University of Trento

Department of Physics

Ph.D Thesis

~ · ~

Academic Year 2022–2023

Machine-Aware Enhancing of Quantum Computers

Supervisor
Prof. Francesco Pederiva

Ph.D. Student
Piero Luchi

Final examination date: July 20, 2023

“Chacun appelle “idées claires” celles qui sont au même degré de confusion que les siennes propres.” M. Proust

Acknowledgments

This research was partially supported by Q@TN grants ML-QForge (PL). The Lawrence Livermore work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 with support from Laboratory Directed Research and Development grant 19-DR-005 LLNL-JRNL-842516. This work was prepared in part by LLNL under Contract DE-AC52-07NA27344 with support from the Laboratory Directed Research and Development grant 19-DR-005.

Abstract

The realization of a computer that exploits quantum - rather than classical - principles represents a formidable scientific and technological challenge. Today, superconducting quantum processors are achieving significant results in simulation and computation capabilities. However, the realization of a fault-tolerant quantum device still poses many technical difficulties. First, it requires the ability to generate high-fidelity gates by exploiting both hardware and software solutions and improvements. Second, it requires the ability to perform quantum error correction. Finally, it is of primary importance to have a high-fidelity qubit readout to actually extract information from the device.

The thesis will focus on the first and last requirements, proposing advances in quantum optimal control protocols for high-fidelity gates and machine-learning based qubit readout. The methods used for these improvements exploit general mathematical machinery that can be specialized for the specific quantum device to obtain (reconfigurable) machine-aware protocols. This is easily achieved by accessing the properties and parameters of the machine. In this dissertation, these techniques are tested on superconducting qubits.

Optimal control protocols make it possible to tailor control signals (in the form of electromagnetic or optical fields) that implement arbitrary unitary transformations in quantum computers. This helps to reduce the depth and hence the noise of quantum circuits. These protocols replace long gate circuits, which are the result of decomposing a unitary operator into a sequence of elementary transformations, with a single application of a customized gate obtained by appropriately optimizing a microwave pulse. However, optimization algorithms can be computationally demanding, especially in contexts where many controls related to parametric variations in the unitary are required. This can negate the benefits of the optimal control approach.

The most common qubit readout technique today is dispersive readout (in the circuit QED architecture), in which the qubit is coupled to a readout resonator. In this approach, the state of the qubit is determined by measuring the quadrature amplitudes of an electromagnetic field transmitted through the resonator. Hardware random thermal noise, gate errors, or qubit decay processes that occur during measurements can reduce readout fidelity. Machine learning techniques and classification schemes could help to restore good fidelity by improving the classification accuracy of the measurement outputs. The Gaussian Mixture Model is the most commonly used classification method due to its ease of use. It uses parametric modeling of the probability distribution of averaged readout output data in terms of a sum of Gaussians to perform a classification of new measurements. However, more advanced techniques can be applied. Some authors have proposed and realized various classification methods based on neural networks trained on the entire output measurement signals instead of their averages, with good results. Another

approach is based on the unsupervised approach of the Hidden Markov Model, which allows a detailed classification of the measurement results and the detection of decay processes that the qubit might undergo during the measurement. These schemes help to improve the accuracy of the classification of the qubit readout measurements.

The present dissertation will follow these two tracks with the common goal of improving the performance of quantum computers.

In the case of quantum optimal control, the application of fitting procedures among a previously computed set of controls could help to reduce computation time. A new control should not be optimized with slow algorithms but simply interpolated from the set of controls already available. In addition, more advanced mathematical techniques can be used. Quantum computers can only perform unitary transformations. Therefore, by pre-computing the controls corresponding to a set of unitary matrices (belonging to $SU(2)$ and $SU(4)$) constructed by sampling their generators, machine learning techniques can then be used to interpolate between them and reconstruct the control for any unitary matrix (of these dimensions). These approaches are tested on the simulation of quantum (nuclear) systems, which is one of the most interesting and promising applications of the capabilities of quantum computers. Regarding the qubit readout part, we expected that the measurement output signal of the qubit system can be exploited to improve the readout procedure. In particular, by exploiting the information contained therein, one can improve the classification of states, making the procedure more parameter-independent and noise-resistant. Depending on the number of qubits or qubit levels, the readout signals are divided into different classes. These signals are noisy and often confused between classes due to thermal fluctuations, instrument noise, and quantum state decay processes. In this dissertation, we investigate how these data can be used to infer the state of the qubit more precisely and to improve the measurements. This can be achieved by applying advanced machine learning protocols, both with a supervised approach, using different realizations of neural networks, and with an unsupervised approach (e.g. autoencoders). Machine learning algorithms, taking advantage of their generalization and universal fitting capabilities, should allow better handling of hidden correlations in the data and provide better classification results of the measurements, as already demonstrated by some preliminary work. Furthermore, the use of unsupervised models paves the way for further studies on the behavior of the qubit in a more speculative way.

Contents

Introduction

The concept of quantum computing finds its origin in the 1980s when some physicists began to hypothesize computational models that integrated the laws of quantum mechanics [1] and with some studies on the quantum Turing machine [2]. Its development expanded in its natural setting of quantum system simulation with the pioneering exposition by Feynmann in 1981 [3] and other works [4]. One of the milestones of its subsequent development is for sure the Shor algorithm, a quantum algorithm for the factorization in prime numbers of composite integers [5]. However, the theoretical study of quantum information and of quantum algorithms requires a device that should be stable, reliable, and usable. Qubits need to be protected from environmental noise that induces decoherence but, at the same time, their states have to be controlled by external controls. Di Vincenzo, in his well-known criteria, summarized the characteristics that the experimental setup should meet to be a "quantum computer" [6] :

1. *A scalable physical system with well-characterized qubits*: This means that the system should be scalable to a large number of qubits, and the qubits should be well-defined and controllable.

2. *The ability to initialize the state of the qubits*: This means that the qubits should be initially set in a known state.

3. *Long coherence times*: This means that the qubits should be able to maintain their quantum state for a long time without being perturbed by external factors.

4. *A universal set of quantum gates*: This means that the system should be able to perform any quantum operation on the qubits.

5. *The ability to measure the qubits*: This means that the system should be able to extract information from the qubits without destroying their quantum state.

6. *The ability to communicate between the qubits*: This means that the qubits should be able to interact with each other in a controlled way.

While progress has been made in developing quantum hardware and software, there is still much work to be done to achieve a practical quantum computer that can solve real-world problems. In particular, current quantum computers, while meeting these requirements, still suffer from noise and decoherence that degrade their performance. Present devices are commonly referred to as *noisy intermediate-scale quantum* (NISQ) devices [7]. These are $\mathcal{O}(100)$ qubit devices that, while having the potential to perform tasks faster than today's classical digital computers, still have a noise level that limits the size of quantum circuits that can be reliably executed. NISQ devices are useful tools for testing current quantum algorithms and exploring many-body quantum physics. The goal for future research in this area is the realization of *fault tolerant* devices that can support a large number of qubits while maintaining sufficiently high qubit quality and fidelity in operations such as quantum gate implementation and measurement.

One of the areas in which quantum computers can have a positive impact is the field of quantum simulation. Classical numerical methods struggle to solve realistic quantum systems. This is due to the exponential growth of the Hilbert space dimension with the number of particles or degree of freedom. This makes it necessary to keep track of the probability amplitudes for all the possible classical configurations of the system with a consequent exponential increase in the memory required to store this information. Furthermore, simulating the temporal evolution of the system requires a number of operations that also increases exponentially with the size of the system. Moreover, peculiar quantum properties, such as the superposition principle, entanglement, or quantum tunneling, make the simulations even more difficult. To tackle these limitations, classical stochastic methods, as *quantum Monte Carlo* algorithms, have been developed. These methods allow evaluating the phase space integrals for many-body quantum systems in a time that scales polynomially with the size of the system. However, these methods perform well when the functions to be integrated vary sufficiently slowly with the relevant variables and, most importantly, do not change sign (this is the well-known "sign problem" [8]). If this happens, the statistical error grows exponentially and the simulation time to compensate for it increases, reducing the advantage of using Monte Carlo methods. There are many other methods developed to solve the dynamics of quantum many-body systems, such as *density functional theory*, *mean-field theories*, *many-body perturbation theories*, or *Green's function-based methods*, etc. [9], each of which, however, has its own limits of applicability. Due to their intrinsic quantum mechanical nature, quantum computers can deal with the quantum behavior of systems under analysis in a natural way. Entanglement or superposition effects are, in this case, inherently present in the device which simulate the quantum system, instead of being information that has to be stored and loaded. To compute the dynamics of a quantum system, we usually need to compute a time evolution operation, which is expressed in terms of the Hamiltonian of the system. The decomposition of the evolution operator with the Trotter-Suzuki expansion into discrete-time steps allows for the efficient implementation of unitaries on quantum computers [10]. The polynomial increase in circuit depth with the desired evolving time and target accuracy may not be feasible on NISQ devices without access to error correction. Nevertheless, NISQ devices play an important role in testing and validating these techniques and algorithms.

This thesis presents machine-aware improvements of current NISQ devices based on superconducting qubits. Tailored mathematical techniques have been developed to mitigate the impact of noise in circuits and to improve the fidelity of qubits readout. The methods used for these improvements exploit general mathematical machinery that can be specialized for the specific quantum device to obtain (reconfigurable) machine-aware protocols. This is easily achieved by accessing the properties and parameters of the specific device in use. This research represents a link between the theoretical study of quantum computing and its experimental/practical implementation and could benefit both sides.

The research focuses on improving NISQ devices in the areas that can be identified with Di Vincenzo's fourth and fifth criteria. On the one hand, it focuses on the improvement of quantum circuits and noise mitigation using optimal control protocols, which can be identified with criterion four. On the other hand, it studies possible improvements of the qubit readout from a software point of view, thus addressing criterion five.

It begins with criterion four. Optimal control protocols can design electromagnetic or optical control pulses that induce the evolutionary unitary of interest in the qubit system. This approach helps to reduce the depth and therefore the noise of quantum circuits. A long circuit, resulting from the decomposition of the unitary operator into a sequence of elementary quantum gates, can be replaced by a single or a few applications of customized gates composed by an appropriately optimized microwave pulse. However, these optimization algorithms can be computationally expensive, especially in contexts where many controls are required due to parametric variations in the unitary. This can spoil the benefits of the optimal control approach. However, applying fitting procedures to a previously computed set of controls could help reduce computation time. A new control should not be optimized with slow algorithms but simply interpolated from the existing set of controls. This results in a speed-up for circuit compilation. In addition, more advanced mathematical techniques can be used. In fact, quantum computers are designed to perform unitary transformations. Therefore, each transformation naturally belongs to the matrix representation of a $SU(2^N)$ Lie group (where N is the number of qubits). One can sample elements of this group and obtain a set of unitary matrices by exponentiating uniformly distributed elements belonging to the corresponding Lie algebra. Then, the controls implementing each sampled transformation can be optimized. Finally, machine learning techniques are used to interpolate between them and, exploiting their generalization properties, reconstruct the control for any unitary matrix. These approaches are tested on the simulation of quantum (nuclear) systems.

The second part focuses on criterion five, the readout. The most common qubit readout technique today is *dispersive readout* (in the circuit QED architecture)[11, 12], in which the qubit is dispersively coupled to a readout resonator, which is a circuit that can store and manipulate microwave photons. The resonator is then probed with a weak microwave signal, which causes the resonator frequency to shift depending on the state of the qubit. By measuring the frequency shift of the resonator, the state of the qubit can be inferred. The advantage of dispersive readout is that it is a non-destructive measurement, meaning that the qubit can be measured multiple times without losing its quantum information. However, the downside is that the measurement is indirect and can be noisy, which can lead to errors in quantum computing operations. Researchers are working on improving the accuracy and speed of dispersive readout to make it more practical for quantum computing applications. Machine learning techniques and classification schemes could help to obtain good fidelity by improving the classification accuracy of the measurement outputs. The Gaussian Mixture Model [13] is the most commonly used classification method due to its ease of use. It uses parametric modeling of the probability distribution of averaged readout output data in terms of a sum of Gaussians to perform a classification of new measurements. However, more advanced techniques can be applied. Some authors have proposed and realized various classification methods based on neural networks trained on the entire output measurement signals instead of their averages, with good results. Another approach is based on the unsupervised approach of the Hidden Markov Model [14], which allows a detailed classification of the measurement results and the detection of decay processes that the qubit might undergo during the measurement. These schemes help to improve the accuracy of the classification of the qubit readout measurements.

In general, exploiting all the output signals should allow for improved readout fidelity. Using the information contained therein, one can improve the classification of states, making the procedure more parameter-independent and noise-resistant. Depending on the number of qubits or qubit levels, the readout signals are divided into different classes. These signals are noisy and often get confused between classes due to thermal fluctuations, instrument noise, and quantum state decay processes. In this dissertation, we investigate how these data can be used to infer the state of the qubit more precisely. This can be achieved by applying advanced machine learning protocols, both with a supervised approach, using different realizations of neural networks, and with an unsupervised approach (e.g. autoencoders). Machine learning algorithms, taking advantage of their generalization and universal fitting capabilities, should allow better handling of hidden correlations in the data and provide better classification results of the measurements. Furthermore, the use of unsupervised models paves the way for further studies on the behavior of the qubit in a more speculative way.

The dissertation has the following structure. In chapter 1 some quantum information basics are introduced. Chapter 2 presents a review of the qubit type used in the thesis, namely the superconducting qubit. In chapter 3 the quantum simulation with quantum computers is discussed. In chapter 4 is given a brief summary of the machine-learning techniques used in the research. Chapter 5 are given results of the control interpolation studies, together with a theoretical application to nuclear system simulation. Finally, in chapter 6 is discussed the application of machine learning algorithm to the superconducting qubit readout and the results of this protocol.

Chapter 1

Basics of Quantum Computing

1.1 Introduction

Quantum computers are a type of computing device that use quantum-mechanical phenomena, such as superposition and entanglement, to perform computations. In contrast to classical computers, which use bits that can only have a value of 0 or 1, quantum computers use quantum bits, or qubits, that can exist in a superposition of states and can represent both 0 and 1 simultaneously. This should, in principle, allow quantum computers to perform certain calculations exponentially faster than classical computers.

In this chapter, some basics about quantum computing will be given. We assume that the reader already has some knowledge of the subject, so the introduction will be concise and will mainly focus on the tools used in the rest of the dissertation. The chapter begins with an analogy between classical and quantum computers to highlight the similarities and differences between the two approaches.

1.2 Classical Computer

A classical computer is a machine that takes strings of bits as input and returns modified strings of bits as output.

The bit is the most basic unit of information in a classical computer. The state space of a single classical bit is $\mathbb{Z}_2 = 0, 1$. Hence, a bit can have only two states, 0 or 1. A realization of a bit is simply a physical system with a two-dimensional state space. In modern computing devices, a bit is usually represented by the two distinct voltage or current levels allowed by a wire or circuit.

The state space of n bits is the direct sum of single bits, namely:

$$\underbrace{\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \ldots \oplus \mathbb{Z}_2}_{n} \equiv \{0,1\}^n. \tag{1.1}$$

So, every string $y = y_n y_{n-1} ... y_1$ of n bits can be in 2^n different combination of zero and one, which are also the number of states that it can represent.

A (classical) computer takes strings of bits as inputs and returns transformed strings as output. The mathematics of classical computers is grounded in the principles of Boolean algebra and digital circuit theory. Boolean algebra is a mathematical framework that deals with true/false or on/off values and logical operations such as AND, OR, and NOT. In digital circuit theory, these logical operations are implemented through electronic components like transistors, which act as switches that can be turned on or off to represent binary values. Formally it can be represented as a function:

$$f : \{0,1\}^n \to \{0,1\}^m, \tag{1.2}$$

where $n(m)$ is the input (output) string dimension. Actually, f is composed of smaller functions that take and transform subsets of bitstrings. Function as "AND", "OR", "NOT" and "XOR" are famous examples of these small functions. They are called *gates* and, assembled together, can create complex classical circuits performing complicated computational tasks. Moreover, it can be demonstrated that a subset of these gates can be used to construct any arbitrary function, hence they are called *universal gate set*. An example is the set {"NOT", "AND"}.

In programming, algorithms are written in high-level languages like Java, Python, or C++, which are then compiled into machine code that can be executed by the computer's processor.

1.3 Quantum Computer

A quantum computer is a machine that takes as input a collection of qubits in an initial state and, after a desired manipulation, returns the qubits in a new output state. The classical information about states can be extracted by performing a measurement of the qubits' output state.

The qubit is the basic unit of information of a quantum computer. It is a controllable and measurable two-level quantum system. Its state space is $\mathbb{C}^2$, where the vector of this space is also required to be normalized. Qubits can be in a superposition of states, representing 0 and 1 simultaneously. According to quantum mechanics, an arbitrary state $|\psi\rangle$ of a qubit can be written as a linear superposition of its basis states, identified by $|0\rangle$ (ground state) and $|1\rangle$ (excited state), namely:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle , \tag{1.3}$$

with $\alpha, \beta \in \mathbb{C}$ subjected to the restriction that $|\alpha|^2 + |\beta|^2 = 1$ or, equivalently, that $\langle\psi|\psi\rangle = || \, |\psi\rangle \, ||^2 = 1$.

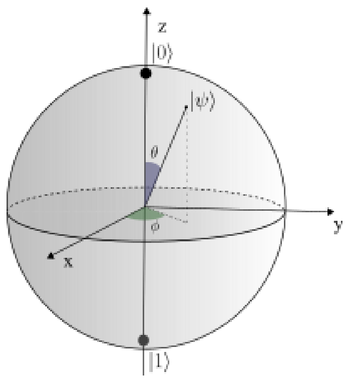


Figure 1.1: Bloch sphere. The "north pole" represent the ground state $|0\rangle$, the "south pole" the first exited state $|1\rangle$. A state $|\psi\rangle$ [Eq. (1.4))] is a point on the surface of the sphere as a function of the basis states $|0\rangle$ and $|1\rangle$.

This restriction allows rewriting Eq. (1.3) as:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle\,, \tag{1.4}$$

where $0 \leq \theta \leq \pi$ and $0 \leq \phi < 2\pi$. From this rewriting is clear that the qubit state space is a sphere of radius 1, which has the name of *Bloch sphere*, see Fig. 1.1. Hence, a qubit, contrary to the classical bit which has only two possible states, is a continuous mixture of the two basis states, so it can virtually attain infinite possible states.

A qubit can be physically realized by many devices, e.g. 1/2-spin particle, ions, polarization states of photons or quantum superconducting circuits. In general, a qubit is a quantum system in which distinct quantum states can be generated and accessed. The state space of n qubits is the tensor product of single qubit spaces, that is:

$$\underbrace{\mathbb{C}^2 \otimes \mathbb{C}^2 \otimes .... \otimes \mathbb{C}^2}_{n}\,. \tag{1.5}$$

So a general state $|\psi\rangle$ is represented as a superposition of basis states $|\psi\rangle = \sum_{i=0}^{n} c_i\,|i\rangle$. In general, we need 2^n complex coefficients to fully characterize the state.

A quantum computer manipulates the state of the qubits to obtain results. The mathematics behind manipulations are linear algebra and quantum mechanics formalisms. Formally a transformation can be represented by a function F:

$$F : |\psi\rangle \to |\psi\rangle'\,. \tag{1.6}$$

For a quantum computer, this function is the action of a unitary operator U on the initial state, i.e $|\psi\rangle' = U|\psi\rangle$. Analogously to the case of the classical computer, this function can be broken down into a sequence of simpler functions acting on a limited number of qubits. These smaller functions are the equivalent of classical gates and are therefore called *quantum gates*. Even in this case an *universal (quantum) gate set* can be defined, such that any unitary matrix U can be approximated by a sequence of gates coming from this set. The result is a *quantum algorithm* often in the form of *quantum circuit* which are a set of instruction that can run on a quantum computer to obtain a certain result. Mathematically, a gate that acts on n qubits is an element of the $SU(2^n)$ special unitary group. The Solovay-Kitaev theorem [15, 16] states that there is a constant c such that any unitary transformation $U \in SU(d)$ can be approximated by a sequence S of $O(\log^c(1/\epsilon)$ quantum gates of a universal gate set with an error $\epsilon < d(U, S)$ where $d(\cdot,\cdot)$ in operator norm. So this result reassures us that the number on gates scales in a manageable trend with the precision ϵ.

1.3.1 Measurements

To extract the information from the qubits after a manipulation, we need to perform a measurement. Generally, in quantum computing, measurement gives us back the probability distribution of the qubits on a computational basis. So given an arbitrary qubit state of Eq.(1.3), the measurement will provide the state $|0\rangle$ with probability $|\alpha|^2$ and state $|1\rangle$ with $|\beta|^2$. A measurement by its nature is a not-unitary and irreversible operation, hence we cannot go back to the state we had before the measurement. The general, definition of measurement comes from the postulates of quantum mechanics [17].

1.3.2 Projective measurements

The most traditional and straightforward description of measurement in quantum mechanics is *projective measurement* [18]. A quantum physical quantity O has an associated operator, or *observable*, $\mathcal{O}$. This can be diagonalized as:

$$\mathcal{O} = \sum_m m M_m, \tag{1.7}$$

where $\{m\}$ are the eigenvalues of $\mathcal{O}$, assumed to be real and discrete, and M_m are the *projection operators* onto the subspace of eigenstates of $\mathcal{O}$. We assume $\mathcal{O}$ to have a non-degenerate spectrum so that the projector is simply a rank 1 operator $M_m = |m\rangle\langle m|$. These are called *von Neumann measurements*. Moreover, projectors form an orthonormal basis, obeying

$$M_m M_{m'} = \delta_{m,m'}. \tag{1.8}$$

Now, the outcome of a measurement of $\mathcal{O}$ is one of its eigenvalues m. The probability to measure the eigenvalue m is

$$\mathcal{P}_m = \mathrm{Tr}\{\rho(t) M_m\}, \tag{1.9}$$

where $\rho(t)$ is the (instantaneous) density matrix (or state matrix). After the measurement, the conditional a-posteriori state becomes:

$$\rho_m(t+T) = \frac{M_m^\dagger \rho(t) M_m}{\mathcal{P}_m}, \tag{1.10}$$

where T is the time to perform the measurement, assumed to be short enough not to allow the system to evolve significantly due to other causes. This rewriting is often called "wavefunction collapse" since it says that the state has been projected by M_m into the corresponding subspace of the total Hilbert space. A consequence of this collapse is that if the measurement is immediately repeated the same result will be given, in fact:

$$\mathcal{P}(m'|m) = \mathrm{Tr}\{\rho_m(t+T) M_{m'}\} = \delta_{m,m'}. \tag{1.11}$$

In the case of pure state, in which $\rho(t) = \langle \psi(t) | \psi(t) \rangle$, Eq. (1.9) and (1.10) become:

$$\mathcal{P}_m = \langle \psi(t) | M_m | \psi(t) \rangle \tag{1.12}$$

and

$$|\psi_m(t)\rangle = \frac{M_m |\psi(t)\rangle}{\sqrt{\mathcal{P}_m}}. \tag{1.13}$$

Qubit measurement example

As an example let's take the qubit state $|\psi\rangle$ of Eq. (1.3). A measurement in the $|0\rangle$, $|1\rangle$ basis, would correspond to have two projectors onto each state, namely $M_0 = |0\rangle\langle 0|$ and $M_1 = |1\rangle\langle 1|$. The the probability of obtaining state $|0\rangle$ is:

$$\begin{aligned} \mathcal{P}_0 &= \langle \psi | M_m^\dagger M_m | \psi \rangle \\ &= \langle \psi | \langle 0 | |0\rangle |\psi\rangle \\ &= (\alpha^* \langle 0| + \beta^* \langle 1|)(\langle 0| |0\rangle)(\alpha |0\rangle + \beta |1\rangle) \\ &= \alpha^* \alpha \langle 0|0\rangle = |\alpha|^2. \end{aligned} \tag{1.14}$$

Moreover, the state $|\psi'\rangle$ after the measurement in $|0\rangle$ is:

$$|\psi'\rangle = \frac{M_0 |\psi\rangle}{\sqrt{\langle \psi | M_0^\dagger M_0 |\psi\rangle}} = \frac{1}{|\alpha|} \langle 0| |0\rangle (\alpha |0\rangle + \beta |1\rangle) = \frac{\alpha}{|\alpha|} |0\rangle. \tag{1.15}$$

Note that $\alpha = |\alpha| e^{i\phi}$ for some phase angle $\phi = [0, 2\pi)$. Consequently, $\alpha/|\alpha| = e^{i\phi}$ is a complex phase. However, as stated in the previous section, the global phase does not matter and we can assume that the state is simply $|0\rangle$.

1.3.3 General formulation of quantum measurements

Despite its importance from a theoretical point of view, the projective measurement framework is generally inadequate or too simplified to describe real measurements. In fact, one rarely performs measurements on the quantum system itself. Instead, one measures the effects on the environment in which the system is embedded. For example, one does not perform a measurement on a qubit using theoretical projectors, but, as in the case discussed in this dissertation (see section.2.4), one relies on the analysis of an electromagnetic field interacting with the qubit itself. Since the field is coupled to the system, their states are correlated and one can infer the state of the system by measuring the field. This is why we need a more general framework.

Assume we have a quantum system we want to measure in an initial state $|\psi(t)\rangle$ and a second quantum system, the *apparatus* or *meter*, in an initial state $|h(t)\rangle$. We can express the initial global unentangled state as:

$$|\Phi(t)\rangle = |h(t)\rangle\,|\psi(t)\rangle\,. \tag{1.16}$$

If we consider the two systems as coupled for a time Δt, they are transformed by a unitary operator $U(\Delta t)$. The global state Eq. (1.16) after the interaction becomes:

$$|\Phi(t+\Delta t)\rangle = U(\Delta t)\,|h(t)\rangle\,|\psi(t)\rangle\,. \tag{1.17}$$

Now, in contrast to the previous case, we measure the state of the apparatus instead of the system itself. We projectively measure the apparatus for a time Δt_M, assuming the evolution of both system and meter is negligible in this time frame. Let's assume that the projection operators for the apparatus are rank-1 operators $M_r = |r\rangle\langle r| \otimes I$, where r is the observed value of the quantity R of the meter. The $|r\rangle$ set forms an orthonormal basis for the apparatus Hilbert space. We can now define the probability of obtaining a specific value r from the measurement of the apparatus, namely:

$$\mathcal{P}_r = \langle h(t)|\,\langle\psi(t)|\,U^\dagger(\Delta t) M_r U(\Delta t)\,|h(t)\rangle\,|\psi(t)\rangle\,. \tag{1.18}$$

As a consequence, the final state becomes:

$$|\Phi(t+\Delta t+\Delta t_M)\rangle = \frac{|r\rangle\langle r|\,U(\Delta t)\,|h(t)\rangle\,|\psi(t)\rangle}{\sqrt{\mathcal{P}_r}} \tag{1.19}$$

Which can be furthermore be rewritten as:

$$|\Phi(t+\Delta t+\Delta t_M)\rangle = \frac{|r\rangle\,A_r\,|\psi(t)\rangle}{\sqrt{\mathcal{P}_r}} \tag{1.20}$$

since the measurement disentangles the system and the meter. Here where we condensed $A_r = \langle r| U(\Delta t) |h(t)\rangle$ which is called *measurement operator.* This given, we can rewrite Eq.(1.18) in the more familar form:

$$\mathcal{P}_r = \langle \psi(t)| A_r^\dagger A_r |\psi\rangle \tag{1.21}$$

1.4 Advantage of Quantum Computers in Quantum System Simulation

One of the main topics of this dissertation is the improvement of quantum systems simulations on quantum computers. Quantum computing has significant potential for simulating quantum systems (e.g., nuclear systems). Classical computers are limited in their ability to simulate complex quantum systems due to the exponential growth of the computational resources required to store all the information about the quantum state. However, quantum computers can efficiently simulate quantum systems and offer the potential to explore phenomena that are difficult to study on classical computers. One of the critical advantages of quantum computing for the simulation of quantum systems is its ability to simulate quantum entanglement efficiently. Quantum computers can also simulate quantum systems with many more particles than classical computers can handle. For example, a quantum computer with only a few hundred qubits could efficiently simulate the behavior of a small molecule, which would be impossible on a classical computer. Another advantage of quantum computing for simulating quantum systems is the ability to simulate quantum systems with continuous variables, such as those found in quantum optics. Classical computers struggle to simulate these systems because of their infinite-dimensional Hilbert spaces, but quantum computers can efficiently simulate these systems with their continuous-variable quantum processors.

In summary, quantum computing has significant potential for the simulation of quantum systems and is an area of active research in quantum physics and computer science. As quantum computing technology advances, we can expect more breakthroughs in quantum simulation and a better understanding of the fundamental workings of the quantum world.

Chapter 2

Elements of Superconducting Qubits and Optimal Control Theory

2.1 Introduction

This section will introduce the type of qubit used in the analyses presented in this work. Qubits can be realized through a wide range of systems. Trapped ions [19, 20, 21], diamond nitrogen-vacancies [22, 23], quantum dots [24, 25], electron spins in silicon [26, 27], ultracold atoms [28, 29], polarized photons [30, 31] are all examples of this. In all these cases, the quantum information (the states $|0\rangle$ and $|1\rangle$) is encoded in natural microscopic quantum systems. Another approach, employed in this dissertation, is to work with *superconducting qubits* [32, 33, 34, 35]. These are, in contrast, macroscopic and lithographically defined objects. Here the information is stored in the degrees of freedom of engineered superconducting circuits with anharmonic oscillatory behavior. Contrary to other qubits, superconducting ones can be realized by exploiting present technologies and expertise in microchip fabrications. So they are relatively easy to obtain and can be controlled and coupled with present electronic devices. There are several types of superconducting qubits, including the transmon qubit, the flux qubit, and the phase qubit. The transmon qubit is the most commonly used superconducting qubit and has been used in many experimental demonstrations of quantum algorithms. Superconducting qubits are typically manipulated using microwave pulses, and read out using microwave or low-frequency electrical measurements. They are often coupled to form larger systems, such as quantum processors and quantum communication networks. They can be built in a broad range of the values of their parameters, e.g. transition frequencies or anharmonicity, to exhibit different behaviors and work in

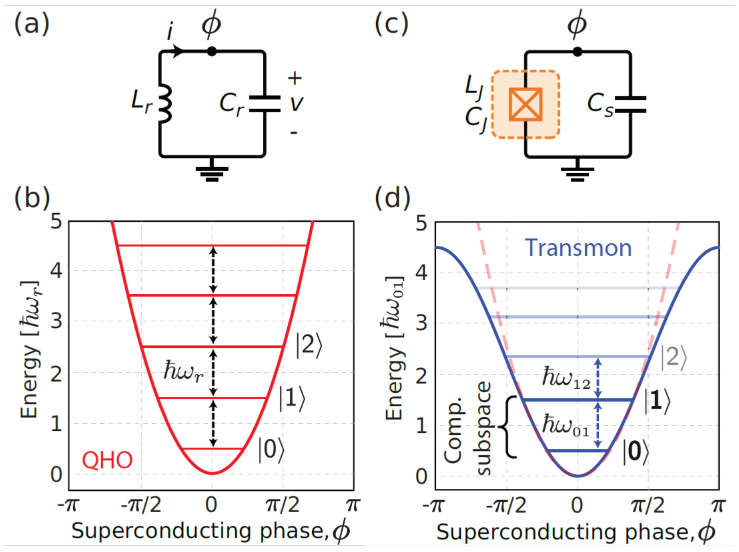


Figure 2.1: Harmonic and anharmonic oscillators. (*a*) A schematic representation of a LC circuit. (*b*) Potential energy of the LC circuit. It presents equidistant levels. (*c*) A schematic representation of a non-linear LC circuit (transmon) where a Josephson junction replaces the classical inductance. (*d*) Potential energy of the transmon. The levels are not equidistant and can be addressed individually. Figure taken from [32]

different regimes. This chapter aims to introduce the basic properties of superconducting qubits and in particular the behavior of the transmon superconducting qubit.

2.2 Basic working principle

To obtain quantum mechanical behavior in a microscopic integrated circuit, the absence of dissipation is the main requirement. Therefore, these devices operate at ultra-low temperatures and must be made of metals that exhibit superconducting behavior at these temperatures. The operating temperature of these devices must be the temperature at which the typical energy k_BT of the thermal fluctuations is less than the energy $\hbar\omega_{01}$ associated with the transition between the $|0\rangle$ and $|1\rangle$ states of the qubit. Typically, this energy ranges from 5 to 20 GHz, so the operating temperature must be around 20 mK. These temperatures are challenging from an engineering point of view, not only because one has to rely on a dilution refrigerator, but more importantly because one has to deal with the temperature gradient of the wires connecting the superconducting qubits at 20 mK to room temperature. This requires a careful setup of electromagnetic noise filtering [36].

Another requirement for quantum signal processing is the use of nonlinear

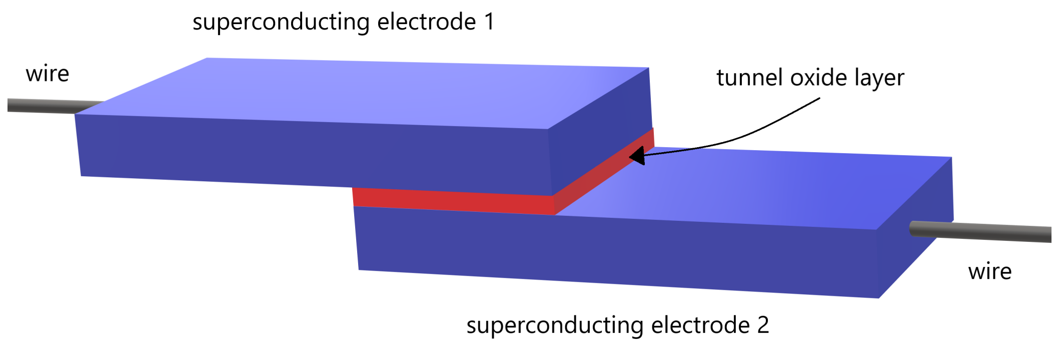


Figure 2.2: Representation of a Josephson junction. The blue plates represent superconducting metal films and the red plane represents an insulating oxide layer. This insulating layer allows discrete charges to tunnel between the superconducting elements, making the junction a nonlinear inductor.

components that are also non-dissipative. The only electronic device that is both nonlinear and non-dissipative at low temperature is the *superconducting tunnel junction* or *Josephson junction*. [34]. This device consists of a sandwich of two superconducting films separated by a thin insulating layer, as shown in Fig. 2.2. This barrier is thin enough to allow the tunneling of discrete charges between the superconducting elements. In particular, the tunneling of Cooper pairs creates an inductive path with strong nonlinearity. This is responsible for the creation of energy levels suitable for a qubit. This oxide layer is typically a $\sim$ 1 nm film of alumina (amorphous aluminum oxide).

The superconducting circuits that make up these types of qubits are fabricated using conventional integrated circuit techniques. They are fabricated on silicon wafers using optical or electron beam lithography and thin film deposition. Thus, Josephson junctions, capacitors, inductors, etc. are micron- or submicron-sized elements connected by wires or transmission lines.

All these elements give rise to quantum circuits that do not obey conventional circuit laws. In fact, collective electronic degrees of freedom such as current and voltage must be treated here as quantum operators that do not necessarily commute. For example, the charge, which in conventional circuits can be represented by a number in $\mathbb{R}$, is here represented by a wave function that gives the probability amplitude of all charge configurations and also the superposition states where the charge is both positive and negative at the same time. So these are devices that, although macroscopic, exhibit well-defined quantum behavior [37, 38].

2.2.1 Linear Quantum LC oscillator

We consider the case of a circuit consisting of an inductor L and a capacitor C in the superconducting regime. The behavior of this device is governed by the harmonic oscillator equation of motion. We take as the "position coordinate" the flux Φ in the inductor, defined as the integral of the voltage $V(t)$.

$$\Phi(t) = \int_{-\infty}^{t} V(t')dt'. \tag{2.1}$$

The "conjugate momentum" is instead the charge Q on the capacitor. Since we are in the quantum mechanical regime, Q and Φ are treated as canonically conjugate quantum operators obeying $[\Phi, Q] = i\hbar$.
The energy is:

$$E(t) = \int_{-\infty}^{t} V(t')I(t')dt'. \tag{2.2}$$

The Hamiltonian of this system is

$$H_{LC} = \frac{\Phi^2}{2L} + \frac{Q^2}{2C}. \tag{2.3}$$

The Hamiltonian can be rewritten in a more convenient form that introduces the *reduced flux* $\phi \doteq 2\pi\Phi/\Phi_0$ and the *reduced charge* $n \doteq Q/2e$, namely

$$H_{LC} = 4E_C n^2 + \frac{1}{2}E_L\phi^2, \tag{2.4}$$

where $E_C = e^2/(2C)$ and $E_L = (\Phi_0/2\pi)^2/L$ had to be defined. E_C represents the charging energy to add each electron of the Cooper pair to the island, while E_L is the inductive energy, where $\Phi_0 = h/(2e)$ is the *superconducting magnetic flux quantum.* The operator n represents the excess number of Cooper pairs on the island, and ϕ is usually called the "gauge-invariant phase" across the junction. Moreover, these two operators obey the commutation relation $[\phi, n] = i$.

Using the second quantization formalism, the Hamiltonian can be conveniently written as:

$$H_{LC} = \hbar\omega_0(a^\dagger a + 1/2), \tag{2.5}$$

where $a^\dagger(a)$ is the creation (annihilation) operator of a single excitation of the oscillator and $\omega_0 = 1/\sqrt{LC}$ is the resonance frequency of the oscillator.

The limit one encounters in using this device as a qubit is its linear nature. In fact, the quantum harmonic oscillator presents equidistant energy levels due to the parabolic shape of its potential. See panel (b) of Fig. 2.1. To obtain a qubit, the transition frequency between $|0\rangle$ and $|1\rangle$ must be sufficiently different from the transition frequency of higher eigenstates. In this way, the transition between the first two states can be precisely controlled, and leakage to higher states can be avoided. To obtain this result, we rely on the previously introduced Josephson junction, which, by replacing the linear inductor L, introduces a non-parabolic potential with non-equidistant levels. See panel (d) of Fig. 2.1.

2.2.2 Non-Linear Quantum LC Oscillator

The current $I(t)$ flowing through the linear inductor of the LC oscillator obey the law:

$$I(t) = \frac{1}{L}\Phi(t). \tag{2.6}$$

In the Josephson junction instead, the current has the following form:

$$I(t) = I_0 \sin[2\pi\Phi(t)/\Phi_0] \tag{2.7}$$

where I_0 is the critical current, a quantity that grows linearly with the tunnel layer area and diminishes exponentially with the tunnel layer thickness. This sinusoidal behavior comes from the inertia of Cooper pairs tunneling across the insulator.

Finally, the linear inductance of the LC circuit is replaced by a Josephson junction, which plays the role of a nonlinear inductor. A schematic representation of the two circuits is given in panels (a) and (c) of Fig.2.1. Now, using the expression for the current Eq.(2.7) in the formula for the energy Eq. (2.2), one can find the nonlinear version of the Hamiltonian in Eq. (2.4):

$$H_{nLC} = 4E_C n^2 - E_J \cos(\phi), \tag{2.8}$$

where $E_C = e^2/(2(C + C_J))$ now includes the Josephson junction self-capacitance, C_J, and $E_J = I_0\Phi_0/2\pi$ is the junction energy.

From this expression, it is clear that the energy potential of the system is no longer parabolic, as for the QHO, but sinusoidal. The two potentials are shown in panels (b) and (d) of Fig. 2.1. Panel (b) shows the typical energy landscape of a quantum LC circuit as a function of the gauge-invariant phase ϕ, while panel (d) shows the energy landscape of the quantum nonlinear LC oscillator. In the latter, the energy levels are no longer equidistant, so they can be addressed individually.

Transmon superconducting qubit

The dynamics of the system is now governed by the relative value of E_C and E_J in Eq.(2.8) or, equivalently, by their ratio E_C/E_J. This ratio can be varied to obtain different behaviors and qubits. In general, in the case $E_J \leq E_C$, the device becomes very sensitive to charge noise. So-called "charge qubit" or "Cooper-pair box" qubits fall into this case [39, 40, 41, 42]. In this work, instead, we focus on the most used $E_J >> E_C$ regime. This is also called *transmon regime.* In this limit, the quantum fluctuations of the superconducting phase ϕ are small, so it can be used to encode quantum information and use this device as a qubit.

The potential energy in Eq. (2.8) can be Taylor expanded to obtain:

$$E_J \cos(\phi) = \frac{1}{2}E_J\phi^2 - \frac{1}{24}E_J\phi^4 + \mathcal{O}(\phi^6). \tag{2.9}$$

The quartic term on the right hand side is responsible for disrupting the harmonic energy structure given by the quadratic term. We define *anharmonicity* as

$$\alpha = \omega_{12} - \omega_{01}, \tag{2.10}$$

where $\omega_{01}(\omega_{12})$ is the transition frequency between the eigenstates $|0\rangle$ and $|1\rangle$ ($|1\rangle$ and $|2\rangle$). This value indicates how different the energies of the two states are. In addition, the negativity of the quartic term of eq.(2.8) shows that the anharmonicity α is negative, so careful fabrication of the device is required to obtain a usable device. In the case of transmons, the anharmonicity is usually set to $\alpha = -E_C \sim 100 - 300$ MHz so that the qubit frequency can be kept in the considerable range $\omega = (\sqrt{8E_JE_C} - E_C)/\tilde{\sim}3 - 6$ GHZ, while the ratio $E_J/E_C \geq 50$ is chosen to be stable in the transmon regime.

Substituting the Taylor expanded potential of Eq. (2.9) in the Hamiltonian (2.8) and passing to the second quantization formalism we can rewrite the Hamiltonian as:

$$H_{tr} = \omega_{01}a^\dagger a + \frac{\alpha}{2}a^\dagger a(a^\dagger a + 1). \tag{2.11}$$

For a complete derivation of this equation see Appendix A.

If the higher levels of the transmon are suppressed due to large $|\alpha|$ or by approrpiate optimal control protocols, this system can be further simplified as:

$$H_{tr} = \omega_{01}\frac{\sigma_z}{2}, \tag{2.12}$$

where σ_z is the Z Pauli matrix. Despite this, these additional layers can actually be used to create qubits with multiple layers, so-called **qudits**, which have been proven to be useful for implementing operations efficiently [43, 44, 45]. In this work the qudits will be extensively used.

2.2.3 Qubits Coupling

To truly exploit the properties that distinguish the quantum computer from its classical counterpart, i.e. entanglement, it is necessary to couple the qubits. In this section, we will give an insight into the capacitive couplings between qubits and between qubits and resonators that are necessary to perform readout.

A general Hamiltonian of two coupled qubits is

$$H = H_{q_1} + H_{q_2} + H_{int}, \tag{2.13}$$

where H_{q_i} is the i^{th} qubit Hamiltonian and H_{int} is the interaction Hamiltonian describing how the qubit variables are connected. In superconducting qubits, the coupling is physically implemented by an electric or magnetic field. The coupling can be capacitive or inductive. In this dissertation, we will focus on the capacitive coupling. To achieve simple capacitive coupling, the two qubits are connected via a capacitor of capacitance C_g, as shown in Fig. 2.3 (a). In this case, the interaction Hamiltonian can be written as

$$H_{int} = C_g V_1 V_2, \tag{2.14}$$

where V_i is the voltage operator of the i^{th} voltage node to be connected. See Fig. 2.3 (a) for a schematic representation.

Taking $V_i = (2e/C_i)n_i$, where C_i are the qubits capacitance, $n_i = Q_i/2e$ is the reduced charge of the ith qubit, and assuming to work in the limit of $C_g \ll C_1, C_2$, we can write the explicit Hamiltonian of the two-qubit coupled system as:

$$H = \sum_{i=1,2} \left[4E_{C,i}n_i^2 - E_{J,i}\cos(\phi_i)\right] + 4e^2\frac{C_g}{C_1C_2}n_1n_2, \tag{2.15}$$

where the first term on the right-hand side is the sum of the Hamiltonian of the individual qubits [See. Eq. (2.8)]. The capacitance needs to be carefully designed to obtain the desired performances [46].

We can rewrite Eq. (2.15) in the second quantization form:

$$H = \sum_{i=1,2} \left[\omega_i a_i^\dagger a_i + \frac{\alpha_i}{2} a_i^\dagger a_i (a_i^\dagger a_i + 1)\right] - g(a_1 - a_1^\dagger)(a_2 - a_2^\dagger). \tag{2.16}$$

See Appendix A for more details. In addition to the direct coupling just presented, superconducting qubits are often coupled by a device that behaves like a quantum harmonic oscillator. These devices, often called *resonator* or *cavity*, have many applications, the most important of which is high-fidelity qubit readout [see section 2.4 and section 6]. But they also serve as cavity buses [47], quantum memory [48, 49], quantum error correction [50], and others. The whole system can be

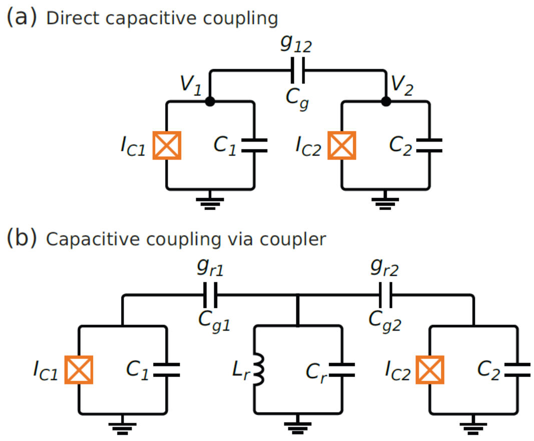


Figure 2.3: Example of qubits couplings. (*a*) Direct capacitive coupling between two qubits. (*b*) Capacitive coupling of two qubits via a coupler. The qubits coupling is enforced by capacitively coupling them to the same LC circuit (cavity or resonator). Figure taken from [32]

described by the cavity quantum electrodynamics (cQED) formalism, a branch of quantum mechanics that studies the interaction between electromagnetic radiation and matter at the quantum level [51, 52, 53]. Figure 2.3 (b) shows a schematic representation of this type of coupling, where two qubits are connected to the same cavity. Its Hamiltonian is

$$\begin{aligned} H &= \sum_{i=1,2}\left[\omega_i a_i^\dagger a_i + \frac{\alpha_i}{2} a_i^\dagger a_i (a_i^\dagger a_i + 1)\right] + \omega_r a_r^\dagger a_r \\ &+ g_{1r}(a_1^\dagger a_r + a_1 a_r^\dagger) + g_{2r}(a_2^\dagger a_r + a_2 a_r^\dagger), \end{aligned} \tag{2.17}$$

where $a_r^\dagger (a_r)$ is the resonator creation (annihilation) operator, ω_r is the resonator frequency, and g_{ir} is the coupling constant between the i^{th} qubit and the resonator.

Clearly, the coupling can be extended to an arbitrary number of qubits and cavities. This will introduce a particular topology in the connectivity, as it is not possible to create an all-to-all coupling. When performing calculations with such devices, this topology must be taken into account.

2.2.4 Noise

So far we have treated qubits as perfect, isolated systems. Of course, this is not true in reality. Qubit states are very fragile systems and suffer from many unintended environmental interactions. These interactions take the form of *noise*, and their effect is generally to reduce the coherence of the qubit and degrade computational performance. These disturbances can be deterministic, resulting from errors in the

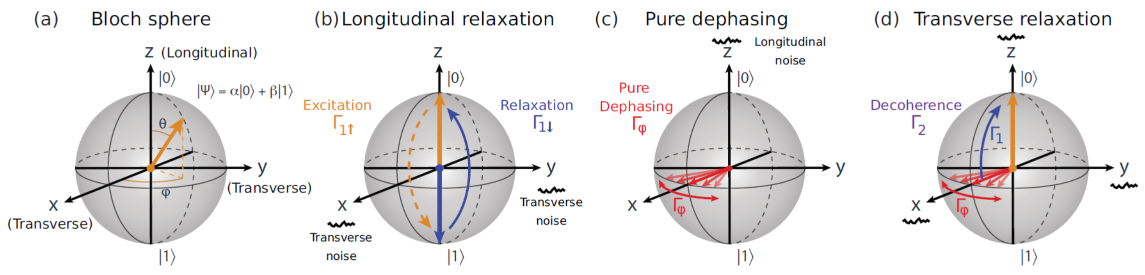


Figure 2.4: Effect of different types of noise on the qubit. (*a*) Bloch sphere representation of a single qubit state space. (*b*) Effect of longitudinal relaxation on the Bloch sphere. (*c*) Effect of pure dephasing on the Bloch sphere. (*d*) Effect of transverse relaxation on the Bloch sphere. Figure taken from [32]

control or measurement system, or stochastic, resulting from random fluctuations in the device components due to imperfections or temperature. In an attempt to reduce this noise, one can act on the one hand by improving the materials and manufacturing of the device, thus reducing the amount of noise produced, and on the other hand on the design to obtain a device less susceptible to noise. It will often be necessary to work within a trade-off between noise reduction and susceptibility to that noise.

Modelling noise

To schematize the influence of noise on the qubit, we use the Bloch sphere representation introduced in Sec. 1.3. We recall that in this representation the state of a qubit is written as a vector pointing to a surface of the unit sphere:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle\,. \tag{2.18}$$

Referring to the Fig. 1.1 and 2.4, the z-axis is the *longitudinal axis* and represents the quantization axis of the qubit for the two states $|0\rangle$ and $|1\rangle$ in the eigenbasis of the qubit. The x-y plane, on the other hand, is the *transverse plane*. The state $|\psi\rangle$ rotates around the longitudinal axis at the qubit frequency ω, so to simplify the visualization we consider the Bloch sphere to rotate around the z-axis at the qubit frequency ω. In this way, the state in the Bloch sphere is stationary, as written in Eq.(2.18). The state can also be rewritten in the form of a density matrix, which takes a particularly simple form for the case of a single qubit:

$$\begin{aligned}\rho &= |\psi\rangle\langle\psi| = (\alpha\,|0\rangle + \beta\,|1\rangle)(\alpha\,\langle 0| + \beta\,\langle 1|)\\ &= \begin{pmatrix}|\alpha|^2 & \alpha\beta^*\\ \alpha^*\beta & |\beta|^2\end{pmatrix},\end{aligned} \tag{2.19}$$

where I is the identity matrix and σ_i with $i = x, y, z$ the pauli matrices.

Three effects of noise on the qubit can be schematized: longitudinal relaxation, pure dephasing, and transverse relaxation. These will be discussed separately in the following paragraphs.

Longitudinal relaxation Longitudinal relaxation, often called "energy decay" or "energy relaxation", is the depolarization along the qubit quantization axis. Thus, it consists of the decay from $|1\rangle$ to $|0\rangle$ or the excitation in the opposite direction. See Fig. 2.4(b) for a schematic representation. The rate of this process is:

$$\Gamma_1 = \frac{1}{T_1} = \frac{1}{T_1^{1\to 0}} + \frac{1}{T_1^{0\to 1}}, \tag{2.20}$$

where $T_1^{1\to 0}$ ($T_1^{0\to 1}$) are the typical exitation times. Due to the low-temperature operating conditions of superconducting qubits, the exitation process is exponentially suppressed by the Boltzmann distribution, in fact $T_1^{0\to 1} = T_1^{1\to 0} e^{-\frac{\omega_{01}}{k_B T}}$ where $\omega_{01} = E_1 - E_0$ is the qubit frequency and T is the temperature (usually in the range $T \sim 2 - 20$mK). From this expression, it is clear that in the depolarization process the rate $1/T_1^{0\to 1}$ is negligible.

The measurement of T_1 is obtained by preparing the qubit in state $|1\rangle$ and sampling the probability of finding it still in state $|1\rangle$ as a function of time. The resulting exponential decay will have a characteristic time T_1 i.e. the time for which the qubit's energy level decays to 63% of its initial value.

Pure dephasing It describes the depolarization in the x-y plane of the Bloch sphere. It leads to a fluctuation of the qubit frequency ω_{01} so that it is no longer equal to the rotation frequency of the Bloch sphere. Thus, the state acquires a precession frequency. In other words, this process explains the loss of the relative phase between the $|0\rangle$ and $|1\rangle$ states. Fig.2.4(c) shows the effect of this process on the Bloch sphere. A state $|x\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, a vector lying on the x-y plane and perpendicular to the z-axis, is randomly rotated by the dephasing noise and after certain time results in a cloud of states spread around the equatorial plane of the Bloch sphere. The relative phase between $|0\rangle$ and $|1\rangle$ is thus completely lost. This is represented by the angle ϕ in the Bloch sphere.

Unlike longitudinal relaxation, pure dephasing is not a resonant phenomenon, so it can be triggered by any noise frequency. Moreover, pure dephasing is not a process involving energy exchange with the environment, so it is in principle reversible. This means that dephasing can be fixed by applying a unitary operation [54].

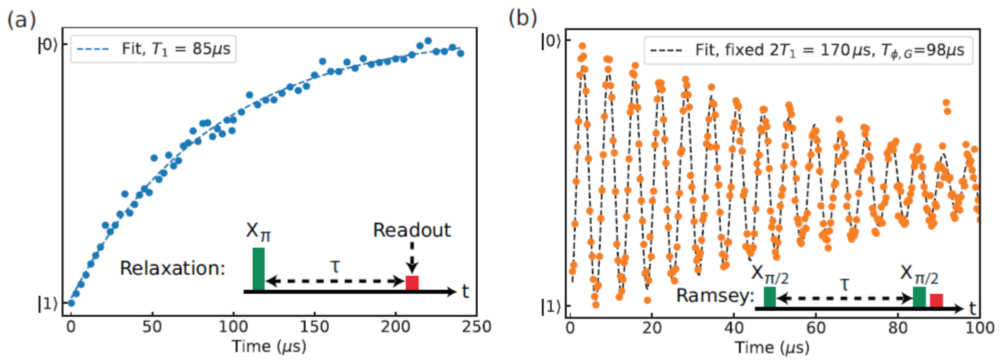


Figure 2.5: Characterization of longitudinal (T_1) and transverse (T_2) relaxation times of a transmon qubit. (a) Measurement of longitudinal relaxation (energy relaxation). The qubit is excited with a $\pi-$pulse and measured after a waiting time τ. For each value, τ, this procedure is repeated to obtain the blue point series. From this series the characteristic decay time T_1 can easily be extracted. (b) Measurement of transverse relaxation (decoherence) by Ramsey interferometry. The qubit is prepared at the equator with a $\pi/2$ pulse that is intentionally detuned from the qubit frequency by $\delta\omega$, causing the Bloch vector to precess in the rotating frame at a rate of $\delta\omega$ around the z-axis. After a time τ, a second $\pi/2$ pulse projects the Bloch vector back to the z-axis, effectively mapping its former position on the equator to a position on the z-axis. The oscillations decay with an approximately (but not exactly) exponential decay function, from which the time T_2 can be derived. Figures taken from [32]

Transverse relaxation Transverse relaxation is the union of the two mechanisms just introduced. It generally describes the loss of coherence of a superposition state. An example is shown in figure 2.4(d). The vector $|\psi\rangle$ fluctuates around the equator due to pure dephasing and tends to decay to the ground state due to longitudinal relaxation. This is a phase-breaking process because once the state has decayed to $|0\rangle$ and no information about the direction it was pointing before is retained, the relative phase of the superposition state is lost. The rate of this process is:

$$\Gamma_2 = \frac{1}{T_2} \tag{2.21}$$

where T_2 is the typical transverse relaxation time. T_2 can be measured by Ramsey interferometry. A superposition state pointing to the equator is prepared with an $X_{\pi/2}$ pulse. The carrier frequency of the pulse is detuned from the qubit frequency by a small amount $\delta\omega$. This causes the vector to precess around the equator at a rate of $\delta\omega$. After a time τ another $X_{\pi/2}$ pulse is applied. This projects the vector back to the z-axis, then the state is measured. The result for increasing time τ is an exponentially decaying oscillatory curve. The typical decay time is T_2. A representation of this measurement is shown in figure Fig.2.5(b)

2.2.5 Effect of noise on the qubit: the Bloch-Redfield model

Considering the standard Bloch-Redfield model [55, 56] for noise, the "noise-less" density matrix of Eq. 2.19 can be modified to describe the effect of the noise on the qubit. This results in the Bloch-Redfield density matrix ρ_{BR}:

$$\rho_{BR} = \begin{pmatrix} 1 + (|\alpha|^2 - 1)e^{-\Gamma_1 t} & \alpha\beta^* e^{i\delta\omega t} e^{-\Gamma_2 t} \\ \alpha^* \beta e^{-i\delta\omega t} e^{-\Gamma_2 t} & |\beta|^2 e^{-\Gamma_1 t} \end{pmatrix}. \tag{2.22}$$

This expression incorporates the longitudinal decay function $e^{-\Gamma_1 t}$, which accounts for longitudinal relaxation of the qubit, the transverse decay function $e^{-\Gamma_2 t}$, which accounts for transverse decay of the qubit and, in addition, an explicit phase accrual $e^{-i\delta\omega t}$, where $\omega = \omega_q - \omega_d$, which generalizes the Bloch sphere picture to account for cases where the qubit frequency ω_q differs from the rotating-frame frequency ω_d.

2.3 Qubit Control

In this section, it will be described how to actually manipulate the qubits in order to implement the desired unitary transformations [57].

Physically, this control is implemented through capacitive coupling between a resonator, or feedline, and the superconducting qubit dipole-field that allows for microwave pulses that implement single-qubit rotations. The qubits are typically designed to respond to microwave signals at specific frequencies, which can be tuned by applying a magnetic field or adjusting the physical parameters of the qubit.

The quantum processor state $|\psi(t)\rangle$ dynamics can be described by the usual Schrödinger equation:

$$|\dot{\psi}(t)\rangle = -iH(u(t))\,|\psi(t)\rangle \tag{2.23}$$

where $\hbar = 1$. Here $H(u(t))$ is the Hamiltonian of the device which depends on a set of control functions $u(t) = u_1(t), u_2(t), ..., u_n(t)$. One can then control the dynamics of the state $|\psi(t)\rangle$ with an appropriate choice of these control functions. The Schrödinger equation (2.23) can be formally solved as:

$$\begin{aligned} |\psi(t)\rangle &= e^{-i\int_0^t H(u(t))dt}\,|\psi(0)\rangle \\ &= U(u(t),t)\,|\psi(0)\rangle \end{aligned} \tag{2.24}$$

where the whole exponential is the *propagator* of the system and is denoted by $U(u(t),t)$. Since H_0 and H_k are Hermitian operators, it follows that $U(t)$ is a unitary operator. In matrix representation, it is an nxn matrix, an element belonging to the Lie group $U(n)$ (or $SU(n)$ for traceless Hamiltonians). Now, given Eq.(2.24), Eq. (2.23) can be rewritten as:

$$\frac{d}{dt}U(u(t),t) = -iH(u(t))U(u(t),t) \tag{2.25}$$

$$U(u(0),0) = \mathbb{I} \tag{2.26}$$

where $\mathbb{I}$ is the identity.

The **reachable set** at time $T > 0$ for system (2.25) is the set $R(T)$ of all unitary matrices $\tilde{U}$ such that there exists a set of controls $u(t) = [u_1(t), u_2(t), ..., u_k(t)]$ in some function space F for which $U(u(T),T) = \tilde{U}$. If we consider all possible time T the reachable set is $R = \bigcup_{T>0} R(T)$. We can state the **controllability condition** in the following way:
Theorem The reachable set R of the system (2.25) is the connected Lie group associated to the Lie algebra $\mathcal{L}$ generated by $\text{span}_{u(t)\in F}\{-iH(u(t))\}$. In short, $R = e^{\mathcal{L}}$.

The Lie algebra $\mathcal{L}$ is the **dynamical Lie algebra** associated with the system. $\mathcal{L}$ is always a subalgebra of $\mathfrak{u}(n)$ [57]. If $dim(\mathcal{L}) = n^2 = \dim(\mathfrak{u}(n))$, it holds $\mathcal{L} = \mathfrak{u}(n)$ and $e^{\mathcal{L}} = R = U(n)$. In particular, the system is said to be **controllable**. This means that every unitary matrix $U(t)$, and ultimately every state $|\psi\rangle$ in Eq. (2.23), can be reached with an appropriate set of controls $u(t)$.

The optimal control problem consists of finding the best shapes of controls $u_i(t)$ such that they transform the system's state in the desired way.

Controllability of state

Having identified the possible $U(t)$ transformations, we are now interested in understanding in which cases these transformations allow the quantum states of the qubits to be manipulated in a desired way.

For this purpose we return to the Schrödinger equation (2.23),

$$|\dot{\psi}(t)\rangle = -iH(u(t))\,|\psi(t)\rangle\,. \tag{2.27}$$

This equation is said to be **pure state controllable** if for each pair of initial and final states, $|\psi(0)\rangle$ and $|\psi(T)\rangle$, there exists a control $u(t)$ such that the solution of Eq. (2.27) at time T, with initial condition $|\psi(0)\rangle$, is $|\psi(T)\rangle$.

Since $|\psi(t)\rangle$ and $|\psi(t)\rangle\,e^{i\phi}$, where ϕ is a phase in $\mathbb{R}$, represent the same physical state, another equivalent condition for controllability can be defined. The system (2.27) is said to be **equivalent state controllable** if, for each pair of initial and final states $|\psi(0)\rangle$ and $|\psi(T)\rangle$, there exists a control $u(t)$ and a phase factor ϕ such that the solution of Eq. (2.27) at time T, with the initial condition $|\psi(0)\rangle$, is $e^{i\phi}|\psi(T)\rangle$.

As anticipated, the solution of Eq.(2.27) is:

$$|\psi(t)\rangle = U(t)\,|\psi(0)\rangle\,.$$

Since $U(t) \in e^{\mathcal{L}}$, i.e. the Lie group identified by the "dynamical Lie algebra" associated to the system, the set of states $|\psi(t)\rangle$ the can be the solution of Eq.(2.27) with initial condition $|\psi(0)\rangle$ is:

$$\mathcal{O}_{|\psi(0)\rangle} \doteq \{U(t)\,|\psi(0)\rangle\,|U(t) \in e^{\mathcal{L}}\}. \tag{2.28}$$

From this point of view, the system is pure state controllable if, for each pair of initial and final states $|\psi(0)\rangle$ and $|\psi(T)\rangle$, there exists an operator $U(t) \in e^{\mathcal{L}}$ such that $|\psi(T)\rangle = U(t)\,|\psi(0)\rangle$. Thus, the conditions allowing the "pure state controllability" of a system can be derived from the analysis of the properties of $e^{\mathcal{L}}$ acting as a **Lie transformation group** on the space of $|\psi(t)\rangle$ states.

Test for controllability

Referring again to [57], it can be stated that the system (2.27) is pure state controllable if $\mathcal{L}$ is either $\mathfrak{su}(n)$ or a Lie algebra conjugate to $\mathfrak{sp}(n/2)$ (i.e. the *symplectic Lie algebra*). The following theorem holds:

Theorem A quantum system (2.27) is pure state controllable if and only if the corresponding dynamical Lie algebra $\mathcal{L}$ satisfies one of the following:

1. $\mathcal{L} = \mathfrak{su}(n)$
2. $\mathcal{L}$ is conjugate to $\mathfrak{sp}(n/2)$
3. $\mathcal{L} = \mathfrak{u}(n)$
4. $\mathcal{L} = \mathrm{span}\{i\mathbb{I}_{n\times n}\} \oplus \tilde{\mathcal{L}}$, with $\tilde{\mathcal{L}}$ the Lie algebra conjugate to $\mathfrak{sp}(n/2)$.

Finally, we consider equivalent state controllability. From a physical point of view, having equivalent state controllability is equivalent to pure state controllability. It turns out that mathematically the conditions are the same.

2.3.1 Optimal control problem

Now we can state the optimal control problem for quantum systems in general and specialize it for qubit manipulation.

The dynamics of the system is described by the controlled Schrodinger equation (2.25). The optimal control problem can be formulated as follows. Given an initial and a final state, $|\psi_0\rangle$ and $|\psi_f\rangle$, find the set of controls $u(t) = u_1(t), u_2(t), ...$ in a set F of functions such that a cost functional J is minimized and Eq. (2.25) is satisfied.

The general formulation of the cost functional J is [58]:

$$J[u(t)] = |\langle\psi_f|\,\mathcal{T}\exp\left\{-i\int_0^T H(u(t'))dt'\right\}|\psi_0\rangle|^2 + f_c(u(t)), \tag{2.29}$$

where $\mathcal{T}$ stands for time-ordered integral, since we are considering a quantum system. The first term imposes to find the controls $u(t)$ that minimize the distance between the desired final state $|\phi_f\rangle$ and the one obtained by the controls at time T, i.e. $|\psi(T)\rangle$. The second term, the function $f_c(u(t))$, on the other hand, is a penalty function that enforces any physical and instrumental constraints of the device.

Another specific cost function, also used in this work, exploits the propagators $U(t)$ of the system, see Eq.(2.28), instead of the states. The idea is to find the controls $u(t)$ such that the propagator $U(u(t))$ defined by these controls is similar to a desired propagator U_T [59]. A reasonable way to implement this constraint is to minimize a properly defined distance between the two propagators. An appropriate distance is the square of the Hilbert-Schmidt norm $||O||^2_{HS} = \mathrm{Tr}\big[O^\dagger O\big]$. So the distance is:

$$D = ||U(u(t)) - U_T||^2_{HS} = 2N - 2\,\mathrm{Re}\,\mathrm{Tr}\big[U(u(t))^\dagger U_T\big], \tag{2.30}$$

where N is the dimension of the $SU(N)$ group in which the propagators take values, or equivalently, the dimension of their matrix form. In order to have a minimum of the norm at 0 instead of $4N$, D is normalized, i.e:

$$\tilde{D} = \frac{1}{4N}D = \frac{1}{2} - \frac{1}{2N}\,\mathrm{Re}\,\mathrm{Tr}\big[U(u(t))^\dagger U_T\big], \tag{2.31}$$

which now attains values in $[0,1]$.
Often in quantum optimal control, instead of minimizing the distance $\tilde{D}$, the **fidelity** is maximized, i.e.:

$$\mathcal{F} = 1 - \tilde{D} = \frac{1}{2} + \frac{1}{2N}\,\mathrm{Re}\,\mathrm{Tr}\big[U(u(t))^\dagger U_T\big], \tag{2.32}$$

such that if $U(u(t)) \equiv U_T$, $\mathcal{F} = 1$.

2.3.2 Minimization algorithms

To actually find the optimal control, one can rely on the discretization of the cost function and then apply standard optimization methods such as Steepest Grandient Descent (SGD). In this thesis, we introduce the method of GRadient Ascent Pulse Engineering, GRAPE for short [58, 60, 59]. The Hamiltonian in $H(u(t))$ of (2.23) has the general form:

$$H(u(t)) = H_0 + \sum_{k=1}^{N_{ctrl}} \epsilon_k(t) H_k, \tag{2.33}$$

where H_0 represents the general Hamiltonian of the quantum processor, which depends on the qubit properties, the qubit couplings, and the general connectivity between the qubits. The set of H_k matrices are the external drive Hamiltonians schematizing the external interaction with the system. These are also device-specific. Finally, $\epsilon_k(t)$ are a set of N_{ctrl} functions describing how the strength of each H_k on the system changes as a function of time. The number N_{ctrl} of such controls depends on the specific hardware or connectivity between qubits.

Plugging Eq. (2.33) into the propagator explicit expression (2.24), it becomes:

$$U(\epsilon(t), t) = \exp\left\{ -i \int_0^T H_0 + \sum_{k=1}^{N_{ctr}} \epsilon_k(\tau) H_k d\tau \right\}. \tag{2.34}$$

The integral can be broken up into adequately short chunks of duration $\Delta\tau$ and the propagator $U(\epsilon(t), t)$ can be written as a multiplication of individual matrices each of duration $\Delta\tau$:

$$\begin{aligned} U &= \exp\left\{ -i\Delta\tau \sum_{j=1}^{N_T} \left[H_0 + \sum_{k=1}^{N_{ctr}} \epsilon_k(j) H_k \right] \right\} && (2.35) \\ &= U_{N_T} U_{N_{T-1}} ... U_2 U_1, && (2.36) \end{aligned}$$

with

$$U_n = \exp\left\{ -i\Delta\tau \left[H_0 + \sum_{k=1}^{N_{ctr}} \epsilon_k(n) H_k \right] \right\} \tag{2.37}$$

and $N_T = T/\Delta\tau$ is the number of time steps the pulse duration T is divided into. To sketch how this works, consider the special case of a cost function defined by $J[\epsilon(t)] = \tilde{\mathcal{D}}$. We simply use the gradient descent algorithm to minimize $J[\epsilon(t)]$, with respect to $\epsilon(t)$. The discrete version of the cost function is

$$\begin{aligned} J[\epsilon(t)] &= \frac{1}{2} - \frac{1}{2N} \operatorname{Re} \operatorname{Tr} \left[U_{targ}^{\dagger} U(\epsilon(t)) \right] && (2.38) \\ &= \frac{1}{2} - \frac{1}{2N} \operatorname{Re} \operatorname{Tr} \left[U_{targ}^{\dagger} (U_{N_T} U_{N_T - 1} ... U_2 U_1) \right], && (2.39) \end{aligned}$$

where we used the idenity (2.35). We compute the derivative of of J with respect to each control $\epsilon_k(t_j)$ and for each time t_j as follow:

$$\frac{\delta J}{\delta \epsilon_k(t_j)} = -2 \operatorname{Re} \operatorname{Tr} \left[U^{\dagger}_{targ} (U_{N_T} U_{N_T-1} ... \frac{\delta U_j}{\delta \epsilon_k(t_j)} ... U_2 U_1) \right]. \tag{2.40}$$

A short calculation returns that

$$\frac{\delta U_k}{\delta \epsilon_k(t_j)} = -i \Delta \tau H_k U_j, \tag{2.41}$$

so we can simply use the usual gradient descend rule to update the value of $\epsilon_k(t_j)$ as:

$$\epsilon_k(t_j) \leftarrow \epsilon_k(t_j) - \eta \frac{\delta J}{\delta \epsilon_k(t_j)}, \tag{2.42}$$

with η a numerical parameter to improve the convergence of the algorithm.

Many other optimization algorithms have been developed as Krotov Method[61], D-MORPH (diffeomorphic modulation under observable-response-preserving homotopy) [62] or CRAB (chopped random basis method)[63].

2.4 Qubit Readout

After implementing the state transformation of the qubits, a reliable and fast readout is important to actually obtain the results of the computation. This section briefly introduces the most common readout framework for superconducting qubits, namely dispersive readout. This technique consists of coupling qubits to a readout cavity/resonator. By setting a resonator frequency detuned from that of the qubit, i.e. working in the *dispersive regime*, the qubit induces a state-dependent frequency shift of the resonator, from which the state of the qubit can be inferred by reading the resonator output signal. In the dispersive readout, the state of the qubit is found by a classical response of the linear resonator. Therefore, when designing and optimizing the readout process, it is necessary to obtain the best signal-to-noise ratio of the microwave signal used to probe the resonator.

2.4.1 Dispersive readout

Instead of interacting directly with the qubit, the readout works by entangling the qubit with a probe (i.e., the cavity observable) and then inferring the state of the qubit by analyzing the output of the signal used to probe the cavity. Thus, there are two conflicting processes to be optimized in the readout process. On the one hand, maximizing the signal-to-noise ratio (SNR) of the microwave signal sent into the cavity and, on the other hand, minimizing the unwanted feedback from the cavity to the qubit.

From Eq.(2.17) we know how to write the Hamiltonian of a qubit coupled with a cavity:

$$\begin{aligned} H &= \omega_q a_q^\dagger a_q + \frac{\alpha_q}{2} a_q^\dagger a_q (a_q^\dagger a_q + 1) + \omega_r a_r^\dagger a_r \\ &\quad + g_r (a_q^\dagger a_r + a_q a_r^\dagger), \end{aligned} \tag{2.43}$$

where ω_q and α_q are the frequency and the anarmonicity of the qubit, ω_r is the cavity frequency and g_r is the coupling constant to the cavity. The dispersive regime consists of a large detuning between qubit and cavity with respect to the coupling constant, i.e. $\Delta = |\omega_q - \omega_r| \ll g_r$. In this case, there is no direct energy exchange between the two systems, and the qubit and the cavity push each other's frequencies. To understand this behavior, we find an approximation of Eq.(2.43) that is valid in the dispersive regime. We perform a Shrieffler-Wolff transformation to the second order [64] yielding:

$$\begin{aligned} H_{disp} &\simeq \hbar\omega_r a_r^\dagger a_r + \hbar\omega_q a_q^\dagger a_q + \alpha a_q^\dagger a_q^\dagger a_q a_q \\ &\quad + \sum_{j=0}^{\infty} \hbar(\Lambda_j + \chi_j a_r^\dagger a_r) |j\rangle \langle j| , \end{aligned} \tag{2.44}$$

where $|j\rangle$ are the eigenstates of the transmon and:

$$\Lambda_j = \chi_{j-1,j}, \quad \chi_j = \chi_{j-1,j} - \chi_{j,j+1} \tag{2.45}$$

$$\chi_{j-1,j} = \frac{jg^2}{\Delta - (j-1)E_C/\hbar}, \tag{2.46}$$

for $j > 0$ and with $\Lambda_0 = 0$ and $\chi_0 = -g^2/\Delta$. The constants χ_j are called the *dispersive shifts* and Λ_j are the *Lamb shifts*. They are the results of vacuum fluctuations [65, 66, 67]

Readout

As mentioned earlier, qubit measurement consists of studying the difference in amplitude and phase that a readout signal acquires when interacting with the qubit. A short microwave tone is sent into the resonator with a *carrier frequency* ω_{RO}. As the result of the interaction with the qubit, the output signal will acquire a phase θ_{RO} and an amplitude A_{RO}, resulting in:

$$s(t) = A_{RO} \cos(\omega_{RO} t + \theta_{RO}). \tag{2.47}$$

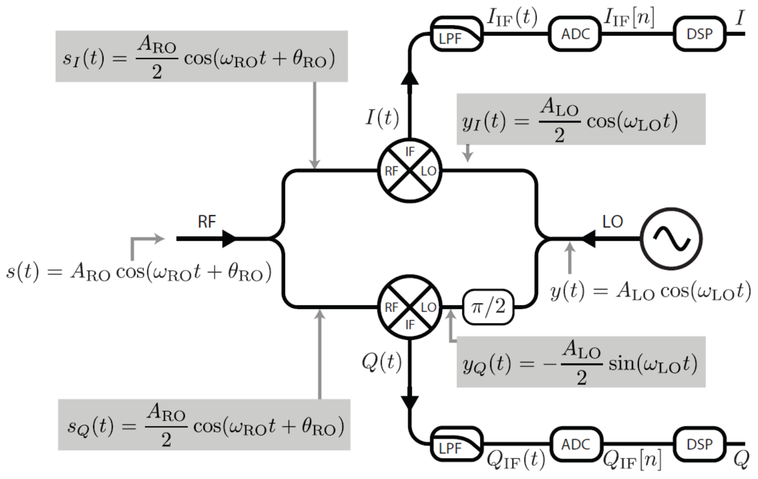

$s_I(t) = \frac{A_{\mathrm{RO}}}{2}\cos(\omega_{\mathrm{RO}}t + \theta_{\mathrm{RO}})$
$I_{\mathrm{IF}}(t)$
LPF
ADC
$I_{\mathrm{IF}}[n]$
DSP
I
$I(t)$
$y_I(t) = \frac{A_{\mathrm{LO}}}{2}\cos(\omega_{\mathrm{LO}}t)$
IF
RF
LO
LO
$s(t) = A_{\mathrm{RO}}\cos(\omega_{\mathrm{RO}}t + \theta_{\mathrm{RO}})$
$\pi/2$
$y(t) = A_{\mathrm{LO}}\cos(\omega_{\mathrm{LO}}t)$
$Q(t)$
$y_Q(t) = -\frac{A_{\mathrm{LO}}}{2}\sin(\omega_{\mathrm{LO}}t)$
$s_Q(t) = \frac{A_{\mathrm{RO}}}{2}\cos(\omega_{\mathrm{RO}}t + \theta_{\mathrm{RO}})$
$Q_{\mathrm{IF}}(t)$
$Q_{\mathrm{IF}}[n]$
Q

Figure 2.6: Schematic representation of In-phase and Quadrature extraction from a qubit readout signal. The figure is taken from [32].

A_{RO} and θ_{RO} are the qubit-state dependent values one needs to measure to deduce in which state the qubit is.

To measure these differences, we rely on calculating the "in-phase", I, and "quadrature", Q, components of the signal $s(t)$. *Homodyne* and *heterodyne* measurements are techniques for finding these components using an analog I-Q mixer. A schematic representation of this process is shown in Fig.2.6. The mixer has a reference local oscillator with a signal $y(t) = A_{LO}\cos(\omega_{LO}t)$. This reference signal and the readout signal $s(t)$ are fed into a mixer via RF and LO mixer ports. The mixer splits both signals into two branches and multiplies them. The I branch consists of the multiplication of the signals $s_I(t) = s(t)/2$ and $y_I(t) = (A_{LO}/2)\cos(\omega_{LO}t)$, while the Q branch considers the signal $s_Q(t) = s(t)/2$ multiplied for a $\pi/2$ phase shifted reference signal $y_Q(t) = -(A_{LO}/2)\sin(\omega_{LO}t)$.

The signals at the I and Q ports represent a frequency that is the sum and difference of the original frequencies. This frequency is usually called *intermediate frequency* $\omega_{IF} = \omega_{RO} \pm \omega_{LO}$. A low pass filter allows to keep only the difference frequency $\omega_{IF} = \omega_{RO} - \omega_{LO}$. The signals are then digitized to obtain the discrete signals $I_{IF}[t]$ and $Q_{IF}[t]$. From the analysis of these signals the state of the qubit can be deduced.

Homodyne demodulation

Homodyne measurement is a way to extract I and Q values from the output signal. In this approach, the local oscillator frequency is chosen to be equal to the carrier frequency, i.e., $\omega_{LO} = \omega_{RO}$. This choice results in $I(t)$ and $Q(t)$ signals that contain zero frequency terms, since $\omega_{IF} = \omega_{LO} - \omega_{RO} = 0$, and twice the carrier frequency, since $\omega_{IF} = \omega_{LO} + \omega_{RO} = 2\omega_{RO}$. Now we perform a demodulation of the signals, which acts as a filter. This consists of a time average of the $I_{IF}(t)$ and $Q_{IF}(t)$.

$$\begin{aligned} I &= \frac{1}{T}\int_0^T s_I(t)y_I(t)dt \\ &= \frac{A_{RO}A_{LO}}{8}\cos(\theta_{RO}), \end{aligned} \tag{2.48}$$

$$\begin{aligned} Q &= \frac{1}{T}\int_0^T s_Q(t)y_Q(t)dt \\ &= \frac{A_{RO}A_{LO}}{8}\sin(\theta_{RO}), \end{aligned} \tag{2.49}$$

where T is the time interval on which the measurement is performed. This time must span an integer number of signal periods to avoid unintended effects in demodulation. The values of I and Q thus obtained can be used to find the amplitude and phase of the output signal:

$$A_{RO} \propto \sqrt{I^2 + Q^2}, \tag{2.50}$$

$$\theta_{RO} = \arctan(Q/I). \tag{2.51}$$

A more immediate use of the I and Q values is to plot them on a plane. In this way, the pairs (I-Q) of different states of the qubit are statistically clustered in different regions of the graph. From the study of this distribution, the state of the qubits of the following measurements can be inferred.

Heterodyne demodulation

In this second approach, the local oscillator frequency is set to a value different from the carrier frequency, i.e. $\omega_{IF} = |\omega_{RO} - \omega_{LO}| > 0$. The signals $I(t)$ and $Q(t)$ now represent sum and difference terms. Demodulation, which is low pass filtering with time averaging, produces the IF signals:

$$\begin{aligned} I_{IF}(t) &= \frac{1}{T}\int_0^T s_I(t)y_I(t) \\ &= \frac{A_{RO}A_{LO}}{8}\cos(\omega_{IF}t + \theta_{RO}), \end{aligned} \tag{2.52}$$

$$\begin{aligned} Q_{IF}(t) &= \frac{1}{T}\int_0^T s_Q(t)y_Q(t) \\ &= \frac{A_{RO}A_{LO}}{8}\sin(\omega_{IF}t + \theta_{RO}), \end{aligned} \tag{2.53}$$

These anlog signals $I_{IF}(t)$ and $Q_{IF}(t)$ are now digitalized usign standard analog-to-digital converters (ADCs), resulting in digital signals:

$$I_{IF}[t_n] = \frac{A_{RO}A_{LO}}{8}\cos(\Omega_{IF}t_n + \theta_{RO}) \tag{2.54}$$

$$Q_{IF}[t_n] = \frac{A_{RO}A_{LO}}{8}\sin(\Omega_{IF}t_n + \theta_{RO}), \tag{2.55}$$

where $t_n = t/\Delta t$ are the sample times of the analog signal with sample period Δt and $\Omega_{IF} = \Omega_{IF}$ is the digital frequency. The standard way to deal with these signals is to perform a digital demodulation by multiplying the signals by a discrete sine and cosine. In this way, we get a single value for the I and Q signals again, such as

$$I = \frac{1}{M}\sum_{t_n=1}^{M} I_{IF}[t_n]\cos(\Omega_{IF}t_n) = \frac{A_{RO}A_{LO}}{16}\cos(\theta_{RO}) \tag{2.56}$$

$$Q = \frac{1}{M}\sum_{t_n=1}^{M} Q_{IF}[t_n]\sin(\Omega_{IF}t_n) = \frac{A_{RO}A_{LO}}{16}\sin(\theta_{RO}). \tag{2.57}$$

Several solutions can be developed to improve the effectiveness of the readout. One can improve the build quality of the system, optimize the readout procedure by trimming the parameters, but also use more complex approaches to readout and the use of machine learning techniques. These will be analyzed in Sec. 6 of this dissertation.

Chapter 3

Quantum Simulations

3.1 Introduction

Quantum mechanics is perhaps the most comprehensive and successful theory available to describe the behavior of the fundamental building blocks of our universe. Over the course of the last century, a number of techniques and computational tools, such as Quantum Monte Carlo [68, 69], molecular dynamics [70], and tensor networks [71], have been developed to tackle specific theoretical models framed in the language of quantum mechanics and effectively account for a wide range of quantum phenomena. Simulating a physical system essentially consists of creating an artificial replica of its properties and dynamic evolution over time. This is achieved through precise mathematical modeling, which maps known information about a system of interest to a set of variables and equations. The resulting mathematical identities, or a computer program designed to solve them, can then be called a *simulator*. This simulator is used to study the behavior of the real system under various conditions, to make predictions, and to test new hypotheses, the only limitations being the validity of the original model and the available computing power.

Due to the exponential scalability of memory and time resources, accurate simulation of most physical systems remains out of reach. This is especially true for systems with strong many-body correlations, typical of quantum mechanics and most interesting situations. In an attempt to address this problem, simulation with a quantum computer has long been proposed. This is based on the fact that a quantum system would be simulated with a device that is itself quantum and, unlike the former, controllable and measurable. There are two paradigms of quantum computer simulation. The first is **analogue quantum computing**. This is based on the idea of encoding quantum information in the continuous variables of a physical system, such as the position or momentum of a particle or the frequency or amplitude of a photon [72, 73, 74]. In this paradigm, the continuous variables are manipulated using analog circuits, such as waveguides, resonators, or interferometers, which allow the manipulation of the quantum state without requiring the discrete control of qubits. This paradigm will not be explored in depth in this dissertation. The second is the well-known **digital quantum computing**, which uses digital circuits to manipulate and process quantum information. Quantum gates are used to manipulate the state of qubits, just as classical gates are used to manipulate the state of classical bits in classical computing [75, 76, 77]. These devices are programmable and general-purpose quantum devices that offer the possibility to simulate a wider range of quantum systems. They are called *universal quantum simulator* because, in principle, they can simulate with arbitrary precision the dynamics of any Hamiltonian that can be mapped onto the quantum register and approximated by a sequence of unitary gates.

This chapter introduces the basics of simulating quantum systems, especially nuclear systems, on quantum computers, with independent and time-dependent Hamiltonians, in the context of digital quantum computing.

3.2 Quantum Simulations

As already mentioned in the previous chapters, the dynamical evolution of a generic quantum system is described by the well-known Schrödinger equation:

$$i\frac{d}{dt}\left|\psi(t)\right\rangle = H\left|\psi(t)\right\rangle , \tag{3.1}$$

where $|\psi(t)\rangle$ is the state of the quantum system at time t, H is the Hamiltonian operator representing the total energy of the system and where we impose $\hbar = 1$. Eq.(3.1) has the usual formal solution:

$$\left|\psi(t)\right\rangle = e^{-iHt}\left|\psi(0)\right\rangle , \tag{3.2}$$

where we call $e^{-iHt} \doteq U(t)$ the propagator of the system.
Matrix exponentiation is a common computational task that arises in various simulation scenarios, especially in the field of quantum mechanics when working in the matrix representation. However, performing matrix exponentiation on classical computers can be computationally expensive, especially when simulating quantum mechanical systems. This is due to the exponential growth of the Hilbert space size of a composite system with the number of subsystems, which leads to a corresponding exponential demand on time and memory resources. In 1982, Richard Feynman proposed that using a controllable quantum mechanical system as a computational resource, rather than a classical object, would provide significant advantages in the simulation of quantum systems. Seth Lloyd later

proved this idea to be essentially correct in 1996 [4], with the only limitation that the systems to be simulated should exhibit only local interactions between their constituent subsystems. We will therefore restrict ourselves to analyzing systems whose Hamiltonian can be written as a sum of locally acting Hamiltonians, i.e:

$$H = \sum_l H_l. \tag{3.3}$$

Fortunately, this is the case for a large number of physical phenomena.

Universal Quantum Simulator

We know that a quantum computer with a unversal set of gates is able to approximate any unitary transformation as mentioned in section.1.3. However, there is no indication of how efficient this approximation is in terms of the number of operations. Lloyd actually proved that this can be done in polynomial time and in memory of the size of the target system if it is a sum of local terms [4]. A general unitary matrix $U = e^{-iHt}$ acting on N qubits can be implemented by a sequence S of $\mathcal{O}(2^{2N})$ elementary operations. This would be exponentially inefficient with the size of U. However, if we consider a Hamiltonian like (3.3) and assume that the dimension m_l of each H_l is much smaller than H, $m_l \ll 2^N$, (i.e., involving only few-body/local interactions), it is clear that the implementation of $U_l = e^{-iH_lt}$ would require only $\mathcal{O}(m_l^2)$ operations. Now, since

$$U = \prod_l^L U_l, \tag{3.4}$$

the total order of operation become $\mathcal{O}(Lm_{max}^2)$, where $m_{max} = \max m_l$. Finally, by using the Suzuki-Trotter decomposition [78, 79] we obtain:

$$U = e^{-i\sum_l H_l t} = \lim_{n\to+\infty} \left(\prod_l^L e^{-iH_l t/n} \right)^n . \tag{3.5}$$

In this way, the unitary matrix U can be arbitrarily well approximated by applying n times the sequence of gates corresponding to the product of local terms for time slices t/n. Obviously, this procedure accumulates a computational error for $n < \infty$. It can be shown that $\forall n$:

$$U(t) = \left(\prod_l^L e^{-iH_l t/n} \right)^n + \mathcal{O}\left(\frac{t^2}{n} \right) . \tag{3.6}$$

Mapping

In quantum theory, each system is associated with a set of operators and an algebra that defines this language. There are three types of particles in nature: fermions, bosons, and anyons. Fermions and bosons are elementary particles that follow the Fermi-Dirac and Bose-Einstein statistics, respectively. Anyons are quasiparticles that exist only in two-dimensional confinement and follow continuous, or anyonic, statistics. Quantum computers work with qubits, which are a set of spin-1/2 particles. Quantum simulation involves mapping the operator of a physical system into the language of quantum computing while preserving the underlying statistics [80, 81]. In this way, an equivalence between the quantum system and the quantum computer is established, and this allows to deduce the behavior of the quantum system by transforming the state of the set of qubits.

In the standard case of two-level qubits quantum computing, the qubit is a 1/2 spin quantum system and can be denoted by its spin orientation, i.e. $|\uparrow\rangle = |0\rangle = (1,0)^T$ and $|\downarrow\rangle = |q\rangle = (0,1)^T$. A N qubit system is constructed from the usual Pauli matrices (for each local qubit site i) $\sigma_x^i, \sigma_y^i, \sigma_z^i$. These operators satisfy the commutation relations $[\sigma_\mu^l, \sigma_\nu^m] = 2i\delta_{lm}\epsilon_{\mu\nu\lambda}\sigma_\lambda^l$ for the $\bigoplus_{i=1}^N \mathfrak{su}(2)_i$ algebra, where $\epsilon_{\mu\nu\lambda}$ is the totally antisymmetric Levi-Civita symbol with $\mu, \nu, \lambda \in \{x, y, z\}$. Now, depending on what particle constitute the system one can choose between different mappings.

Fermions Fermions, one of three types of particles in nature, are elementary particles that follow the Fermi-Dirac statistics. In the second quantized notation, N fermions are represented by the fermionic operators $f_i^\dagger$ and f_i, which create and annihilate a fermion in the i^{th} mode or site. The fermionic operators obey Pauli's exclusion principle and the antisymmetric nature of the fermion wave function, leading to an algebra defined by the anticommutators $\{f_i, f_j\} = 0$ and $\{f_i^\dagger, f_j\} = \delta_{ij}$. There are several mappings to describe a fermionic system using the standard model of quantum computers. Examples are the Jordan-Wigner transformation [82], the Bravyi-Kitaev transformation [83], or Ball-Verstraete-Cirac transformation [84]. We briefly introduce the Jordan-Wigner transformation, the oldest and most intuitive mapping. This mathematical tool allows us to map a system of interacting fermions to an equivalent system of of interacting qubits, or vice versa. Thus, to map a fermionic state to a spin state, e.g. $|1100\rangle \leftrightarrow |\uparrow\uparrow\downarrow\downarrow\rangle$, one must use the following mapping:

$$f_i \leftrightarrow \left(\prod_{l=1}^{j-1} -\sigma_z^l\right)\sigma_-^j, \tag{3.7}$$

$$f_i^\dagger \leftrightarrow \left(\prod_{l=1}^{j-1} -\sigma_z^l\right)\sigma_+^j, \tag{3.8}$$

where $\sigma_\pm^j = (\sigma_x^j \pm i\sigma_y^j)/2$.

Bosons Bosonic creation (annihilation) operators $b_i^\dagger(b_i)$ satisfy the commutation relations $[b_i, b_j] = 0$, $[b_i, b_j^\dagger] = \delta_{ij}$ in an infinite-dimensional Hilbert space. Although it is impossible to simulate infinite space on a quantum computer, one is often interested in studying finite excitation modes above the ground state, making the use of the entire infinite dimensional Hilbert space unnecessary. In a finite-dimensional basis, the boson operators obey the commutation relations [85]:

$$[\bar{b}_i, \bar{b}_j] = 0, \tag{3.9}$$

$$\left[\bar{b}_i, \bar{b}_j^\dagger\right] = \delta_{ij}\left[1 - \frac{N_b+1}{N_b!}(\bar{b}_i^\dagger)^{N_b}(\bar{b}_i)^{N_b}\right], \tag{3.10}$$

where N_b is the maximum truncated excitation number corresponding to the i^{th} bosonic site or mode. Given $|n_i\rangle$ the quantum state with n bosons in site i and $b_i^\dagger b_i |n_i\rangle = n_i |n_i\rangle$ with $n_i = 0, ..., N_b$, the creation operator can be written

$$\bar{b}_i^\dagger = \sum_{n=0}^{N_b-1} \sqrt{n+1}\sigma_-^{n,i}\sigma_+^{n+1,i} \tag{3.11}$$

where the pair (n, i) indicates the qubit n that represents the i^{th} site. b_i is simply the complex conjugate of $b_i^\dagger$. These operators can be used with a mapping, e.g. a *binary* or *compact encoding*, where each $|n\rangle$ is written in terms of binary strings:

$$|0\rangle_i = |\uparrow_0, \downarrow_1, ..., \downarrow_{N_b}\rangle_i\,, \tag{3.12}$$

$$|1\rangle_i = |\downarrow_0, \uparrow_1, ..., \downarrow_{N_b}\rangle_i\,, \tag{3.13}$$

$$\vdots \tag{3.14}$$

$$|N_b\rangle_i = |\downarrow_0, \downarrow_1, ..., \uparrow_{N_b}\rangle_i\,, \tag{3.15}$$

where $N(N_b+1)$ qubits are needed for the simulation (with N the number of sites) [86, 87].

3.2.1 Quantum Simulation Recipe

A generic quantum system can be simulated on a quantum computer using the following steps.

1. *Hamiltonian Definition.* Obtain a Hamiltonian H that contains the dynamical information necessary to describe the quantum system to be simulated.

2. *Mapping.* Find an encoding of the degrees of freedom of the system in the N qubits. It is necessary to establish a one-to-one relationship between the states of physical systems and quantum processors, and it is essential to choose the most efficient mapping that eliminates potential errors. This may involve, for example, eliminating the effects of quantum noise. This mapping is usually straightforward for physical systems consisting of collections of spin-1/2 objects, since they obey Pauli algebra.

3. *Suzuki-Trotter decomposition.* Choose the appropriate number of Suzuki-Trotter steps n to obtain the desired precision in implementing $U = \exp(-iHt)$ for a given t as eq. (3.6).

4. *Encoding.* Translate each $U_l = e^{-iHt/n}$ into smaller unit operations that can be implemented on the quantum computer. The entire quantum circuit implementing U will be the iteration of the Suzuki Trotter step n times.

5. *Initial state preparation.* Select the initial state to start the simulation. Often the initial state is $|0\rangle$ for all qubits, because this is the most natural and easiest state to prepare in quantum computers, but other states can be set.

6. *Measurement.* After the application of a quantum circuit, it is necessary to have an appropriate set of measurements at the end of the procedure to extract expectation values of the relevant observable quantities on the evolved quantum state of the qubits. The probability distribution of the quantum processor on some computational basis is usually measured. Then, thanks to the map between the quantum system and the qubits, the dynamics of the degrees of freedom of the system under analysis can be traced.

Encoding

Encoding is a rather delicate step and can be done in two ways, which have already been discussed in previous chapters but will now be more clearly defined for the specific case of quantum simulations.

Elementary Gate Decomposition In this approach, each propagator $U_l(t/n)$ is decomposed into a sequence of simpler, elementary gates that can be physically implemented on a quantum computer [88]. This process is necessary because most physical quantum computers are limited in the types of gates they can directly implement, and thus more complex gates must be built from simpler ones. This set of gates must be a *universal gates set* (See. Sec. 1.3), meaning that there is a sequence of these gates that can approximate any arbitrary propagator U.

Examples of such elementary gates that are physically realizable are the Pauli-X, Pauli-Y, Pauli-Z, Hadamard, and Phase gates. The decomposition of gates can be done automatically using software tools that use mathematical algorithms to analyze the structure of the gate and determine an optimal decomposition into simpler gates. This process is important in quantum circuit design and helps ensure that quantum algorithms can be executed efficiently and accurately on real quantum hardware.

This approach is not perfect and has some drawbacks. First of all, gate decomposition can result in a long sequence of gates, i.e. a large "circuit depth". This can make the circuit more susceptible to errors due to noise and decoherence, as well as increase the time required to perform the computation.

On the other hand, it can make it difficult to optimize circuits. The process of gate decomposition can result in a large number of possible gate sequences, which can make it difficult to find the optimal circuit for a given quantum operation. This can be particularly challenging for large and complex operations, where finding the optimal decomposition might require significant computational resources. Third, it has limited applicability: Gate decomposition is not suitable for all quantum operations. Certain operations, such as non-unitary operations or unitaries of highly entangled systems, may not be easily decomposable into a sequence of simpler gates. In these cases, alternative techniques may be needed. Finally, the complexity of error correction may increase. This is because errors can occur at any point in the gate sequence, and correcting these errors requires a more complex error correction scheme that can detect and correct errors at multiple points in the circuit.

Optimal Control Techniques Another less used but powerful approach is optimal control, which was introduced in section 2.3.1. In the context of quantum computing, optimal control is used to design and implement quantum gates that perform specific operations on qubits. The goal is to find the control function that minimizes a given cost function while satisfying certain constraints. The cost function is typically a measure of how far the driven quantum state is from the desired state, and the constraints can be physical limits on the control parameters due to physical constraints. In the case of quantum simulations, the target unitary propagator is the trotter-suzuki slice from Eq. (3.4):

$$U(t/n) = \prod_{l=0}^{L} e^{-iH_l t/n}. \tag{3.16}$$

Using GRAPE (See Sec. 2.3) or other optimization algorithm one can finally find the set of controls $\epsilon_k(t)$ which satisfy Eq.(2.35). In the superconducting circuits the controls $\epsilon_k(t)$ are microwave pulses interacting directly with the k^{th} qubit and driving it in the desired way.

Quantum optimal control enables the design and implementation of quantum gates that are more robust and accurate than those based on simpler control techniques. In fact, it can condense the use of many gates into just one, reducing the amount of time the qubits are subject to noise and decoherence. By optimizing the control parameters, quantum computers can perform complex computations more efficiently and accurately, leading to faster and more powerful quantum algorithms.

However, there are some drawbacks to this technique. Quantum optimal control algorithms can be computationally expensive, especially for large qubit systems or long gate operations. In addition, quantum optimal control techniques may not scale well to larger quantum systems because the computational complexity of the optimization problem grows exponentially with the number of qubits. This may limit the practical utility of these techniques for large-scale quantum computing applications.

3.3 An Example: Nulcear System

3.3.1 Nuclear Physics Background

We present an example of a quantum simulation of a nuclear system according to Ref. [89]. This is the spin dynamics of two neutrons interacting with a simple potential derived from *chiral effective field theory* (χ-EFT), taking into account the leading order (LO) of the expansion.

χ-EFT is a theoretical framework for describing the interactions between subatomic particles, particularly those involving the strong nuclear force [90]. The adjective *chiral* comes from the fact that this approach is based on the principle of chiral symmetry, which is a fundamental property of the strong force that arises from the fact that the masses of the quarks that make up protons and neutrons are much smaller than the energy scale at which the strong force operates. Chiral EFT is a low-energy, perturbative approach in which the interactions between particles are expanded in powers of their momenta and masses. This expansion is controlled by a small parameter called the chiral symmetry breaking scale, which is

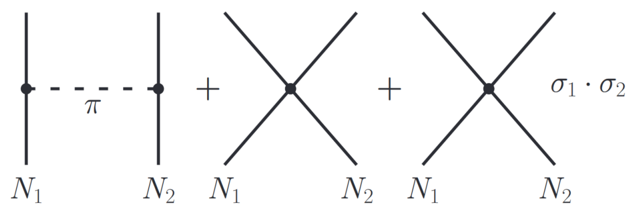


Figure 3.1: Schematic description of the leading-order nucleon-nucleon interaction. The first diagram represents a single pion exchange process, the middle diagram depicts a spin-independent interaction, and the right one a spin-dependent contact term.

related to the mass of the pion, the lightest meson particle. One of the key features of chiral EFT is that it can be used to systematically calculate the properties of hadrons (particles made up of quarks) and their interactions with other particles, such as photons or other hadrons. Chiral EFT has been successful in describing a wide range of phenomena in nuclear physics, such as the structure of the deuteron, nucleon-nucleon scattering, and the properties of light nuclei. The forces between nucleons are a remnant of the color interactions from the underlying quantum chromodynamics theory.

We are interested in the two-nucleon system. The first-order χ-EFT expansion captures the features of the neutron interaction, including a tensor-like spin-dependent component, in agreement with experimental measurements [91, 92]. In general, single pion exchange is the main interaction mechanism at medium distances ($\approx 2e - 15m$), while all other processes (multiple pion exchange or heavier meson exchange) at shorter distances can be recombined into a spin-dependent contact force. A schematic representation with Feynmann diagrams of this interaction is reported in Fig. 3.1. The LO Hamiltonian H_{LO} obtained with EFT is $H_{LO} = T + V_{SI} + V_{SD}$, where T is the kinetic energy, V_{SI} is the spin-independent (SI) part of the two-nucleon potential, acting on the spatial degree of freedom of the system, and V_{SD} the spin-dependent (SD) part, acting on the spin degrees of freedom. The SD potential takes into account vector and a tensor force, namely:

$$V_{SD}(\mathbf{r}) = A^{(1)}(\mathbf{r}) \sum_{\alpha} \sigma^1_\alpha \sigma^2_\alpha + \sum_{\alpha\beta} \sigma^1_\alpha A^{(2)}_{\alpha\beta}(\mathbf{r}) \sigma^2_\alpha, \tag{3.17}$$

where $\mathbf{r}$ represents the two neutrons relative position in Cartesian coordinates, σ^k_α for $\alpha = x, y, z$ are the Pauli matrices acting on spin $k = 1, 2$ and the functions $A^{(1)}(\mathbf{r})$ and $A^{(2)}_{\alpha\beta}(\mathbf{r})$ are functions coming from EFT expansion. Their explicit form can be recovered from Ref. [93, 94] and in Appendix B.1 the form used in the present work is reported.

3.3.2 Time-independent simulation

We now consider a quantum simulation in which the Hamiltonian does not depend on time. In particular, we consider the artificial case where the two neutrons are fixed in space and we are interested only in the spin dynamics.

The propagator $U(\delta s) = e^{-iH_{LO}\delta s}$ of the system under analysis can be decomposed into its two components:

$$\begin{aligned} U(\delta s) &= [e^{-i(T+V_{SI})\delta s}][e^{-iV_{SD}(\mathbf{r})\delta s}] \\ &= U_{SI}(\delta s)U_{SD}(\delta s, \mathbf{r}). \end{aligned} \tag{3.18}$$

Since we are considering fixed particles, the dynamical part can be neglected and we can only consider the SD propagator:

$$U_{SD} = e^{-iV_{SD}\delta s}. \tag{3.19}$$

The simulation is done at a simulated device level, so it simulates the output one should get with a real device. We will also assume that we are working with a four-level transmon qubit (i.e. a qudit). Let us now follow the steps in section.3.2.1 to make a quantum simulation on a quantum computer.

Step 1: In this case the Hamiltonian is simply $H_{LO} = V_{SD}$.

Step 2: The map chosen in the case of a two-neutron system on a 4-level qudit transmon is the one that identifies each level of the transmon with one of the four spin configurations of the system. Explicitly:

$$|\uparrow\uparrow\rangle \Leftrightarrow |0\rangle\,, \tag{3.20}$$
$$|\uparrow\downarrow\rangle \Leftrightarrow |1\rangle\,, \tag{3.21}$$
$$|\downarrow\uparrow\rangle \Leftrightarrow |2\rangle\,, \tag{3.22}$$
$$|\downarrow\downarrow\rangle \Leftrightarrow |3\rangle\,. \tag{3.23}$$

Step 3: This is already satisfied by having the short time propagator of Eq. (3.19) with a δt adequate to the simulation. In this case, we use the representation of $U_{SD}(\delta t)$ in terms of a $4x4$ matrix. *Step 4*: We use GRAPE algorithm to find the controls that drive the 4-level transmon in the proper way to implement the propagator $U_{SD}(\delta t)$. This is done by using the *Quantum Toolbox in Python* (QuTiP) [95, 96]. Using the built-in function *optimize_pulse_unitary*, implementing GRAPE algorithm, we are able to compute the optimal pulse implementing $U_{SD}(\delta t)$ in the transmon. See Tab. 3.1 for the algorithm specifications

Step 5: The initial state is simply one of the four spin configurations, in this case, $\psi(0) = |\uparrow\uparrow\rangle = |0\rangle$.

Step 6: No actual measurements are made since this is a simulated device, but the occupancy probabilities of the transmon levels at each time step are analyzed.

Transmon anharmonicity α	200 MHz
Pulse duration	100 ns
Pulse sampling frequency	32 GHz
Number of pulse time-steps	3200
Fidelity error	10^{-4}
initial guess for pulses	zero function

Table 3.1: Parameters for GRAPE algorithm implemented in Qutip

We know that at each time step the state is:

$$|\psi(j\delta t)\rangle = \left(\prod_{z=0}^{j} e^{-iV_{SD}\delta t}\right)|\psi(0)\rangle\,, \tag{3.24}$$

Introducing the computational basis $|\xi\rangle\,, \xi = 0, 1, 2, 3$ for measuring the quantum processor, the i^{th} occupation probability at time-step j is given by:

$$P_i(j\delta t) = |\,\langle\xi|\psi(j\delta t)\rangle\,|^2. \tag{3.25}$$

Now we can run the simulation. We take an arbitrary relative position of the two neutrons and derive the corresponding Hamiltonian H_{syst}. Now we want to calculate two evolutions. First, the exact reference evolution. To get this, we calculate the propagator $U_{SD} = e^{-iH_{syst}\delta t}$ for a time step δt of 0.01 $[MeV^{-1}]$. We apply it 100 times to the initial state, obtaining for each time step j the state described by Eq. (3.24). Projecting the $|\psi(j\delta t)\rangle$ state onto the computational base, we obtain the occupation probability as a function of time. This is plotted as solid lines in Fig.3.2. Second, we compute the simulated output of the four-level transmon implementing the same U_{SD}. To do this, we simply optimize the controls corresponding to the propagator with GRAPE and then reconstruct the propagator using the right-hand side of Eq.(2.35), which we report here for clarity:

$$U_{recon} = \exp\left\{-i\delta t\sum_{j=1}^{N_T}\left[H_0 + \sum_{k=1}^{N_{ctr}}\epsilon_k[j]H_k\right]\right\}. \tag{3.26}$$

This way we have the propagator that the controls $\epsilon_k(t)$ would implement in the actual machine. This allows us to test its behavior. Applying U_{recon} to the initial state as the exact case, we can find the spin dynamics, again in terms of occupancy probabilities. This is shown in Fig. 3.2 as dots.

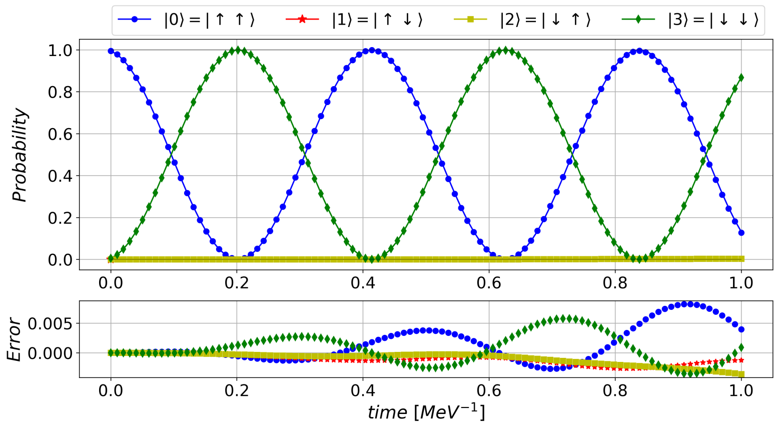


Figure 3.2: Spin dynamics for two neutrons fixed in space in terms of occupation probability. Solid lines represent the exact evolution. Dots represent the subsequent application of controls

3.3.3 Time-dependendet simulation

There are cases in which the Hamiltonian depends on time, or on parameters that are themselves time-dependent. In the two-neutron example just introduced, if one wants to carry out the complete evolution with the neutrons not fixed in space, it is necessary to consider that the Hamiltonian changes at each position of the neutron (i.e. at each time step). To handle this, following always Ref.[89], we make use of the approximation that the simulation of the SD and SI part can be carried out separately. Under this approximation, the SI part acts only on the spatial part of the system, while the SD part acts only on the spin degrees of freedom. Given a complete set of states $|r, s_1, s_2\rangle \equiv |r\rangle \otimes |s_1, s_2\rangle$, normalized as $\langle r, s_1 s_2 | r', s'_1 s'_2\rangle = \delta(r - r')\delta_{s'_1 s_1}\delta_{s'_2 s_2}$, with $|r\rangle$ the (relative) position state, and $|s_1, s_2\rangle$ the spin state of the system, we can project the state $|\psi(s)\rangle$ onto this basis at an evolved time $s + \delta s$ as

$$\begin{aligned}\langle r, s_1, s_2|\psi(s+\delta s)\rangle \quad \simeq \quad & \sum_{s'_1, s'_2} \int d^3r' \, \langle r| \, U_{SI}(\delta s) \, |r'\rangle \times \\ & \langle s_1, s_2| \, U_{SD}(\delta s, \mathbf{r}) \, |s'_1, s'_2\rangle \, \langle r', s'_1, s'_2\rangle \, \psi(s).\end{aligned} \tag{3.27}$$

Therefore, for an infinitesimal time step, one can advance the spatial part and then, keeping the neutron position fixed, advance the spin part of the wave function. We exploit this approximation to use a hybrid computing protocol, or *co-processig protocol*, in which the spatial part of the system is simulated with classical algorithms on a classical computer, while the spin dynamics part is performed by a quantum processor. This protocol relies on the saddle point approximation of the path integral of the SI part [97]. Under this approximation, we neglect the quantum fluctuations and compute the classical trajectory of the particle, knowing that it is the most probable.

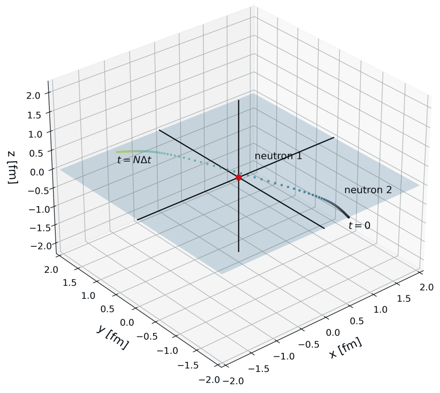
In the following, we adopt this approach and simulate the trajectory of the two neutrons using a classical differential equation integrator algorithm. At the same time, we advance the spin dynamics by a simulation of quantum computer results as in the previous section of the time-independent simulation. Thus, the evolved spin state is obtained by applying the short-time propagator $U_{SD}(\mathbf{r}(s))$ to the "instantaneous" spin state at each time step of the spatial trajectory, i.e., corresponding to the "instantaneous" neutrons' relative position $\mathbf{r}(s)$. We are interested in the occupation probability of each spin configuration at each time step, starting from an initial configuration $|s_1^0, s_2^0\rangle$,

The hybrid classical-quantum simulation procedure for every time-step starting from the initial condition is:

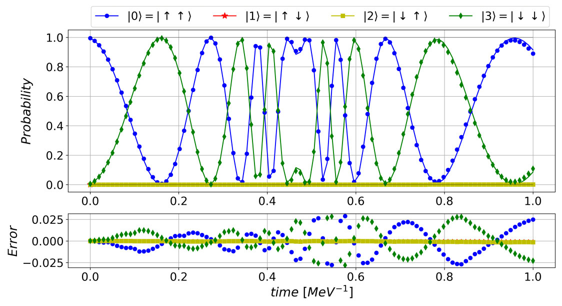
1. Update the relative position $\mathbf{r}_i$ $i > 0$ of the two neutrons with a classical algorithm on a classical computer.
2. Evaluate $\hat{U}_{SD}(\delta t, \mathbf{r}_i)$ for the new position $\mathbf{r}_i$.
3. Optimize the control pulses $\epsilon_R(t)$ and $\epsilon_I(t)$ implementing $\hat{U}_{SD}(\delta t, \mathbf{r}_i)$.
4. Update the spin state $|s_i\rangle$ on the quantum processor using the just optimized control pulses.

Spatial trajectory is obtained by solving the Newton equation with a simple Crank–Nicolson algorithm.

An example of this simulation is shown in Fig. 3.3. Panel (a) shows the trajectory of the second particle with respect to the first (on which the coordinates are fixed); the colors represent the simulation time. Panel (b) shows the spin dynamics in terms of the occupation probability (3.24) along the spatial trajectory shown in panel (a). The total simulation time, taking into account the optimization time of the controls, is about 980 seconds. In the following chapters, a method to drastically reduce this time will be presented and tested.



(a) Neutrons trajectory



(b) Spin dynamics

Figure 3.3: Representation of the neutrons dynamics. *Panel (a)*: A single realization of a classical spatial trajectory for two neutrons obtained solving their equation of motion with a simple Euler algorithm. The origin of axes is fixed on one particle. *Panel (b)*: Spin dynamics for the system following the trajectory of panel (a) in terms of probability to find a particular spin configuration. In the lower panel is represented the error for each time step.

Chapter 4

Machine Learning Techinques

4.1 Introduction

Machine learning is a subfield of artificial intelligence that focuses on developing algorithms and statistical models that enable computer systems to automatically learn and improve from experience without being explicitly programmed for the specific task at hand. In other words, machine learning algorithms are designed to analyze and learn from data to make predictions or decisions about new data. There are three major types of machine learning: *supervised learning*, *unsupervised learning*, and *reinforcement learning.* Supervised learning involves training an algorithm on labeled data, where the algorithm is given inputs and expected outputs. Unsupervised learning involves training an algorithm on unlabeled data, where the algorithm tries to find patterns or structures in the data. Reinforcement learning involves training an algorithm to make decisions based on feedback from its environment. Common applications of machine learning include image and speech recognition, natural language processing, recommendation systems, and predictive modeling.

In this dissertation, supervised and unsupervised machine learning algorithms are applied to optimal qubit control and qubit readout. In this chapter, we give a short summary of the algorithm used in Chapter 5 and 6.

4.1.1 Supervised algorithms

Gaussian mixture model (GMM) is a special case of mixture models. This algorithm is able to approximate the distribution of data as a weighted superposition of (multi-dimensional) Gaussian distributions [13, 98]. After the optimization of the GMM's parameters on the dataset consisting of N classes, a new (multi-dimensional) point is attributed to one out of the N classes based on the higher probability that it belongs to one of the Gaussians of the GMM.

Feed-forward Neural Network (FFNN) are the simplest class of neural networks. Trained over a labeled dataset, they are capable of classifying new inputs. Formally the neural network implements a closed-form parametrized function, N_θ, which maps input in a space $\mathcal{X} \subseteq \mathbf{R}^m$ into a space $\mathcal{Y} \subseteq \mathbf{R}^n$ which encodes in some way the information on the classes the inputs are divided into. Optimal data classification is obtained by adjusting the parameters θ using optimization algorithms. This is obtained by minimizing some type of loss function l between the correct label $\mathbf{y}^i$ of input $\mathbf{x}^i$ and the neural network predicted label $\hat{\mathbf{y}}_i = N_\phi(\mathbf{x}^i)$, namely:

$$\min_\theta \sum_i l\left(\mathbf{y}^i, N_\phi(\mathbf{x}^i)\right) \ . \tag{4.1}$$

This optimization commonly employs the well-known *back-propagation* algorithm [99, 100].

4.1.2 Unsupervised

Autoencoders are neural networks designed to learn, via unsupervised learning procedures, efficient encoding of data [100, 101, 102]. This encoding is achieved by adjusting the network's weights and biases to regenerate the input data. It is composed of a first part, the encoder, which learns to map the input data into a lower dimensional representation (the latent space), ignoring insignificant features or noise, and a second part, the decoder, that, conversely, is trained to reconstruct the original input from the low dimensional encoding in the latent space. Autoencoders perform dimensionality reduction and feature learning.

Mathematically, the autoencoder is a model composed of two closed-form parametrized functions, the encoder f_{θ^e} and the decoder g_{θ^d}. The parameters $\theta = [\theta^e, \theta^d]$ need to be optimized to perform the correct inputs reconstruction. These functions are defined as:

$$\begin{aligned} f_{\theta^e} &: \mathcal{X} \to \mathcal{L} \\ g_{\theta^d} &: \mathcal{L} \to \mathcal{X} \ . \end{aligned}$$

The function f_{θ^e} takes an input $\mathbf{x}^i \in \mathcal{X} \subseteq \mathbf{R}^m$ from the data-set $\{\mathbf{x}^1, \mathbf{x}^2, ...\}$ and maps it into the feature-vector $\mathbf{h}^i \in \mathcal{L} \subseteq \mathbf{R}^p$ with $p < m$ i.e. $\mathbf{h}^i = f_{\theta^e}(\mathbf{x}^i)$. Conversely, the decoder function, g_{θ^d} maps the feature-vector $\mathbf{h}^i$ back into the input space, giving a reconstruction $\tilde{\mathbf{x}}^i$ of the input $\mathbf{x}^i$.

The parameters θ of the autoencoder are optimized such that the model minimizes the reconstruction error $l(\mathbf{x}, \tilde{\mathbf{x}})$, i.e. a measure of the discrepancy of the reconstructed input from the original one. The general minimization problem is, therefore:

$$\min_{\theta} \sum_{i} l\left(\mathbf{x}^i, g_{\theta^d}(f_{\theta^e}(\mathbf{x}^i)\right). \tag{4.2}$$

Again, this is optimized with the already mentioned *back-propagation* algorithm.

Chapter 5

Improving Quantum Simulation with Optimal Controls Interpolation

5.1 Introduction

The main drawback of using optimization algorithms (such as GRAPE) to compute optimized control sequences is the computational cost. In fact, the time required to compute a control grows exponentially with the number of qubits involved, i.e. the dimension of the propagator matrix. Furthermore, the control pulses are not transferable, i.e. any change in the unitary operator requires a completely new optimization. This is true if there is any parametric dependence of the Hamiltonian, and a study of the system as a function of the parameter value is required. This has already been introduced in Sec. 3.3.2, where the simulation of a scattering process of two nucleons interacting through a spin/isospin dependent interaction, as captured in the by the leading order (LO) in the Chiral Effective Field Theory expansion, implies that the control pulses implementing the instantaneous spin dynamics have to be computed at each time step of the neutron physical trajectory. This tends to neutralize the advantages in terms of computational speedup that quantum computation brings to the simulation of quantum processes.

To mitigate this problem, one can act in several directions. For instance, one could improve the optimization algorithms to make them faster, but the exponential scaling remains. If the physics of the system under analysis permits, one might decompose the target unitary transformation into a short sequence involving fewer qubit transformations at a time. This reduces the exponential scaling and one has to find the optimal counter of several transformations with reduced dimensionality. However, this does not eliminate the problem of having to recompute the controls at each time step in the case of time-dependent simulations.

In this section, we will propose a method to solve, at least partially, this problem for time-dependent Hamiltonians. In essence, this method simply consists of computing in advance a set of controls for a set of parameter values of the Hamiltonian of interest, and then deriving new controls for arbitrary values of these parameters by appropriately interpolating between the controls in the data set. This makes it possible to bypass the optimization algorithm after deriving a sufficient number of controls in advance. This method will be applied to the neutron-neutron scattering of Sec. 3.3.2. Then, a more general implementation using neural networks and Lie group theory will also be shown and tested.

5.2 Control Pulse Reconstruction method

We recall from Eq. (2.35) that GRAPE algorithm works by finding the optimal controls $\epsilon_k(t)$ such that the following relation is satisfied, within an appropriate threshold:

$$U_{targ} \quad = \quad \exp\left\{-i\Delta\tau\sum_{j=1}^{N_T}\left[H_0+\sum_{k=1}^{N_{ctr}}\epsilon_k(j)H_k\right]\right\}, \tag{5.1}$$

where H_0 is the qubits system Hamiltonian, H_k the drive Hamiltonians, N_{ctr} the number of controls, N_T the number of discrete time-steps, of duration $\Delta\tau$, into which the controls are divided and U_{targ} is the target unitary we want to implement.

An explicit computation of the pulses for each value of the Hamiltonian's parameters can be avoided using the following general control pulse reconstruction (CPR) method. Given the Hamiltonian $H_{syst}(\Lambda)$, where $\Lambda = [\lambda_1, \lambda_2, \ldots, \lambda_K]$ is a point in the K-dimensional space of parameters characterizing the Hamiltonian, and assuming that one wants to implement the unitary transformation $U_{syst}(\delta s, \Lambda) = \exp\{-i\delta s H_{syst}(\Lambda)\}$ on the quantum device, one should:

1. Solve Eq. (5.1) (using GRAPE or another equivalent optimization algorithm) for a discrete grid of the values of Λ and store the resulting control pulses. The result is a family of controls, each of which implements the specific unitary transformation, $U_{syst}(\delta s, \Lambda)$, for a specific value of Λ.

2. Perform a fit of the pulses either in terms of a function (e.g. a polynomial fitting) or as an expansion over a basis (e.g. Fourier transform of the pulses). Let us call $C = [c_1, c_2, \ldots, c_M]$ the M coefficients of the fit (e.g the polynomial coefficient or some type of coefficient relative to the expansion over a basis).

3. Find a mathematical relationship between the coefficients of the fit C and the parameters Λ, i.e. $C = f(\Lambda) = [f_1(\Lambda), f_2(\Lambda), \ldots, f_M(\Lambda)]$.

4. Reconstruct the pulses for an arbitrary $\tilde{\Lambda}$ via the following procedure: select the $\tilde{\Lambda}$ values of interest, recover the fit parameters $\tilde{C}$ through the mathematical relationship $C = f(\Lambda)$ and reconstruct, finally, the control pulses identified by $\tilde{C}$ inverting the fit relation.

5. Repeat point 4 in the simulation loop to recover all the needed control pulses.

5.2.1 CPR Method Realization

In this section, we describe in detail the actual implementation of the CPR method introduced above.

First of all, let's define the grid of Λ values. Each component λ_i of Λ, with $i = 1, ..., K$, is a continuous parameter of the Hamiltonian $H_{syst}(\Lambda)$ (and the correspondent propagator $U_{syst}(\Lambda)$) defined on a real interval I_i. Let's divide each I_i in Z_i discrete values, obtaining Z_i discrete values of λ_i, namely $\lambda_i^{a_i}$ for $a_i = 1, ..., Z_i$. Let's define Λ_m to be a K-dimensional grid of all possible combination of these parameters, i.e $\Lambda_m^{a_1,a_2,\ldots,a_K} = [\lambda_1^{a_1}, \lambda_2^{a_2} ..., \lambda_K^{a_k}]$ with $a_1 \in \{1, 2, ..., Z_1\}$, $a_2 \in \{1, 2, ..., Z_2\}$, ..., $a_K \in \{1, 2, ..., Z_K\}$. Λ_m is the discrete grid of value of the parameters Λ.

In the second place, we define the expansion C. The pulses have shapes that cannot be easily interpreted in terms of elementary functions, since they could contain multiple frequency components. This is true especially for large Hamiltonians where increasingly complex controls are observed. To deal with this type of signals an obvious choice for the expansion is the use of the Fourier transform. In this case, the fit parameters C are the components of the control pulse spectra.

Lastly, the mathematical relationship $C = f(\Lambda)$ is a multi-linear interpolation of each component of vector C over the grid Λ_m.

To summarize, in this work the CPR method takes the following form:

1. For each element $\Lambda' \in \Lambda_m$, corresponding to a single and unique combination $[\lambda_1^{a_1}, \lambda_2^{a_2} ..., \lambda_K^{a_k}]$, we can compute, via Eq. (5.1), the N_{ctrl} controls $\epsilon_z(t, \Lambda')$, with $z = 1, ..., N_{ctrl}$, implementing the specific transformation identified by $U_{syst}(\Lambda')$. Let $\mathcal{E}_z(t, \Lambda_m)$ be the discrete set of all $\epsilon_z(t, \Lambda')$ for all $\Lambda' \in \Lambda_m$.

2. Let $C_z(\omega, \Lambda') = [c_{z,1}(\omega_1, \Lambda'), .., c_{z,M}(\omega_M, \Lambda')]$ be the (discrete) Fourier transform of $\epsilon_z(t, \Lambda')$ truncated to its M^{th} component. $C_z(\omega, \Lambda')$ is interpreted as the expansion coefficients vector over the basis (i.e. the frequency components ω_i). Obviously, as a result of a Fourier transform, each C_z is a complex vector $C_z = C_z^{real} + iC_z^{imag}$. Let $\mathcal{C}_z(\omega, \Lambda_m)$ be the discrete set of all $C_z(\omega, \Lambda')$ for all $\Lambda' \in \Lambda_m$.

3. In order to obtain a new vector $\tilde{C}_z(\omega, \tilde{\Lambda})$ for an arbitrary $\tilde{\Lambda} \notin \Lambda_m$, we use a linear multi-variate interpolation of each component $c_{z,i}$ between the values of the subset $\mathcal{C}_z(\omega_i, \Lambda_m)$ over the K-dimensional grid Λ_m. This defines the mathematical relationship $C = f(\Lambda)$ that links the parameters Λ to the expansion coefficients C_z. The interpolation is performed separately for the real and imaginary parts of the vector $\tilde{C}_z(\omega, \tilde{\Lambda})$ and the results are cast together. This is carried out for every $z = 1, .., N_{ctrl}$.

4. The control pulse reconstruction procedure becomes the following: the N_{ctrl} control pulses for an arbitrary $\tilde{\Lambda} \notin \Lambda_m$ are recovered by interpolating, for each control, the real and imaginary part of the new coefficients $\tilde{C}_z(\tilde{\Lambda})$ through the linear multi-variate interpolation of the set $\mathcal{C}_z(\omega_i, \Lambda_m)$ for $i = 1, ..., M$, over the discrete K-dimensional grid Λ_m. Then, the so obtained $\tilde{C}$ is transformed back to the time domain with the inverse Fourier transform obtaining the correspondent new controls $\tilde{\epsilon}_z(t, \tilde{\Lambda})$ which are not elements of the original sets $\mathcal{E}_k(t, \Lambda_m)$.

The purpose of the Fourier transform of the original data set $\mathcal{E}_z(t, \Lambda_m)$, in Step 2, is that, in the cased considered, the number of spectral components relevant to the reconstruction of a control is far less than the number of time steps of the control. This, even could not be true in general, is true in the case under analysis. In fact, many spectral components are close to zero, so the expansion over the Fourier basis offers a significant reduction in the number of elements that need to be interpolated using the CPR method. In practice, in this work, the spectra were all truncated at the M^{th} component for which the average (of all spectra) was three orders of magnitude smaller than the higher averages, and at the same time when it was less than 0.001. This allowed us to have a M that was two or three orders of magnitude smaller than the number of time steps of the controls (depending on the configuration). In addition, the Fourier transform offers the advantage of being able to use the CPR method with noisy controls by acting as a high frequency filter. This could be the case for some kind of optimization initial conditions which, however, are not used in this work.

5.2.2 CPR Method Characterization

Scaling

To test the scaling of the method, we study the fidelity of the interpolated controls in three cases: 1) the number of quantum levels/qubits involved, 2) the number K of Hamiltonian parameters and 3) the Λ_m grid spacing. This will be tested on a 1D transverse-field Ising model with periodic boundary conditions (i.e in a closed spin chain configuration). We chose the Ising model because it is a simple and well-known model, easy to be mapped onto the quantum processor, being at the same time a not trivial system. The transverse-field Ising Hamiltonian is:

$$H_{Ising} = \sum_{i=1}^{N} J_i \sigma_i^z \otimes \sigma_{i+1}^z + h \sum_{i=1}^{N} \sigma^x, \tag{5.2}$$

with N the number of spins/sites of the chain, σ_i^z and σ_i^x the z and x Pauli matrices for the i^{th} spin/site, J_i the coupling terms between nearest spins and h the external field intensity. Note that we set $N+1 \equiv 1$ to obtain the closed chain. The propagator is written:

$$U_{Ising} = e^{-iH_{Ising}\delta s}. \tag{5.3}$$

for an arbitrary (small) time interval δs.

The CPR system is here tested on a $N_q = 2, ..., 5$ qubits system with a closed ring topology, described by the following Hamiltonian:

$$\begin{aligned} H_0 = \quad & \sum_{i=1}^{N_q} \omega_i b_i^\dagger b_i + \frac{\alpha_i}{2} b_i^\dagger b_i (b_i^\dagger b_i - 1) + \\ & + \sum_{i=1}^{N_q} g_{i,i+1} (b_i^\dagger b_{i+1} + b_i b_{i+1}^\dagger), \end{aligned} \tag{5.4}$$

where ω_i are the qubits frquencies, α_i are the qubits anharmonicity, $g_{i,i+1}$ are the coupling constant between connected qubits and clearly $N_q + 1 \equiv 1$ to implement the closed ring topology. The values used in this work are reported in Tab.5.1.

Each spin has two states, the spin up, $|\uparrow\rangle$, and the spin down, $|\downarrow\rangle$. The mapping of the Ising model in the quantum processor is therefore done simply by identifying each spin with a single two-level qudit of the quantum processor Hamiltonian H_0. Using Eq. (5.1) is now possible to compute the controls $\epsilon_z(t)$ that implement U_{Ising} in the quantum device. For this experiment, we use a configuration that considers a control for each qubit (but many other configurations can be realized).

We consider the parameters $J_i, \forall i \in [1, N]$ and h all taking values in the same real interval $I = [0.2, 2]$. We discretize I in Z discrete values so to have the discrete values $J_i^{a_i}$ and h^b with $a_i, b \in \{1, .., Z\}$. We can now define Λ_m as the $N+1$-dimensional grid of all possible combination of $J_i^{a_i}$ and h^b, i.e. $\Lambda_m^{a_1,...,a_N,b} = [J_1^{a_1}, .., J_N^{a_N}, h^b]$. We can now compute the N_{ctrl} sets of controls, $\mathcal{E}_z(t, \Lambda_m)$, and the corresponding set of truncated spectra, $\mathcal{C}_z(\omega, \Lambda_m)$, for $z \in \{1, .., N_{ctrl}\}$. New controls can be now finally found following the CPR procedure described above.

	Q 1	Q 2	Q 3	Q 4	Q 5
$\omega/2\pi$ [GHz]	5.114	4.914	4.714	4.614	4.514
$\alpha/2\pi$ [GHz]	-0.33	-0.33	-0.33	-0.33	0.33
$g_{i,j}/2\pi$ [GHz]	0.0038	0.0038	0.0038	0.0038	0.0038

Table 5.1: Values of qubit parameters for Eq. 5.4 used in the Ising model analysis.

Fidelity vs. number of qudits Increasing the number N of spins also increases the dimension of the matrix U_{Ising}. The optimization becomes harder and the control pulses incorporate higher and higher frequencies. Therefore is necessary to study how the CPR method scales with the dimension of the system.

We take Ising models with increasing number N of spins with two parameters, the values h and $J_i = J$ for every $i = 1, ..., N$. The number Z of subintervals of I is set to 9. Hence, in this case, we have a 2-dimensional grid $\Lambda_m^{a,b} = [J^a, h^b]$ with $a, b = 1, ..., 9$. We prepare the CPR method dataset computing $\mathcal{E}_z^N(t, \Lambda_m)$ for $N = 2, .., 5$. The average fidelity of the datasets computed with GRAPE is $0.981 \pm 1.4e - 4$ for all N. The scaling is tested with the following analysis. We sample new values $\tilde{J}$ and $\tilde{h}$ from a uniform probability distribution in the interval I and, based on that values, we compute the controls both with CPR method, $\tilde{\epsilon}_z^N(t, [\tilde{J}, \tilde{h}])$, and GRAPE algorithm, $\epsilon_z^N(t, [\tilde{J}, \tilde{h}])$. We then compute the mean square error (MSE), ε, between the two controls. The MSE ε is defined as:

$$\varepsilon = \frac{1}{n_{ts}} \sum_{i=1}^{n_{ts}} \left(\tilde{\epsilon}_z^N(t_i, [\tilde{J}, \tilde{h}]) - \epsilon_z^N(t_i, [\tilde{J}, \tilde{h}]) \right)^2, \tag{5.5}$$

where n_{ts} is the number of time steps t_i into which the signals are actually divided. Moreover, we reconstruct the propagator, $\tilde{U}_{Ising}([\tilde{J}, \tilde{h}])$, induced by the interpolated controls $\tilde{\epsilon}_z^N(t, [\tilde{J}, \tilde{h}])$, plugging them into the right hand side of Eq. (5.1). In this way we can compute the fidelity (Eq.(2.32)) between the exact propagator $U_{Ising}([\tilde{J}, \tilde{h}])$ defined by Eq.(5.3), and the one induced by the controls, i.e. $\tilde{U}_{Ising}([\tilde{J}, \tilde{h}])$, found with Eq. (5.1), so to check their degree of similarity. This procedure is repeated 50 times for every N and the results are averaged. In Fig. 5.1 the plot of the reconstruction fidelity vs. the MSE is reported. As expected, the reconstruction fidelity decreases, on average, as the number of spins increases while the error increases. However, the MSE remains very limited in absolute value and the fidelity decrease remains small compared to the dataset's average fidelity of 0.981. Moreover, the decreasing trend suggests that fidelity degradation slows down as the number of spins increases.

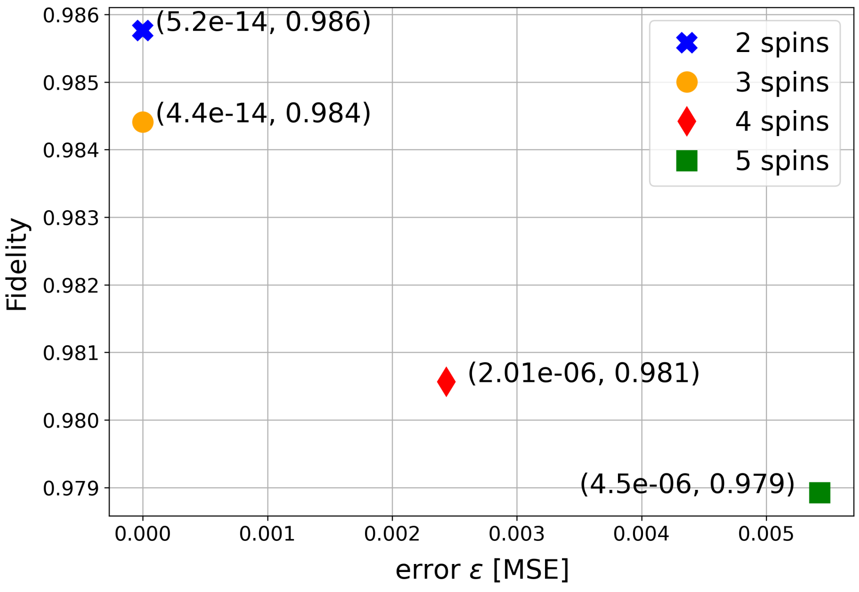


Figure 5.1: The x-axis represents the average mean squared error (MSE) between the controls optimized with GRAPE and the controls interpolated with the CPR method for the same sampled set of the parameters h and J. The y-axis, instead, represents the average fidelity between the exact propagator U_{Ising} and the reconstructed one, $\tilde{U}_{Ising}$ with the right-hand side of Eq. (5.1). Each point refers to a chain with a different spin number N.

Fidelity vs. number of Hamiltonian's parameters Hamiltonian can depend on multiple parameters λ_i so it is of interest to study the scaling of CPR method with respect to this number. We study its behavior on a 3 spins chain taking as λ_i parameters J_1, J_2, J_3 and h. Each of them varies in the interval I divided in $Z = 9$ discrete values. We compute the first datasets $\mathcal{E}_k^1(t, \Lambda_m^1)$ varying only h and fixing J_1, J_2, J_3 equal to 1, the second datasets, $\mathcal{E}_k^2(t, \Lambda_m^2)$, varying only h and J_1 while fixing J_2, J_3 equal to 1, and so on till all the four parameters are used. The time for dataset computation grows (exponentially) with the number of parameters since the number of controls in a dataset is $\prod_i^{n_{par}} Z$, where n_{par} is the number of parameters in use. The fidelity of the dataset controls is 0.981 for each n_{par}. Then, as in the previous paragraph, we take new random values $\tilde{h}, \tilde{J}_1, \tilde{J}_2, \tilde{J}_3$ and we interpolate the new controls $\tilde{\epsilon}_z^{n_{par}}(t)$ with CPR method for increasing n_{par}. We then reconstruct the propagator $\tilde{U}_{Ising}$ induced by these controls via Eq.(5.1) and we compute its fidelity with respect to the exact one obtained via Eq.(5.3) for the same sampled parameters values. The process is done 50 times for each $n_{par} = 1, 2, 3, 4$ and the results are averaged. In Tab. 5.2 the results are reported. In the first column the average fidelity of the controls in the datasets $\mathcal{E}_k^{n_{par}}(t, \Lambda_m^{n_{par}})$

n_{par}	aver. fid. dataset	aver. fid. sampling
1	0.981 ± 1.3 e-4	0.981 ± 2.4 e-4
2	0.981 ± 1.2 e-4	0.981 ± 5.4 e-4
3	0.981 ± 6.4 e-4	0.981 ± 1.2 e-3
4	0.981 ± 4.5 e-5	0.982 ± 2.3 e-4

Table 5.2: Average fidelity for a 3-spin Ising chain for increasing number of Hamiltonian parameters. First column: average fidelity of the datasets controls. Second column: average fidelity of CPR interpolated controls with uniformly sampled J_i and h.

is reported. In the second column, the average fidelity of the controls-induced propagators for random sampling of J_i and h is reported. It can be seen that the average fidelity for the interpolated controls remains high and comparable with the fidelity of the dataset (obtained with GRAPE). The data suggest that the CPR method is capable of obtaining controls with a good fidelity for every configuration, showing a good behavior for increasing number of Hamiltonian's parameters.

Fidelity vs. density of Λ_m grid Another test to characterize the method concerns the behaviour of the reconstruction fidelity as function of the interval I discretization density. We take a 3 spins closed chain with two parameters, h and $J_i = J$ for every $i = 1, 2, 3$, and we compute the datasets $\mathcal{E}_k^Z(t, \Lambda_m)$ for the interval I divided in $Z = 3, 5, 10$ and 15 subintervals. The dataset fidelity was again set to 0.981 for every case. As in the previous case we sample a new value of the parameters, we interpolate a new control and we compute the fidelity of the reconstructed propagator with respect to the exact one. This is repeated 50 times for each Z and the results averaged. In Tab. 5.3 the results are reported using the same format as the previous case. As expected, the CPR method fidelity is low for small Z as the linear interpolation become imprecise since is performed over distant grid points. However, the analysis indicates that the fidelity saturates quickly with increasing discretization density Z.

Computational cost and time

With regard to calculation time, careful considerations must be made. In fact, the method requires calculating a certain number of controls in advance, and this can result in a prohibitive calculation time.

The average time t_G to obtain a control with GRAPE depends exponentially on the number of Hamiltonian dimensions (i.e. quantum device levels or qubit in use) and depends linearly on the number of control's time-steps (i.e. the sampling

n_{par}	aver. fid. dataset	aver. fid. sampling
3	0.981 ± 4.7 e-4	0.952 ± 4.1 e-3
5	0.981 ± 2.7 e-4	0.977 ± 1.9 e-3
10	0.982 ± 1.9 e-4	0.983 ± 3.5 e-4
15	0.982 ± 9.5 e-4	0.983 ± 1.5 e-4

Table 5.3: Average fidelity for a 3-spin chain with two parameters of the Hamiltonian, $J = J_1 = J_2 = J_3$ and h for increasing parameters interval I spacing Z. The data format is the same as the Tab. 5.2.

frequency multiplied by the control time duration). As an example, in Fig. 5.2 the time t_G to compute 1600 time-steps controls for an increasing number of device quantum levels is reported. Instead, the average time, t_{CPR}, that CPR method takes to compute a single control is 0.78 seconds regardless of the number of levels/qubits, since is a simple interpolation. These values refer to a mid-range computer with a 2.9 GHz processor with 4 cores and 8 GB of RAM.

The total number A of controls in a general controls dataset $\mathcal{E}_k(t, \Lambda_m)$ is:

$$A = \sum_{i=1}^{K} Z_i, \tag{5.6}$$

with K the number of Hamiltonian parameters λ_i and Z_i the number of discrete values of interval I_i. The total time T_G to compute the A controls with GRAPE is clearly:

$$T_G = t_G A = t_G \sum_{i=1}^{K} Z_i. \tag{5.7}$$

To gain a computational advantage in using CPR method over using GRAPE, the total number P of controls one want to get from the use of CPR method in a simulation must be larger than A. Possibly, this difference should be high.

Let's make two examples assuming we have an advantage if $P > A$. Let $t_G = 53.6$ s (See Fig.5.2) and $A = 500$. To obtain an advantage, the time to compute P controls with GRAPE, must be greater than the time to compute them with the CPR method plus the time to compute in advance A controls of the dataset. So $t_G P > (t_{CPR}P + t_G A)$, from which is easy to derive that $P > 507$.

Conversely, Let $P = 1000$ be the number of controls one uses in a simulation. Using GRAPE, one would take $t_G P = 53,6s \times 1000 = 53600s \approx 14h$. Otherwise, using the CPR method with $A = 500$, one takes $t_G A + t_{CPR} P = 53,6s \times 500 +$

$1000 \times 0.78s = 27508 \approx 7.6h$. Actually, the advantage is even higher because thanks to a careful choice of the initial guess of the GRAPE algorithm and by exploiting parallel computing routines, the time to compute the A controls of the dataset can be considerably reduced. This is in general not possible without using the CPR method because the controls are computed in series for the subsequent values that the parameters of the Hamiltonian take during the dynamics of the system under analysis.

Thus, in general, the use of the CPR method requires careful consideration and the choice of the appropriate trade-off between dataset size, precision and calculation time.

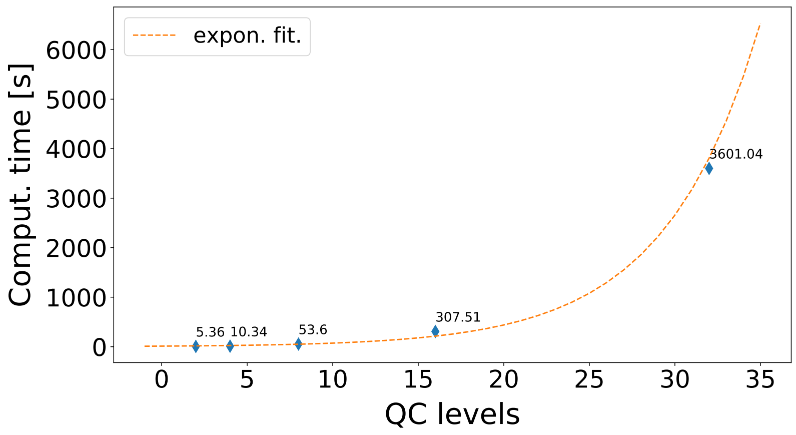


Figure 5.2: Computational time vs. the total number of levels of a quantum device. The single time data (diamond markers) are computed for 2^N QC levels with $N = [1, 2, 3, 4, 5]$ (which identify the total number of levels for 1,2,3,4 or 5 qubits). Computational time grows exponentially with the number of levels, as exponential fitting highlights (dashed line).

5.2.3 Time-dependent simulations with CPR Mehtod: Two neutrons dynamics

Having defined and characterized the CPR method, we now wish to present its application to a concrete case of quantum simulation. Let us therefore consider the two-neutron system introduced in Sec. 3.3 and in particular the time-dependent case of Sec.3.3.2.

For the scope of this work, we assume to work with a single four-level qubit, as in Ref [89, 44]. The Hamiltonian of whole the system takes a particular form:

$$H^{qd} = \omega b^\dagger b + \frac{\alpha}{2} b^\dagger b(b^\dagger b - 1) + \epsilon(t)(b^\dagger + b) \tag{5.8}$$

where we used $\omega = 5.114/2\pi\ GHz$ and $\alpha = -0.33\ GHz$. To simplify the optimization procedure, we move to the rotating frame close to the drive frequency ω_d [103, 44, 89] obtaining the Hamiltonian:

$$\begin{aligned} H^{qd} = \quad & \Delta b^\dagger b + \frac{\alpha}{2} b^\dagger b (b^\dagger b - 1) + \\ & \epsilon_I(t)(b^\dagger + b) + i\epsilon_Q(t)(b - b^\dagger), \end{aligned} \tag{5.9}$$

where $\Delta = \omega - \omega_d$ is the detuning of the qubit frequency ω from the drive frequency ω_d and $\epsilon_I(t)$ ($\epsilon_Q(t)$) is the in-phase (quadrature) component of the original control $\epsilon(t)$ in the rotating frame. $\Delta = 0$ since we choose $\omega_d = \omega$.

The use of this qudit approach aims to demonstrate how this optimal control protocol is particularly useful in the case of multi-level devices, where standard gate-based approach is not naturally applicable. Multi-level qudits, although technically more difficult to realize, offer the advantage of having more levels to encode information [104, 105]. This makes it possible to reduce the total number of usual qubits since more space is already available in the icreased number of levels. Consequently less couplings between qudis are needed, which are often source of noise and error. Furthermore, with the analysis of the present section, we would like also to show how CPR method can be applied naturally to this type of devices.

As already presented in those sections, the hybrid classical-quantum simulation procedure, for every time-step starting from the initial condition, is:

1. Update the relative position $\mathbf{r}_i$ $i > 0$ of the two neutrons with a classical algorithm on a classical computer.
2. Evaluate $\hat{U}_{SD}(\delta t, \mathbf{r}_i)$ for the new position $\mathbf{r}_i$.
3. Optimize the control pulses $\epsilon_R(t)$ and $\epsilon_I(t)$ implementing $\hat{U}_{SD}(\delta t, \mathbf{r}_i)$.
4. Update the spin state $|s_i\rangle$ on the quantum processor using the just optimized control pulses.

Where the spatial trajectory of the particles is obtained by solving the Newton equation with a simple Crank–Nicolson algorithm relying on the saddle point approximation.

In the aforementioned procedure, Step 3 is the bottleneck of the simulation. In fact, the optimization algorithm is computationally expensive depending exponentially on the dimension of the unitary propagator. Moreover, specifically in this application, $U_{SD}(\delta s, \mathrm{r})$ depends on the position of the particles and, consequently, the correspondent controls must be computed each time-step of the simulation. Furthermore, since many simulations with different initial positions, momenta and spin states must be performed to fully characterize the dynamics of the system, the number of these controls increases considerably. To mitigate this problem, we replace, in Step 3, the GRAPE optimization with the CPR method.

First of all, we fix the origin of the coordinates system on one particle. We use the spherical coordinate (r, ϕ, θ) to represent the relative position $\mathbf{r}$ of the second particle. In this way, the dependence of the spin propagator shift from $\mathbf{r}$ to (r, ϕ, θ). We need now to define the grid Λ_m to compute the sets $\mathcal{E}_z(t, \Lambda_m)$ with $z = I, Q$ indicating respectively the dataset of $\epsilon_I(t)$ and $\epsilon_Q(t)$. In this case, the radial distance, r, is taken in the interval $I_r = [0.001, 2.8]$ fm, where most of the interesting part of the neutron-neutron dynamics takes place, and the interval is discretized in $Z_r = 20$ equidistant points. The azimuthal angle ϕ is taken in its natural domain $I_\phi = [-\pi/2, \pi/2]$ divided in $Z_\phi = 25$ points. The longitudinal angle θ, instead, takes value in the range $I_\theta = [0, \pi/2]$ discretized in $Z_\theta = 13$ points. The range of θ is narrowed to $[0, \pi/2]$ instead of the usual $[-\pi, \pi]$ because we exploit the symmetries of the quantum system. In fact, the controls of the interval $[-\pi/2, -\pi]$ sector are the same as the I_θ interval, while the controls of the $[0, -\pi/2]$ and $[\pi/2, \pi]$ sectors have the same controls as the I_θ interval but reversed. Finally, Λ_m the three-dimensional grid of all possible combinations of $\Lambda' = [r^{a_1}, \phi^{a_2}, \theta^{a_3}]$ with $a_1 \in \{1, ..., Z_r\}, a_2 \in \{1, ..., Z_\phi\}$ and $a_3 \in \{1, ..., Z_\theta\}$.

We compute with GRAPE the datasets $\mathcal{E}_z(t, \Lambda_m)$ with an average fidelity of 0.9999 ± 0.00007. We exploit parallel computing routines to optimize multiple controls at once. The first control is computed for $\Lambda' = [r^1, \phi^{13}, \theta^1] = [0.001, 0, 0]$ taking a all zero initial guess. All the other controls are found using the first optimized control as the initial guess. This makes it possible to obtain a continuously varying family of controls, avoiding the solutions falling into different local minima. This makes the interpolation more effective.

We test this procedure at a *device-simulation level* so once we have reconstructed the appropriate controls $\tilde{\epsilon}_{I(Q)}(t)$, we do not send them into an actual quantum computer but we reconstruct the corresponding U_{SD}^{rec}, with the appropriate form of the right-hand side of Eq. (5.1), and we use it to advance the spin state as $|s_{i-1}\rangle = U_{SD}^{rec} |s_i\rangle$. The simulation is performed without adding a noise model. This is done to isolate the effect of the introduced method on the simulation and to better evaluate and characterize the results. In principle, however, the optimal control approach could be set to optimize the controls with the noise model to obtain noise-resistant controls, but this is beyond the scope of this work. The Eq.(5.1) to obtain U_{SD}^{rec} in this case specializes as:

$$
\begin{aligned}
U^{rec}_{SD}(\delta s, [r_i, \phi_i, \theta_i]) &= \exp\left\{ -i\delta\tau \sum_{t_i=0}^{N_\tau} \frac{\alpha}{2} b^\dagger b(b^\dagger b - 1) + \tilde{\epsilon}_R(t_i)(b^\dagger + b) \right. \\
&\quad \left. + i\tilde{\epsilon}_I(t_i)(b - b^\dagger) \right\}. \qquad (5.10)
\end{aligned}
$$

Finally, concerning the initial conditions, in this work we chose $\mathbf{r}_0 = [r^x_0, r^y_0, r^z_0] = [0.5, -2, 0.5]$ fm, an initial velocity of the particle of $\mathbf{v} = [v^x_0, v^y_0, v^z_0] = [0.5, -1, 0]$ fm/MeV and $|s_0\rangle = |0\rangle \equiv |\uparrow\uparrow\rangle$.

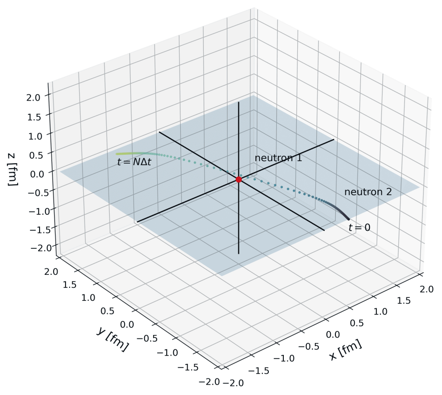
Summarizing, the complete simulation procedure with CPR method, for each time-step starting from initial conditions, becomes:

1. Update the particles relative position $\mathbf{r}_{i+1}$ with a classical algorithm.
2. Use the CPR method to interpolate the controls correspondent to new position $\mathbf{r}_{i+1}$ (expressed in terms of the parameters $(r_{i+1}, \phi_{i+1}, \theta_{i+1})$).
3. Obtain $U^{rec}_{SD}(\delta s, [r_{i+1}, \phi_{i+1}, \theta_{i+1}])$ with Eq. (5.10)
4. Use U^{rec}_{SD} to update the spin state as $|s_{i+1}\rangle = U^{rec}_{SD}|s_i\rangle$.

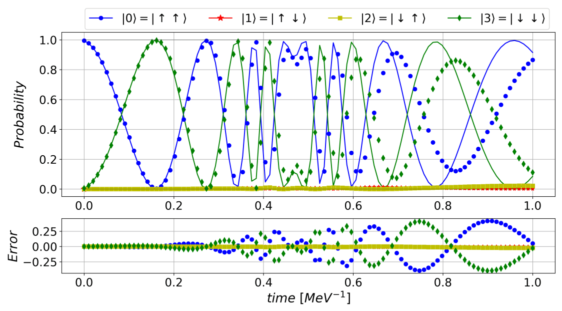
Spatial trajectory is obtained by solving the Newton equations of motion of the particles with a simple Crank–Nicolson algorithm.

Results

We carry out the simulation, following the procedure just introduced, for a 100 timesteps trajectory with a $\delta s = 0.01$ MeV^{-1}. The results of this simulation are shown in Fig. 5.3. The panel (a) reports the trajectory of the second particle with respect to the first (on which the coordinates are fixed). The colors represent the quantum system's time. The panel (b) reports the spin dynamics in terms of occupation probability, computed with Eq. (3.25), along the spatial trajectory represented in panel (a). The solid lines represent the exact spin dynamics found using the exact propagators $U_{SD}(\delta s, \mathbf{r}_i)$, instead the dots represent the spin state for subsequent applications of $U^{rec}_{SD}(\delta s, [r_i, \phi_i, \theta_i])$ starting from initial state $|s_0\rangle$ with the controls obtained with CPR method. It therefore represents the dynamics we would obtain using the controls in the quantum processor (without the noise of the real device). As can be seen, the dotted spin dynamics follows the exact reference dynamics till half of the simulation. This is due the fact that in that position, the velocity of the particles is high and the reconstruction becomes imprecise. This results in a an accumulation of errors at the end of the simulation



(a) Neutrons spacial trajectory



(b) Spin dynamics

Figure 5.3: Representation of the neutrons dynamics. *Panel (a)*: A single realization of a classical spatial trajectory for two neutrons obtained by solving their equation of motion with a Crank-Nicolson scheme starting from a specified initial condition. The origin of axes is fixed on one particle. *Panel (b)*: Spin dynamics in terms of occupation probability corresponding to the trajectory of the panel (a). For every timestep the values are found using Eq. (3.24), as for the case in Sec. 3.3.2

which however does not spoil the dynamic itself but only the delay. This treasure us that this imprecision can be reduced with shorter δs. The average fidelity between the reconstructed and exact propagators is 0.9997 ± 0.0005.

In general, the accuracy of the method can be modulated using different configurations of the CPR method elements. A finer discretization of the intervals results in a increased accuracy. This discretization can also be not evenly spaced. It can be finer only in correspondence of points where Hamiltonian changes most rapidly with respect to its parameters and coarser otherwise. In addition, as in the case just presented, one can take advantage of the symmetries of the quantum system under study to greatly reduce the number of controls to be computed beforehand. In general then, one can find a good trade-off between accuracy and computational cost.

As we mentioned earlier, the main advantage of CPR method is the shorter time required to derive the controls necessary to implement a unitary transformation. We report the data of the simulation. We use 800 time-step controls (i.e. controls with a duration of 50 ns and a signal sampling frequency of 16 GHz). The GRAPE computation time for this case is $t_G = 5.12$ s. The total number A of controls in the datasets $\mathcal{E}_{I(Q)}(t, \Lambda_m)$ is $Z_r Z_\phi Z_\theta = 6500$. The total computational time for this dataset, using parallel computing on a medium-level computer with 2.9 GHz CPU, 4 cores and 16 GM of RAM, is $T_G = 10400$ seconds. This means that each control took on average 1.6 seconds instead of the 5.12 seconds that would take in a serial computation. This means that by exploiting parallel computing and using the same initial condition for every control optimization, we are able to reduce, on average, this time making the CPR method even more efficient. Having more CPU cores, it would be even more efficient. The time to compute a single step of the simulation with GRAPE in this setup, for controls of 800 time-steps and fidelity of 0.9999 , is 6.43 ± 0.02 seconds. CPR method shrinks this time to 0.913 ± 0.002 seconds. Hence, from the computational time point of view, we gain approximately an order of magnitude in using CPR method (for this specific configuration). The CPR method however becomes preferable to GRAPE optimization if the time T_G to compute in advance the whole dataset plus the time $t_C PR$ to compute the P controls needed in the simulation is less than the time to compute the same P controls using GRAPE. Formally, P should satisfy $T_G + t_{CPR}P < t_G P$. This is true if $P > T_G/(t_G - t_{CPR}) \approx 2396$. So, in this configuration, we obtain a net advantage in using CPR method if the number of needed control during the simulation is greater than 2396. This amount is not prohibitive since, to characterize the dynamics of the system, or to calculate quantities such as the cross-section, one needs to calculate many trajectories with different initial conditions of position and spin, so this limit is easily reached.

5.3 Lie Group Theory and Control Pulse Reconstruction

Let us now consider the problem of control interpolation from a more abstract and general point of view. It is known that the propagators $U(\delta t)$ of a quantum system belong to the unitary group $U(N)$, or, neglecting a global phase which does not change the expected values of operators, to its subgroup, the special unitary group $SU(N)$ [106, 107]. If these groups are represented as matrix groups, $SU(N)$ is the group of all unitary matrices with determinants equal to one. The theory of Lie groups says that the Lie group $SU(N)$ has an associated Lie algebra $\mathfrak{su}(N)$. This algebra has a basis of elements, $\{g_i\}$. Given this basis, we can construct every element of the $\mathfrak{su}(N)$ algebra as a linear combination of $\{g_i\}$. An element of the corresponding Lie group $SU(N)$ can be found by exponentiating the arbitrary sum of this basis set, explicitly:

$$U = e^{i\sum_l^{N_g} c_l g_l} \tag{5.11}$$

where N_g is the number of generators and c_l are constants taking values in the real interval $[-\pi, \pi]$.

Now it is also known that the propagators of closed quantum systems belong to the $SU(n)$ groups with the appropriate dimension n determined by their Hilbert space [106]. The idea is therefore to exploit this property to build a method capable of providing the control pulses for any arbitrary propagator. Since, by definition, all propagators of a quantum system are naturally contained in the corresponding $SU(n)$ group, we can consider the elements of the group abstractly and later give them a physical meaning. So we randomly sample a set of group elements, compute the corresponding controls, and then train a neural network to connect them. In this way, the neural network should learn how to generate the controls for each element of the group and, consequently, for each quantum system propagator, we want to implement. This method is therefore system-independent and, after a training process, could help to speed up the compilation of quantum circuits for the simulation of quantum systems in the optimal control framework.

5.3.1 Lie Algebra Based Control Pulse Reconstruction

The group theory framework can be exploited to set up a method to reconstruct the controls of an arbitrary propagator $U(\delta t)$ in matrix form for a set of qubits. This method will be called Lie Algebra based Control Pulse Reconstruction (LA-CPR).

The idea works as follows:

1. Compute a set of group elements $\{U_j\}$ using Eq. (5.11) sampling each c_j from a uniform distribution in $[-\pi, \pi]$.
2. Compute the controls $\epsilon_j^k(t)$ for each propagator U_i to obtain a dataset of controls $\{\epsilon_j^k(t)\}$ (with $k = 1, ..., N_c$ where N_c is the number of device's controls).
3. Define a feed-forward neural network and train it using, as input, the propagator's dataset $\{U_j\}$ and, as output, the controls dataset $\{\epsilon_j^z(t)\}$.
4. Given a new arbitrary propagator $\tilde{U}$ not present in the original dataset, we can find its controls $\tilde{\epsilon}^z(t)$, by plugging it into the neural network just trained.

The initial guess for every optimization is the control corresponding to the matrix obtained imposing all $c_l = 1$. With this choice a group element U consisting of a equally weighted sum of generators is obtained. This appears to be a reasonable choice to have the basic control from which to derive all others.

The actual implementation of the method requires a preparation phase before its use. These are stated explicitly in the following:

Preparation : As the CPR method described in previous section, LA-CPR needs some information in advance. We need to provide the experimental parameters of the device in use, i.e. the specific Hamiltonian H_0 of the qubits system and the N_c control Hamiltonians H_c. Let assume the system has dimension N, so its propagators are elements of $SU(N)$ group. Its associated Lie algebra is $\mathfrak{su}(N)$.

1. Define the basis set $\{g_l\}$ of $\mathfrak{su}(N)$. E.g. for $\mathfrak{su}(2)$ the basis set consist of $\{g_l\}_{l=1,...,3} = \{i\sigma_1, i\sigma_2, i\sigma_3\}$ where σ_l are the Pauli matricex. Instead, for $\mathfrak{su}(3)$, $\{g_l\}_{l=1,...,8}$ is the set of Gell-Mann matrices λ_l.

2. Using Eq. (5.11) compute a set of matrices $\{U_j\}$ sampling c_l elements from a uniform distribution in the interval $[-\pi, \pi]$

3. Using GRAPE algorithm Eq. (5.1), compute the controls $\epsilon_j^k(t)$ for $k = 1, ..., N_c$ for each U_j in the dataset of matrices. We thus obtain N_c datasets of controls.

4. Prepare the inputs and the outputs for the neural network. Since the neural network can only take vectors as inputs, each matrix U_i is flattened stacking the real and the imaginary part of the matrix in a single vector, i.e. $U_j^{flat} = [\operatorname{Re} U_j^{1,1}, ... \operatorname{Re} U_j^{N,N}, \operatorname{Im} U_j^{1,1}, ... \operatorname{Im} U_j^{N,N}]$. The outputs instead are the controls themselves, which are already in the shape of single vectors. All data are normalized between 0 and 1, as common practice in machine learning application.

5. Define N_c feed-forward neural network,$f_k(\mathbf{x})$ with appropriate architecture. This network is then trained to link each U_j^{flat} to the corresponding control $\epsilon_j^k(t)$, i.e. $\epsilon_i^k = f_k(U_i^{flat})$.

Use After the preparation of the method, its use is straightforward.

1. Take any arbitrary matrix $\tilde{U} \in SU(N)$ not in the training dataset.

2. Flatten it in a vector, $\tilde{U} \to \tilde{U}^{flat}$.

3. Plug $\tilde{U}^{flat}$ in the N_c neural networks to obtain the corresponding controls $\tilde{\epsilon}^k = f_k(\tilde{U}^{flat})$ with $k = 1, ..., N_c$.

5.3.2 Results

We report here some results and analysis of the LA-CPR method.

We test this method on a system $N_q = 1, 2, 3$ qubits with one control for each qubit in a closed ring configuration. Hence we have $N_c = N_q$. The Hamiltonians are:

$$\begin{aligned} H_0 &= \sum_{i=1}^{N_q} \omega_i b_i^\dagger b_i + \frac{1}{2}\alpha_i b_i^\dagger b_i (b_i^\dagger b_i - 1) + \\ &\quad g_{i,i+1}(b_i^\dagger b_{i+1} + b_{i+1}^\dagger b_i), \end{aligned} \tag{5.12}$$

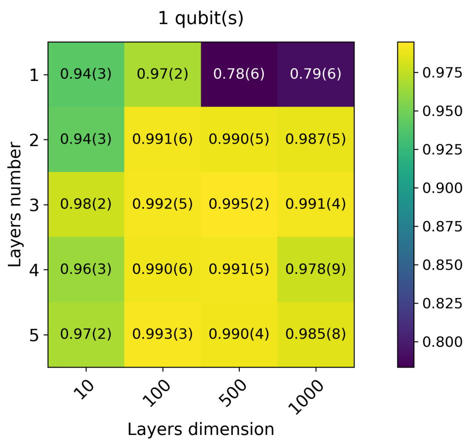
$$H_c = \sum_{k=1}^{N_c} \epsilon_k(t)(b_k^\dagger + b_k) \tag{5.13}$$

where the values of the variables are the same of Tab. 5.1.

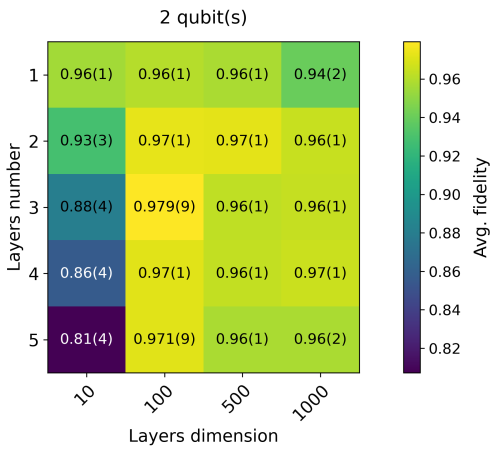
To test the performance of the LA-CPR method in the different configurations, we use the fidelity with the following procedure. After training a neural network (with a given architecture), we use it to reconstruct the controls $\epsilon_k^j(t)$ for a set of propagators U_j not present in the training dataset, we compute the propagators U_j^{recon} that $\epsilon_k^i(t)$ implement by plugging them into Eq. 5.1 and finally we compute the fidelity $\mathcal{F}(U_i, U_i^{recon})$ between each pair (U_i, U_i^{recon}). In this way, we can quantify the quality of the controls reconstructed by the neural network.

Study of the Neural Network Architecture

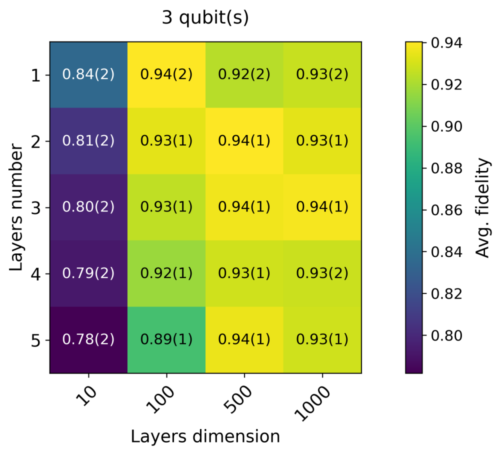
To identify the best neural network architecture for any number N_q of qubits, the performance of neural networks with different depths and widths is studied. A neural network with $N_l = \{1, 2, 3, 4, 5\}$ layers of size $N_D = \{10, 100, 500, 1000\}$ is built and trained on a dataset of 2000 elements. Then the average fidelity of the reconstructed controls is computed on a test dataset of 250 elements. This procedure is repeated 5 times for each combination of N_l and N_D and the results are averaged. For each repetition, the training dataset is randomly sampled from a 20000-element dataset, the test dataset is computed using Eq. (5.11), and the neural networks are initialized with random weights. The values of the input and output data sets are normalized in the interval $[0, 1]$. The activation functions are the sigmoid for the output layer and the swish function for all others. Training is performed by minimizing a mean square error loss function using the Adam algorithm. Also, the maximum number of epochs is set to 100, but an early stopping callback is set to stop the training if the loss does not decrease for 6 epochs in a row. The results of this procedure for different numbers of qubits are shown in Fig. 5.4. Different architectures give different results, especially for 2 and 3 qubits. In the case of one qubit, Fig. 5.4a, the worst results are obtained with a single-layer neural network, but with a large size. Instead, the best combination is 2 layers of 500 neurons each, or (2,500) for short. Instead, in the case of two and three qubits, Fig. 5.4b5.4c, the worst results are obtained for a deep network with 10 neurons for each layer. This is probably due to inefficient training. The best combination is (3,100) for the two-qubit case and (2,500) for the three-qubit case.



(a) One-qubits case. Best architecture: 3 layers of 500 neurons each.



(b) Two qubits case. Best architecture: 3 layers of 100 neurons each.



(c) Three qubits case. Best architecture: 2 layers of 500 neurons each.

Figure 5.4: Average fidelity for neural networks with different combinations of layer size and number of layers in the case of 1,2 and 3 qubits. The architecture that produces the highest average fidelity can be clearly spotted in each case.

Dataset analysis

After identifying the best architecture for each neural network, we can study how the fidelity of the reconstructed control varies for datasets of different sizes. To do this, we take the neural network with the optimal architecture and train it with datasets of increasing size $D = \{100, 500, 2000, 10000, 20000\}$. As in the previous case, we do this 5 times for each dataset size and average the results. Each time, the training data set is randomly sampled from the entire data set of 20000 elements. All other training specifications are the same as in the previous case. The results are shown in Fig. 5.5. The x-axis shows the size of the dataset (in logarithmic scale). The dataset dimension used for training is highlighted in the lower part of the graph. The y-axis instead shows the average fidelity. The three lines represent the fidelity for the three cases as a function of the dataset dimension. The red text on the left reports the optimal architecture used for each case. The one-qubit case shows good fidelity for all values. This is due to the shape of the control, which is not too complicated. In this case, the dataset dimension is stopped at 2000 elements because the fidelity performance was already saturated. The two-qubit case achieves good results for the largest dataset. The three-qubit case achieves worse results than the other cases due to the more complex form of the controls. The controls present in the dataset have been optimized by setting a fidelity of 0.99, so the results obtained by the LA-CPR method are good since the decrease in performance is very limited for all cases. In general, as expected, all the behaviors show an increasing trend with the dimension of the dataset but with a decreasing speed.

5.3.3 Application to quantum system simulation

The purpose of the LA-CPR method is to quickly and easily obtain the control pulses that implement an arbitrary unitary propagator on a quantum computer. Random propagators were tested in the previous section, but the real interest would be in implementing propagators that describe the dynamics of quantum systems. To test this, we consider again the case of the two-neutron system introduced in Sec. 3.3 and tested with the CPR method in Sec. 5.2.3. We recall that the spin-dependent part of the propagator, $U_{SD}(\delta t, \mathbf{r})$, depends on the relative position, $\mathbf{r}$, of the two particles. The matrix $U_{SD}(\delta t, \mathbf{r})$ is, by construction, a unitary matrix belonging to $SU(4)$, so the ability of the LA-CPR method to provide the controls corresponding to this propagator can be tested. To do this, we use the usual procedure: we construct a set of N propagators, $\{U^i_{SD}(\delta t, \mathbf{r})\}$, randomly sampling the positions $\{\mathbf{r}^i = [r^i_x, r^i_y, r^i_z]\}$ from a uniform distribution for each r_x, r_y and r_z (but such that the norm is less than 1.5 nm). We then plug these propagators into the neural networks of the LA-CPR method, flattening them ap-

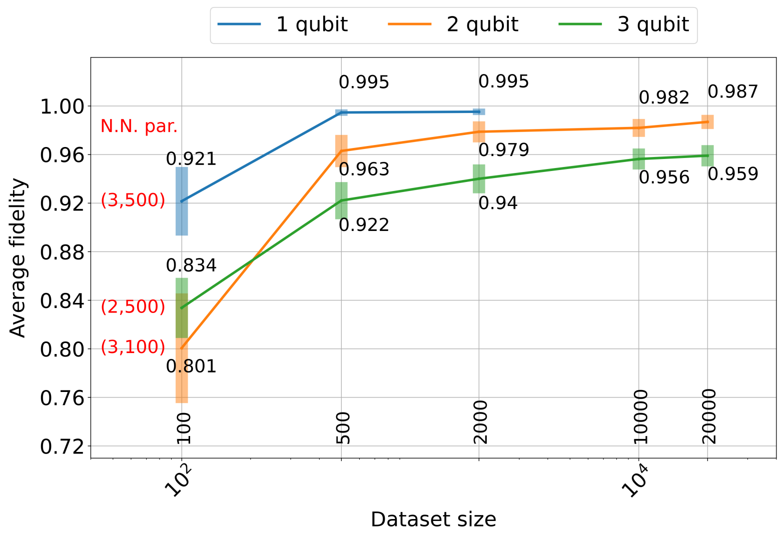


Figure 5.5: Average fidelity and dataset dimension for the one, two and three qubit setup using the optimal architecture (red text on the left). The one qubit case is stopped at 2000 elements because the fidelity was already saturated.

propriately to fit into the input layer, and thus derive the set of corresponding controls $\{\epsilon^i(t;\mathbf{r}^i)\}_k$, where $k = [1, ..., N_{ctrl}]$, as always the index of the number of controls present in the qubit system used. Plugging the controls into Eq. (5.1), one can reconstruct the set of propagators, $\{U^i_{recon}\}$, that these controls implement in the qubit system, and consequently calculate the fidelity between the original propagators and the reconstructed ones.

We followed this procedure with a set of $N = 1000$ propagators and obtained an average fidelity of 0.97 ± 0.05, which is only 0.01 off the average fidelity of the LA-CPR method for the case of two qubits (i. e. $SU(4)$ matrices) for random matrices, see Fig. 5.5, and only 0.03 from the average fidelity of the training dataset for the same configuration, which is 0.99.

This analysis confirms that the LA-CPR method allows to obtain the controls that implement any unitary transformation belonging to a $SU(2^{N_q})$ group on a set of N_q qubits. Thus, LA-CPR emerges as a promising method to achieve speedup in the simulation of quantum systems, where a large number of controls related to different propagators are required.

Chapter 6

Qubit Readout with Autoencoders

6.1 Introduction

In order to have a working quantum computer, an important requirement is a high-fidelity readout of the qubits. This corresponds to Di Vincenzo's fifth criterion. The measurement of the qubits is necessary to extract the information about the state of the qubits after a computation, especially for observables that are very sensitive to it (see e.g. [108] for an extreme case of this).

The currently most common qubit readout technique for superconducting qubits is the **dispersive readout** which couples the qubit to a readout resonator. In this approach, the state of the qubit is determined by measuring the changes in phase and amplitudes of an electromagnetic field transmitted through the resonator [109, 110, 12, 11]. Hardware, random thermal noise, decoherence introduced by the measurement process itself, or qubit decay processes that occur during measurements may reduce the readout fidelity. In addition to a careful design of the system parameters [111, 112] or improvement in fabrication processes extending qubits coherence time [113, 114], readout fidelity can be enhanced through the use of machine-learning techniques that works by improving the classification accuracy in assining the readout signal to the right state of the qubit. Many different techniques and architectures can be used at this scope. Gaussian mixture model [98] is the most commonly used classification method given its ease of use. It exploits parametric modeling of the averaged readout signals probability distribution in terms of a sum of Gaussians to perform a classification of each measurement. In [115, 116, 117, 118], instead, the authors developed and implemented various classification methods based on neural networks trained on the full dynamics of the measurement, instead of on their averages, obtaining good results. Another more speculative approach uses the hidden Markov model proposed in [14], which allows for a detailed classification of the measurement results and detection of the decay processes that the qubit could undergo during the measurement.

In this chapter, a novel semi-unsupervised machine learning classification method based on autoencoder pre-training applied to the heterodyne readout signal of a superconducting qubit [109, 119] (See Sec. 2.4) is presented. As mentioned in Chapter 4, autoencoders are a type of artificial neural network designed to encode efficiently a set of data by learning how to regenerate them from a synthetic encoded representation [120, 121]. The encoding process automatically isolates the most relevant and representative features of the input dataset, i.e. those features which allow for the most faithful reconstruction of input data while neglecting noise and non-relevant details [122, 123]. Hence, this method exploits this characteristic of autoencoders and performs data classification not on the readout signals or on their time average, but on their encoded representation produced by autoencoder training. In this chapter, the performance of this method is presented, together with a comparison with the other most common classification method. It is shown how this method can enhance the state classification of readout signals, especially for short readout times where other more traditional methods have worse performance and, in general, shows a more stable performance for a broad range of measurement time lengths. Clearly, the most significant improvements are obtained with a combination of hardware and software improvements, as obtained by the authors in Ref. [124], however, in this dissertation, the focus will be only on software improvement on present machines.

6.2 Heterodyne readout of transmon qubit

We consider a transmon-type qubit coupled to a detuned resonator (i.e. a quantum harmonic oscillator) in the context of a strong projective dispersive measurement scheme [110, 12]. Due to the qubit interaction, the readout resonator undergoes a frequency shift whose value depends on the qubit state. This dependency can be exploited to perform measurements of the qubit state in the dispersive regime i.e. when the detuning of qubit and resonator is large relative to their mutual coupling strength [125]. Once the resonator is irradiated with a specific microwave pulse, the registered transmitted signal will incorporate different amplitude and phase shifts based on the qubit's state. The demodulation procedure can extract such information from the signal, discriminating between qubit states.

Our setup consists of a superconducting qubit controlled by the Quantum Orchestration Platform (QOP) programming environment (Q.M Technologies Ltd.) through the QUA programming language based on python [126]. In this setup, a signal of amplitude A and frequency ω_r is sent into the readout resonator. In

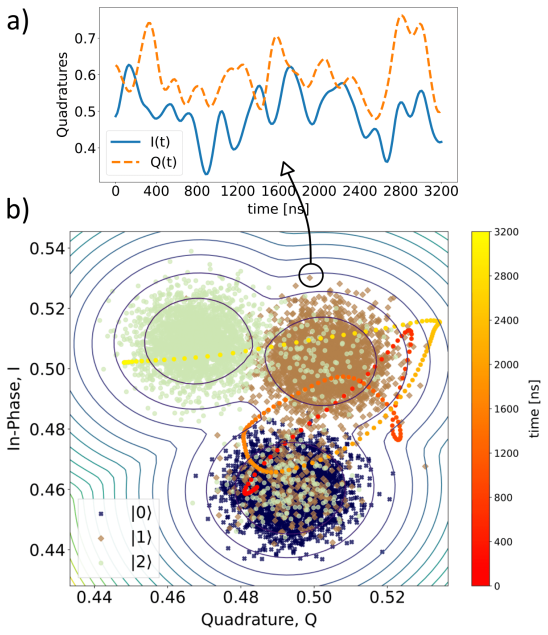


Figure 6.1: Pictorial representation of qubit readout data. *Panel a* Example of in-phase, I(t), and quadrature, Q(t), components of heterodyned signal of a single shot obtained via sliced demodulation (as described in Sec.6.3). The average of these signals is a single point in the I-Q plane below. *Panel b* Example of the whole dataset. Each point is the time average of a measurement represented in the I-Q plane for qubit states 0,1, and 2. The lines represent the 2D Gaussian contour plot (see Sec. 6.4) for the 3 Gaussian distribution. The dotted red-yellow line is an example of a measurement signal represented in the I-Q plane. The colors represent the time evolution (in nanoseconds).

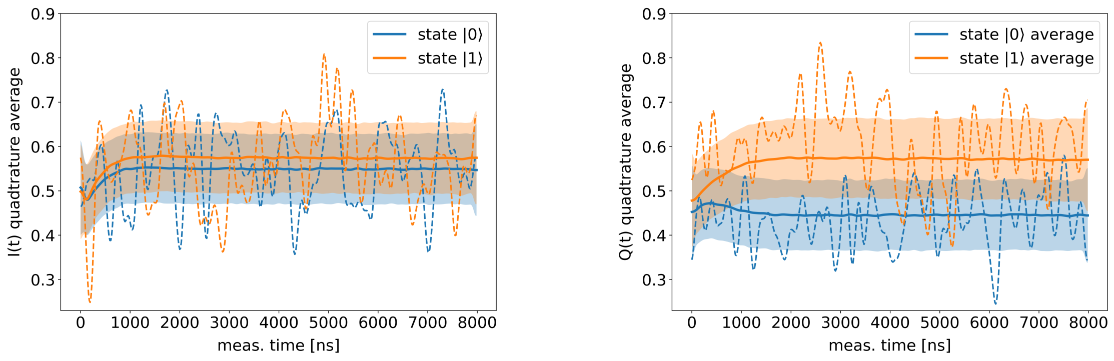
interacting with the system, this signal is modulated by the resonator's response. The output signal is then filtered, amplified, and down-converted to an intermediate frequency $\omega_{IF} = \omega_r - \omega_{LO}$ through a signal mixer, with ω_{LO} the frequency of the local oscillator (an electronic component needed by the mixer to change signals frequency). Finally, it has to be demodulated to extract information about the qubit state that the readout signal acquired in the interaction. To do this we recall the demodulation equations 2.52 defined in Sec. 2.4 i.e.:

$$I = \frac{1}{T_m}\int_0^{T_m} r(\tau)\cos(\omega_{IF}\tau)d\tau \tag{6.1}$$

$$Q = -\frac{1}{T_m}\int_0^{T_m} r(\tau)\sin(\omega_{IF}\tau)d\tau, \tag{6.2}$$

where the readout signal is denoted by $r(\tau)$ and T_m is the integration time.

This will be denoted *full demodulation* because it is performed by integrating over time intervals T_m and returns a single value for the I and Q components for each qubit readout signal. In this way, each measurement can be represented as a point in the I-Q plane. Thanks to the qubit state-dependent frequency shift, these points will accumulate in different zones of the I-Q plane. An example is displayed in panel (b) of Fig. 6.1 where the points for a three-level qutrit are reported.



(a) In-phase, $I(t)$ signals (b) Quadrature, $Q(t)$ singals

Figure 6.2: Average readout trajectories for state $|0\rangle$ and $|1\rangle$ in both quadrature. Solid lines represent the mean of all trajectories in the data set for state $|0\rangle$ (blue) and state $|1\rangle$ (orange). The shaded regions represent the standard deviation of the average for each timestep. The dashed line instead represents an example of a single trajectory.

However, an alternative approach can be employed. This is denoted *sliced demodulation* and it collect the digitalized I(t) and Q(t) signals (See Eq (2.54)) so it retains "slices" of the full demodulation. In this way, instead of having a single point (I, Q) we have the full dynamics of the measurement $(I(t), Q(t))$.

From the practical point of view, each measurement is obtained by preparing the device in states (e.g. $|0\rangle$ or $|1\rangle$) and then by measuring it immediately, storing the obtained signals. The selection of the time windows Δt for the sliced demodulation requires careful consideration. The demodulation time-step Δt should span an integer number of periods of the readout signal to avoid imprecise demodulation. The frequency of the readout signal is $\omega_{IF} = 60$ MHz, so its period is $1/\omega_{IF} \approx 16$ ns. For this reason, in this work, we took a time window $\Delta t = 16$ ns. Hence, each readout signals $(I(t), Q(t))$ have a point every 16 ns. The length of the measurement, T_m is also an essential parameter. Here we choose to consider measurements of increasing length starting from 800 ns up to 8000 ns, corresponding to discrete signals whose number of elements spans from 50 to 500, to study the efficiency of the classification methods in different configurations. The collection of $I(t)$ and $Q(t)$ signals are then smoothed with a window smoothing algorithm with a Hanning window of 50 timestep length to remove some noise.

In Fig. 6.1, panel (a), the $I(t)$ and $Q(t)$ signals of a single readout signal are represented as examples. Averaging these signals we obtain a single point in the I-Q plane as represented in the panel (b). The red-yellow line in panel (b) represents the $I(t)$ and $Q(t)$ signals plotted together as a trajectory (state-path trajectory). The color gradient represents time. In Fig. 6.2, instead, the average signals $\langle I(t)\rangle\rangle$ and $\langle Q(t)\rangle$ (solid lines) are reported together with the standard deviation for each timestep (shaded range). The same graph also shows individual readout signals (dashed lines). As can be seen, the machine's noise is high as the standard deviation zones heavily overlap. One of the aims of this work, however, is to show how the proposed method can deal with this noise and improve, in any case, the classification of the measures.

In principle, the sliced demodulation should retain information that otherwise is lost in the averaging process of the full demodulation. This information will be exploited in this work to increase state detection accuracy. Usually, in full demodulation, the readout accuracy is adjusted and maximized by tuning the readout length, i.e. the demodulation integration time T_m. The aim is to obtain clouds of points (as in Fig. 6.1) with a distribution that is as Gaussian as possible to use the Gaussian Mixture Model to perform the classification. In fact, short integration times produce poorly distinguishable states, while for long times, the qubit states tend to decay during the measurement, which produces a non-Gaussian data distribution and, again, low classification accuracy. In contrast to full demodulation, sliced demodulation retains more information about the qubit state measurements and, in principle, allows for increased accuracy of the state classification. Moreover, as will be observed in this work, it reduces the dependence of the classification result on T_m since the data do not need to be Gaussian distributed.

It should be mentioned that data preparation is not error-free. Indeed, it may happen that, due to control errors or environmental coupling, the state ($|0\rangle$,$|1\rangle$ or $|2\rangle$) that is expected is not actually prepared. There will therefore be cases in which a state, although labeled with a certain state, actually belong to another one. So the classification will not be 100% accurate even with very sofisticated method because the device, and hence the dataset, suffers from this flaw.

6.3 Model: Neural Network with Autoencoder type Pre-training

In this work, we propose a classification model based on a neural network with an autoencoder pre-training which we denote "*PreTraNN*". It is composed of two sections.

The first section consists of an encoder f_{θ^e} whose parameters θ^e are pre-trained in advance as an autoencoder over the input dataset. The encoder consists of two layers with L_1 and L_2 neurons and a third layer, the latent layer, with L_H neurons. The decoder g_{θ^d} necessary for the pre-training has the same structure as the encoder but is in reverse order. Given a input of dimension d, we always set $L_1 = \frac{3}{4}d$, $L_2 = \frac{2}{4}d$ and $L_H = \frac{1}{4}d$. The activation functions are the *sigmoid* for the first layer of the encoder (and the last layer of the decoder) and the *tanh* function for all the internal layers. The choice of internal layer size is explained in Appendix C.1.1 while the complete specifications of the autoencoder are reported in Appendix C.2.

The second section is a feed-forward neural network, N_ϕ, dependent on a set of parameters ϕ which works as a classifier taking as inputs the feature-vector of the encoder and, as outputs, the exact labels of the readout signals. It is composed of two hidden layers with L_{N_1} and L_{N_2} neurons, respectively, and an output layer with a number of neurons equal to the number of data classes. Given d the dimension of the input, we set $L_{N_1} = 2d$ and $L_{N_2} = d$. The activation functions are *tanh* for the internal layers and the *softmax* for the last layer, commonly employed for classification purposes.

The assignment of the label $\mathbf{y}^i$ to a qubit readout signal $\mathbf{x}^i(t)$ works as follows:

1. The discrete signal $\mathbf{x}^i$ is flattened by stacking the I and Q components in a single one-dimensional vector, i.e $\mathbf{X}^i = [\mathbf{x}^i_I, \mathbf{x}^i_Q]$ so it can be plugged into the neural network.

2. The input $\mathbf{X}^i$ is transformed in the feature-vector $\mathbf{h}^i$ via the encoder function, i.e. $\mathbf{h}^i = f_{\theta^e}(\mathbf{X}^i)$.

3. The feature-vector $\mathbf{h}^i$ is plugged into the feed-forward neural network N_ϕ to be assigned to one out of the three classes. Formally, $N_\phi(\mathbf{h}^i) = \hat{\mathbf{y}}^i$ where $\hat{\mathbf{y}}^i$ is the predicted label for the input $\mathbf{X}^i$.

A pictorial representation of the PreTraNN classification working principle is displayed in Fig. 6.3.

Training

The training is performed separately for the two sections that compose the PreTraNN model.

The autoencoder is trained first. The dataset is composed by inputs $\mathbf{x}^i$ with $i = 1, 2, ..., M$, representing the 2D trajectories in the I-Q plane. The neural network architecture requires a one-dimensional vector input so $\mathbf{x}^i$ need to be flattened, stacking the I and the Q components in a single one-dimensional vector.

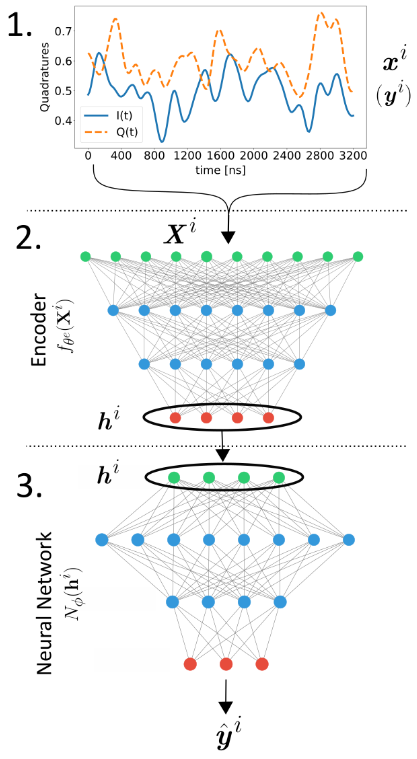


Figure 6.3: Pictorial representation of the working principle and the architecture of the PreTraNN method described in Sec. 6.3. *Section 1*: Example of the measurement signal $\mathbf{x}(t)$ we want to classify with PreTraNN. *Section 2*: The input $\mathbf{x}(t)^i$ is flattened to obtain $\mathbf{X}^i$, plugged into the encoder, previously trained as an autoencoder, and transformed into its encoded representation $\mathbf{h}^i$. *Section 3*: The latent layer of the encoder,$\mathbf{h}^i$ is passed into a feed-forward neural network trained to assign the label $\hat{\mathbf{y}}^i$.

So we compose a new dataset of $\mathbf{X}^i = [\mathbf{x}_I^i \ \mathbf{x}_Q^i]$. The parameters $\theta = [\theta^e, \theta^d]$ of the autoencoder $A_\theta(\mathbf{x}^i) = g_{\theta^d}(f_{\theta^e}(\mathbf{X}^i))$ are trained by minimizing Eq. (4.2) where we choose as loss function l the mean square error

$$l = \frac{1}{d} \sum_{t=1}^{d} \left(X^i[t] - \hat{X}^i[t] \right)^2, \tag{6.3}$$

with d the length of the input data $\mathbf{X}^i$ and $\hat{\mathbf{X}}^i = A_\theta(\mathbf{X}^i)$ the reconstructed input.

In a second step, the neural network N_ϕ is trained taking as inputs the feature-vectors $\mathbf{h}^i$ of the encoder f_{θ^e} and, as output, the real labels $\mathbf{y}^i$ of the corresponding $\mathbf{x}^i(t)$. The optimal network's parameters ϕ are obtained by minimizing Eq. (4.1) where the loss function l is chosen to be the cross entropy loss function, widely used in classification.

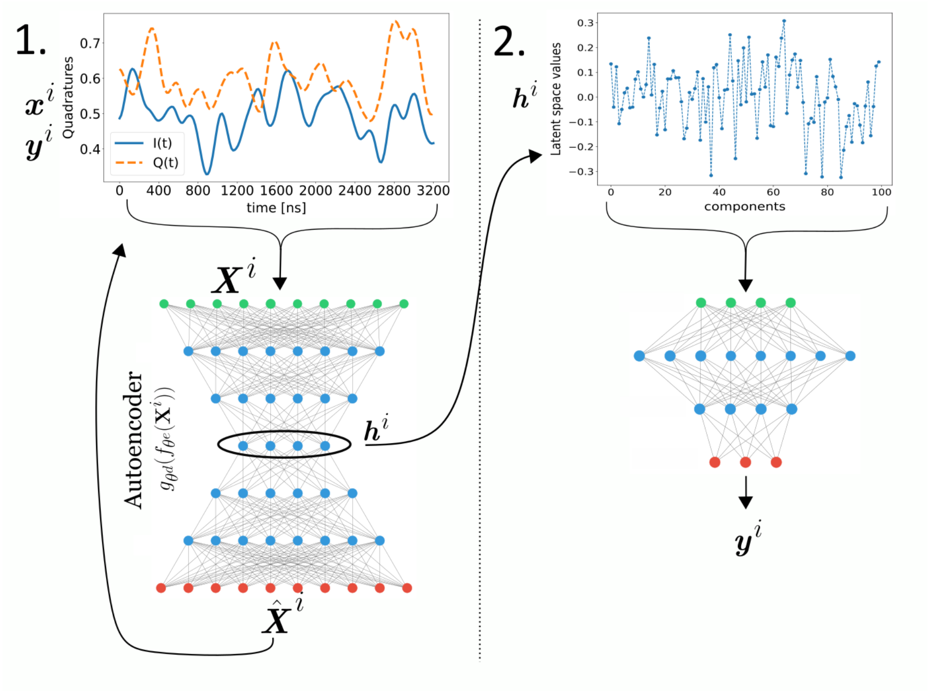


Figure 6.4: Pictorial representation of PreTraNN training described in Sec. 6.3. *Section 1*: The autoencoder is trained to reconstruct the measurement signals. This should train the network to extract the relevant features from each temporal chunk. *Section 2*: After the training, the decoder part of the network is removed, and the encoded representation of data (represented in the plot at the top right) is used as the train input dataset for the second section of the PreTraNN model which is trained to classify them into the correct class $\mathbf{y}^i$

A pictorial explanation of the PreTraNN training procedure is depicted in Fig. 6.4 while a complete specification of the autoencoder structure is reported in Appendix C.2.

Note that although the use of the term 'pre-training', PreTraNN is not a pre-trained model in the general sense. We do not use a bulk neural network pre-trained on a vast quantity of data, attaching to it new layers which are then trained on our specific classification problem. In PreTraNN model, we take as "pre-trained neural network" the encoder part of an autoencoder that was previously trained over our specific readout data.

6.4 Standard Methods and Metrics

The result of the proposed PreTraNN model are compared with two state-of-the-art methods introduced above: the Gaussian mixture model (GMM) and a simple feed-forward neural network (FFNN).

The GMM is trained directly on I-Q points, averages of the readout signal.

The FFNN is, instead, trained over the readout signals dataset, taking as input the flattened vectors $\mathbf{X}^i = [\mathbf{x}^i_I, \mathbf{x}^i_Q]$ and, as outputs, their labels $\mathbf{y}^i$. The architecture of the FFNN consists of two inner layers of dimension $L_{FF_1} = 2d$ and $L_{FF_2} = d$, with d the input dimension, and an output layer. The activation functions are the *tanh* for the internal layer and the *softmax* for the output layer. The structure of the FFNN is the same as the second section of the PreTraNN. The only difference is that while the PreTraNN neural network takes as input the readout signal encoded in the latent space, the $\mathbf{h}^i$ vector, the FFNN takes directly the signals $\mathbf{X}^i$.

6.4.1 Metrics

To measure the accuracy of the classification systems, we utilize the "classification accuracy", i.e. the probability that each signal is attributed to the correct label (i.e. the correct state of the qubit). This classification is obtained as a percentage of correctly attributed signals out of their total number (for each state). The global accuracy is the average of the accuracies of each state.

6.4.2 Datasets

As already mentioned, two versions of the same dataset are used in this work. Each measurement is a two-dimensional $\mathbf{x}^i(t) = [I^i(t), Q^i(t)]$ trajectory that, flattened to form the $\mathbf{X}^i$ inputs (See Sec. 6.3), will form the dataset for the PreTraNN and FFNN. The dataset for the GMM, on the other hand, is obtained by time-averaging each $\mathbf{x}^i(t)$ measurement so as to obtain two values that can be represented in I-Q space (an example of which is shown in panel (b) Fig. 6.1). The dataset is then shuffled and split into train and test datasets in a 75% - 25% proportion.

The size of the dataset impacts the accuracy of the method and needs some consideration to avoid under-fitting or unnecessarily long training times. Such considerations are drawn in Appendix C.1.2.

We emphasize that the readout data all come from the same device. Although it may be interesting to study a multi-device classification system, in general, different devices may show differences in the average behavior of readout trajectories, due to construction or control differences. This clearly makes the training more challenging and it could spoil the results.

6.5 Results

The purpose of this section is to demonstrate how the feature extraction capability of the autoencoder helps improve the effectiveness of qubit readout. So, specifically, how the PreTraNN method performs better than other commonly used methods for readout, namely GMM and a simple FFNN. In this section, PreTraNN and the benchmark methods are compared in terms of classification accuracy and their overall performance is studied.

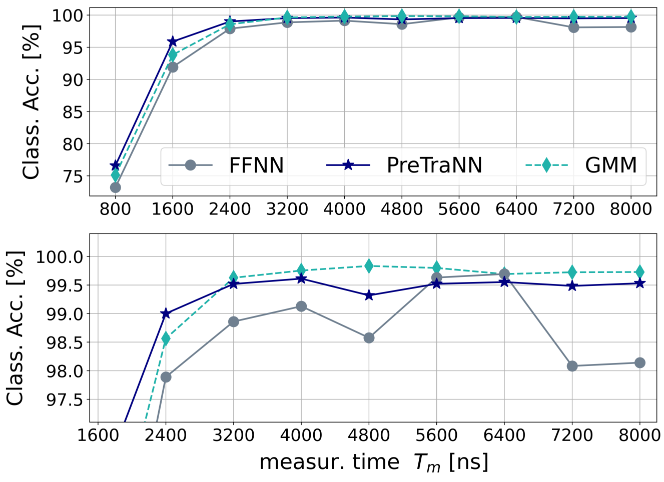
In addition, to deepen the analysis, the application of the models is extended to two readout configurations. The first is the readout of the usual two-level qubit and the second is the readout of a three-level qutrit. This analysis will give an idea of the good scalability of PreTraNN for multiple levels readout.

6.5.1 Two-state qubit readout

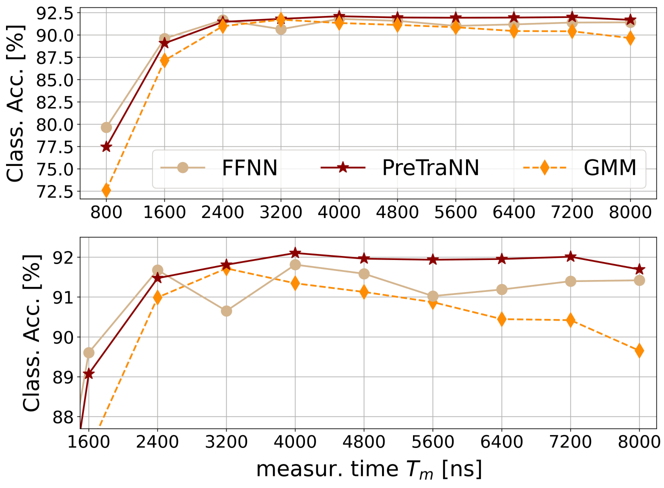
In this case, the qubit is prepared and immediately measured in state $|0\rangle$ and $|1\rangle$. The dataset consists of 16000 readout signals (8000 for each state) and it is split into train and test subsets in 75%/25% proportions. Consideration on the choice of the dataset are drawn in Appendix C.1.2. The PreTraNN, FFNN and GMM setup is the one defined in Sec. 6.3 and Sec. 6.4.

Classification accuracy

We start by showing our results for the classification accuracy of the three methods for increasing measurement length T_m to compare their performance in different cases. All experiments are computed 10 times and averaged. We report the state classification accuracy for each state separately in Fig. 6.5 and the global classification between state $|0\rangle$ and $|1\rangle$ in Fig. 6.6.



(a) *Upper panel*: Classification accuracy for state $|0\rangle$ by the three methods as a function of the measurement time. *Lower panel*: a zoom on the 2400-8000 ns part of the plot.



(b) *Upper panel*: Classification accuracy for state $|1\rangle$ of the three methods as a function of the measurement time. *Lower panel*: a zoom on the 2400-8000 ns part of the plot.

Figure 6.5: Classification accuracy comparison, for state $|0\rangle$ and $|1\rangle$ separately, between Gaussian Mixture Model (GMM), the simple feed-forward neural network (FFNN) and the PreTraNN method . The readout time T_m spans from 800 ns to 8000 ns.

We start by considering Fig. 6.5. In the upper figure, the classification accuracy of $|0\rangle$ state for the three models as a function of measurement length is shown, in the lower figure, the same information is reported but for $|1\rangle$ state. First of all, it can be noted that, for short measurements, all models deteriorate their performance. This behavior should be attributed to the fact that, for short measurement times, the points distributions heavily overlap, preventing all methods, and especially GMM, from fitting them appropriately with two Gaussians (see Fig. 6.7 for an illustrative example). For middle and long measurement times, instead, the GMM performs, respectively, better and worse for state $|0\rangle$ and state $|1\rangle$ than the other two methods. Moreover, the state $|0\rangle$ classification accuracy remains high and stable for long measurements, while that of state $|1\rangle$ presents a descending trend at longer times. This behavior has a simple explanation: the pure qubit excited state (e.g. the $|1\rangle$ state) have leakage to the ground state (the $|0\rangle$ state) at a much higher rate than the opposite direction. As a consequence, there is an asymmetry in the data points distributions. This results in states prepared as $|1\rangle$ to be spotted on state $|0\rangle$ distribution due to the decay process, while the reverse is much more unlikely. Therefore the GMM, fitting the distribution with two Gaussians, can not handle this asymmetry performing very differently in the two cases. The number of signals decaying during the measurement procedure increase with the measurement time and, in fact, the accuracy of state $|1\rangle$ drops for long times. Instead, FFNN has a fluctuating trend, and it performs often worse than GMM. We can speculate that this behavior derives from the fact that, for very large inputs, the training is more difficult, and a simple FFNN does not converge adequately. This suggests that FFNN is not completely adequate for this purpose. On the contrary, the PreTraNN method shows very stable behavior for both states even for long measurement times. It not only uses all the "histoy" of the measurements but also exploit the feature extraction of the autoencoder.

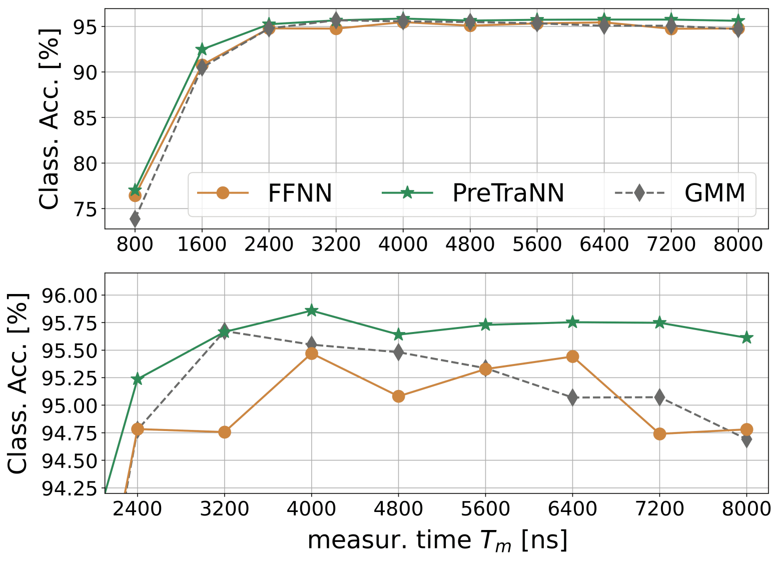


Figure 6.6: Global classification accuracy between state $|0\rangle$ and $|1\rangle$ for increasing measurement time T_m. The accuracy obtained with PreTraNN method is higher (or at most equal) to the ones obtained with GMM and FFNN.

In Fig. 6.6, the global discrimination accuracy between state $|0\rangle$ and $|1\rangle$ is reported. It is obtained averaging the accuracies of $|0\rangle$ and $|1\rangle$ states. In this global case, the PreTraNN method outperforms the GMM and the FFNN methods for every measurement time (except for a measurement time of 3200 ns where GMM's an PreTraNN's accuracies coincide). The considerations of the previous case also apply here.

It can also be noted that GMM accuracy has a global maximum at 3200 ns. As mentioned before, for the GMM to work well, the distribution of I-Q points for each qubit state must be as "Gaussian" and distinguishable as possible. It happens that, for short measurement times, the points distributions overlap since the qubit-resonator response is still in a transient state, while, for long times, decaying processes come into play which makes the distribution skewed. Therefore, we can deduce that the length of 3200 ns produces the least overlapping distributions that allow the GMM to reach the greatest accuracy. This measurement time is therefore the one that should be set for the readout in case of GMM use. The PreTraNN method makes the need for this adjustment less strict since it works well for a larger interval of the experimental parameters T_m. In general, it can be seen that, in PreTraNN method, the classification accuracy is only increasing or constant. As a consequence, the trimming is faster and easier since the need of finding the maximum accuracy is removed.

We want also to stress that in other works, such as Ref. [115], the readout accuracy may be greater than the one reported here. As described above, the machine used for this work has a certain level of error in preparing state $|1\rangle$. This, however, is of secondary importance since the purpose of the present work was not to present new hardware over-performing the current state-of-the-art one, but only to propose a method to improve readout in the present machines. Thus, interest was primarily focused on improving the performance of a given machine from the software point of view.

The classification obtained with PreTraNN, not only improves the classification accuracy but also better reproduces the actual distribution of data. In Fig. 6.7 the comparison of the GMM and PreTraNN labeling result on data with different read-out times is reported. The labeling for the FFNN is similar to the PreTraNN one, so it was omitted for clarity purposes. The first column shows data with the actual labels (represented by colors) as they were prepared in the quantum device. The second and third columns, instead, represent the same data but labeled according to GMM and PreTraNN, respectively. The same analysis is performed for short, medium and long times (rows of the figure). As anticipated, we conclude again that the GMM misses the classification for short times, dividing simply in half the overlapping distributions, while the PreTraNN provides a considerably more realistic and accurate classification. The two distributions of points overlapping

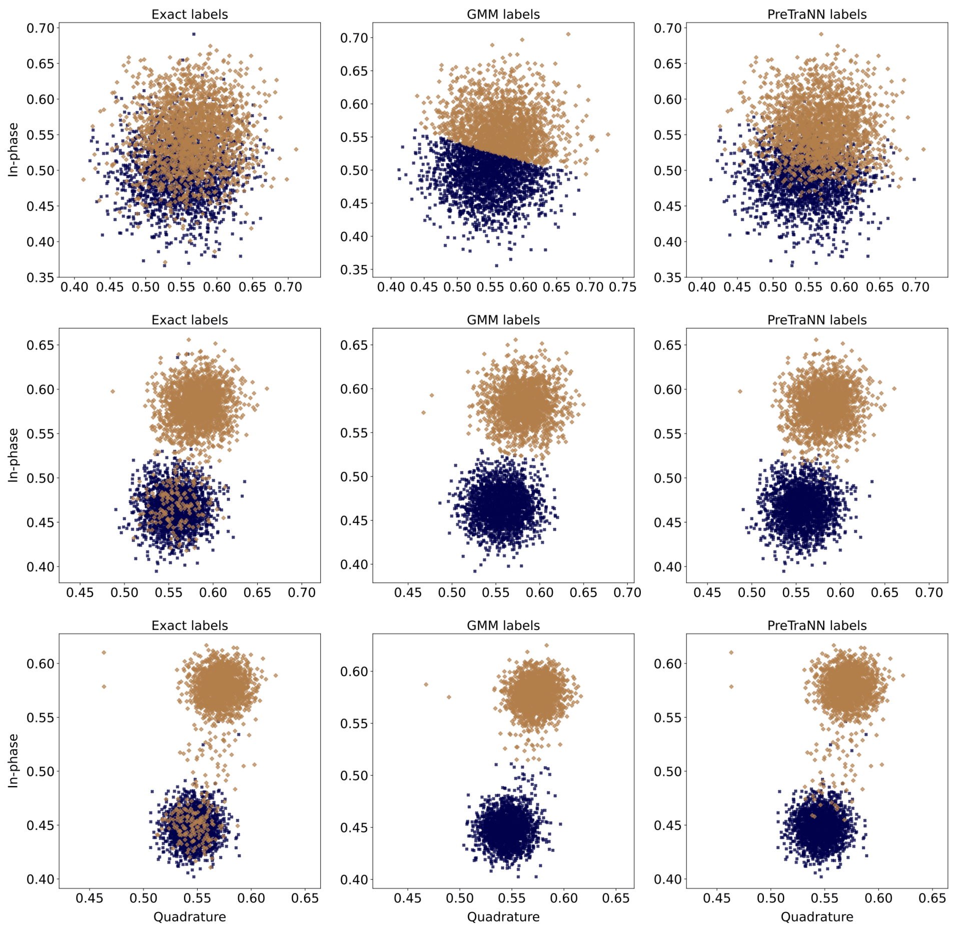


Figure 6.7: Pictorial representation of the dataset with exact, GMM's and PreTraNN's labeling. Each point is the time average of the I(t) and Q(t) signals. The actual label, i.e. the prepared state, is represented in the first column. The GMM and PreTraNN methods labels are represented in the second and third columns.

can now be spotted again.

The exact labels show the asymmetry in the data distribution due to the decay of the excited state: many $|1\rangle$-labeled points lay in $|0\rangle$ distribution. The comparison between the labels highlights that there are many points belonging to state $|1\rangle$ that even PreTraNN fails to recognize. Probably many of those points result from the imperfect calibration of the π-pulse used to prepare the state $|1\rangle$ on the machine.

Another important measure to take into account is the confusion matrix, which helps to visualize the classification performances of the three methods in comparison. In Fig. 6.8 are reported the confusion matrices for the three methods in three different measurement length setups. Each row reports the confusion matrices of the three models for a specific measurement length. Clearly, the best confusion matrices are those obtained for long times and with PreTraNN model.

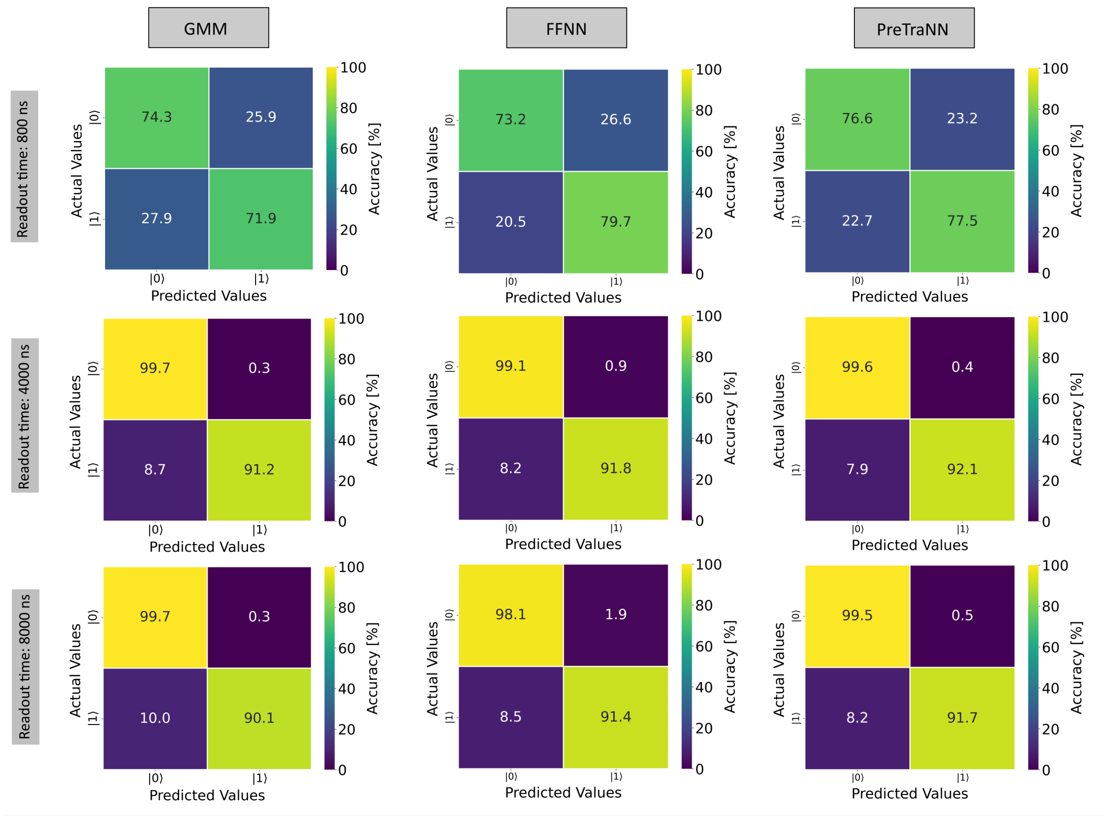


Figure 6.8: Confusion matrices for classification between states $|0\rangle$ and $|1\rangle$ for the three methods for short, medium and long readout times.

Computational cost and scaling

The higher structural complexity of the PreTraNN architecture means training and classification times longer than those required by GMM. In the following, we report the results together with some consideration on the scaling of the method.

The training for every neural network is performed with the "early stopping" approach to avoid over- or under-fitting. Instead of fixing the number of epochs, the training is stopped when the accuracy of the model does not increase for two epochs in a row. In Fig. 6.9 the results are reported. The upper table shows the training time of each model with respect to the readout length T_m for a 16000 elements dataset. The lower table, instead, shows the average time for a single input classification for each method. In both cases, the times are represented in

input batch size	1	100	10000
Classif. time PreTraNN [s]	0.04200	0.04300	0.22400
Classif. time GMM [s]	0.00012	0.00013	0.00043

Table 6.1: Classification times for PreTraNN and GMM as a function of inputs batch size. Every reported time is the result of an average of 100 experiments. The FFNN method is not reported because its behavior follows PreTraNN's.

logarithmic scale to better spot possible trends. Times are reported in seconds and refer to a mid-range laptop computer with 4 cores and 8 GB of RAM.

Considering the training time, it can be noted that the PreTraNN method takes a significantly longer time than the parameter estimation for GMM (from 2 to 3 orders of magnitude) but not much more than the FFNN, despite the two training stages of the PreTraNN. As one might expect, the training time of non-GMM methods increases as the inputs measurement time increases. In fact, long measurement times correspond to wider neural networks and, therefore, longer optimizations.

From the classification time point of view, we see that the times of the PreTraNN to label a single data (0.039 and 0.042 seconds, respectively) are almost equal and much longer than GMM's (0.00013 seconds). Moreover, for each method, the classification time does not depend on the measurement length.

It is important to specify that the classification time of an inputs batch of size S is not S times the classification time of a single input. We report the actual classification times as a function of batch size in Tab. 6.1.

Based on this data, some considerations can be made. First of all, we can assert that the training for PreTraNN and FFNN remains easily manageable by any computer, even for the longest measurement times. In fact, the training times, although much larger than the GMM, remain very small in absolute value. In general, the training process is not a problem since is done in advance.

Instead, more careful considerations are needed on the classification time side. If only an offline classification is needed, there are no stringent time constraints, and the model could be considered fast enough for some applications. If one instead needs a real-time or online readout on the machine, the classification times must be below the qubit lifetime. Since state-of-the-art superconducting transmon qubits have a lifetime of 200-500 microseconds [127, 128], in principle, we want a classification time that is well below these values, possibly on the order of tens or hundreds of nanoseconds. For this goal, neither the GMM nor PreTraNN have

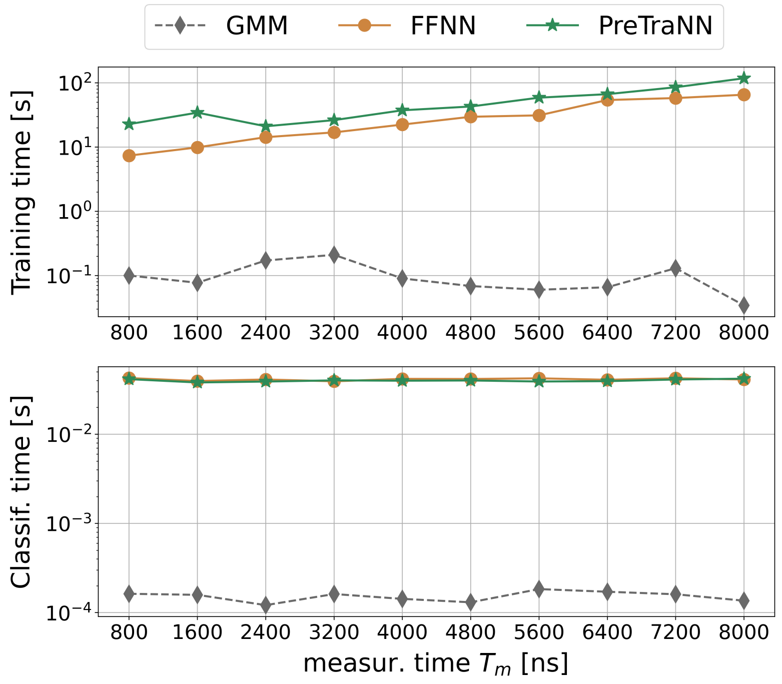


Figure 6.9: Training and classification times for GMM, FFNN and PreTraNN methods. The times are reported in seconds for a middle-range laptop computer. *Upper panel*: Training time in function of the measurement time (i.e. the length of the inputs). *Lower panel*: Classification time. The average time is 0.00013 seconds for GMM, 0.039 seconds for FNN and 0.037 seconds for PreTraNN.

the necessary characteristics, under the conditions used in this work. Of course, the use of more powerful computers, might be reduced the classification time by a few orders of magnitude. Moreover, an FPGA or an ASIC implementation could improve even more the efficiency of the classification step or also improve the training process by implementing it in an online way. See Ref. [129, 130, 131, 132].

In general, the ability to perform short-time measurement classification (with higher accuracy) is of great interest in quantum computing. The proposed approach allows for a good accuracy for short measurements compared to GMM. This can be exploited for real-time control systems, e.g., quantum orchestration platforms, leading to measurement speed-up or reducing computational time in error correction routines. Attention must be paid to the classification speed of the system. However, the longer time required to perform classification can be compensated, at least partially, by shorter measurements (as short as 1000 ns) than those of the GMM (4000 ns) while achieving the same classification accuracy. The PreTraNN performs well regardless of readout time, allowing one to potentially skip the readout time T_m trimming. Moreover, this method can be utilized for standard two-level qubits or, conversely, extended to arbitrary numbers of levels or qubits with slight modifications in its structure and by simply using different datasets. All this considered, the proposed method offers a promising approach to exploit short measurements that disturb the device as little as possible with less computational effort.

6.5.2 Three-state qutrit

In this case study, we exploit the possibility of accessing the higher quantum levels of superconducting qubits. We prepare and measure the qubit in $|0\rangle$,$|1\rangle$ and $|2\rangle$ state and store the obtained data. The whole dataset consists of 24000 elements (8000 for each state) divided into 75% train data and 25% test data. The architecture of the models is the same as in the previous case (and as defined in Sec. 6.3 and Sec. 6.4). The only difference between the two cases is the number of classes in the dataset. This allows to show the good scaling properties of the model.

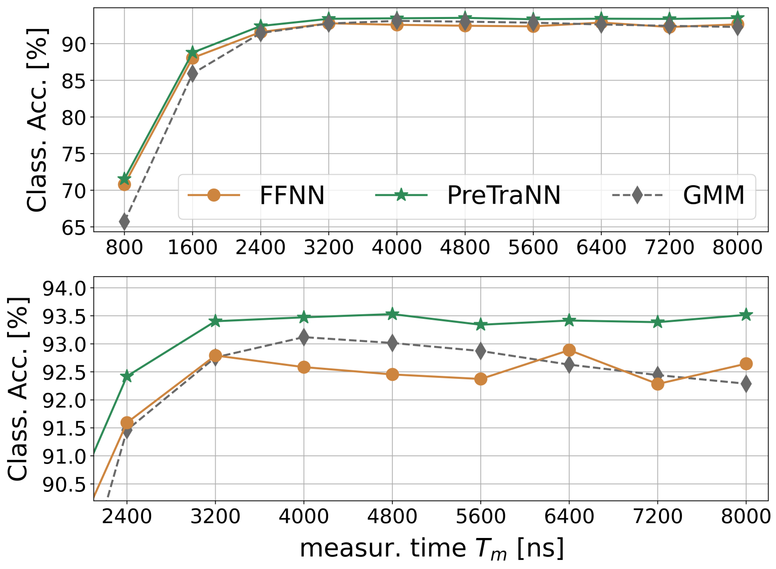


Figure 6.10: Global classification accuracy for $|0\rangle$,$|1\rangle$ and $|2\rangle$ states classification for a qutrit.

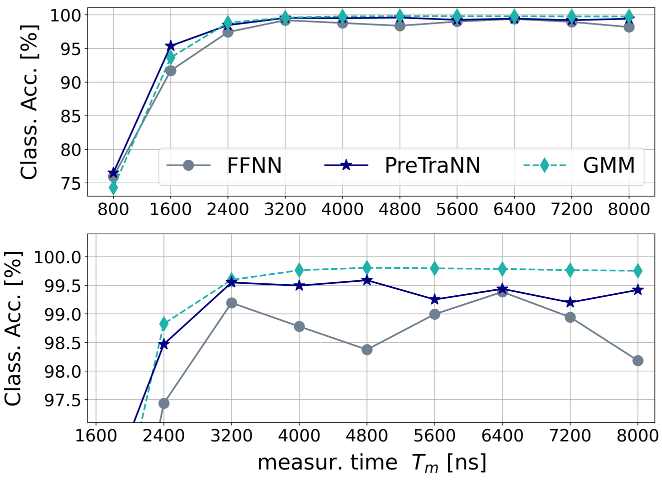
Classification accuracy

In Fig. 6.10 the results for the global accuracy are presented. The PreTraNN method achieves better classification performance for every measurement time.

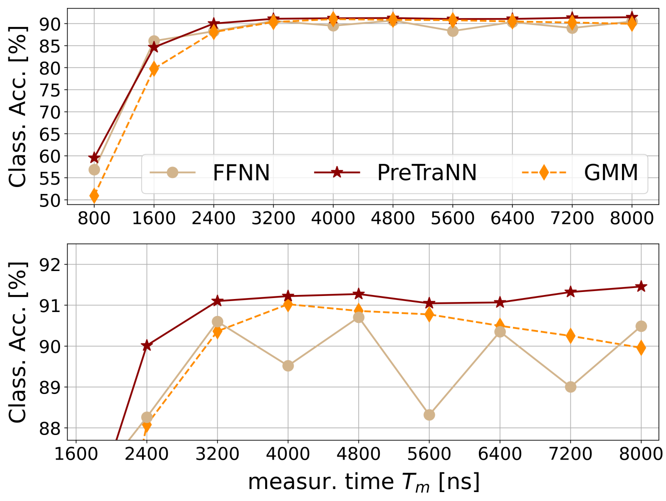
Again the GMM accuracy presents an increasing and decreasing trend with a maximum located at 4000 ns, while the FFNN, notwithstanding a reduction in the fluctuating trend, obtains a lower classification accuracy than the other two methods possibly due to training difficulties for high dimensional datasets. The PreTraNN instead presents a stable accuracy as a function of measurement time.

In Fig. 6.11a,6.11b,6.11c we show the classification accuracy for, respectively, state $|0\rangle$,$|1\rangle$ and $|2\rangle$. The lower panel of each figure is a zoom on the 2400-8000 ns part of the plot to better see the details. Even in this configuration, we can see the same trends as in the 2-level case. All methods show bad results for short times, especially GMM, and the FFNN still exhibits a seesaw pattern that makes it poorly suited to the task. Again, GMM performs better than PreTraNN in state $|0\rangle$ and worse in state $|1\rangle$ classification due to the data distribution asymmetry. For state $|2\rangle$ the difference between GMM and PrTranNN is even higher since the state $|2\rangle$ can decay not only on the state $|0\rangle$ but also on state $|1\rangle$.

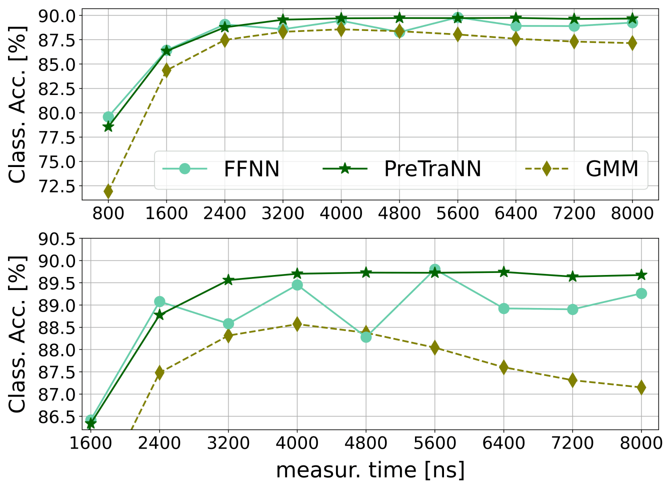
We can also study the performances of PreTraNN as a function of the number of qudit levels. This will show us of how the method scales with the number of points clouds. To achieve this, we compute the difference in percentage points (p.p.) between the global classification accuracy of the PreTraNN and that of the other methods. In Fig. 6.12 the difference in (p.p.) between the PreTraNN global accuracy and GMM's global accuracy for the two and three-level cases for every measurement time T_m is reported. The lower panel zooms on the middle and long times range. The average values for all T_m are highlighted on the right panels. An increasing value of this difference, as the system levels increase, suggests a possible increasing advantage in using the PreTraNN method for increasing system levels. In this case, we see that this trend can be clearly seen. Fig. 6.13 reports the same calculation referred to FFNN method. Here, the trend is also clear for both the whole set of measurement times and the medium-long range.



(a) *Upper panel*: State $|0\rangle$ classification accuracy for the three methods as a function of the measurement time in the case of a qutrit. *Lower panel*: zoom on the medium-long times.



(b) *Upper panel*: State $|1\rangle$ classification accuracy for the three methods as a function of the measurement time in the case of a qutrit. *Lower panel*: zoom on the medium-long times.



(c) *Upper panel*: State $|2\rangle$ classification accuracy for the three methods as a function of the measurement time. *Lower panel*: zoom on the medium-long times.

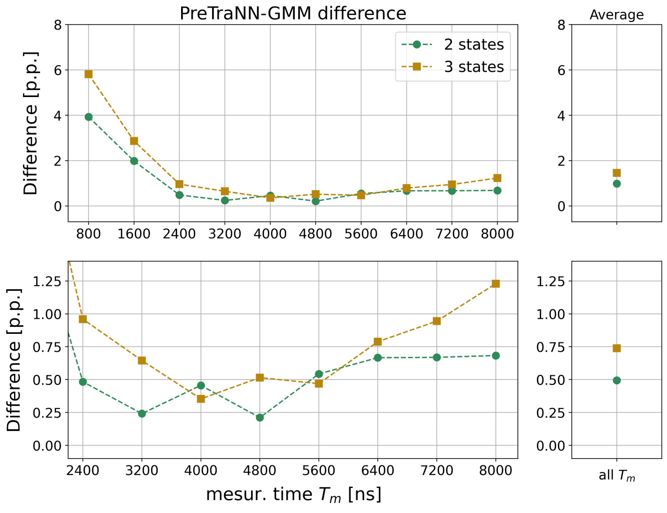


Figure 6.12: Difference in percentage points [p.p.] between the accuracy of PreTraNN and GMM for the qubit and qutrit cases for different measurement time T_m. The lower panel reports the analysis only for medium-long times. The small panels on the right show the average of all values of the respective plot on the left.

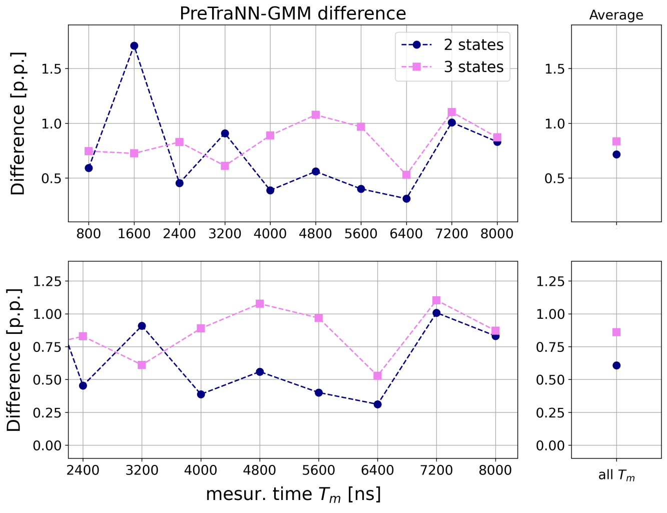


Figure 6.13: Difference in percentage points [p.p.] between the accuracy of PreTraNN and FFNN for the 2 or 3 qubit state case. The lower panel reports the analysis only for medium-long times. The small panels on the right show the average value of the respective plot on the left.

This analysis suggests the existence of a marginal increase in the effectiveness of PreTraNN compared to the other two methods as the classes of the dataset increase (i.e., as the dataset complexity rises). In other words, the difference in the global classification accuracy between PreTraNN and GMM or between PreTraNN and FFNN is bigger, on average, in the case of the three classes dataset, corresponding to qutrit readout data.

This analysis, although limited to 2 and 3 classes problem, suggests that the PreTraNN method should scale well as the qudit dimension increase. We can assume that it also scales well with the number of qubits since it also reduces to a multiclass dataset, but further analysis to better characterize the performance is needed.

Furthermore, PreTraNN requires only minimal structural modifications for different qudit dimensions. One only needs to adjust the number of output nodes in the last stage of the network and use an appropriate dataset with a different number of classes. While the training times rise due to the increased dataset size (training time grows linearly with the dataset dimension), the classification time remains the same as the previous 2-state case.

6.6 Conclusion

In this chapter, it has been shown that a feed-forward neural network with autoencoder pre-training allows a robust qubit readout classification scheme with high accuracy and low dependence on the experimental device feature values. It allows for a consistent classification performance even for short readout times, unlike the more traditional schemes affected by overlapping measurement results. It obtains good results also for longer measurement time where GMM method decrease its efficiency due to energy relaxation processes and a simple feed-forward neural network becomes difficult to train properly resulting in fluctuating results.

In addition, the proposed method allows for good classification on shorter measures, achieving a measurement speedup. More importantly, this measurement speedup is helpful for real-time control systems, e.g., quantum orchestration platforms or quantum error correction, where we need to disturb the system as little as possible.

In general, it was shown that the proposed method performs well for all measurement times, helping in increasing classification results from a software point of view. On the other side, the classification times for a single measure are higher than standard methods but can be improved with more optimized FPGA and ASIC implementations. Lastly, the proposed approach can be readily extended to an arbitrary number of states (or, possibly, a number of qubits) with minimal modification of the model structure and obtaining marginally increasing performances.

Conclusions

In this thesis, we presented strategies for machine-aware improvement of present NISQ quantum computers based on superconducting qubits. We focused on improvements related to two quantum computing requirements that we can identify with two of Di Vinceno's six quantum computing criteria.

First, we described quantum optimal control (QOC) algorithms that allow finding customized pulses that transform the state of qubits without resorting to deep circuits based on standard gates. This can be identified as the fourth criterion. In this context, we have also introduced two methods to speed up the generation of these pulses, the Control Pulse Reconstruction (CPR) method and the Lie Algebra-based Control Pulse Reconstruction (LA-CPR) method. The general idea of both is to compute in advance a finite set of controls corresponding to a discretization of the parameter values of the system, and then to interpolate between them to extract new controls for new parameter values. The CPR method is based on a multidimensional interpolation of the controls on the grid of parameters on which the unit work to be implemented in the quantum computer depends. The second method, on the other hand, is more general and is based on describing the propagators as elements of the Lie group $SU(n)$, and the interpolation is done by training a feed-forward neural network. These methods have been applied to cases of quantum simulations, in particular the spin dynamics of a two-neutron system, and it has been shown how the optimal control approaches, extended by pulse reconstruction methods, can be a powerful framework for quantum computations. In the second part of the paper, however, we address Di Vincenzo's fifth criterion, i.e., how to achieve high-fidelity readout of superconducting qubits. This was done not by hardware improvements, but by using machine learning methods to improve the classification of individual readout measurements. The most common classification methods based on Gaussian Mixture Model (GMM) and Feed Forward Neural Networks (FFNN) and a new method based on autoencoder-type pre-training of a neural network (PreTraNN) were presented. These methods were applied to the heterodyned readout signals of a three-level transmon qubit (i.e., a "qutrit"). In general, the PreTraNN method allows for a robust qubit readout classification scheme with high accuracy and low dependence on the experimental device feature values. This improvement can be attributed to the feature extraction capability of the autoencoder-type pre-training. The comparison was made with FFNN and GMM methods, which in all cases performed worse than PreTraNN. In general, the new method is presented as a promising approach to improve the readout of NISQ computers without hardware modification, but with a highly reconfigurable algorithm adaptable to a wide range of machines, provided they rely on the same readout method.

Appendix A

Derivation of transmon Hamiltonian

A.1 Transmon Hamiltonian Derivation

In Sec. 2.2 we introduced the Hamiltonian of a transmon device in Eq.(2.8) and we gave its representation in terms of the second quantization formalism. This result is not straightforward and it requires careful manipulation of the equation, taking into account the non-commutativity of the creation, $a^\dagger$, and annihilation, a, operators.

We start by defining the creation and annihilation operators of the transmon in terms of phase and charge zero-point fluctuations:

$$n = in_{zpf}(a^\dagger - a) \text{ with } n_{zpf} = \left(\frac{E_J}{32E_C}\right)^{\frac{1}{4}} \tag{A.1}$$

$$\phi = \phi_{zpf}(a^\dagger + a) \text{ with } \phi_{zpf} = \left(\frac{2E_C}{E_J}\right)^{\frac{1}{4}}. \tag{A.2}$$

Plugging the just introduced equation and the Taylor expansion of the potential Eq.(2.9) into Eq.(2.8), we obtain:

$$\begin{aligned} H &= -4E_C n_{zpf}^2(a^\dagger - a)^2 - E_J(1 - \frac{1}{2}\phi_{zpf}^2(a^\dagger + a)^2 && \text{(A.3)}\\ &\quad + \frac{1}{24}\phi_{zpf}^4(a^\dagger + a)^4 + ...). && \text{(A.4)}\end{aligned}$$

The previous expression can be recast as:

$$H \approx \sqrt{8E_C E_J}\left(a^\dagger a + \frac{1}{2}\right) - E_J - \frac{E_C}{12}(a^\dagger + a)^4. \tag{A.5}$$

Now, the quartic term on the right-hand side has to be carefully considered. Taking into account that we are in a non-commutative algebra, it becomes:

$$\begin{aligned}(a^\dagger + a)^4 &= (a^\dagger + a)^2(a^\dagger + a)^2 \\ &= (a^\dagger a^\dagger + a^\dagger a + aa^\dagger + aa)(a^\dagger a^\dagger + a^\dagger a + aa^\dagger + aa).\end{aligned} \tag{A.6}$$

This can be further expanded in :

$$\begin{aligned}= \quad & a^\dagger a^\dagger a^\dagger a^\dagger + \\ & + aa^\dagger a^\dagger a^\dagger + a^\dagger aa^\dagger a^\dagger + a^\dagger a^\dagger aa^\dagger + a^\dagger a^\dagger a^\dagger a + \\ & + a^\dagger aa^\dagger a + aa^\dagger aa^\dagger + a^\dagger a^\dagger aa + aaa^\dagger a^\dagger + a^\dagger aaaa^\dagger + aa^\dagger a^\dagger a + \\ & + a^\dagger aaa + aa^\dagger aa + aaa^\dagger a + aaaa^\dagger + \\ & + aaaa\end{aligned} \tag{A.7}$$

We use the rotating wave approximation (RWA) consisting in neglecting the rotating terms (i.e. the one with an unbalanced number of $a^\dagger$ and a). This can alternatively be seen as a first-order perturbation theory in which we neglect second-order terms. In the RWA, Eq. (A.7) reduces to the third row of Eq. (A.7):

$$\approx a^\dagger aa^\dagger a + aa^\dagger aa^\dagger + a^\dagger a^\dagger aa + aaa^\dagger a^\dagger + a^\dagger aaaa^\dagger + aa^\dagger a^\dagger a. \tag{A.8}$$

Now, recalling that the operators obey the relations:

$$[a^\dagger, a] = 1 \;\Rightarrow\; a^\dagger a = aa^\dagger + 1 \tag{A.9}$$

$$[a, a^\dagger] = -1 \;\Rightarrow\; aa^\dagger = a^\dagger a - 1, \tag{A.10}$$

we can rewrite each term of Eq.(A.8) as:

$$\begin{aligned}a^\dagger aa^\dagger a &= (a^\dagger a)^2 \\ aa^\dagger aa^\dagger &= (a^\dagger a)^2 + 2a^\dagger a + 1 \\ a^\dagger a^\dagger aa &= (a^\dagger a)^2 - a^\dagger a \\ aaa^\dagger a^\dagger &= (a^\dagger a)^2 + 3a^\dagger a + 2 \\ {}^\dagger aaaa^\dagger &= (a^\dagger a)^2 + a^\dagger a \\ aa^\dagger a^\dagger a &= (a^\dagger a)^2 + a^\dagger a\end{aligned}$$

Summing all the terms we finally obtain the desired approximation for the quartic term, namely:

$$(a^\dagger + a)^4 \approx 6(a^\dagger a)^2 + 6a^\dagger a + 3. \tag{A.11}$$

Plugging Eq.(A.11) into the transmon Hamiltonian Eq. (A.5), neglecting constant terms and fixing the anharmonicity $\alpha = -E_C$, we finally obtain the transmon Hamiltonian in:

$$H_{tr} = \omega_0 a^\dagger a + \frac{\alpha}{2} a^\dagger a(a^\dagger a + 1) \tag{A.12}$$

A.2 Transmon-Transmon Coupling Hamiltonian Term

In Sec. 2.2.3, Eq. (2.15), which considers a capacitive coupling between two transmons, is transformed into its second quantization representation. The first term corresponds to the case already discussed in this Appendix A. The second term, the coupling term, is transformed as follows. Considering again the phase and charge zero point fluctuation expression of eq. (A.1), and inserting it into the corresponding term of (2.15), we obtain:

$$\begin{aligned} 4e^2\frac{C_g}{C_1C_2}n_1n_2 &= 4e^2\frac{C_g}{C_1C_2}(in_{zpf})(in_{zpf})(a_1^\dagger - a_1)(a_2^\dagger - a_2) \\ &\quad -g(a_1^\dagger - a_1)(a_2^\dagger - a_2), \end{aligned} \tag{A.13}$$

where $g = 4e^2\frac{C_g}{C_1C_2}n_{zpf}^2$. Hence, we obtain the coupling term of Eq.(2.16).

Appendix B

Nuclear Theory

B.1 Spin Dependent Potential in neutron-neutron potential

We rely on Ref. [94] to obtain the explicit form of $A^{(1)}(\mathbf{r})$ and $A^{(2)}_{\alpha\beta}(\mathbf{r})$ of the SD neutron-neutron interaction at LO of chiral EFT in coordinate space given by Eq.(3.17). They are:

$$A^{(1)}(\mathbf{r}) = C_1\delta_{R_0}(\mathbf{r}) - Y_\pi(r)\left(1 - e^{-(r/R_0)^4}\right), \tag{B.1}$$

$$A^{(2)}_{\alpha\beta}(\mathbf{r}) = T_\pi(r)\left(3\frac{r_\alpha r_\beta}{r^2} - \delta_{\alpha\beta}\right)\left(1 - e^{-(r/R_0)^4}\right), \tag{B.2}$$

The SI part instead can be written as $V_{SI} = C_0\delta_{R_0}(\mathbf{r})$. In all these expressions, C_0 and C_1 are experimental constants fit to reproduce some quantity (e.g. the s-wave nucleon-nucleon phase shifts),

$$\delta_{R_0}(\mathbf{r}) = \frac{1}{\pi\Gamma(3/4)R_0^3}\exp\{(-r/R_0)\} \tag{B.3}$$

is the regulated Dirac function, $Y_\pi(r)$ is the Yukawa function, i.e.:

$$Y_\pi(r) = \frac{m_\pi^3}{12\pi}\left(\frac{g_a}{2f_\pi}\right)^2\frac{\exp(-m_\pi r)}{m_\pi r}, \tag{B.4}$$

and

$$T_\pi(r) = \left(1 + \frac{3}{m_\pi r} + \frac{3}{m_\pi^2 r^2}\right)Y_\pi(r) \tag{B.5}$$

where g_a, f_π and m_π are the axial-vector coupling constant, the pion exchange decay constant and the pion mass.

Appendix C

PreTraNN Scaling

C.1 Numerical Consideration on the Autoencoders

C.1.1 Autoencoder's latent space dimension

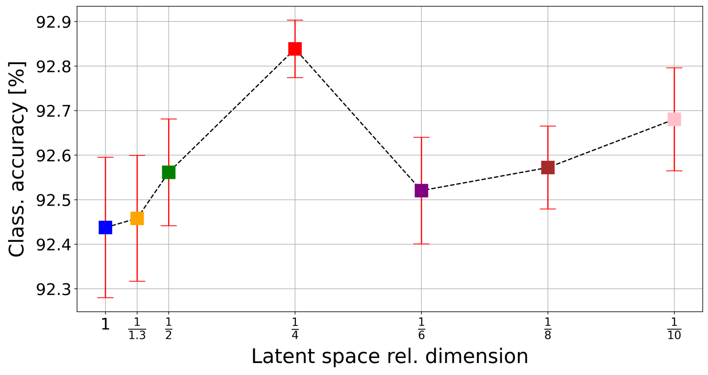


Figure C.1: PreTraNN global classification accuracy for the 3-state case with 2400 ns readout inputs as a function of the latent space dimension. The higher accuracy is reached at 1/4 the input dimension.

In the design of the architecture of a neural network, there is no solid theoretical guidance. However one might rely on a heuristic and ”trial and error” attitude based on experience. To make the procedure more quantitative, one can also vary the structure in an automated way and study how its metrics change accordingly. In this way, one can identify, within a certain degree of approximation, the architecture that works best for some specific problem.

In the case of the autoencoder, the main parameter is its latent space size. In principle, a latent space that is too small is not sufficient to perform expressive encoding, while too large of a latent space increases the computational cost without

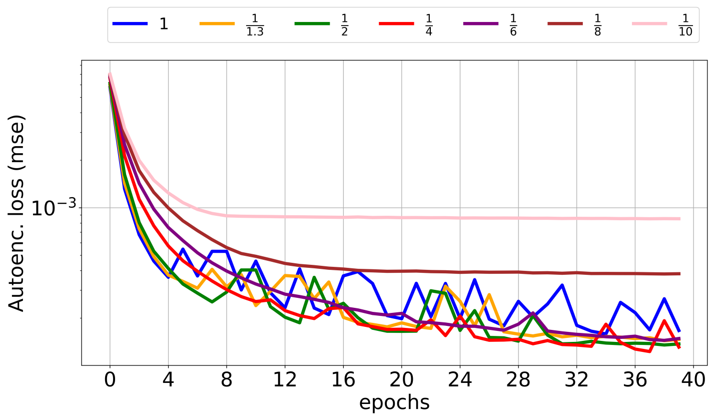


Figure C.2: Autoencoder training loss function as a function of training epochs for different latent space relative dimensions. Too large latent dimensions (1, 1/1.3 1/2 times the input size) present a fluctuating behavior and are useless for feature extraction, while too small latent dimensions do not allow an effective encoding and their loss function remains high (1/8 and 1/10 the input size).

extracting in a compact way information from the dataset. In the limiting case of a latent space equal to the input space, the neural network becomes equivalent to applying an identity to the inputs.

In this appendix, we describe the procedure used in our work to identify the best autoencoder structure. We took the PreTraNN with trajectories of 2400 ns (150 time-steps of 16 ns, i.e. inputs dimension of 300 values), and trained it for different values of latent space. We started from a latent dimension equal to the input dimension and gradually went down to one-tenth of it. The dimension of the other two inner layers was set linearly interpolating between the size of the input and latent space. The decoder had the same structure but reversed. Contextually, three properties of PreTraNN were studied as a function of latent dimension: the global classification accuracy, the autoencoder training loss and autoencoder training time. To obtain more consistent results, for each latent dimension the training was repeated 10 times with different samplings of the dataset and the properties values was averaged.

In Fig. C.1 the PreTraNN global classification accuracy for decreasing latent space dimension is reported. The abscissa shows the size of the latent space in terms of fractions of the input length (so that the information extracted from this case can be scaled directly to the other input lengths). The greatest accuracy, moreover with the smallest error bars, is achieved with a latent space whose size is one-fourth that of the input space. In absolute terms, the classification accuracy is quite stable for every latent space dimension but an increasing trend from 1 to 1/4 can be clearly spotted.

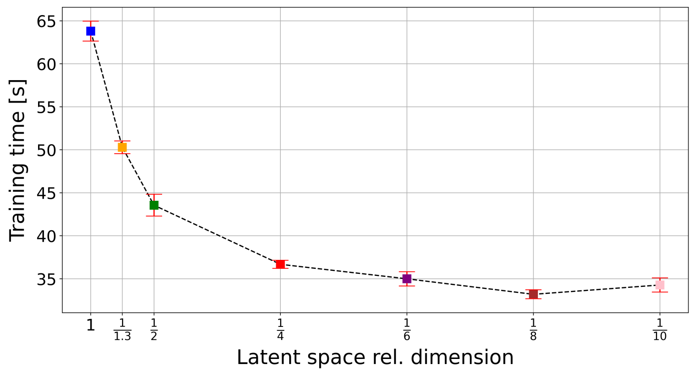


Figure C.3: Autoencoder training time in function of latent space relative dimension. Clearly larger latent spaces correspond to neural networks with more parameters and thus longer training times.

In Fig.C.2 the loss function values (mean squared error) during the training of the autoencoder for different latent space dimensions is represented. For large latent space sizes, the training converges faster for the first epochs but then assumes a fluctuating trend. For latent spaces that are small (e.g. 1/10, 1/8 the size of the input), on the other hand, convergence stalls at much higher values of the loss function. Thus the best values are 1/2, 1/4 and 1/6 of the input length.

In Fig. C.3 the training time in seconds is reported. Clearly, the training time decreases as the latent space decreases, since the number of network parameters decreases. A short training time is preferable.

Given this PreTraNN behavior, we can choose the latent space dimension making a trade-off between the reported metrics. The value which maximizes the classification accuracy having at the same time good loss function convergence and (relatively) short training time is a latent dimension of 1/4 the size of the inputs. This is the value chosen to carry out the analysis in this work. The dimension of the 2 internal layers is set linearly interpolating between the latent space and the input dimensions.

C.1.2 Dataset size and convergence

In order to obtain a good training convergence that maximizes classification accuracy an adequate dataset is needed. Small datasets are fast to train but usually produce inadequate classification accuracies, while large ones have the opposite behavior. At the same time, the growth of the classification accuracy capability is marginally decreasing with increasing dataset size. Here we report some analysis on the behavior of the PreTraNN as a function of the dataset dimension studying the same three properties introduced in the previous section i.e. loss function,

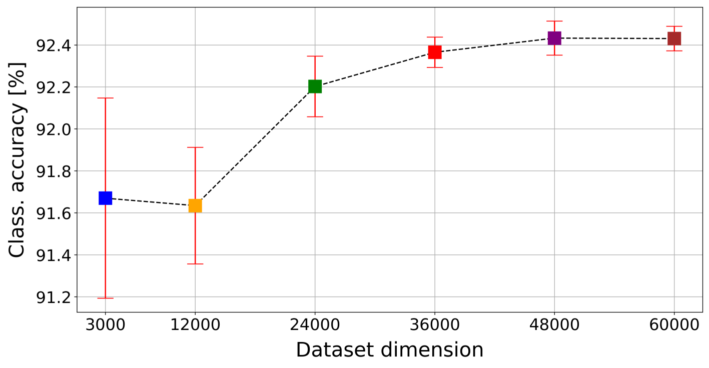


Figure C.4: Global classification accuracy of the PreTraNN as a function of the number of dataset elements.

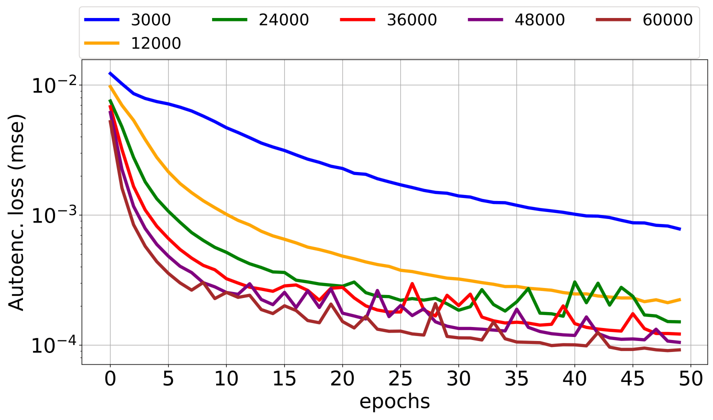


Figure C.5: Autoencoder loss (mean square error) as a function of the epochs for increasing dataset size.

classification accuracy and training time. Even in this case we took the PreTraNN with 2400 ns measurement signals (150 time-steps of 16 ns, i.e. inputs dimension of 300 values) with a latent space of 75 neurons, and trained it for different dataset dimensions. We started from a training dataset of 3000 elements (1000 elements for each class) and gradually increase its dimension to 60000 elements (with 75% of them dedicated to training). For each dataset dimension, the training was repeated 10 times with different sampling of the dataset and the properties values were averaged.

In Fig. C.4 the global classification accuracy as a function of the dataset size is reported. It can be seen that the accuracy increases as the dataset grows even if with decreasing speed.

Fig. C.5 represents the loss function values (mean squared error) during the training of the autoencoder for different configurations. The trend is quite neat.

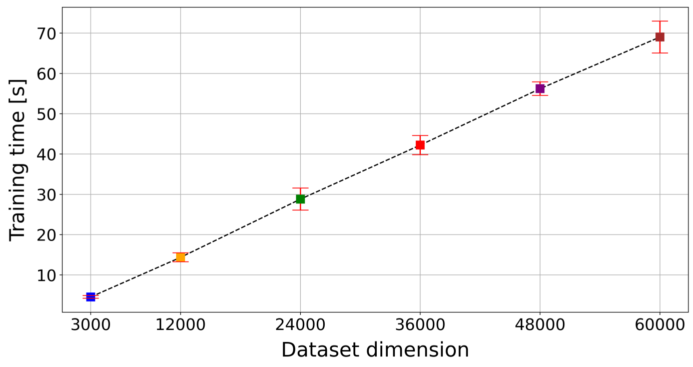


Figure C.6: Training time for increasing dataset dimension.

The larger the dataset the better the convergence, although for large data sets the convergence becomes more unstable.

In Fig. C.6 the training time in seconds is reported. As expected, the training time increase linearly with the dataset dimension. A short training time is preferable.

Given these results, the trade-off between accuracy, loss function, and training time, in order to maximize effectiveness and minimize cost, was identified in the 24000-item dataset for the three-state case and the 16000-item dataset for two-state case.

C.2 Models specifications

We report here the complete characterization of the autoencoder, the PreTraNN, the FFNN and the GMM models and their procedure of training.

In this work, the building and training of the neural network are performed via the python package *Keras* [133]. For the GMM instead the *sklearn* python package [134].

Autoencoder In every configuration employed in this work, the encoder is composed of an input layer, a first hidden layer and a second hidden layer connected to the latent layer. The decoder, on the other hand, has the same structure but is mirrored. So it has a first hidden layer connected to the latent layer, a second hidden layer and finally an output layer. We employ a full connectivity network implemented with the *Dense* layer specification in Keras. In Tab. C.1 all the information on the network is reported.

The training is performed using the *Adam* stochastic optimization algorithm [135] with the standard configuration implemented in Keras. The loss function

	Layer	Size	Activ. funct.	Keras type
Encoder	input	L	sigmoid	Dense
	1^{th} hidden	L3/4	tanh	Dense
	2^{nd} hidden	L2/4	tanh	Dense
	latent	L/4	tanh	Dense
Decoder	1^{th} hidden	L2/4	tanh	Dense
	2^{nd} hidden	L3/4	tanh	Dense
	output	L	sigmoid	Dense

Table C.1: Autoencoder's specifications. The "Size" column represents the number of neurons for each layer in a fraction of the input dimension L. The "Keras type" column reports the type of Keras layer employed.

is the *mean square error*. The training is performed with the *Early Stopping* procedure that stops the training if the loss does not decrease for two epochs in a row.

FFNN and PreTraNN's second stage The second stage of the PreTraNN is a simple feed-forward neural network. It is composed of an input layer (of the same dimension as the latent layer of the autoencoder), a first hidden layer and a second hidden layer connected to the output layer. The dimension C of the output layer depends on the number of classes we are doing the classification with. Hence, $C = 2$ for qubit classification of Sec. 6.5.1, while $C = 3$ for qutrit classification of Sec. 6.5.2. The connectivity between the neurons is full. The optimization algorithm is the *Adam*. The loss function is the *cross-entropy*, suitable for classification purposes. The training is performed with the *Early Stopping* procedure that stops the training if the loss does not decrease for two epochs in a row. Other information is summarized in Tab. C.2. The structure of FFNN model is the same but with a number of input neurons equal to the dataset dimension instead of the latent layer dimension.

Gaussian Mixture Model The GMM is implemented with *sklearn* package with the standard build-in parameters specifying only the number of classes of the input dataset.

Layer	Size	Activ. funct.	Keras type
Input	L/4 (L)	tanh	Dense
1^{th} hidden	L2/4 (2L)	tanh	Dense
2^{nd} hidden	L/4 (L)	tanh	Dense
Output	C	softmax	Dense

Table C.2: Structure and specifications of PreTraNN's second section (FFNN) network with Keras. L is the dataset inputs length, C is the dimension of the output layer which change based on the number of classes.

C.3 Autoencoder features

In this Appendix, we give examples of the two important autoencoder features: input regeneration and latent space values. Fig. C.7 shows an example of 3200 ns (i.e. 400 components) input reconstruction done by the autoencoder. The solid lines represent the original input (divided into the two quadratures), while the lines with markers represent the output of the autoencoder, i.e., the regeneration of the input from its synthetic representation in the latent space of the autoencoder. It can be seen that the reconstruction is quite faithful to the original.

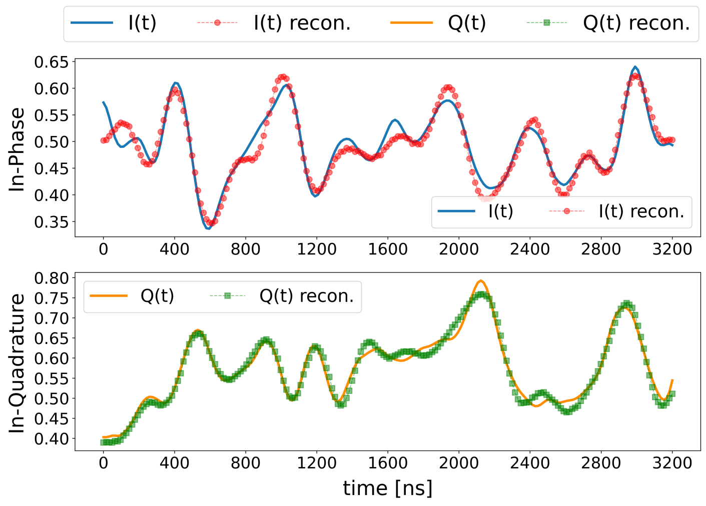


Figure C.7: An example of input regeneration made by the autoencoder. In both panels, the solid lines represent the measurement signal divided into its two quadratures, respectively In-phase (I) and In-Quadrature (Q). The lines with markers, instead, represent the input reconstruction made by the autoencoder.

The latent space representation is presented in Fig. C.8. The thin colored lines represent the latent space values of different inputs while the thick black line is the average of such lines. It can be seen that the latent space vectors for the two states are somewhat different on average. Both have 0 on average but those for $|0\rangle$ have larger fluctuations and a bit of structure. In particular, in both plots specific points where all the $\mathbf{h}^i$ vectors follow a definite trend (e.g., the points around 20 and 60 for state $|0\rangle$) can be spotted. These differences are the ones that allow the increase in classification performance shown in Sec 6.5.

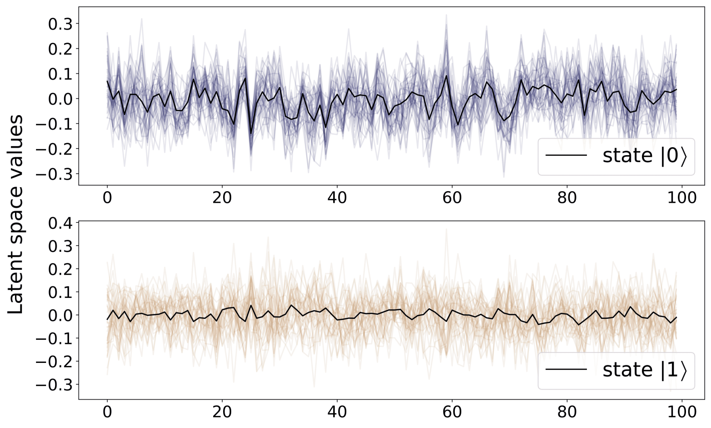


Figure C.8: Representation of latent space of the autoencoder for state $|0\rangle$ (upper) and $|1\rangle$ state (lower). In both panels, the colored lines are the latent space representation (i.e. $\mathbf{h}^i$ vector) of inputs for state $|0\rangle$ or $|1\rangle$. The solid black lines represent instead the average of these values.

One might wonder how inputs reconstruction varies as the latent representation varies. To answer this question we can proceed as follows. We use the encoder to obtain the latent representation of an input, we then vary slightly only one of its values, and finally, we plug the modified latent vector into the decoder to obtain its "reconstruction". We do this several times by varying slightly the input each time. Fig. C.9 depicts the result of this procedure. The thick lines represented the correct reconstruction of an input (divided into I and Q components) while the thin lines represent the reconstruction for increasing values of the 20th component of the latent representation. We can see that by slowly varying this value, we obtain a slowly varying family of reconstructions.

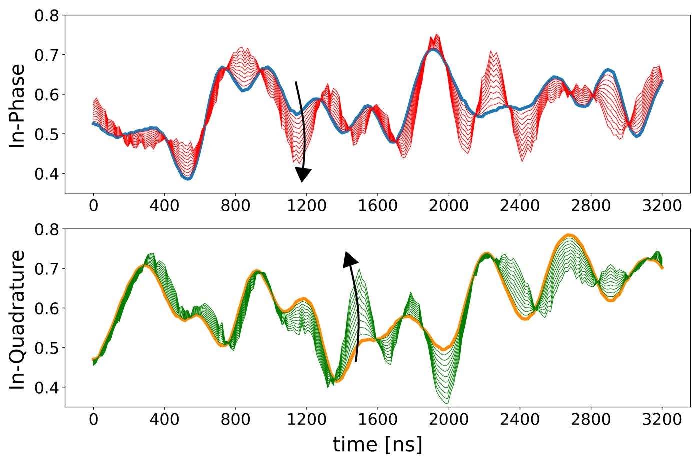


Figure C.9: The figure depicts an example of how the input reconstruction varies if a single value of the latent representation is varied slightly. The upper panel represents the in-phase component, lower panel the quadrature one. In both panels, the thick lines are the original "correct" input reconstruction, and the thin lines represent the reconstructions obtained by slowly varying a single value of the latent representation. In both panels, arrows are used to indicate the direction of changes induced by increasing the latent value.

Bibliography

[1] David Kaiser. *How the hippies saved physics: science, counterculture, and the quantum revival.* WW Norton & Company, 2011.

[2] David Deutsch. “Quantum theory, the Church–Turing principle and the universal quantum computer”. In: *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 400 (1985), pp. 117–97.

[3] Richard Phillips Feynman. “Simulating physics with computers”. In: *International Journal of Theoretical Physics* 21 (1999), pp. 467–488.

[4] Seth Lloyd. “Universal Quantum Simulators”. In: *Science (New York, N.Y.)* 273 (Sept. 1996), pp. 1073–8. DOI: `10.1126/science.273.5278.1073`.

[5] P.W. Shor. *Algorithms for quantum computation: discrete logarithms and factoring.* 1994. DOI: `10.1109/SFCS.1994.365700`.

[6] David DiVincenzo and IBM. “The Physical Implementation of Quantum Computation”. In: *Fortschritte der Physik* 48 (Mar. 2000). DOI: `10.1002/1521-3978(200009)48:9/113.0.CO;2-E`.

[7] John Preskill. “Quantum Computing in the NISQ era and beyond”. In: *Quantum* 2 (Jan. 2018). DOI: `10.22331/q-2018-08-06-79`.

[8] Matthias Troyer and Uwe-Jens Wiese. “Computational Complexity and Fundamental Limitations to Fermionic Quantum Monte Carlo Simulations”. In: *Phys. Rev. Lett.* 94 (17 May 2005), p. 170201. DOI: `10.1103/PhysRevLett.94.170201`. URL: `https://link.aps.org/doi/10.1103/PhysRevLett.94.170201`.

[9] David J Thouless. *The quantum mechanics of many-body systems.* Courier Corporation, 2014.

[10] Andrew Daley et al. “Practical quantum advantage in quantum simulation”. In: *Nature* 607 (July 2022), pp. 667–676. DOI: `10.1038/s41586-022-04940-6`.

[11] Jay Gambetta et al. "Quantum trajectory approach to circuit QED: Quantum jumps and the Zeno effect". In: *Phys. Rev. A* 77.1 (2008), p. 012112. DOI: 10.1103/PhysRevA.77.012112.

[12] R Bianchetti et al. "Dynamics of dispersive single-qubit readout in circuit quantum electrodynamics". In: *Phys. Rev. A* 80.4 (2009), p. 043840. DOI: 10.1103/PhysRevA.80.043840.

[13] Brian Everitt. *Finite mixture distributions*. Springer Science & Business Media, 2013. DOI: 10.1002/9781118445112.stat06216.

[14] Luis A Martinez, Yaniv J Rosen, and Jonathan L DuBois. "Improving qubit readout with hidden Markov models". In: *Phys.Rev. A* 102.6 (2020), p. 062426. DOI: 10.1103/PhysRevA.102.062426.

[15] Christopher M. Dawson and Micheal A. Nielsen. "The Solovay-Kitaev algorithm". In: *Quant. Info. Compt.* 6 (1 Jan. 2006), pp. 81–95. URL: https://dl.acm.org/doi/10.5555/2011679.2011685#sec-ref.

[16] Tien Trung Pham, Rodney Van Meter, and Clare Horsman. "Optimization of the Solovay-Kitaev algorithm". In: *Phys. Rev. A* 87 (5 May 2013), p. 052332. DOI: 10.1103/PhysRevA.87.052332. URL: https://link.aps.org/doi/10.1103/PhysRevA.87.052332.

[17] David J Griffiths and Darrell F Schroeter. *Introduction to quantum mechanics*. Cambridge university press, 2018.

[18] Howard M Wiseman and Gerard J Milburn. *Quantum measurement and control*. Cambridge university press, 2009.

[19] J. I. Cirac and P. Zoller. "Quantum Computations with Cold Trapped Ions". In: *Phys. Rev. Lett.* 74 (20 May 1995), pp. 4091–4094. DOI: 10.1103/PhysRevLett.74.4091. URL: https://link.aps.org/doi/10.1103/PhysRevLett.74.4091.

[20] D. Leibfried et al. "Quantum dynamics of single trapped ions". In: *Rev. Mod. Phys.* 75 (1 Mar. 2003), pp. 281–324. DOI: 10.1103/RevModPhys.75.281. URL: https://link.aps.org/doi/10.1103/RevModPhys.75.281.

[21] R. Blatt and Christian Roos. "Quantum Simulations with Trapped Ions". In: *Nature Physics* 8 (Apr. 2012), pp. 277–284. DOI: 10.1038/nphys2252.

[22] R. Hanson, O. Gywat, and D. D. Awschalom. "Room-temperature manipulation and decoherence of a single spin in diamond". In: *Phys. Rev. B* 74 (16 Oct. 2006), p. 161203. DOI: 10.1103/PhysRevB.74.161203. URL: https://link.aps.org/doi/10.1103/PhysRevB.74.161203.

[23] Jingfu Zhang, Swathi S. Hegde, and Dieter Suter. “Efficient Implementation of a Quantum Algorithm in a Single Nitrogen-Vacancy Center of Diamond”. In: *Phys. Rev. Lett.* 125 (3 July 2020), p. 030501. DOI: 10.1103/PhysRevLett.125.030501. URL: https://link.aps.org/doi/10.1103/PhysRevLett.125.030501.

[24] A. Imamog¯lu et al. “Quantum Information Processing Using Quantum Dot Spins and Cavity QED”. In: *Phys. Rev. Lett.* 83 (20 Sept. 1999), pp. 4204–4207. DOI: 10.1103/PhysRevLett.83.4204. URL: https://link.aps.org/doi/10.1103/PhysRevLett.83.4204.

[25] Jason R. Petta et al. “Coherent Manipulation of Coupled Electron Spins in Semiconductor Quantum Dots”. In: *Science* 309 (2005), pp. 2180–2184. DOI: 10.1126/science.1116955.

[26] Bruce E. Kane. “A silicon-based nuclear spin quantum computer”. In: *Nature* 393 (1998), pp. 133–137. DOI: 10.1038/30156.

[27] Rutger Vrijen et al. “Electron-spin-resonance transistors for quantum computing in silicon-germanium heterostructures”. In: *Phys. Rev. A* 62 (1 June 2000), p. 012306. DOI: 10.1103/PhysRevA.62.012306. URL: https://link.aps.org/doi/10.1103/PhysRevA.62.012306.

[28] Immanuel Bloch, Jean Dalibard, and Wilhelm Zwerger. “Many-body physics with ultracold gases”. In: *Rev. Mod. Phys.* 80 (3 July 2008), pp. 885–964. DOI: 10.1103/RevModPhys.80.885. URL: https://link.aps.org/doi/10.1103/RevModPhys.80.885.

[29] Christian Gross and Immanuel Bloch. “Quantum simulations with ultracold atoms in optical lattices”. In: *Science* 357.6355 (2017), pp. 995–1001. DOI: 10.1126/science.aal3837.

[30] Emanuel Knill, Raymond Laflamme, and Gerard J. Milburn. “A scheme for efficient quantum computation with linear optics”. In: *Nature* 409 (2001), pp. 46–52. DOI: 10.1038/35051009.

[31] Stefanie Barz. In: 48.8 (Mar. 2015), p. 083001. DOI: 10.1088/0953-4075/48/8/083001. URL: https://dx.doi.org/10.1088/0953-4075/48/8/083001.

[32] Philip Krantz et al. “A quantum engineer’s guide to superconducting qubits”. In: *Applied Physics Reviews* 6 (June 2019), p. 021318. DOI: 10.1063/1.5089550.

[33] Michel H Devoret and Robert J Schoelkopf. “Superconducting circuits for quantum information: An outlook”. In: *Science* 339.6124 (2013), pp. 1169–1174.

[34] LC Gupta and Manu S Multani. *Selected topics in superconductivity.* Vol. 1. World Scientific, 1993.

[35] SE Rasmussen et al. “Superconducting circuit companion—an introduction with worked examples”. In: *PRX Quantum* 2.4 (2021), p. 040204.

[36] John M. Martinis and M. Nahum. “Effect of environmental noise on the accuracy of Coulomb-blockade devices”. In: *Phys. Rev. B* 48 (24 Dec. 1993), pp. 18316–18319. DOI: `10.1103/PhysRevB.48.18316`. URL: `https://link.aps.org/doi/10.1103/PhysRevB.48.18316`.

[37] A. O. Caldeira and A. J. Leggett. “Influence of Dissipation on Quantum Tunneling in Macroscopic Systems”. In: *Phys. Rev. Lett.* 46 (4 Jan. 1981), pp. 211–214. DOI: `10.1103/PhysRevLett.46.211`. URL: `https://link.aps.org/doi/10.1103/PhysRevLett.46.211`.

[38] Anthony J Leggett. “Testing the limits of quantum mechanics: motivation, state of play, prospects”. In: *J. Phys: Condens. Matter* 14.15 (2002), R415. DOI: `10.1088/0953-8984/14/15/201`.

[39] T. Duty et al. “Coherent dynamics of a Josephson charge qubit”. In: *Phys. Rev. B* 69 (14 Apr. 2004), p. 140503. DOI: `10.1103/PhysRevB.69.140503`. URL: `https://link.aps.org/doi/10.1103/PhysRevB.69.140503`.

[40] Denis Vion et al. “Manipulating the quantum state of an electrical circuit”. In: *Science* 296.5569 (2002), pp. 886–889. DOI: `10.1126/science.1069372`.

[41] J. Q. You and Franco Nori. “Quantum information processing with superconducting qubits in a microwave field”. In: *Phys. Rev. B* 68 (6 Aug. 2003), p. 064509. DOI: `10.1103/PhysRevB.68.064509`. URL: `https://link.aps.org/doi/10.1103/PhysRevB.68.064509`.

[42] Yasunobu Nakamura, Yu A Pashkin, and JS Tsai. “Coherent control of macroscopic quantum states in a single-Cooper-pair box”. In: *nature* 398.6730 (1999), pp. 786–788. DOI: `10.1038/19718`.

[43] Michael J. Peterer et al. “Coherence and Decay of Higher Energy Levels of a Superconducting Transmon Qubit”. In: *Phys. Rev. Lett.* 114 (1 Jan. 2015), p. 010501. DOI: `10.1103/PhysRevLett.114.010501`. URL: `https://link.aps.org/doi/10.1103/PhysRevLett.114.010501`.

[44] Xian Wu et al. “High-Fidelity Software-Defined Quantum Logic on a Superconducting Qudit”. In: *Phys. Rev. Lett.* 125 (17 Oct. 2020), p. 170502. DOI: `10.1103/PhysRevLett.125.170502`. URL: `https://link.aps.org/doi/10.1103/PhysRevLett.125.170502`.

[45] L Dicarlo et al. “Preparation and Measurement of Three-Qubit Entanglement in a Superconducting Circuit”. In: *Nature* 467 (Sept. 2010), pp. 574–8. DOI: 10.1038/nature09416.

[46] R. Barends et al. “Coherent Josephson Qubit Suitable for Scalable Quantum Integrated Circuits”. In: *Phys. Rev. Lett.* 111 (8 Aug. 2013), p. 080502. DOI: 10.1103/PhysRevLett.111.080502. URL: https://link.aps.org/doi/10.1103/PhysRevLett.111.080502.

[47] Martin Sandberg et al. “Tuning the field in a microwave resonator faster than the photon lifetime”. In: *Applied Physics Letters* 92 (May 2008), p. 203501. DOI: 10.1063/1.2929367.

[48] Mathieu Pierre et al. “Storage and on-demand release of microwaves using superconducting resonators with tunable coupling”. In: *Applied Physics Letters* 104 (June 2014). DOI: 10.1063/1.4882646.

[49] Yi Yin et al. “Catch and Release of Microwave Photon States”. In: *Physical review letters* 110 (Mar. 2013), p. 107001. DOI: 10.1103/PhysRevLett.110.107001.

[50] Nissim Ofek et al. “Extending the lifetime of a quantum bit with error correction in superconducting circuits”. In: *Nature* 536 (July 2016). DOI: 10.1038/nature18949.

[51] Alexandre Blais et al. “Circuit quantum electrodynamics”. In: *Reviews of Modern Physics* 93 (May 2021). DOI: 10.1103/RevModPhys.93.025005.

[52] Alexandre Blais et al. “Cavity quantum electrodynamics for superconducting electrical circuits: An architecture for quantum computation”. In: *Physical Review A* 69 (Feb. 2004). DOI: 10.1103/PhysRevA.69.062320.

[53] Christopher Axline et al. “An architecture for integrating planar and 3D cQED devices”. In: *Applied Physics Letters* 109 (July 2016), p. 042601. DOI: 10.1063/1.4959241.

[54] Jonas Bylander et al. “Noise spectroscopy through dynamical decoupling with a superconducting flux qubit”. In: *Nature Physics* 7 (May 2011), pp. 565–570. DOI: 10.1038/nphys1994.

[55] F. Bloch. “Generalized Theory of Relaxation”. In: *Physical Review - PHYS REV X* 105 (Feb. 1957), pp. 1206–1222. DOI: 10.1103/PhysRev.105.1206.

[56] A. G. Redfield. “The Theory of Relaxation Processes”. In: *IBM Journal of Research and Development* 1 (Feb. 1957), pp. 19–31. DOI: 10.1147/rd.11.0019.

[57] Domenico d’Alessandro. *Introduction to quantum control and dynamics.* Chapman and hall/CRC, 2021.

[58] David Goodwin and Ilya Kuprov. “Modified Newton-Raphson GRAPE methods for optimal control of spin systems”. In: *The Journal of Chemical Physics* 144 (May 2016), p. 204107. DOI: 10.1063/1.4949534.

[59] Katharine W. Moore Tibbetts et al. “Exploring the tradeoff between fidelity and time optimal control of quantum unitary transformations”. In: *Phys. Rev. A* 86 (6 Dec. 2012), p. 062309. DOI: 10.1103/PhysRevA.86.062309. URL: https://link.aps.org/doi/10.1103/PhysRevA.86.062309.

[60] Benjamin Rowland and Jonathan Jones. “Implementing quantum logic gates with gradient ascent pulse engineering: Principles and practicalities”. In: *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences* 370 (Oct. 2012), pp. 4636–50. DOI: 10.1098/rsta.2011.0361.

[61] Sophie Shermer and Pierre Fouquieres. “Efficient Algorithms for Optimal Control of Quantum Dynamics: The”Krotov” Method unencumbered”. In: *New Journal of Physics* 13 (July 2011), p. 073029. DOI: 10.1088/1367-2630/13/7/073029.

[62] Jason Dominy and Herschel Rabitz. “Exploring families of quantum controls for generating unitary transformations”. In: *Journal of Physics A: Mathematical and Theoretical* 41 (May 2008), p. 205305. DOI: 10.1088/1751-8113/41/20/205305.

[63] Tommaso Caneva, Tommaso Calarco, and Simone Montangero. “Chopped random-basis quantum optimization”. In: *Phys. Rev. A* 84 (2 Aug. 2011), p. 022326. DOI: 10.1103/PhysRevA.84.022326. URL: https://link.aps.org/doi/10.1103/PhysRevA.84.022326.

[64] Jens Koch et al. “Charge-insensitive qubit design derived from the Cooper pair box”. In: *Phys. Rev. A* 76 (4 Oct. 2007), p. 042319. DOI: 10.1103/PhysRevA.76.042319. URL: https://link.aps.org/doi/10.1103/PhysRevA.76.042319.

[65] Willis E. Lamb and Robert C. Retherford. “Fine Structure of the Hydrogen Atom by a Microwave Method”. In: *Phys. Rev.* 72 (3 Aug. 1947), pp. 241–243. DOI: 10.1103/PhysRev.72.241. URL: https://link.aps.org/doi/10.1103/PhysRev.72.241.

[66] A Fragner et al. “Resolving Vacuum Fluctuations in an Electrical Circuit by Measuring the Lamb Shift”. In: *Science (New York, N.Y.)* 322 (Dec. 2008), pp. 1357–60. DOI: 10.1126/science.1164482.

[67] H. A. Bethe. “The Electromagnetic Shift of Energy Levels”. In: *Phys. Rev.* 72 (4 Aug. 1947), pp. 339–341. DOI: 10.1103/PhysRev.72.339. URL: https://link.aps.org/doi/10.1103/PhysRev.72.339.

[68] James Gubernatis, Naoki Kawashima, and Philipp Werner. *Quantum Monte Carlo Methods*. Cambridge University Press, 2016.

[69] J. Carlson et al. "Quantum Monte Carlo methods for nuclear physics". In: *Rev. Mod. Phys.* 87 (3 Sept. 2015), pp. 1067–1118. DOI: 10.1103/RevModPhys.87.1067. URL: https://link.aps.org/doi/10.1103/RevModPhys.87.1067.

[70] James M Haile et al. "Molecular dynamics simulation: elementary methods". In: *Computers in Physics* 7.6 (1993), pp. 625–625.

[71] Simone Montangero, Evenson Montangero, and Evenson. *Introduction to tensor network methods*. Springer, 2018.

[72] Liang-Hui Du, J. Q. You, and Lin Tian. "Superconducting circuit probe for analog quantum simulators". In: *Phys. Rev. A* 92 (1 July 2015), p. 012330. DOI: 10.1103/PhysRevA.92.012330. URL: https://link.aps.org/doi/10.1103/PhysRevA.92.012330.

[73] 1-Michael Reiner et al. "Emulating the one-dimensional Fermi-Hubbard model by a double chain of qubits". In: *Phys. Rev. A* 94 (3 Sept. 2016), p. 032338. DOI: 10.1103/PhysRevA.94.032338. URL: https://link.aps.org/doi/10.1103/PhysRevA.94.032338.

[74] Zohreh Davoudi et al. "Towards analog quantum simulations of lattice gauge theories with trapped ions". In: *Phys. Rev. Res.* 2 (2 Apr. 2020), p. 023015. DOI: 10.1103/PhysRevResearch.2.023015. URL: https://link.aps.org/doi/10.1103/PhysRevResearch.2.023015.

[75] G. Ortiz et al. "Quantum algorithms for fermionic simulations". In: *Phys. Rev. A* 64 (2 July 2001), p. 022319. DOI: 10.1103/PhysRevA.64.022319. URL: https://link.aps.org/doi/10.1103/PhysRevA.64.022319.

[76] J. Casanova et al. "Quantum Simulation of Interacting Fermion Lattice Models in Trapped Ions". In: *Phys. Rev. Lett.* 108 (19 May 2012), p. 190502. DOI: 10.1103/PhysRevLett.108.190502. URL: https://link.aps.org/doi/10.1103/PhysRevLett.108.190502.

[77] Valentina Amitrano et al. "Trapped-ion quantum simulation of collective neutrino oscillations". In: *Phys. Rev. D* 107 (2 Jan. 2023), p. 023007. DOI: 10.1103/PhysRevD.107.023007. URL: https://link.aps.org/doi/10.1103/PhysRevD.107.023007.

[78] Naomichi Hatano and Masuo Suzuki. "Finding exponential product formulas of higher orders". In: *Quantum annealing and other optimization methods*. Springer, 2005, pp. 37–68.

[79] Benjamin Jones et al. "Optimising trotter-suzuki decompositions for quantum simulation using evolutionary strategies". In: (July 2019), pp. 1223–1231. DOI: 10.1145/3321707.3321835.

[80] Francesco Tacchino et al. "Quantum computers as universal quantum Simulators: state-of-the-art and perspectives". In: *Advanced Quantum Technologies* 3.3 (2020), p. 1900052.

[81] Kishor Bharti et al. "Noisy intermediate-scale quantum algorithms". In: *Rev. Mod. Phys.* 94 (1 Feb. 2022), p. 015004. DOI: 10.1103/RevModPhys.94.015004. URL: https://link.aps.org/doi/10.1103/RevModPhys.94.015004.

[82] Michael A Nielsen et al. "The Fermionic canonical commutation relations and the Jordan-Wigner transform". In: *School of Physical Sciences The University of Queensland* 59 (2005).

[83] Sergey B. Bravyi and Alexei Yu. Kitaev. "Fermionic Quantum Computation". In: *Annals of Physics* 298.1 (2002), pp. 210–226. ISSN: 0003-4916. DOI: https://doi.org/10.1006/aphy.2002.6254. URL: https://www.sciencedirect.com/science/article/pii/S0003491602962548.

[84] F Verstraete and J I Cirac. "Mapping local Hamiltonians of fermions to local Hamiltonians of spins". In: *Journal of Statistical Mechanics: Theory and Experiment* 2005.09 (Sept. 2005), P09012. DOI: 10.1088/1742-5468/2005/09/P09012. URL: https://dx.doi.org/10.1088/1742-5468/2005/09/P09012.

[85] C. D. Batista and G. Ortiz. "Algebraic approach to interacting quantum systems". In: *Advances in Physics* 53.1 (2004), pp. 1–82. DOI: 10.1080/00018730310001642086.

[86] Rolando Somma et al. "Quantum Simulations of Physics Problems". In: *International Journal of Quantum Information* 01.02 (2003), pp. 189–206. DOI: 10.1142/S0219749903000140.

[87] Nicolas Sawaya et al. "Resource-efficient digital quantum simulation of d-level systems for photonic, vibrational, and spin-s Hamiltonians". In: *npj Quantum Information* 6 (Dec. 2020). DOI: 10.1038/s41534-020-0278-0.

[88] Farrokh Vatan and Colin Williams. "Optimal quantum circuits for general two-qubit gates". In: *Phys. Rev. A* 69 (3 Mar. 2004), p. 032315. DOI: 10.1103/PhysRevA.69.032315. URL: https://link.aps.org/doi/10.1103/PhysRevA.69.032315.

[89] Eric T Holland et al. "Optimal control for the quantum simulation of nuclear dynamics". In: *Phys. Rev. A* 101.6 (2020), p. 062307. DOI: 10.1103/PhysRevA.101.062307.

[90] R. Machleidt and D. Entem. “Chiral effective field theory and nuclear forces”. In: *Phys. Rep-rev. Sect. of Phys. Lett.* 503 (May 2011). DOI: 10.1016/j.physrep.2011.02.001.

[91] C. D. Goodman et al. “Gamow-Teller Matrix Elements from 0°(p, n) Cross Sections”. In: *Phys. Rev. Lett.* 44 (26 June 1980), pp. 1755–1759. DOI: 10.1103/PhysRevLett.44.1755. URL: https://link.aps.org/doi/10.1103/PhysRevLett.44.1755.

[92] W. G. Love and M. A. Franey. “Effective nucleon-nucleon interaction for scattering at intermediate energies”. In: *Phys. Rev. C* 24 (3 Sept. 1981), pp. 1073–1094. DOI: 10.1103/PhysRevC.24.1073. URL: https://link.aps.org/doi/10.1103/PhysRevC.24.1073.

[93] A. Gezerlis et al. “Local chiral effective field theory interactions and quantum Monte Carlo applications”. In: *Phys. Rev. C* 90 (5 Sept. 2014), p. 054323. DOI: 10.1103/PhysRevC.90.054323. URL: https://link.aps.org/doi/10.1103/PhysRevC.90.054323.

[94] I. Tews et al. “Quantum Monte Carlo calculations of neutron matter with chiral three-body forces”. In: *Phys. Rev. C* 93 (2 Feb. 2016), p. 024305. DOI: 10.1103/PhysRevC.93.024305. URL: https://link.aps.org/doi/10.1103/PhysRevC.93.024305.

[95] J.R. Johansson, P.D. Nation, and Franco Nori. *QuTiP: An open-source Python framework for the dynamics of open quantum systems.* 2012. DOI: https://doi.org/10.1016/j.cpc.2012.02.021. URL: https://www.sciencedirect.com/science/article/pii/S0010465512000835.

[96] J.R. Johansson, P.D. Nation, and Franco Nori. “QuTiP 2: A Python framework for the dynamics of open quantum systems”. In: *Computer Physics Communications* 184.4 (2013), pp. 1234–1240. ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2012.11.019. URL: https://www.sciencedirect.com/science/article/pii/S0010465512003955.

[97] V Smirnov. “On the estimation of a path integral by means of the saddle point method”. In: *Journal of Physics A: Mathematical and Theoretical* 43 (Oct. 2010), p. 465303. DOI: 10.1088/1751-8113/43/46/465303.

[98] Douglas A Reynolds. “Gaussian mixture models”. In: *Encycl. Biometr.* 741.659-663 (2009). DOI: 10.1007/978-1-4899-7488-4_196.

[99] Raúl Rojas. *Neural networks: a systematic introduction.* Springer Science & Business Media, 2013.

[100] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning.* MIT press, 2016. DOI: 10.1007/s10710-017-9314-z.

[101] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation Learning: A Review and New Perspectives". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (2013), pp. 1798–1828. DOI: 10.1109/TPAMI.2013.50.

[102] Ajay Shrestha and Ausif Mahmood. "Review of Deep Learning Algorithms and Architectures". In: *IEEE Access* 7 (2019), pp. 53040–53065. DOI: 10.1109/ACCESS.2019.2912200.

[103] Susanna Kirchhoff et al. "Optimized cross-resonance gate for coupled transmon systems". In: *Phys. Rev. A* 97 (4 Apr. 2018), p. 042348. DOI: 10.1103/PhysRevA.97.042348. URL: https://link.aps.org/doi/10.1103/PhysRevA.97.042348.

[104] Michael H Goerz et al. "Charting the circuit QED design landscape using optimal control theory". In: *npj Quantum Information* 3.1 (2017), p. 37. DOI: 10.1038/s41534-017-0036-0.

[105] Matthew Neeley et al. "Emulation of a Quantum Spin with a Superconducting Phase Qudit". In: *Science* 325.5941 (2009), pp. 722–725. DOI: 10.1126/science.1173440.

[106] Howard Georgi and Kannan Jagannathan. *Lie Algebras in Particle Physics.* Vol. 50. 1982, pp. 1053–1053.

[107] Brian C. Hall. *Lie Groups, Lie Algebras, and Representations.* Springer Chams, 2013. DOI: https://doi.org/10.1007/978-3-319-13467-3.

[108] A. Roggero and A. Baroni. "Short-depth circuits for efficient expectation-value estimation". In: *Phys. Rev. A* 101 (2 Feb. 2020), p. 022328. DOI: 10.1103/PhysRevA.101.022328.

[109] Alexandre Blais et al. "Cavity quantum electrodynamics for superconducting electrical circuits: An architecture for quantum computation". In: *Phys. Rev. A* 69.6 (2004), p. 062320. DOI: 10.1103/PhysRevA.69.062320.

[110] A. Wallraff et al. "Approaching unit visibility for control of a superconducting qubit with dispersive readout". In: *Phys. Rev. Lett.* 95.6 (2005), p. 060501. DOI: 10.1103/PhysRevLett.95.060501.

[111] Theodore Walter et al. "Rapid high-fidelity single-shot dispersive readout of superconducting qubits". In: *Phys. Rev. App.* 7.5 (2017), p. 054020. DOI: 10.1103/PhysRevApplied.7.054020.

[112] Y. Sunada et al. "Fast Readout and Reset of a Superconducting Qubit Coupled to a Resonator with an Intrinsic Purcell Filter". In: *Phys. Rev. App.* 17 (4 Apr. 2022), p. 044016. DOI: 10.1103/PhysRevApplied.17.044016.

[113] Alexander PM Place et al. “New material platform for superconducting transmon qubits with coherence times exceeding 0.3 milliseconds”. In: *Nature Comm.* 12.1 (2021), pp. 1–6. DOI: 10.1038/s41467-021-22030-5.

[114] Ani Nersisyan et al. “Manufacturing low dissipation superconducting quantum processors”. In: *2019 IEEE International Electron Devices Meeting (IEDM).* 2019, pp. 31–1. DOI: 10.1109/IEDM19573.2019.8993458.

[115] Easwar Magesan et al. “Machine learning for discriminating quantum measurement trajectories and improving readout”. In: *Phys. Rev. Lett.* 114.20 (2015), p. 200501. DOI: 10.1103/PhysRevLett.114.200501.

[116] Alireza Seif et al. “Machine learning assisted readout of trapped-ion qubits”. In: *Jour. Phys. B* 51.17 (Aug. 2018), p. 174006. DOI: 10.1088/1361-6455/aad62b.

[117] Benjamin Lienhard et al. “Deep-Neural-Network Discrimination of Multiplexed Superconducting-Qubit States”. In: *Phys. Rev. App.* 17 (1 Jan. 2022), p. 014024. DOI: 10.1103/PhysRevApplied.17.014024.

[118] David Quiroga, Prasanna Date, and Raphael Pooser. “Discriminating Quantum States with Quantum Machine Learning”. In: *2021 IEEE International Conference on Quantum Computing and Engineering (QCE).* IEEE. 2021, pp. 481–482. DOI: 10.1109/ICRC53822.2021.00018.

[119] Xian Wu et al. “High-fidelity software-defined quantum logic on a superconducting qudit”. In: *Phys. Rev. Lett.* 125.17 (2020), p. 170502. DOI: 10.1103/PhysRevLett.125.170502.

[120] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning.* Vol. 4. 4. Springer, 2006. DOI: 10.1108/03684920710743466.

[121] Yoshua Bengio, Aaron Courville, and Pascal Vincent. “Representation Learning: A Review and New Perspectives”. In: *IEEE Trans.Patt. Analy. and Mach. Intell.* 35.8 (2013), pp. 1798–1828. DOI: 10.1109/TPAMI.2013.50.

[122] Luca Pasa and Alessandro Sperduti. “Pre-training of recurrent neural networks via linear autoencoders”. In: *Adva. Neur. Infor. Process. Syst.* 27 (2014).

[123] Bun Theang Ong, Komei Sugiura, and Koji Zettsu. “Dynamic pre-training of deep recurrent neural networks for predicting environmental monitoring data”. In: *2014 IEEE International Conference on Big Data (Big Data).* IEEE. 2014, pp. 760–765. DOI: 10.1109/BigData.2014.7004302.

[124] Liangyu Chen et al. “Transmon qubit readout fidelity at the threshold for quantum error correction without a quantum-limited amplifier”. In: *arXiv preprint arXiv:2208.05879* (2022). DOI: 10.48550/arXiv.2208.05879.

[125] Sigmund Kohler. “Dispersive readout: Universal theory beyond the rotating-wave approximation”. In: *Phys. Rev. A* 98.2 (2018), p. 023849. DOI: 10.1103/PhysRevA.98.023849.

[126] *Quantum Orchestration platform.* 2021. URL: https://qm-docs.qualang.io/introduction/qop_overview.

[127] Morten Kjaergaard et al. “Superconducting qubits: Current state of play”. In: *An. Rev. Cond. Matt. Phys.* 11 (2020), pp. 369–395. DOI: 10.1146/annurev-conmatphys-031119-050605.

[128] He Liang Huang et al. “Superconducting quantum computing: a review”. In: *Sci. Chi. Info. Sci.* 63.8 (2020), pp. 1–32. DOI: 10.1007/s11432-020-2881-9.

[129] Isaac Westby et al. “FPGA acceleration on a multi-layer perceptron neural network for digit recognition”. In: *Jour. Supercomput.* 77.12 (2021), pp. 14356–14373. DOI: 10.1007/s11227-021-03849-7.

[130] Rijad Sarić et al. “FPGA-based real-time epileptic seizure classification using Artificial Neural Network”. In: *Biomed. Sig. Process. Contr.* 62 (2020), p. 102106. DOI: 10.1016/j.bspc.2020.102106.

[131] Yutana Jewajinda and Prabhas Chongstitvatana. “FPGA-based online-learning using parallel genetic algorithm and neural network for ECG signal classification”. In: *ECTI-CON2010: The 2010 ECTI International Confernce on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology.* IEEE. 2010, pp. 1050–1054.

[132] Shubham Gandhare and B Karthikeyan. “Survey on FPGA architecture and recent applications”. In: *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN).* IEEE. 2019, pp. 1–4. DOI: 10.1109/ViTECoN.2019.8899550.

[133] *Keras website.* https://keras.io/.

[134] *Keras website.* https://scikit-learn.org/stable/index.html.

[135] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

List of Figures

List of Tables