a more in-depth analysis, focusing on the necessary bond dimension to describe particular systems. Finally, in Section 3.2.3, we study the GBS protocol, focusing on the necessary Fock space cutoff for a correct description of the system, the correctness of the simulation and the dependence on the bond dimension.

For the whole section we use a singular value cut ratio of $\epsilon = 10^{-9}$.

### 3.2.1 Correctness checks

When we develop a simulator, we must be sure that its results are correct. For this reason, we analyze *how close* the MPS-simulated state is to the correct one. We use a measure of closeness, the *fidelity* $\mathcal{F}$, between two quantum states $|\phi\rangle$, $|\psi\rangle$:

$$\mathcal{F}(|\phi\rangle, |\psi\rangle) = \left| \langle\phi|\psi\rangle \right|^2. \tag{3.1}$$

This measure is defined in $\mathcal{F} \in [0,1]$, and the two quantum states are the same if their fidelity is 1. To better appreciate the scaling around 1, in this analysis we employ the *Bures distance* instead of the fidelity. It is defined as:

$$B(|\phi\rangle, |\psi\rangle) = 2(1 - \sqrt{\mathcal{F}(|\phi\rangle, |\psi\rangle)}). \tag{3.2}$$

In this way, we are able to look at the error in representing the state, as we see in the following.

Since we must be able to exactly simulate a quantum state to perform these checks, we are limited in the number of qubits $n$. Furthermore, we recall that the definition of circuit depth is present in Section 1.3.1. We start by investigating these two cases:

- The QVOLUME quantum circuit [55], developed by IBM, to quantify the capability of a quantum computer. This circuit is composed of $d$ layers, where each layer is formed by a permutation of the qubits labels and the application of random 2-qubit gates. In the following, we choose to set the depth equal to the number of qubits, i.e. $d = n$. The important feature of this circuit is that generates an highly-entangled state. Thus, it should be a difficult circuit to simulate for QC-MPS, since MPS are used to represent quantum state with a *limited* amount of entanglement, controlled by the bond dimension $\chi$;

- The W quantum circuit, which for $n$ qubits results in the state:

$$|W\rangle = \frac{1}{\sqrt{n}} \left( |10\ldots0\rangle + |010\ldots0\rangle + \cdots + |0\ldots010\ldots0\rangle + \cdots + |0\ldots01\rangle \right). \tag{3.3}$$

  It is a uniform superposition of states that present 1 on a qubit and 0 on all the others. It is a well-known state entangled state, which is easy to prepare. Moreover, it is possible to write it analytically in MPS representation [56].

In Figure 3.1, we observe how the Bures metric changes in these two cases. Since the QFT is the main ingredient of many quantum algorithms, we monitor the evolution of the metric also after its application. We can observe that, in the QVOLUME case, the application of the QFT does not increase the bond dimension for a correct simulation, since the starting state is already highly entangled, requiring the maximum bond dimension for the given number of qubits, i.e. $\chi = 2^{\lfloor \frac{n}{2} \rfloor} = 2^7 = 128$. In contrast, we can observe that the QFT does increase the difficulty of representing the state in the W case.

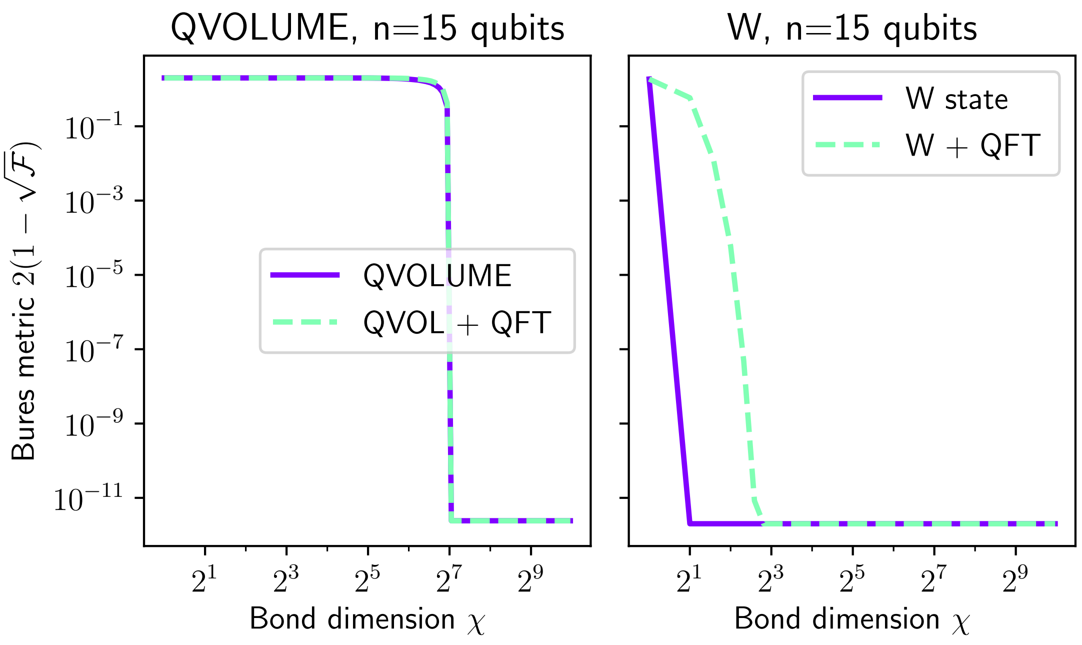


**Fig. 3.1:** Correctness checks for $n = 15$ qubits, showing the Bures distance as a function of the bond dimension $\chi$ in a log-log scale. Moreover, we compare the Bures metric after the application of the QFT. (left) The QVOLUME circuit drops in the Bures distance once the bond dimension of the MPS can capture all the entanglement. (right) The circuit to create a W state does not encounter high entanglement and a low bond dimension is sufficient.

However, these two very specific cases are at the boundaries of an "easy" state and a "difficult" one for MPS. For this reason, we decide to proceed in the analysis using random circuits with a fixed depth, since they can be interpreted as a more unbiased case of study. With a random circuit with a fixed depth, we denote a quantum circuit where we apply random 1-qubit and 2-qubit gates, until the depth of the circuit is the chosen one. In Figure 3.2 we show the necessary bond dimension to correctly describe a random quantum circuit, as a function of the depth $d$. We consider a quantum circuit correctly described if the Bures metric $B$ is smaller than $10^{-9}$, where the metric is averaged over 50 random circuits. We can observe that the simulator is fully capable of describing the random circuits, and that the necessary bond dimension is strongly dependent on the depth of the circuit $d$. This dependency is the expected behavior since with a high depth we apply more gates, and thus increase the entanglement. Indeed, we notice that the application of the QFT can increase a lot the necessary bond dimension. For example, the circuit with $d = 8$ was correctly described with a bond dimension $\chi = 2^5 = 32$, but after the QFT the necessary bond dimension increased to $2^7 = 128$, the maximum bond dimension for the given system. We are able to reach the maximum bond dimension even without the application of the QFT if the depth is sufficiently high, i.e. for $d > 11$.

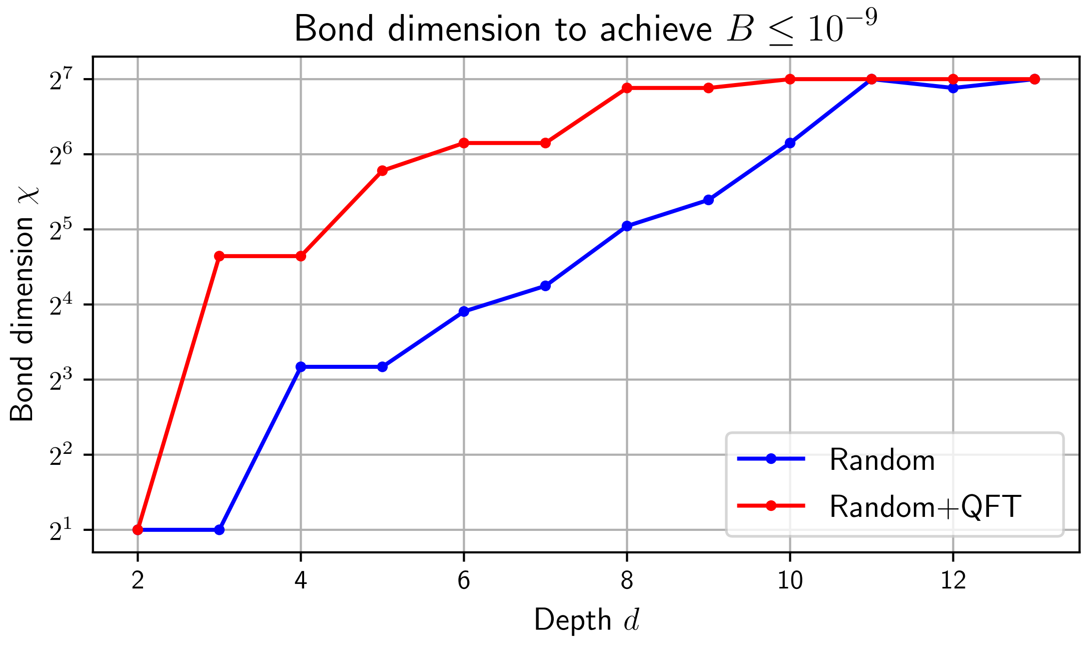


**Fig. 3.2:** Bond dimension $\chi$ necessary to exactly describe a random quantum circuit as a function of the circuit depth $d$. We consider the MPS simulation of a quantum circuit to be exact when the Bures metric $B$ is smaller than $10^{-9}$. Each data point is obtained from a Bures metric averaged over 50 circuits.

We are now confident that the simulator is able to reproduce truthfully the evolution of a quantum circuit, and we can further proceed in the analysis. In particular, in the next section, we analyze the time scaling of the simulation.

### 3.2.2 Time scaling

As we have already observed in Section 1.4, simulating a quantum circuit is a computationally very intensive task even on a supercomputer. To appreciate the limitations of this approach we use the qiskit state-vector simulator to simulate the preparation of a $W$ state, followed by a QFT. We observe in Figure 3.3 that the implementation time is exponential in the number of qubits, as expected, and quickly becomes out of reach even parallelizing the processes. We notice two different exponential scaling, where the change is for $n \sim 20$. Our interpretation of the change of scaling is that it depends probably on CPU constraints, but we did not investigate any further. To better appreciate the difference between the different number of threads we present on the right the ratio between the computational time with $m \in \{8, 16, 32, 64, 128\}$ and 128 threads. We notice a speedup only with respect to 8 and 16 threads, suggesting that the optimal number of threads for this task is 32.

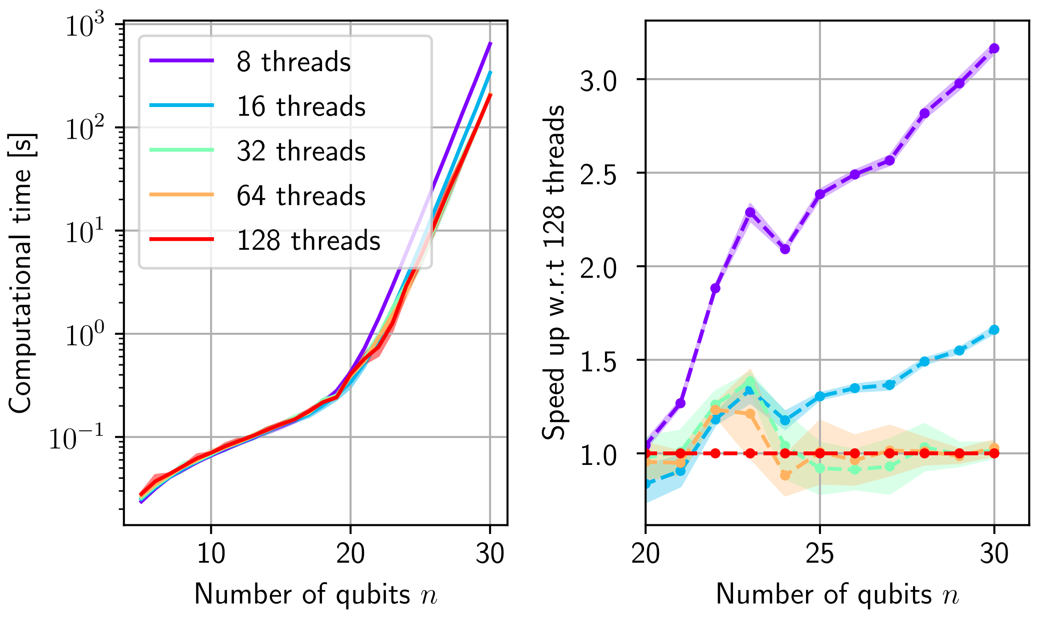


**Fig. 3.3:** Time scaling of the exact simulation of the application of the QFT to a W quantum state using the qiskit state-vector simulator on m100, with different numbers of threads. (left) We present the results in a y-log scale to highlight the exponential behavior. The simulation has been repeated 10 time for each data point, and we thus show the average with the shaded standard deviation. The standard deviation is, however, not easily seen in the log scale. (right) Ratio between the computational time with $m \in \{8, 16, 32, 64, 128\}$ and 128 threads for $n > 20$.

We benchmark the MPS code developed in Fortran on the Cineca cluster, and precisely on the Marconi 100 (m100) supercomputer. We tested the code together with the qiskit MPS simulator, to prove its efficiency. In Figure 3.4, we observe the results. We simulated the preparation of a $W$ state and the subsequent application of a QFT: the same circuit of Figure 3.3. We select a bond dimension $\chi = 10^4$, which guarantees that the maximum bond dimension is not reached, and so we keep a faithful representation of the state. We checked that $\chi = 10^4$ is not reached throughout the simulations. Even though the qiskit simulator is faster for a reduced number of qubits, namely for $n \leq 18$, the Fortran code outperforms it for larger $n$. In particular, we can easily state that the developed code is more efficient since the regime in which we are interested when we use the MPS simulation is $n \gg 1$. Furthermore, analyzing the slope $\gamma$ of the presented lines for $n > 20$, we can observe a significant difference:

$$\gamma_{qiskit} = (2.134 \pm 0.007)s^{-1} \qquad \gamma_{Fortran} = (1.53 \pm 0.02)s^{-1}.$$

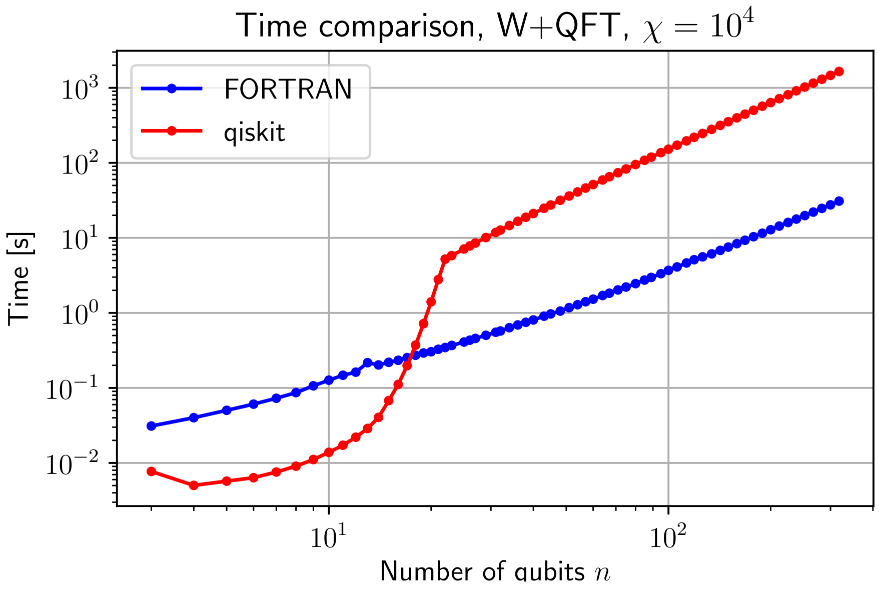


**Fig. 3.4:** Simulation time for the MPS simulation with both the qiskit and the Fortran code. We present the plot on a log-log scale, to better visualize the power-law behavior. The simulation is composed of the creation of a $W$ state and the subsequent application of the QFT.

The results achieved up to now are really promising to continue our analysis. However, we underline that for the MPS simulator, the problem is not the number of qubits, but the necessary bond dimension. For this reason, we analyze now the time scaling at a fixed number of qubits $n = 20$, by varying the maximum bond dimension. We use the QVOLUME circuit, since we have seen that is a difficult circuit to represent by MPS, and we are able to observe the scaling properties. In Figure 3.5, we can observe that for $2^4 \leq \chi \leq 2^8$ the behavior is, as expected, a power law. We use this interval to perform a linear fit in the log-log scale, obtaining the coefficient:

$$\gamma_\chi = (2.40 \pm 0.02)s^{-1}.$$

The Bures metric of the states was checked through the simulation, and displays a behavior analogous to the one presented in the left of Figure 3.1. It remains around zero for $\chi < 2^{\lfloor \frac{n}{2} \rfloor}$, which in this case is $2^{10} = 1024$, and then the state is correctly described for $\chi \geq 2^{10}$ with a Bures metric $B < 10^{-11}$.

In Chapter 2 we learned that the number of coefficients needed to describe for a quantum state of fixed system size in MPS form scales with $O(\chi^2)$. Now, we can observe the computational scaling in terms of the computational time, which depends for example on the matrix multiplication. Indeed, the scaling coefficient is affected by the matrix multiplication scaling with the Strassen algorithm [57], which scales as $O(m^{2.807})$, with $m$ the order of the matrix. For this reason, in Section 3.3 we work for minimizing the number of operations needed to simulate a quantum circuit and present the advantages that we can achieve through parallelization. Furthermore, we also recall that in each two-qubits gate application a SVD is applied, which scales as $O(m^3)$.

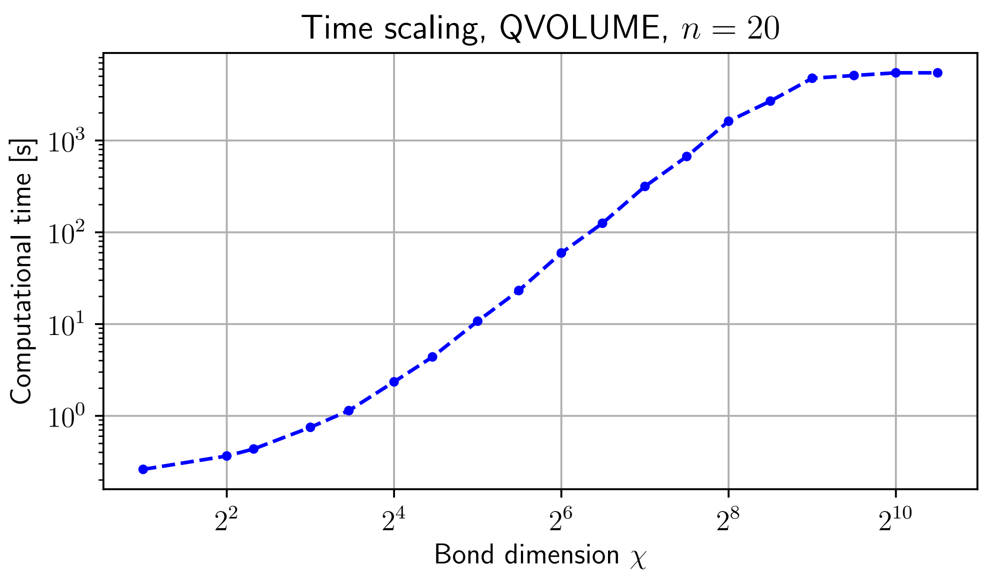


**Fig. 3.5:** Time scaling for the QVOLUME algorithm w.r.t. the bond dimension, in a log-log scale. We can observe that for $2^4 \leq \chi \leq 2^8$ the scaling is a power law, and in particular, we have $O(\chi^{2.40\pm0.02})$. The description of the state is exact for $\chi \geq 2^{10}$.

## 3.2.3 Gaussian Boson Sampling

In this section, we analyze the protocol introduced in Section 1.5, the Gaussian Boson sampling. We aim at minimizing the resources needed for the simulation, by studying the system with different Fock space cutoffs and bond dimensions. We observe which is the error induced by a fixed bond dimension in the estimation of the hafnian of a matrix, as defined in Section 1.5.3.

Before starting the analysis, let us introduce the *occupation profile*. With occupation profile we denote the distribution of the number of photons in a GBS experiment. Namely, we perform $10^4$ measurement of the system, and for each measurement we compute the total number of photons. For example, if in a system with $n = 5$ qumodes we measure the state $|0, 5, 0, 1, 2\rangle$ the total number of photons would be $n_{ph} = 8$. Then, we plot the probability to measure a state with given $n_{ph}$, obtaining a plot analogous to Figure 3.6.

### Fock space cutoff

First, we need to search for the Fock space cutoff is the most suitable for our system. Increasing the Fock dimension is very computationally demanding since it affects all the tensor operations. For this reason, we analyze how the occupation

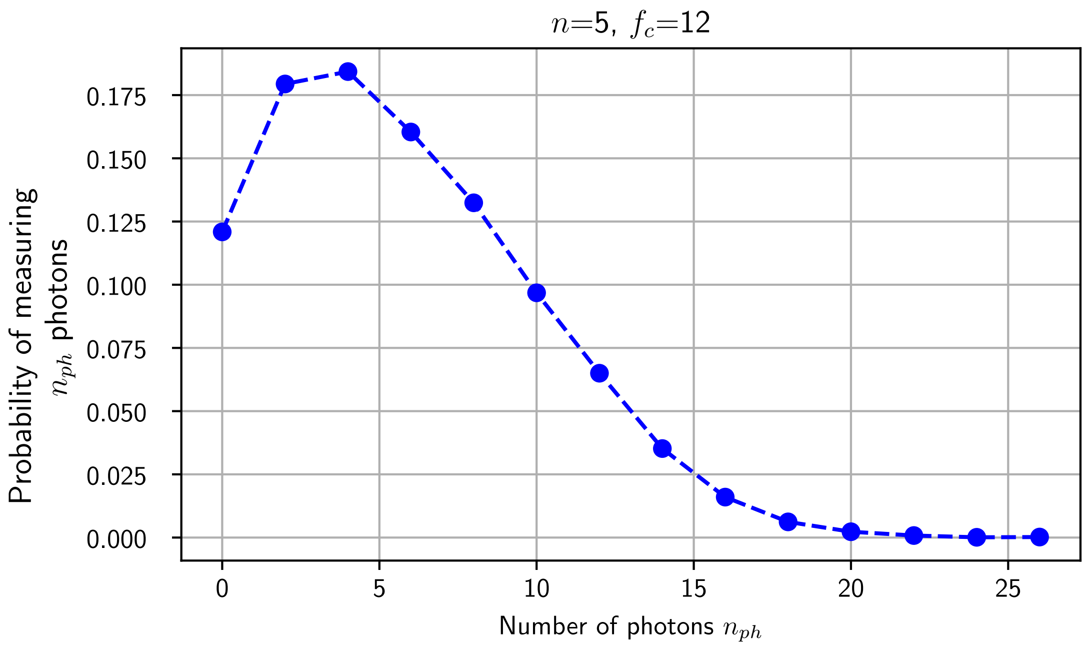


**Fig. 3.6:** Example of an occupation profile plot of a GBS experiments with $n = 5$ qumodes and a Fock space cutoff of $f_c = 12$. On the x-axis we show the number of photons measured, while on the y-axis the relative number of measurements, normalized to the total number of measurements.

profile changes when we modify the Fock space cutoff to discover a heuristic relation to identify the minimal meaningful cutoff. In Figure 3.7 we show how the average of the occupation profile distribution changes as a function of the Fock space cutoff $f_c$. We can notice how increasing the cutoff influences this average, up to a certain threshold, which increases with the number of modes $n$ of the system. By analyzing the derivative of the average, we can suggest that $f_c \simeq 15$ is a good compromise between a correct description of the system and a computationally sustainable simulation. Furthermore, we can observe the fact that the curves corresponding to $n = 10, 20$ are less smooth than the others. This behavior can be explained by the bond dimension $\chi$, which may not be high enough to correctly describe the system. Just to recall the difficulties of the exact simulation, the most computationally demanding point in the plot would require to store $30^{20}$ coefficients, which correspond to a system with $\sim 98$ qubits.

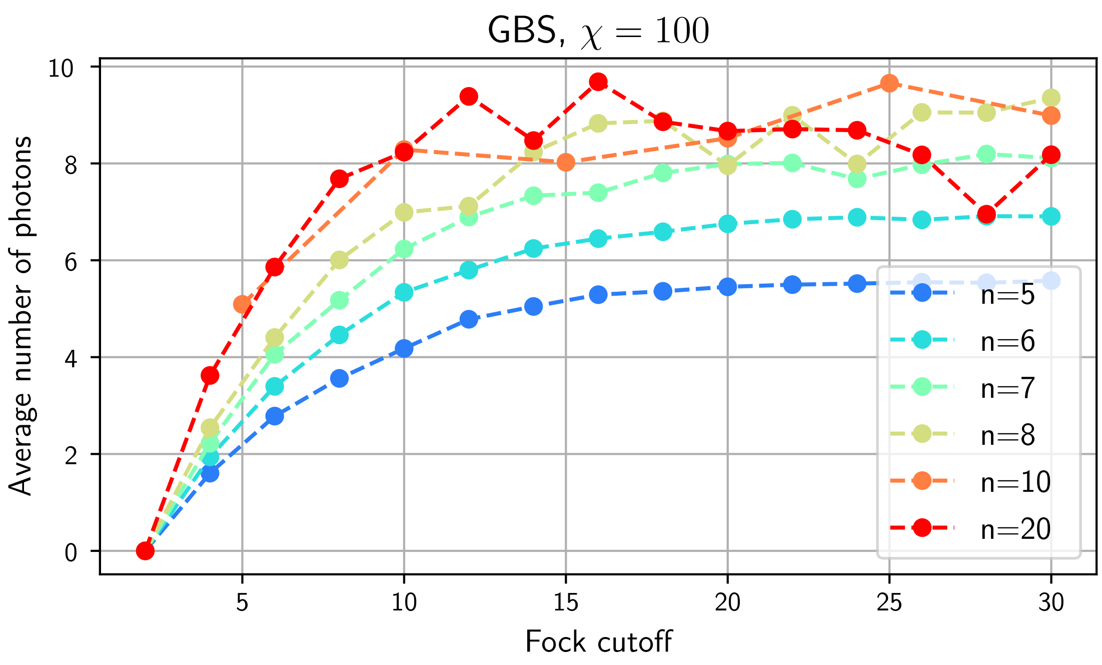


**Fig. 3.7:** Average number of photons w.r.t. the Fock space cutoff in the Gaussian Boson Sampling. We show the results for different number of qumodes $n$. We can observe that the average number of photons increases up to a plateau.

### Hafnian estimation

In Section 1.5.3 we presented the GBS protocol, focusing on the possibility of computing the Hafnian of a matrix through photon-click measurements. In this section, we analyze the error in the Hafnian estimation for small systems that can be easily simulated classically. For the classical computation we use the python library the walrus [58], an optimized package developed by Xanadu. We compute the error over 10 different Hafnian matrices, and take the average as an indicator. We also monitor the entanglement of the system, to check if the MPS technique is suitable for this simulation. In particular, we consider the maximum of the entanglement along the MPS chain, and then take the average along the different matrices. We show the results of this study in Figure 3.8. The bond dimension influences the error in the Hafnian estimation. By plotting the data in a log-log scale, we can notice a power law behavior, with exponent $\gamma_{error}^{gbs} = (-0.4731 \pm 0.008)$. As expected, the bond entropy entanglement $E_B$ grows with the bond dimension, with an exponent of $\gamma_{ent}^{gbs} = (0.577 \pm 0.005)$. It is important to notice that the derivative of both quantities at the right border is non-zero, and they will so continue to vary if we further increase $\chi$. This behavior means that the system is not perfectly described by the given bond dimension because it is not high enough. The GBS protocol needs a high bond dimension even for small systems, like $n = 6$ qumodes. We can so claim that GBS is a suitable platform to prove quantum supremacy, and that it can not be simulated easily with MPS. However, we can truncate the bond dimension and analyze the system with a limited amount of entanglement.

#### Occupation profile

After understanding how to tune the parameters of the simulation we can finally analyze the occupation profile of the Gaussian boson sampling protocol more in detail. We analyze how it varies, in particular increasing the number of qumodes as much as possible. We stick to a Fock space cutoff of $f_c = 15$ and a bond dimension $\chi = 100$. From Figure 3.9, we notice that the distribution is characterized by a single mode. It is more probable to measure states with a reduced number of photons, but as we increase the number of qumodes the maximum shifts to the right. However, this is not true passing from $n = 10$ to $n = 20$. We can hypothesise that the bond dimension $\chi = 100$ is not enough to describe the larger system, since it displays a behavior different from the expected one. For this reason, on the right plot of the figure, we report the distribution of the singular values cut in the simulation, due to the truncation procedure explained in Section 2.3. We notice that for $n = 5$ the ratio between the singular values cut and the largest singular value kept is very small, but the distribution shifts to the right while we increase the number of qumodes. In particular, we observe that for $n = 20$ we discard a relevant number of non-negligible singular values. Therefore, we can not trust the occupation profile for $n = 20$, and we should increase the bond dimension to obtain meaningful results. Again, the GBS protocol confirms the fact that it is a suitable platform for the quantum supremacy claim, being difficult to simulate for our platform. Even though we did not reach a final conclusion in this domain due to the time constraint for the thesis, further efforts will be spent to analyze the GBS protocol. Indeed, after the improvement that is presented in the next section, we may be able to simulate bigger systems in a manageable computational time.

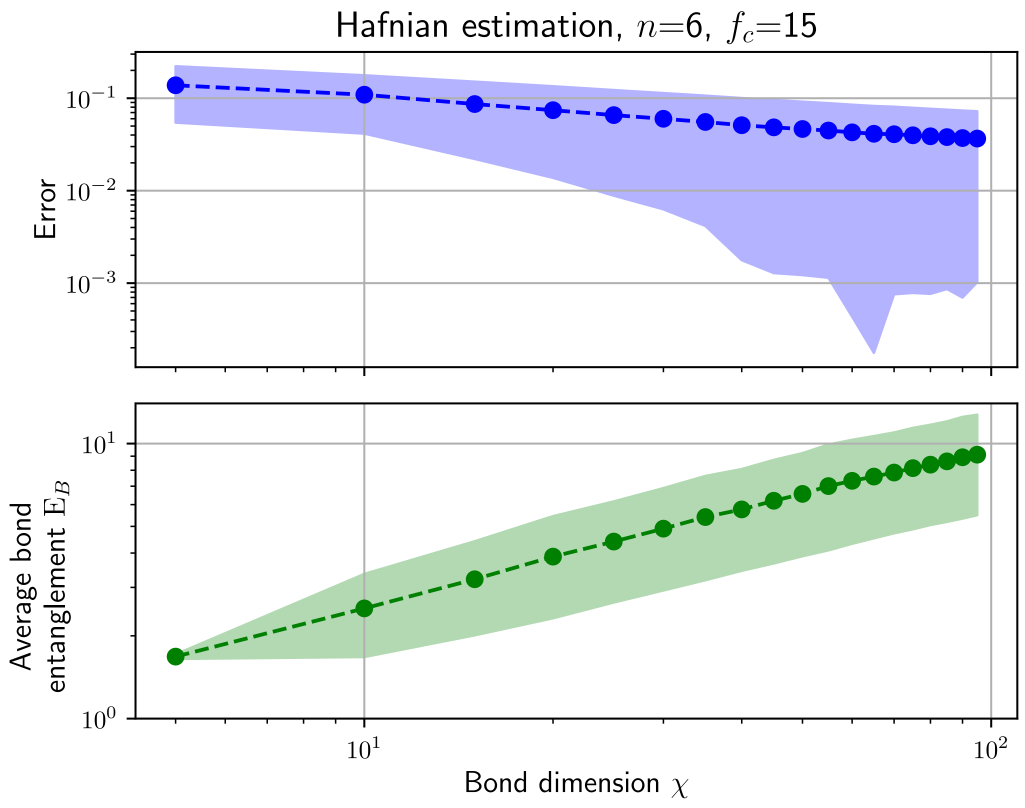


**Fig. 3.8:** Estimation of the Hafnian of a matrix through GBS protocol with 6 qumodes and a Fock space cutoff $f_c = 15$. The plots share the x-axis. The shaded area represent the standard deviation of the quantity, obtained over 10 repetitions of the experiment. On the top plot, we show the evolution of the error as a function of the bond dimension $\chi$ on a log-log scale. We can notice a power-law behavior, with the error decreasing. It is interesting to notice that the error does not reach zero for $\chi < 100$. On the bottom, we present the average bond entropy entanglement $E_B$ as a function of the bond dimension. We notice that the entanglement is growing as a power law with the bond dimension, and that it does not reach a maximum for $\chi < 100$. The whole set of simulations lasted 10:33:14 hours on the Marconi supercomputer.

## 3.3 Future development and improvement

In this section, we overview the possible improvement that we can implement in the simulator. Even though these advancements are not yet available, and will not be implemented in the work related to this thesis, it is nevertheless instructive

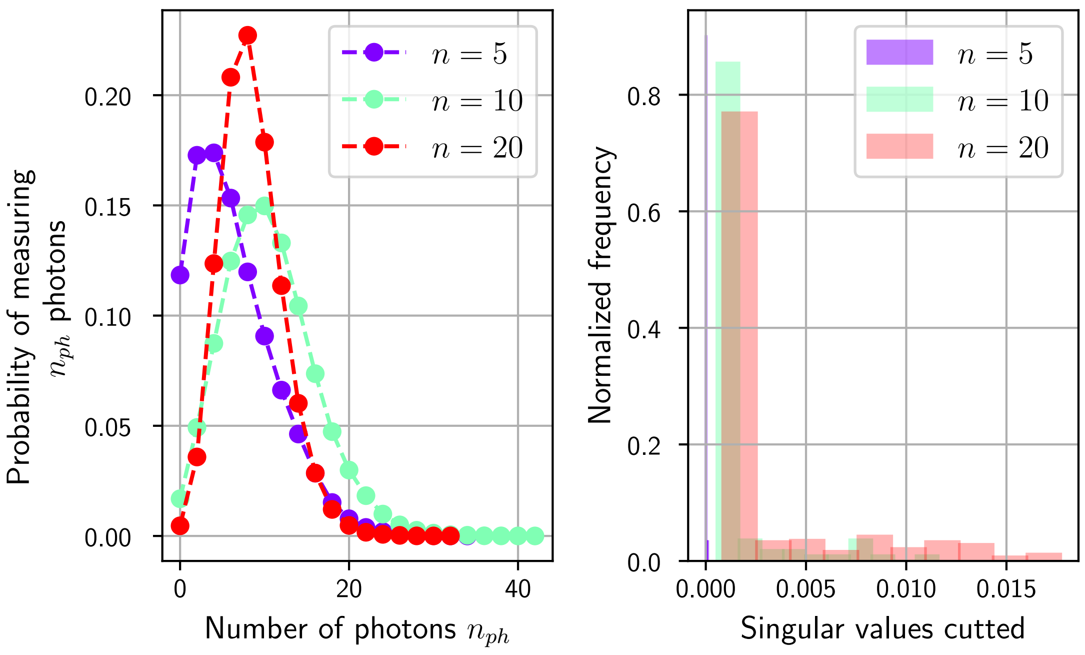


**Fig. 3.9:** These plots are produced with a maximum bond dimension $\chi = 100$ and a Fock space cutoff $f_c = 15$. On the left, we plot the occupation profile for the GBS protocol with increasing number of qumodes $n$. The number of average photons increases with $n$ from 5 to 10, but then it decreases. On the right, we report the distribution of the singular values cut during the simulation. We recall that on the x-axis is plotted the ratio between a singular value cut and the largest singular value kept.

to discuss them, since they will be present in the final product. First, in Section 3.3.1 we discuss how to transform a quantum circuit to minimize the number of gates, taking into account that we do not have constraints in the unitary matrix that represents a quantum gate. Then, in Section 3.3.2 we discuss the advantages and disadvantages of the parallelization of the code among different processes, discussing which is the best architecture for the tensor network quantum simulator.

### 3.3.1 Gate merging

In a real quantum computer only a few quantum gates, forming a universal set, are implemented in the hardware. For this reason, all the remaining gates are decomposed into a combination of this universal set. However, when working

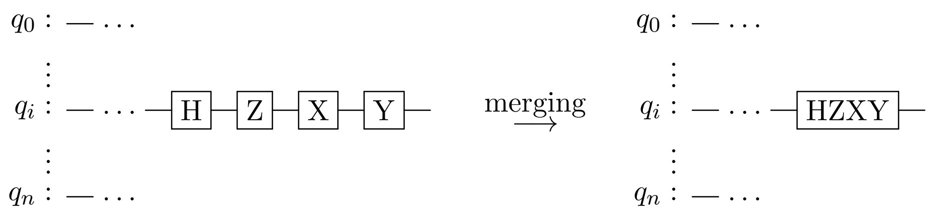


**Fig. 3.10:** Simple example of single-qubit gates merging. This process is advantageous, since instead of performing 4 contraction between the state, tensor of dimension $(\chi \times \chi \times 2)$, and the gates, tensors of dimension $(2 \times 2)$, we first contract together with the gates. Then, we only have to perform a single contraction with the state.

with a simulator we have no such constraint. The gates are represented as unitary matrices, and their application, as seen in Section 2.3, is simply the contraction between two tensors. We can apply *any* unitary matrix, regardless of its physical implementation.

Following this intuition, we can strongly simplify the tensor network which represents a quantum circuit *before* simulating it. For example, if we want to simulate the circuit in Figure 3.10 and proceed with the algorithm presented in Chapter 2, we would need to perform 4 contraction between the state, a tensor of dimension $(\chi \times \chi \times 2)$, and the gates, tensors of dimension $(2 \times 2)$. However, we can first contract the gates together, and then apply only a single contraction with the state.

Merging one-qubit gates is, however, the first trivial step. We can further simplify the network, by executing the following steps:

1. Contract together adjacent one-qubit gates;
2. Contract one-qubit gates with the adjacent two-qubit gate;
3. Contract together two-qubits gates that share *both* the application qubits. It is important to restrict the contraction in such a way since we are still limited to the application of at most two-qubit gates.

The steps above are presented in the tensor network notation in Figure 3.11.

### 3.3.2 Parallelization

In Section 3.2.2 we analyzed the time scaling of the simulator. By using a supercomputer like m100, we can handle matrices of relatively big size in small

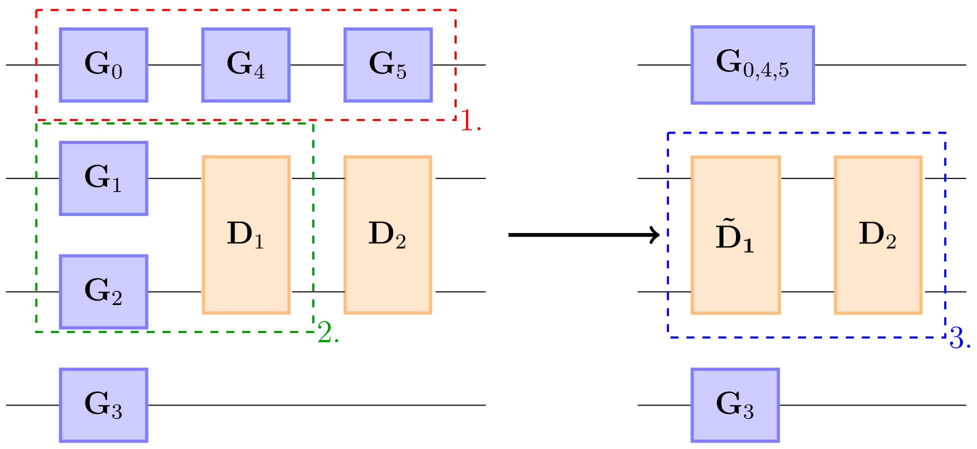


**Fig. 3.11:** Procedure to simplify a quantum circuit, by contracting the gates before the contraction with the quantum state. First (1.), we contract together adjacent one-qubit gates. Then (2.), we contract these one-qubit gates with their adjacent two-qubit gate. Finally (3.), we contract two-qubit gates which act on the same two qubits.

computational times. As an example, if we are working with a maximum bond dimension $\chi = 100$ then a single tensor has a dimension $(2 \times 100 \times 100) = 2 \cdot 10^4$, which is easily managed in the RAM of the device. Even the most computationally intensive task, the contraction of 2-sites gates, can be simply reduced to the multiplication of matrices with at most $(2 \times 2 \times 100 \times 100) = 4 \cdot 10^4$ elements. As already stated, it is possible that the bottleneck of the simulation is not in the single tensor contraction, but rather in the *number of tensor contractions*. From now on, we denote this quantity as the number of operations. We could assume that the parallelization of the operations speeds up the computations. We call a set of parallel operations *parallel cycle*. The number of parallel cycle and operations is the same in the case of a serial code. This statement, however, is not trivial. In the canonical TEBD technique, presented in Section 2.3.1, the parallelization always produces a speedup, since we have to apply two-site operators to the full chain. In the quantum circuit case, however, we encounter a circuit structure that is not regular or periodic. This means that parallelization is not a straightforward improvement. For this reason, we investigate some example cases to check if the introduction of a parallel code through Message Passing Interface (MPI) [59] produces a sufficient improvement. This topic is particularly important, since the parallelization on the quantum hardware is still an open question, and we may take inspiration from this work to tackle that problem.

Moreover, parallelizing the MPS evolution is not trivial. To minimize the truncation error we set the center of orthogonality at each two-site contraction. This approach is not possible if we want to apply more two-site operators *in parallel*. We do introduce a new algorithm, which creates a *fake* orthogonality center [60]. The algorithm, presented in Figure 3.12, is the following:

1. We must store the singular values after each SVD. We denote the singular values between site $k$ and $k+1$ as $\Lambda^{[k]}$. We denote $(\Lambda^{[k]})^{-1} = \tilde{\Lambda}^{[k]}$.

2. Insert an identity $\mathbb{1} = (\Lambda^{[k]})^{-1}\Lambda^{[k]}$ in the link preceding the application of a two-site operator. This means that if we want to apply a CNOT between sites $(1,2)$ and $(4,5)$ we insert the identity only link $(3)$.

3. Contract $\Lambda^{[k]}$ to the tensor on the right side, in our example $(\mathbf{U_4})$. Due to the contraction, these tensors are no longer unitary but locally mimic the orthogonality center, since the sites on the left and on the right are unitary.

4. Apply all the two-site operators in parallel, including the truncation. It is important to notice that the singular values are not necessarily valid anymore from a global perspective. The left-right unitary structure is a priori destroyed, arguing that the truncation of singular values is a non-unitary operation.

5. Perform a series of QR decomposition, taking care of $(\Lambda^{[k]})^{-1}$ to bring back the unitary structure.

It is important to underline that, even though points $1. - 4.$ can be performed in parallel, point 5. must be serial.

Now that we have defined a parallel algorithm, we rigorously enounce when a set of operations is parallelizable. Given a quantum circuit $\mathcal{Q}$ with $n$ qubits we denote an operation on the qubits set $q_\alpha$ at time $t$ as $o_t^{q_\alpha}$. Then, a set of subsequent operations $\mathcal{O} = \{o_t^{q_\alpha}\}_{t=1,2,\cdots,n_{op}}^{\alpha=1,\cdots,n_{op}}$ is parallelizable if and only if the intersection of all the $q_\alpha$ is empty.

$$\mathcal{O} = \{o_t^{q_\alpha}\}_{t=1,2,\cdots,n_{op}}^{\alpha=1,\cdots,n_{op}} \text{ is parallelizable } \iff \bigcap_{\alpha=1}^{n_{op}} q_\alpha = \varnothing. \tag{3.4}$$

In other words, operations in the same *layer* of the quantum circuit are parallelizable. This means that we have an upper bound to the number of parallel operations, which is the number of sites of the MPS $n$. Instead, if we consider the case of a circuit obtained after the transformation of the previous section,

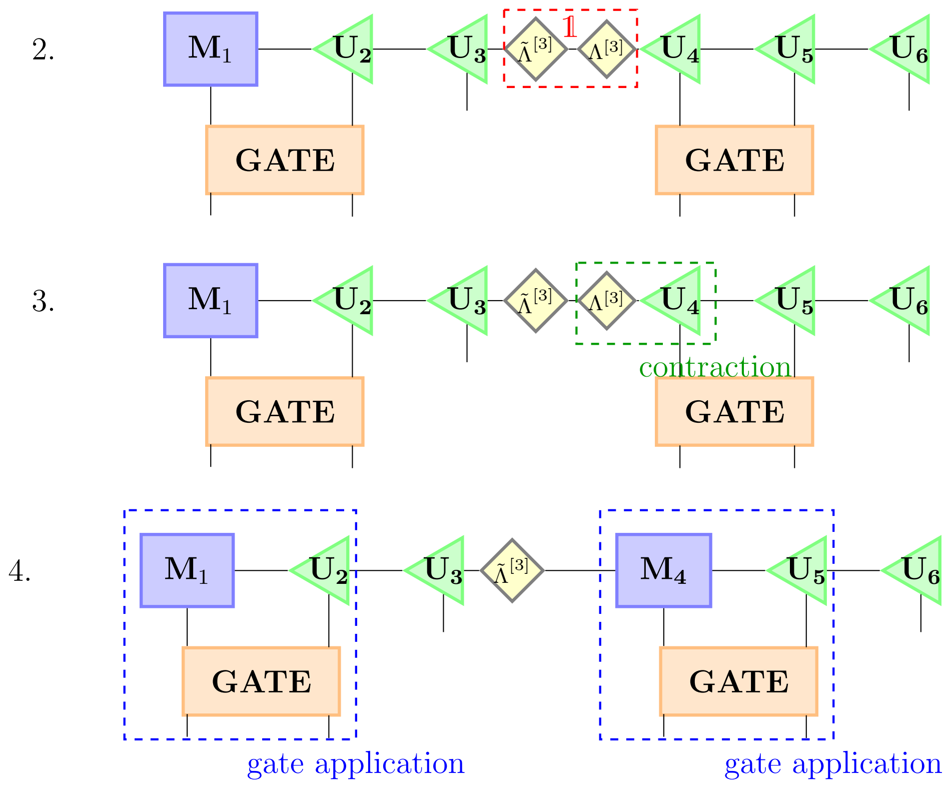


**Fig. 3.12:** Parallel algorithm for the application of gates. We follow the numbering of the text, thus starting from step 2. First, we insert an identity $\mathbb{1} = (\Lambda^{[k]})^{-1}\Lambda^{[k]} = \tilde{\Lambda}^{[k]}\Lambda^{[k]}$ in the link preceding the application of a two-site operator. We recall that the $\Lambda^{[k]}$ are the diagonal matrices obtained by the singular values after the SVD decompositions. In the above example, this means inserting $\tilde{\Lambda}^{[3]}\Lambda^{[3]}$ on the third link, since the gates are applied on tensors $(\mathbf{M_1}, \mathbf{U_2})$, and $(\mathbf{U_4}, \mathbf{U_5})$. We do not insert the identity before the tensor $\mathbf{M_1}$ since it is the first of the chain, and the true orthogonality center. Then, on step 3, we contract $\Lambda^{[k]}$ with the tensor on the right, $\mathbf{U_{k+1}}$ ($\mathbf{U_4}$ in the example). In this way, we are mimicking an orthogonality center in the new tensor $\mathbf{M_4}$. Finally, in step 4, we apply in parallel the operators, as explained in Section 2.3. It is then necessary to apply a series of QR decomposition to come back to step 2.

the circuit is composed of only two-qubits gates. In this scenario, the maximum number of parallel operations is $n/2$.

We proceed with the first case of analysis, a GHZ circuit. We compare the number of operations in the serial case, which coincides with the number of gates in the circuit, with the number of parallel cycles set in the parallel case. The non-local version of this circuit is composed of subsequent CNOT, each of which is connected to the same qubit. The topology of this circuit is not parallelizable, as we can see in the left plot of Figure 3.13. However, in the MPS simulator, we cannot use non-local operators, and so we need to first linearize the circuit using qiskit transpiler. We can observe in the right plot of Figure 3.13 that there actually is an improvement in the number of cycles needed. Indeed, by looking at the slope of the lines we can compute the scaling with the number of qubits:

$$\gamma_{seq}^{ghz-lin} = (1.055 \pm 0.007), \qquad \gamma_{par}^{ghz-lin} = (1.026 \pm 0.002).$$

As we expect from the plot, the scaling is the same (linear) even though the number of parallel cycles is less than the serial one. It is not a problem that the number of parallel cycles on the right plot is higher than the number on the left plot since the simulator is not able to reproduce a general, non-local GHZ circuit, but only its linearized counterpart.

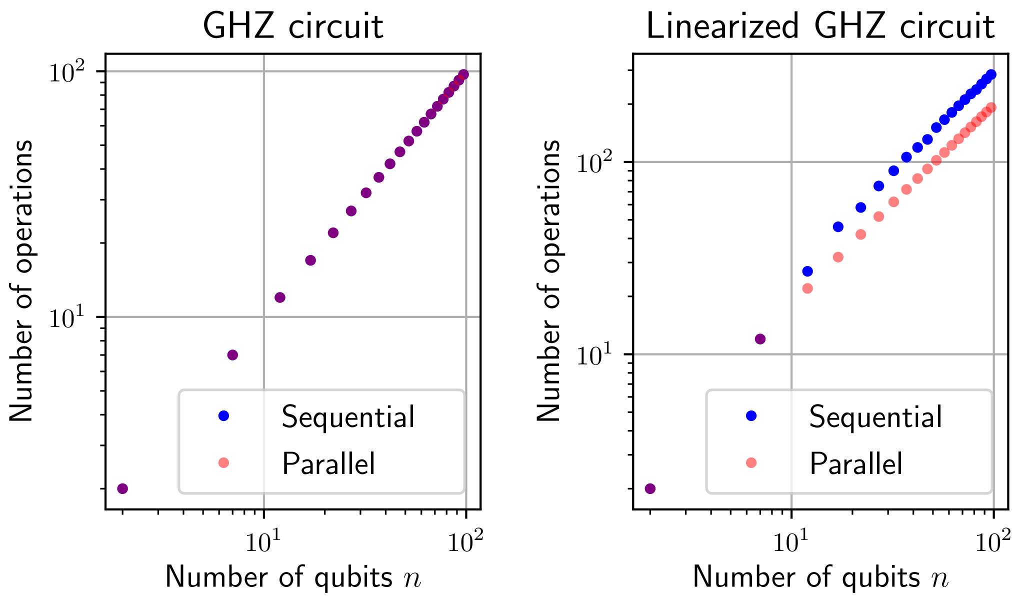


**Fig. 3.13:** Comparison between the number of cycles needed to obtain a GHZ state starting from the void state in log-log scale. On the left, we use the non-local version of the circuit, usually found in the literature, while on the right we show the results for the circuit linearized through the qiskit transpiler.

We can analyze a further example: the Quantum Fourier Transform. We tackle the problem with two different approaches, as before. We look at the number of cycles of the non-local QFT and those of the linearized one introduced in Section 1.3.2. In Figure 3.14 we can observe that, even though in the non-local case a parallelization is not worth the effort, the linearized case greatly benefits from it. Indeed, optimal planning of the circuit also brings an improvement in the number of cycles with respect to the non-local case. In particular, we can understand the scaling by looking at the slope of the lines in log-log scale:

$$\gamma_{seq}^{qft-nl} = (1.86 \pm 0.02) \qquad \gamma_{seq}^{qft-lin} = (2.0 \pm 0.0) \qquad \gamma_{par}^{qft-lin} = (1.13 \pm 0.02)$$

The parallelization strongly affected the scaling behavior, passing from a quadratic to an almost linear one.

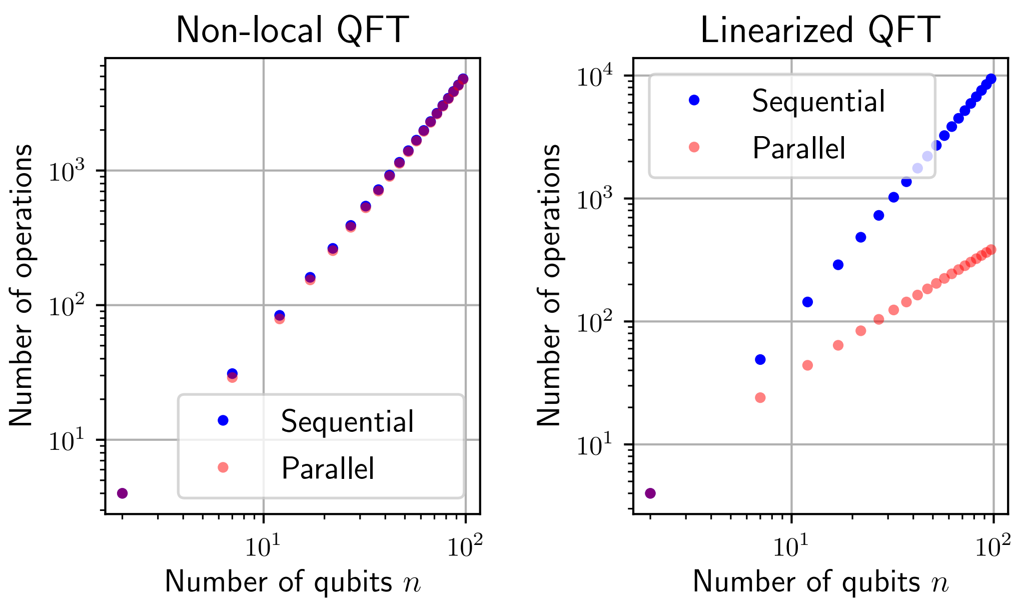


**Fig. 3.14:** Comparison between the number of cycles needed for a QFT circuit in log-log scale. On the left, we use the non-local version of the circuit, usually found in the literature, while on the right we show the results for the linearized circuit.

Even though these results seem very promising we still have to remember that the cardinality of the set of cycles $\mathcal{O}$ applied in parallel is also bounded by the number of parallel processes available in the machine. For this reason, we improve this analysis, focusing on two possible architecture:

- A cartesian structure, shown in Figure 3.15. In this technique, we divide the MPS chain into equal chunks, and each worker performs computations only on his subset of the system. To make operations between boundaries possible we perform a copy of the first tensor of the $(i+1)$-th subset into the $i$-th subset. These copies are called *halo regions*. The advantage of this approach is that we only need to perform communications between workers when we modify the boundaries.
- A master-workers (MW) approach, shown in Figure 3.16. In this procedure a worker, called the *master*, administrate the computations, sending them to the other workers when they are free. The advantage of this technique is that we have fewer idle workers, but we need to communicate for each computation.

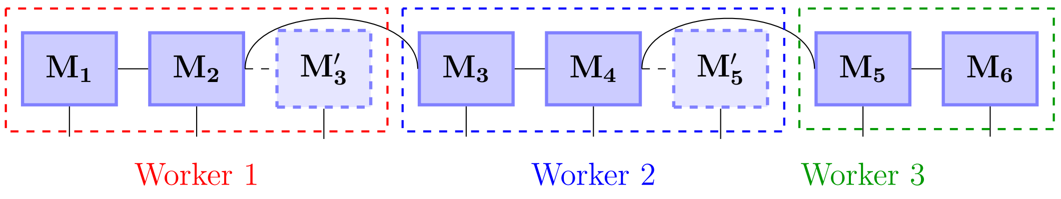


**Fig. 3.15:** Cartesian parallel architecture. We divide the MPS chain into equal chunks between workers 1, 2 and 3. We perform a copy of the first tensor, identified as $\mathbf{M'_j}$, of worker $(i+1)$ in worker $i$ to compute operations across boundaries. This architecture requires communications between workers each time one of the boundary tensors is modified. We can perform at most $w$ operations in parallel, where $w$ is the total number of workers. If no operation is applied to tensors assigned to worker $i$ on a certain layer that worker remains idle.

To check which of these two architectures is better suited for the quantum simulator, we analyze the number of parallel cycles and communications in two different cases. First, we improve the analysis started of the QFT, then we look at the behavior using random circuits.

In Figure 3.17 we observe the number of cycles needed to apply the QFT with either of the architectures. We repeat the analysis for different number of threads,

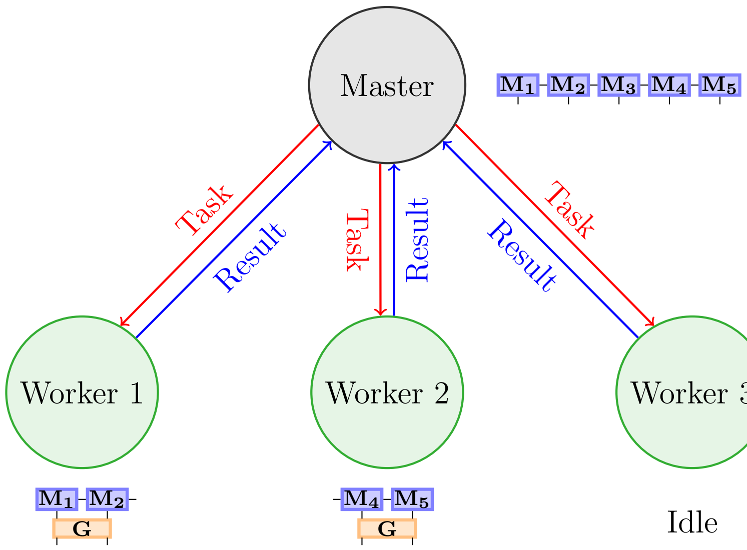


**Fig. 3.16:** Master-workers architecture. In this procedure, the master keeps all the information about the system, the full MPS chain. At each step, it sends a task, i.e. the application of a gate, to a free worker. The workers perform the computations in parallel, and then sends back to the master the results. In this approach, it is more difficult to have idle processes, even though it is possible. The problem is that for each task the worker must communicate with the master twice, once for receiving the data and once to send back the results.

i.e. 4, 16 and 64. We can notice, differently from the ideal version in Figure 3.14, the algorithm displays a different scaling with respect to the sequential one only up to a certain number of qubits, which depends on the number of available threads. This behavior is particularly clear in the master-worker approach with 16 threads. Around $n = 30$ there is a change in the line slope. For every number of threads the master-worker approach perform the simulation with fewer parallel cycles. However, we must keep into account the number of communications involved in the parallel algorithm, displayed in the lower plot. We notice that the number of communication for a given circuit is independent of the number of threads in the master-worker procedure, while it is strongly affected by the number of threads in the cartesian one. Indeed, fewer threads require less communications, as expected. We so need to look for a maximum on the performance for the cartesian approach, which balances the number of cycles and communications. To find out the best technique for this quantum circuit we must balance the information from Figure 3.17. In particular, it is beneficial to look for the fraction $\eta_{max}$ of resources that a communication requires such that the MW approach become more operational

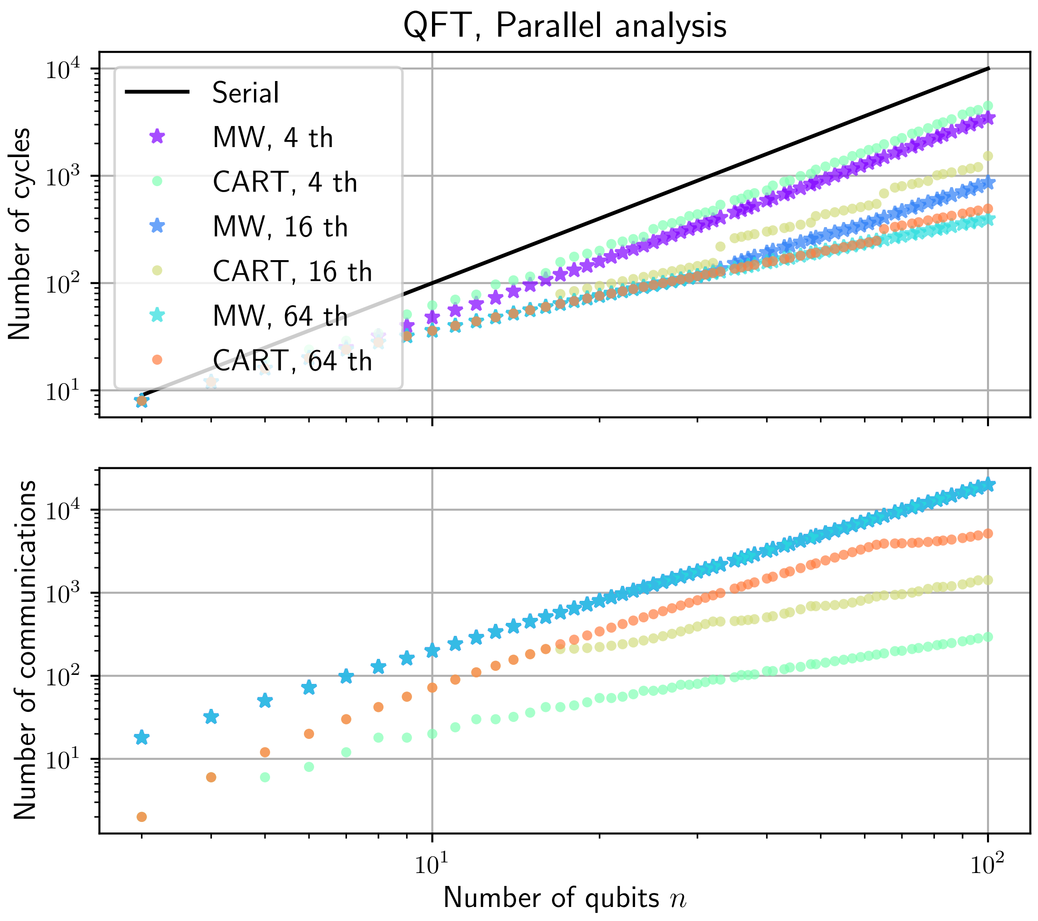


**Fig. 3.17:** Comparison between the master-worker and the cartesian parallel approach for the QFT circuit. We present the results on a log log scale, and the two plots share the x-axis and the legend. On the upper plot, we notice the number of cycles needed to simulate the circuit. We can observe that the slopes of the data points follow the serial one after a certain number of qubits, which depends on the number of threads th. On the bottom plot, we report the number of communications involved in the parallelization versus the number of qubits. This number is independent of the number of threads in the MW case, while it varies in the cartesian one.

intensive than the cartesian one. We can formally define $\eta_{max}$ by denoting the number of cycles (communications) for the MW approach with $n^{mw}_{cyc(com)}$ and with $n^{cart}_{op(com)}$ for the cartesian one:

$$\eta_{max} \quad | \quad n^{cart}_{cyc} + \eta_{max} n^{cart}_{comm} < n^{mw}_{cyc} + \eta_{max} n^{mw}_{comm}. \tag{3.5}$$

The result is thread-dependent: for 64 threads we have $\eta_{max} = 6 \cdot 10^{-3}$, while for 4 threads $\eta_{max} = 6 \cdot 10^{-2}$. This discussion can be further deepened. For example, the communication scales quadratically with the bond dimension, while the operations scale cubically. And, on top on that, the computational time can be greatly influenced by any prefactor in front of the scaling. Therefore, even though the previous analysis is instructive, we should study the algorithm once it is implemented.

We repeat the analysis in the case of random quantum circuits, presenting the results in Figure 3.18. We generated random circuit with $n$ qubits and a depth of $2n$ layers. Each configuration is an average over 10 realization of the circuit. We can observe that the differences between the MW and cartesian technique are less accentuated in these plots. Indeed, the number of cycles needed in the parallel approach is different only in the case of 16 threads, and that difference is smaller then in the QFT case. However, the differences in the number of communications persists: the MW approach requires more communications rather than the cartesian one. If we assign a weight to the communications, as we did in Equation (3.5), it is clear that the cartesian method requires less computational resources, and it is so preferable to the MW.

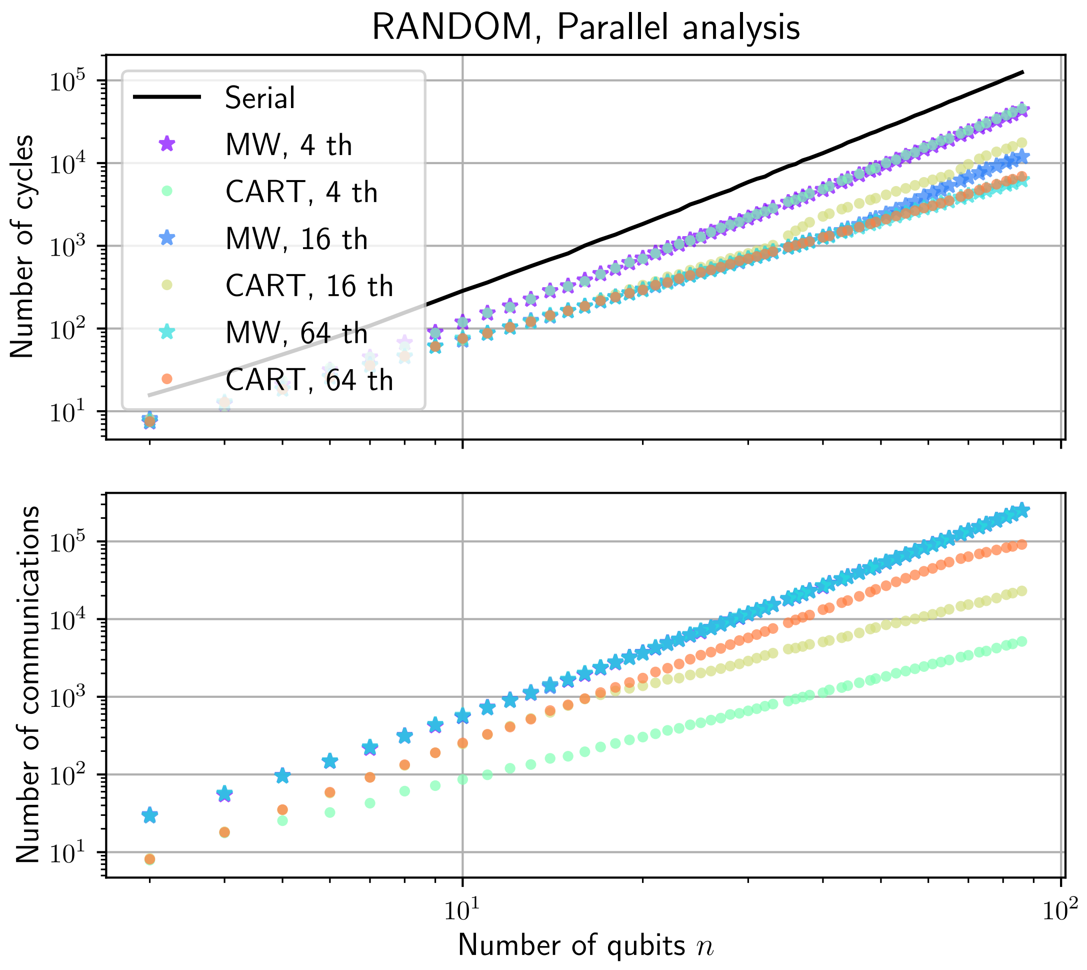


**Fig. 3.18:** Comparison between the master-worker and the cartesian parallel approach for random circuits. We present the results on a log log scale, and the two plots share the x-axis and the legend. On the upper plot, we notice the number of cycles needed to simulate the circuit. In this case, the number of parallel cycles is the same for the two methods in all cases but with 16 threads. On the bottom plot, we report the number of communications as a function of the number of qubits. We notice that the number of communications needed in the MW case are constant for a given circuit, and thus the star-shaped markers are superimposed.

# Conclusions

In this thesis, we implemented and explored a classical quantum simulator for quantum gates and photonic circuits. First, we introduced the concepts of quantum computation, starting from the definition of its fundamental unit, the qubit. We presented the procedure to evolve a quantum state in the quantum computing paradigm, i.e. through quantum gates. We also presented the continuous variables quantum computing paradigm using photonic modes as fundamental units. We focused on the Gaussian Boson sampling protocol, which was recently used in an experiment that claims quantum supremacy [12]. We listed the different methods to simulate these quantum evolutions, highlighting the limitation of such an approach in the exact case. For this reason, we introduced efficient methods to compress the information of a quantum state and simulate its evolution, namely the tensor network methods.

We described thoroughly the representation of quantum states using the Matrix Product State technique, a particular tensor network method suitable for describing one-dimensional chains, such as a quantum circuit. This approach let us evolve systems with many more qubits or qumodes. We listed the different operations that we can apply on Matrix Product States and the underlying sources of errors. We briefly discuss the Time Evolving Block Decimation procedure, which serves as the basis of our algorithm.

We developed a Quantum Computer simulator based on Matrix Product States, called QC-MPS. We wrote the numerical core in Fortran, to push the performances of the simulator as much as possible, and developed a python interface for easiness of use, which supports packages already established in the community, such as qiskit and strawberry fields. This simulator applies to both the quantum circuit and the quantum linear optic circuit cases. We then tested its correctness and computational time scaling on different quantum circuits, concluding that the simulator is working correctly and that the time scaling is as expected. We then focused on studying the Gaussian Boson sampling circuit circuit: we benchmarked the effect of the Fock cutoff necessary to describe the system faithfully. We monitored the evolution of the occupation profile using an increasing number of modes. From the simulations, we discovered a far more complex domain than the qubit's one, which will require further effort to be fully explored.

Finally, we discussed possible ways to improve the efficiency of the simulator by parallelizing the workload. First, we showed a way to reduce the number of operations in a quantum circuit, by contracting one-qubit gates and adjacent two-qubit gates. We debated the advantages and disadvantages of different Message Passing Interface architectures, concluding that the cartesian architecture is the most appropriate for our task. Even though the master-workers approach generally involves less parallel operations, the advantage can vanish due to more communications.

The implementation of the parallel algorithm is beyond the scope of this work, but future effort will be spent in this direction. Furthermore, additional studies will be addressed to better characterize the Gaussian Boson sampling protocol, and linear optical circuit in general. The simulator will be used for many different topics. For example, the possibility of simulating many qubits will enable us to study quantum error correction codes, or analyze the entanglement scaling in quantum machine learning models.

In conclusion, we developed an efficient tensor network quantum simulator, that, when completely mature, will be freely available for the scientific community.

# Bibliography

[1] Kent A Peacock. *The quantum revolution: a historical perspective.* Greenwood Publishing Group, 2008. DOI: `10.5860/choice.45-6849`.

[2] David J Griffiths and Darrell F Schroeter. *Introduction to quantum mechanics.* Cambridge University Press, 2018. DOI: `10.1017/9781316995433`.

[3] R Feynman. *Simulating Physics with Computers, 1982, reprinted in: Feynman and Computation.* 1999. DOI: `10.5860/choice.45-6849`.

[4] Michel H Devoret, Andreas Wallraff, and John M Martinis. "Superconducting qubits: A short review". In: *arXiv preprint cond-mat/0411174* (2004). URL: `https://arxiv.org/abs/cond-mat/0411174`.

[5] Hartmut Häffner, Christian F Roos, and Rainer Blatt. "Quantum computing with trapped ions". In: *Physics reports* 469.4 (2008), pp. 155–203. DOI: `https://doi.org/10.1016/j.physrep.2008.09.003`.

[6] Mark Saffman. "Quantum computing with atomic qubits and Rydberg interactions: progress and challenges". In: *Journal of Physics B: Atomic, Molecular and Optical Physics* 49.20 (2016), p. 202001. DOI: `10.1088/0953-4075/49/20/202001`. URL: `https://doi.org/10.1088/0953-4075/49/20/202001`.

[7] Jeremy L O'brien, Akira Furusawa, and Jelena Vučković. "Photonic quantum technologies". In: *Nature Photonics* 3.12 (2009), pp. 687–695. DOI: `https://doi.org/10.1038/nphoton.2009.229`.

[8] Javier Rodriguez-Laguna. "Real space renormalization group techniques and applications". In: *arXiv preprint cond-mat/0207340* (2002). URL: `https://arxiv.org/abs/cond-mat/0207340`.

[9] Simone Montangero, Montangero, and Evenson. *Introduction to Tensor Network Methods.* Springer, 2018. DOI: `https://doi.org/10.1007/978-3-030-01409-4`.

[10] L. Tagliacozzo et al. "Scaling of entanglement support for matrix product states". In: *Phys. Rev. B* 78 (2 July 2008), p. 024410. DOI: `10.1103/PhysRevB.78.024410`. URL: `https://link.aps.org/doi/10.1103/PhysRevB.78.024410`.

[11] Jacob Biamonte and Ville Bergholm. "Tensor networks in a nutshell". In: *arXiv preprint arXiv:1708.00006* (2017). URL: `https://arxiv.org/abs/1708.00006`.

[12] Han-Sen Zhong et al. "Quantum computational advantage using photons". In: *Science* 370.6523 (2020), pp. 1460–1463. DOI: `10.1126/science.abe8770`.

[13] Michael A Nielsen and Isaac Chuang. *Quantum computation and quantum information.* 2002. DOI: `https://doi.org/10.1017/CBO9780511976667`.

[14] Scott Aaronson and Daniel Gottesman. "Improved simulation of stabilizer circuits". In: *Physical Review A* 70.5 (2004), p. 052328. DOI: `10.1103/PhysRevA.70.052328`. URL: `https://link.aps.org/doi/10.1103/PhysRevA.70.052328`.

[15] Sergey Bravyi et al. "Simulation of quantum circuits by low-rank stabilizer decompositions". In: *Quantum* 3 (2019), p. 181. DOI: `10.22331/q-2019-09-02-181`.

[16] Christoph Adami and Nicolas J Cerf. "Quantum computation with linear optics". In: *NASA International Conference on Quantum Computing and Quantum Communications.* Springer. 1998, pp. 391–401. DOI: `https://doi.org/10.1007/3-540-49208-9_36`.

[17] Emanuel Knill, Raymond Laflamme, and Gerald J Milburn. "A scheme for efficient quantum computation with linear optics". In: *nature* 409.6816 (2001), pp. 46–52. DOI: `https://doi.org/10.1038/35051009`.

[18] Pieter Kok et al. "Linear optical quantum computing with photonic qubits". In: *Reviews of modern physics* 79.1 (2007), p. 135. DOI: `10.1103/RevModPhys.79.135`. URL: `https://link.aps.org/doi/10.1103/RevModPhys.79.135`.

[19] Craig S Hamilton et al. "Gaussian boson sampling". In: *Physical review letters* 119.17 (2017), p. 170501. DOI: `10.1103/PhysRevLett.119.170501`. URL: `https://link.aps.org/doi/10.1103/PhysRevLett.119.170501`.

[20] CS Peirce. *A Boolean algebra with one constant, Collected papers of Charles sanders Peirce, eds. C. Hartshorne and P. Weiss, vol. 4.* 1933. URL: `http://www.math.uic.edu/~kauffman/Penrose.pdf`.

[21] Wolfgang Tittel et al. "Violation of Bell inequalities by photons more than 10 km apart". In: *Physical Review Letters* 81.17 (1998), p. 3563. DOI: `10.1103/PhysRevLett.81.3563`.

[22] Albert Einstein, Boris Podolsky, and Nathan Rosen. "Can quantum mechanical description of physical reality be considered complete?" In: *Physical review* 47.10 (1935), p. 777. DOI: `10.1103/PhysRev.47.777`. URL: `https://link.aps.org/doi/10.1103/PhysRev.47.777`.

[23] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition.* 10th. USA: Cambridge University Press, 2011, p. 109. ISBN: 1107002176.

[24] Don Coppersmith. "An approximate Fourier transform useful in quantum factoring". In: *arXiv preprint quant-ph/0201067* (2002). URL: `https://arxiv.org/abs/quant-ph/0201067`.

[25] Peter W Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *Proceedings 35th annual symposium on foundations of computer science.* Ieee. 1994, pp. 124–134. DOI: `10.1109/SFCS.1994.365700`.

[26] Giuliano Benenti, Giulio Casati, and Giuliano Strini. *Principles of Quantum Computation and Information-Volume II: Basic Tools and Special Topics.* World Scientific Publishing Company, 2007. DOI: `https://doi.org/10.1142/5838`.

[27] Mikhail Smelyanskiy, Nicolas PD Sawaya, and Alán Aspuru-Guzik. "qHiPSTER: The quantum high performance software testing environment". In: *arXiv preprint arXiv:1601.07195* (2016). URL: `https://arxiv.org/abs/1601.07195`.

[28] Koen De Raedt et al. "Massively parallel quantum computer simulator". In: *Computer Physics Communications* 176.2 (2007), pp. 121–136. DOI: `https://doi.org/10.1016/j.cpc.2006.08.007`.

[29] Daniel Gottesman. "The Heisenberg representation of quantum computers". In: *arXiv preprint quant-ph/9807006* (1998). URL: `https://arxiv.org/abs/quant-ph/9807006`.

[30] Simon Anders and Hans J Briegel. “Fast simulation of stabilizer circuits using a graph-state representation”. In: *Physical Review A* 73.2 (2006), p. 022334. DOI: `10.1103/PhysRevA.73.022334`. URL: `https://link.aps.org/doi/10.1103/PhysRevA.73.022334`.

[31] Daniel Gottesman, Alexei Kitaev, and John Preskill. “Encoding a qubit in an oscillator”. In: *Physical Review A* 64.1 (2001), p. 012310. DOI: `10.1103/PhysRevA.64.012310`.

[32] Kevin E Cahill and Roy J Glauber. “Ordered expansions in boson amplitude operators”. In: *Physical Review* 177.5 (1969), p. 1857. DOI: `10.1103/PhysRev.177.1857`. URL: `https://link.aps.org/doi/10.1103/PhysRev.177.1857`.

[33] P Král. “Displaced and squeezed Fock states”. In: *Journal of Modern Optics* 37.5 (1990), pp. 889–917. DOI: `https://doi.org/10.1080/09500349014550941`.

[34] MS Kim et al. “Entanglement by a beam splitter: Nonclassicality as a prerequisite for entanglement”. In: *Physical Review A* 65.3 (2002), p. 032323. DOI: `10.1103/PhysRevA.65.032323`. URL: `https://link.aps.org/doi/10.1103/PhysRevA.65.032323`.

[35] William R Clements et al. “Optimal design for universal multiport interferometers”. In: *Optica* 3.12 (2016), pp. 1460–1465. DOI: `https://doi.org/10.1364/OPTICA.3.001460`.

[36] F. Verstraete, V. Murg, and J.I. Cirac. “Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems”. In: *Advances in Physics* 57.2 (Mar. 2008), pp. 143–224. ISSN: 1460-6976. DOI: `10.1080/14789940801912366`. URL: `http://dx.doi.org/10.1080/14789940801912366`.

[37] Román Orús. “A practical introduction to tensor networks: Matrix product states and projected entangled pair states”. In: *Annals of Physics* 349 (Oct. 2014), pp. 117–158. ISSN: 0003-4916. DOI: `10.1016/j.aop.2014.06.013`. URL: `http://dx.doi.org/10.1016/j.aop.2014.06.013`.

[38] J. Eisert. *Entanglement and tensor network states.* 2013. arXiv: `1308.3318 [quant-ph]`. URL: `https://arxiv.org/abs/1308.3318`.

[39] Jens Eisert, Marcus Cramer, and Martin B Plenio. “Colloquium: Area laws for the entanglement entropy”. In: *Reviews of Modern Physics* 82.1 (2010), p. 277. DOI: `10.1103/RevModPhys.82.277`.

[40] Matthew B Hastings. “An area law for one-dimensional quantum systems”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2007.08 (2007), P08024. DOI: `https://doi.org/10.1088/1742-5468/2007/08/P08024`.

[41] Pietro Silvi. *Tensor Networks: a quantum-information perspective on numerical renormalization groups*. 2012. arXiv: `1205.4198 [quant-ph]`. URL: `http://hdl.handle.net/20.500.11767/4293`.

[42] Mikel Sanz. “Tensor Networks in Condensed Matter”. PhD thesis. Apr. 2011. DOI: `10.14459/2011md1070963`.

[43] Roger Penrose. “Applications of negative dimensional tensors”. In: *Combinatorial mathematics and its applications* 1 (1971), pp. 221–244. URL: `http://www.math.uic.edu/~kauffman/Penrose.pdf`.

[44] Glen Evenbly. *Tutorial 3: Gauge Freedom*. URL: `https://www.tensors.net/tutorial-3` (visited on 02/12/2021).

[45] Sebastian Paeckel et al. “Time-evolution methods for matrix-product states”. In: *Annals of Physics* 411 (Dec. 2019), p. 167998. ISSN: 0003-4916. DOI: `10.1016/j.aop.2019.167998`. URL: `http://dx.doi.org/10.1016/j.aop.2019.167998`.

[46] Pietro Silvi et al. “The Tensor Networks Anthology: Simulation techniques for many-body quantum lattice systems”. In: *SciPost Physics Lecture Notes* (Mar. 2019). ISSN: 2590-1990. DOI: `10.21468/scipostphyslectnotes.8`. URL: `http://dx.doi.org/10.21468/SciPostPhysLectNotes.8`.

[47] Ulrich Schollwöck. “The density-matrix renormalization group in the age of matrix product states”. In: *Annals of physics* 326.1 (2011), pp. 96–192. DOI: `https://doi.org/10.1016/j.aop.2010.09.012`. URL: `https://www.sciencedirect.com/science/article/pii/S0003491610001752`.

[48] Rohit Chandra et al. *Parallel programming in OpenMP*. Morgan kaufmann, 2001. ISBN: 978-1-55860-671-5.

[49] EM Stoudenmire and Steven R White. “Minimally entangled typical thermal state algorithms”. In: *New Journal of Physics* 12.5 (2010), p. 055026. DOI: `10.1088/1367-2630/12/5/055026`. URL: `https://doi.org/10.1088/1367-2630/12/5/055026`.

[50] Guifré Vidal. "Efficient simulation of one-dimensional quantum many-body systems". In: *Physical review letters* 93.4 (2004), p. 040502. DOI: 10.1103/PhysRevLett.93.040502. URL: https://link.aps.org/doi/10.1103/PhysRevLett.93.040502.

[51] Naomichi Hatano and Masuo Suzuki. "Finding exponential product formulas of higher orders". In: *Quantum annealing and other optimization methods*. Springer, 2005, pp. 37–68. DOI: https://doi.org/10.1007/11526216_2.

[52] Héctor Abraham et al. *Qiskit: An Open-source Framework for Quantum Computing*. 2019. DOI: 10.5281/zenodo.2562110.

[53] Héctor Abraham et al. *Qiskit terra standard gates*. URL: https://github.com/Qiskit/qiskit-terra/tree/main/qiskit/circuit/library/standard_gates. (accessed: 10.08.2021).

[54] Nathan Killoran et al. "Strawberry fields: A software platform for photonic quantum computing". In: *Quantum* 3 (2019), p. 129. DOI: 10.22331/q-2019-03-11-129.

[55] Andrew W Cross et al. "Validating quantum computers using randomized model circuits". In: *Physical Review A* 100.3 (2019), p. 032328. DOI: 10.1103/PhysRevA.100.032328.

[56] Amandeep Singh Bhatia and Ajay Kumar. "Quantifying matrix product state". In: *Quantum Information Processing* 17.3 (2018), pp. 1–16. DOI: https://doi.org/10.1007/s11128-017-1761-1.

[57] V Strassen CM970438RF. "Gaussian elimination is not optimal". In: *Numer. Math* 13 (1969), pp. 354–356. DOI: https://doi.org/10.1007/BF02165411.

[58] Brajesh Gupt, Josh Izaac, and Nicolás Quesada. "The Walrus: a library for the calculation of hafnians, Hermite polynomials and Gaussian boson sampling". In: *Journal of Open Source Software* 4.44 (2019), p. 1705. DOI: https://doi.org/10.21105/joss.01705.

[59] Brandon Barker. "Message passing interface (mpi)". In: *Workshop: High Performance Computing on Stampede*. Vol. 262. 2015. URL: http://cac.cornell.edu/Education/training/StampedeJan2015/IntroMPI.pdf.

[60] EM Stoudenmire and Steven R White. "Real-space parallel density matrix renormalization group". In: *Physical review B* 87.15 (2013), p. 155137. DOI: `10.1103/PhysRevB.87.155137`. URL: `https://link.aps.org/doi/10.1103/PhysRevB.87.155137`.