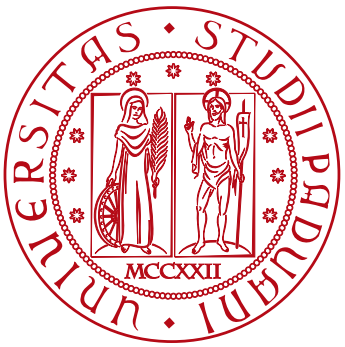

UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Fisica e Astronomia "Galileo Galilei"

Master Degree in Physics of Data

Final Dissertation

Gradient-based quantum optimal control on superconducting qubit systems

Thesis supervisor

Prof. Simone Montangero

Thesis co-supervisors

Dr. Felix Motzoi

Prof. Matteo Rizzi

Candidate

Samuele Piccinelli

Academic Year 2021/2022

Abstract

Quantum technologies are expected to help solve many of today's global challenges, revolutionizing several fields such as computing, sensing and secure communications. In this regard, the need for precise manipulation of the dynamics of a quantum system and its optimization has given rise to the field of quantum control theory. In the search for optimal controls, accurate derivatives are a possible method to traverse and ultimately converge in quantum optimization landscapes.
In this work we study an efficient algorithm for computing analytically-exact derivatives by formulating the control problem in the basis that diagonalizes the control Hamiltonian and applying a specific Trotterized time propagation scheme. The method is numerically verified for a system of superconducting transmon qubits in the few- and many body regime using matrix product states. The comparison between the results obtained using an exact dynamics via Krylov subspace methods shows how the approximate dynamics ultimately sets a trade-off between computational complexity and quality of the final solutions.

JÜLICH
Forschungszentrum

Contents

Notation

Multidimensional quantities are denoted in bold: vectors with lowercase letters ($\boldsymbol{x}$), higher order tensors with uppercase letters ($\boldsymbol{X}$).
The following abbreviations will be used:

Chapter 1

Introduction

Since the discovery of the black body radiation law by Max Planck in 1900, quantum theory has lead to profound and radical changes in our society in terms of technological progress and collective philosophical thought. With the advent of the theory of wave mechanics counter-intuitive concepts like entanglement and superposition were introduced: such phenomena, so distant from our day-to-day experience, are now fundamental cornerstones of quantum information processing, a field which has experienced over the past three decades an impressive growth. This second quantum revolution, sparked by Feynman's famous paper [1], has interested academia as well as industry in a race to practical quantum advantage, i.e. to build programmable quantum devices capable of solving a problem that no classical computer can solve in any feasible amount of time.

The physical realization of controllable quantum devices is however an extremely challenging engineering task, since the fundamental building blocks of computation - the qubits - are highly susceptible to any external source of noise. Up to these days, this is the greatest obstacle preventing any concrete breakthrough towards real world quantum applications. A common trait for technologies relying on quantum laws such as metrology, communications or computation, is thus the need for stable and precise control of quantum systems, capable of limiting the noise and synthesize a specific transformation with a given precision. This is the field of study of optimal control theory, which will be the central topic of this thesis.

Organization

After the introduction given in Chapter 1, in Chapter 2 we will deal with the fundamentals of quantum computation, namely the quantum logical bit, quantum gates and quantum hardware, focusing on superconducting qubits. In Chapter 3 we will discuss the equations governing the time evolution and frame transformations, which allow for a simpler representation of the dynamics of a quantum system. We will then concentrate in Chapter 4 on the numerical simulation and optimization concepts that will be used throughout this thesis. In Chapter 5 we will give an introduction to optimal control theory, describing some common derivative based methods for optimal control and focusing on a particular methodology aimed at reducing computational cost and time. In Chapter 6 we will give a brief introduction on tensor network methods, which will be later used to expand our discussion into the many-body regime. In Chapter 7 we will present the results of this thesis: finally, in Chapter 8 conclusions will be drawn and possible future prospects will be discussed.

Chapter 2

Universal quantum computation

This chapter is aimed at giving an overview of the main principles and notation used in Quantum Computing (QC) which we will use in the following chapters. In the last section we will describe one of the most prominent platform for constructing a multi-qubit quantum processor, namely superconducting qubits. For an extensive and recent overview of the physics of quantum information we refer to [2]: parts of this chapter have been based on [3].

2.1 Gate-based quantum computing

2.1.1 The quantum bit

A qubit is a two-level system whose quantum mechanical state displays phase coherence between two basis states, $|0\rangle$ and $|1\rangle$. The definition of coherence lies at the heart of quantum computation: given a generic state of a qubit as linear superposition of the two basis states

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \qquad\qquad \alpha, \beta \in \mathbb{C}\ , |\alpha|^2 + |\beta|^2 = 1\ , \tag{2.1}$$

we can define the relative quantum phase φ as

$$\varphi \equiv \arg(\alpha^\star \beta)\ , \tag{2.2}$$

where we denote with * complex conjugation. A state with a finite value of φ is said to be coherent. The complex coefficients α and β are analogous to amplitudes of a physical wave, and the vector $(\alpha, \beta)^\intercal$ is referred to as the wave vector.

An arbitrary qubit state can also be expressed via its density matrix form,

$$\hat{\rho} = |\psi\rangle \langle\psi| = \begin{pmatrix} |\alpha|^2 & \alpha\beta^\star \\ \alpha^\star\beta & |\beta|^2 \end{pmatrix} . \tag{2.3}$$

The diagonal entries of the density matrix (populations) give the effective probabilities of the respective basis states, while the off-diagonal elements (coherences) represent the phase coherences between these same states.

A qubit state is often represented by an arrow, called the Bloch vector, in the Bloch sphere (see Fig. 2.1). Conventionally the qubit quantization axis sits on the z axis and the north (south) pole represent the $|0\rangle$ ($|1\rangle$) state: the longitudinal component of the Bloch vector corresponds to the polarization of the qubit, whereas the transverse component corresponds to the coherence between the two basis states. In spherical coordinates, a generic quantum

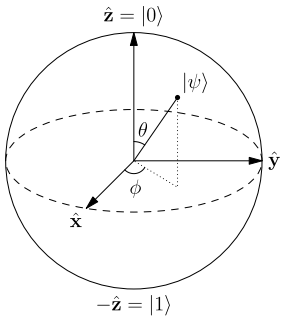

Figure 2.1: Bloch sphere representation of the qubit state.

state can be represented using a Block vector $\vec{s}$ of the form

$$\vec{s} = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta) , \tag{2.4}$$

where θ (ϕ) is the polar (azimuthal) angle. The Bloch vector is related to the density matrix via the Pauli matrices,

$$\hat{\rho} = \frac{1}{2}(\mathbb{1} + s_x\hat{\sigma}_x + s_y\hat{\sigma}_y + s_z\hat{\sigma}_z) \tag{2.5}$$

$$= \begin{pmatrix} \cos^2(\theta/2) & \mathrm{e}^{-i\phi}\cos(\theta/2)\sin(\theta/2) \\ \mathrm{e}^{i\phi}\cos(\theta/2)\sin(\theta/2) & \sin^2(\theta/2) \end{pmatrix} . \tag{2.6}$$

Practically, a qubit is (for the majority of cases) implemented by the two lowest states of a quantum system, such as (artificial or natural) atoms. This subspace is called the computational subspace: in general, any Hilbert space whose dimension is truncated into two can be used as a qubit.
Finally, the dynamics of the qubit state are often conveniently described in the rotating frame (see Chapter 3) which allows one to focus on the dynamics of interest, effectively eliminating trivial evolution via a unitary transformation.

2.1.2 Superposition and measurement

The fundamental difference between quantum and classical computation lies in two requirements of quantum mechanics that QC is able to simultaneously satisfy. From the superposition principle of quantum mechanics immediately follows the possibility of creating states without a classical analogue (Eq. (2.1)). Quantum operations are described by the unitary group [4], in contrast to digital computations and probabilistic computation, which are based on permutations and stochastic matrices respectively.
The second constraint is that the basis states must be distinguishable: this distinguishability in turn ensures that any intermediary states have no interpretation as measurable elements of reality and fundamentally separates QC from models of conventional analog computation.
To produce an output we have to make a measurement on the qubit, which will only have two possible results, either $|0\rangle$ or $|1\rangle$. The state information can can only be obtained from a large set of perfectly identical qubits and reconstruct, at least in principle, the exact coefficients of the original $|\psi\rangle$ via quantum tomography techniques. In particular, the information extracted will be proportional to the number of measurements taken, thus returning to a case equivalent to a classical system of many bits. Colloquially speaking, the advantage of QC can be summarized in that the complex numbers defining $|\psi\rangle$ remain available during the quantum process - i.e. at any instant preceding the measurement - and are thus capable of influencing its evolution.

2.1.3 Quantum gates

A quantum gate is a discrete control acting on qubits inducing the unitary evolution of the quantum states of the qubits: quantum computation is basically a series of quantum gate operations. If we consider a closed quantum system described by the Hamiltonian $\hat{H}$, a gate operation is implemented by engineering the system Hamiltonian (see Chapter 5) such that the resulting (unitary) evolution of the qubits implements the target gate.
A set of universal quantum gates is any set of gates to which any operation possible on a quantum computer can be reduced: in other words, a set is universal if any unitary operation can be expressed as a finite sequence of gates from the set. An example is given by the single-qubit gates (the rotation operators $\{R_i(\theta)\}_{i=x,y,z}$, the phase shift gate $P(\phi)$) and the CNOT gate. The set of rotations is critical to QC because the gates create the intermediate non-computational (non-basis) states discussed in the last section. Moreover, the Solovay–Kitaev theorem guarantees that for unitaries on a constant number of qubits any quantum operation can be approximated by a sequence of gates from the finite universal set.
The set composed by Hadamard, phase shift and CNOT form a group called

the Clifford group, which is particularly relevant for QC, particularly for quantum error correction. The Clifford set alone is not universal and can be efficiently simulated on a probabilistic classical computer by the Gottesman–Knill theorem. The addition of a T gate makes however the set universal.
The unitary matrices form of quantum gates is not here reported for brevity but can be easily found in most quantum information science textbooks, as e.g. [5].

2.2 Quantum hardware

A viable implementation of a quantum computer has to meet a set of requirements known as the DiVincenzo criteria [6, 7]: a quantum computer has to operate on an easily extendable set of well characterized qubits (1) whose coherence times are long enough to allow coherent operation (2) and whose initial state can be set (3): moreover, the qubits of the system can be operated on logically with a universal set of gates (4) and the final state can be measured (5). Moreover, an essential aspect to enable the operation of any quantum computer is the possibility of efficiently correcting errors, which are inevitable and much more likely than in classical computers.
Today quantum processors are implemented using a range of physical systems containing quantum degrees of freedom that can be controlled. An exhaustive overview of this platforms is outside the scope of this thesis: we will instead concentrate on one on the most flexible and perhaps promising QC technologies, i.e. Superconducting (SC) qubits which in the last years have been the focus of research efforts of companies such as Google [8] and IBM [9].

2.2.1 Superconducting qubits

In SC qubits the information is stored in the quantum degrees of freedom of anharmonic oscillators constructed from superconducting circuit elements.
In superconducting materials electrons are effectively attracted to each other and combine as Cooper pairs, i.e. bosonic particles with charge and mass double that of an electron, which take energy (not available thermally at low temperatures) to break up. Historically, the first type of superconducting qubit discovered was the Cooper Pair Box, consisting of a superconducting island that can (cannot) possesses an extra Cooper pair of charge $2e$, which is identified as the ground state $|0\rangle$ (excited state $|1\rangle$).
However, the Cooper Pair Box is particularly sensitive the noise caused by fluctuating charges nearby, the so called charge noise, which led to the design of a qubit with higher-order energy levels, the transmon (from transmission-line shunted plasma oscillation qubit). At the price of sacrificing anharmonicity, i.e. the difference between the states transition frequencies, charge

noise is suppressed while still allowing the lowest two levels to be addressed as a qubit. The quantum states are now encoded in oscillations of Cooper pairs across a tunnel junction between two superconducting islands, with the excited $|1\rangle$ state oscillating at a higher frequency than the ground $|0\rangle$. We will now derive the transmon Hamiltonian in a more formal way.

Starting from an LC circuit, we quantize its Hamiltonian, obtaining the Quantum Harmonic Oscillator (QHO)

$$\hat{H} = \frac{\hat{Q}^2}{2C} + \frac{\hat{\Phi}^2}{2L} \,, \tag{2.7}$$

which is analogous to that of a mechanical harmonic oscillator, with mass $m = C$, and resonant frequency $\omega = 1/\sqrt{LC}$. The operators $\hat{Q}$, $\hat{\Phi}$, just like position an momentum, satisfy the commutation relation

$$\left[\hat{Q}, \hat{\Phi}\right] = i\hbar \,. \tag{2.8}$$

By defining the reduced flux $\phi = 2\pi\Phi/\Phi_0$ and the reduced charge $n = Q/2e$, where $\Phi_0 = h/2e$ is the superconducting magnetic flux quantum, Eq. (2.7) can be more easily written as

$$\hat{H}_{\mathrm{QHO}} = 4\mathcal{E}_C\hat{n}^2 + \frac{1}{2}\mathcal{E}_L\hat{\phi}^2 \,, \tag{2.9}$$

where $\mathcal{E}_C = e^2/2C$ is the charging energy required to add each electron of the Cooper-pair to the island and $\mathcal{E}_L = (\Phi_0/2\pi)^2$ is the inductive energy. Moreover, the quantum operator $\hat{n}$ is the excess number of Cooper-pairs on the island, and $\hat{\phi}$ the reduced flux across the inductor. These two operators form a canonical conjugate pair, obeying the commutation relation $\left[\hat{n}, \hat{\phi}\right] = i$. The solution to this eigenvalue problem gives an infinite series of eigenstates $\{|k\rangle\}_{k=0,1,\ldots}$ whose corresponding eigenenergies $\mathcal{E}_k$ are all equidistantly spaced, $\mathcal{E}_{k+1} - \mathcal{E}_k = \hbar\omega_r$, with $\omega_r = \sqrt{8\mathcal{E}_L\mathcal{E}_C} = \sqrt{LC}$ the resonant frequency of the system.
In order to be able to drive transitions between energy levels independently we introduce a level of anharmonicity (i.e. non linearity), which consists in requiring for the transition frequencies $\omega_q^{0\to1}$ and $\omega_q^{1\to2}$ to be sufficiently different from one another to be individually addressable. This allows to properly define a computational subspace consisting of only two energy states in between which transitions can be driven without also exciting other levels in the system. In practice, the amount of anharmonicity sets a limit on how short the pulses used to drive the qubit can be. Experimentally, this is obtained by replacing the linear inductor of the QHO with a Josephson junction, playing the role of a nonlinear inductor. This results in a modified Hamiltonian in the form

$$\hat{H}_{\mathrm{TSN}} = 4\mathcal{E}_C\hat{n}^2 - \mathcal{E}_J\cos\hat{\phi} \,, \tag{2.10}$$

where the Josephson energy $\mathcal{E}_J = I_0\Phi_0/2\pi$ replaces the inductive energy from the QHO. The regime $\mathcal{E}_J \gg \mathcal{E}_C$ is characterized by a low sensitivity to charge noise: to reach it, the junction is shunted with a large capacitor, obtaining a circuit commonly known as the transmon qubit.
We can now define the creation and annihilation operators in terms of the zero-point fluctuations of charge n_0 and phase ϕ_0,

$$n_0 = \left(\frac{\mathcal{E}_J}{32\mathcal{E}_C}\right)^{1/4} \qquad \phi_0 = \left(\frac{2\mathcal{E}_C}{\mathcal{E}_J}\right)^{1/4} , \tag{2.11}$$

as

$$\hat{n} = in_0(\hat{b}^\dagger + \hat{b}) \qquad \hat{\phi} = \hat{\phi}_0(\hat{b}^\dagger + \hat{b}) , \tag{2.12}$$

where $\hat{b}^\dagger = \sum_j \sqrt{j+1}\,|j\rangle\,|j+1\rangle$ denotes the transmon annihilation operator. We will now operate a few approximations. First, we note that $\phi \ll 1$ because in the transmon regime $\mathcal{E}_J/\mathcal{E}_C \gg 1$: we can thus expand the term $\cos\hat{\phi}$ of Eq. (2.10) in series obtaining

$$\hat{H}_{\text{TSN}} \approx \sqrt{8\mathcal{E}_C\mathcal{E}_J}\left(\hat{b}^\dagger\hat{b} + \frac{1}{2}\right) - \mathcal{E}_J - \frac{\mathcal{E}_C}{12}(\hat{b}^\dagger + \hat{b})^4 . \tag{2.13}$$

We now expand the terms of the transmon operator $\hat{b}$ and drop the fast-rotating terms characterized by an uneven number of $\hat{b}$ and $\hat{b}^\dagger$, neglecting constants that have no influence on the transmon dynamics. We define $\omega_0 = \sqrt{8\mathcal{E}_C\mathcal{E}_J}$ and identify $\delta = -\mathcal{E}_C$ as the transmon anharmonicity, finally coming to

$$\hat{H}_{\text{TSN}} = \left(\omega_0 + \frac{\delta}{2}\right)\hat{b}^\dagger\hat{b} + \frac{\delta}{2}(\hat{b}^\dagger\hat{b})^2 . \tag{2.14}$$

From this expression, it is immediate to see how the transmon levels have energy spacings differing by the anharmonicity δ as $\omega_{j+1} - \omega_j = \omega + \delta j$, $\omega = \omega_0 + \delta$ which corresponds to the drive frequency of the transmon qubit for the $\omega_1 - \omega_0$ transition.
This discussion, far from being exhaustive, can be further extended to other topics concerning superconducting circuits in [3, 10].

We now focus on the problem of a system of $N = 2$ transmon qutrits dispersively coupled through a linear microwave resonator activated by a microwave field. We report, as done in Appendix E of [11], the expression of the effective Hamiltonian where the cavity is removed adiabatically and the qutrit-cavity coupling is replaced with an effective qutrit-qutrit coupling as in [12]. The starting point is to model each transmon as an anharmonic Duffing oscillator as in Eq. (2.14) and describe the transmon cavity coupling

via the Jaynes-Cummings model, which in the absence of control is given by

$$\hat{H}_0 = \sum_{j=1,2} \omega b_j^\dagger b_j + \frac{\delta_j}{2} b_j^\dagger b_j (b_j^\dagger b_j - 1) + \omega_r a^\dagger a \tag{2.15}$$

$$+ \sum_{j=1,2} g_j (a_j b_j^\dagger + a^\dagger b_j) \ , \tag{2.16}$$

where ω_r is the cavity resonance frequency and g_j are transmon-cavity couplings.
First, we eliminate the cavity by going to a frame rotating at ω_r (see Chapter 3) via $\mathcal{R} = \exp\left[-i\omega_r(b_1^\dagger b_1 + b_2^\dagger b_2 + a^\dagger a)\right]$. This gives

$$\hat{H} = \sum_{j=1,2} \Delta_j b_j^\dagger b_j + \frac{\delta_j}{2} b_j^\dagger b_j (b_j^\dagger b_j - 1) + \sum_{j=1,2} g_j (a b_j^\dagger + a^\dagger b_j) \ , \tag{2.17}$$

where $\Delta_j = \omega_j - \omega_r$. We then perform a Schrieffer-Wolff transformation [13] using

$$S = \sum_{j=1,2} \frac{g_j}{\Delta_j} (a b_j^\dagger - a^\dagger b_j) \tag{2.18}$$

in order to eliminate the cavity. The resulting Hamiltonian, once any constant energy shifts have been removed and the cavity neglected, is

$$\hat{H} = \sum_{j=1,2} \tilde{\omega}_j b_j^\dagger b_j + \frac{\delta_j}{2} b_j^\dagger b_j (b_j^\dagger b_j - 1) + J(b_1^\dagger b_2 + b_1 b_2^\dagger) \ . \tag{2.19}$$

Here we see that the transmon-cavity coupling has been replaced with an effective transmon-transmon coupling where

$$J = \frac{g_1 g_2 (\Delta_1 + \Delta_2)}{\Delta_1 \Delta_2}, \qquad \tilde{\omega}_j = \omega_j + \frac{g_j^2}{\Delta_j} \tag{2.20}$$

is now the dressed transmon state (a term we will clarify in Section 3.2.1). As a final step we perform a second rotation $\mathcal{R}' = \exp\left[-i\tilde{\omega}_2(b_1^\dagger b_1 + b_2^\dagger b_2)\right]$ such that the detuning becomes $\Delta = \tilde{\omega}_2 - \tilde{\omega}_1$.
We finally come to

$$\hat{H}_{\text{eff}}(t) = \Delta b_1^\dagger b_1 + \sum_{j=1,2} \frac{\delta_j}{2} b_j^\dagger b_j (b_j^\dagger b_j - 1) + J(b_1^\dagger b_2 + b_1 b_2^\dagger) \tag{2.21}$$

$$+ \Omega(t)(b_1^\dagger + b_1) \ , \tag{2.22}$$

a central equation of this work. Note that we added a direct drive on the first transmon $H^c(t) = \Omega(t)(b_1^\dagger + b_1)$ [12], which will be later applied also to the other qutrits when considering $N > 2$.

Chapter 3

Quantum dynamics and frame transformations

The Schrödinger equation governs the wave function of a quantum-mechanical system. Solving it is often a non-trivial task (due to e.g. multiple qubit couplings and time-dependence), even with numerical techniques which are still limited by an exponential scaling with number of qubits. One of the most diffuse methods developed to simplify such problem is to apply unitary transformations to the Hamiltonian of the system: such an operation can result in a simplified version of the Schrödinger equation which has nonetheless the same solution as the original and can be efficiently calculated (generally polynomial in the number of qubits). Moreover, this process has the added benefit of allowing for a better conceptual understanding of the physical processes in act.
In this chapter we first derive the time evolution equations governing quantum dynamics and successively discuss the time-dependent transformations that make the aforementioned tractable representations possible. We will use [14] as a reference point.

3.1 Quantum dynamics

The time evolution of a state vector in the quantum mechanical Hilbert space for an isolated system is governed by the Liouville-von Neumann equation

$$i\frac{d}{dt}\hat{\rho}(t) = \left[\hat{H}(t), \hat{\rho}(t)\right] \ , \tag{3.1}$$

where $\hat{H}(t)$ is the Hamiltonian operator, which is in general a function of time t, and $\hat{\rho}$ is the density matrix describing the state of the system: if additionally the initial state is pure one can alternatively rewrite it to describe

the dynamics of the plain state $|\psi(t)\rangle$. In that case the time evolution is determined by the Schrödinger equation,

$$i\hbar\frac{d}{dt}\,|\psi(t)\rangle = \hat{H}(t)\,|\psi(t)\rangle \ . \tag{3.2}$$

This equation is valid in the Schrödinger picture, where the state evolves over time, while the operators acting on it do not. However, the matrix nature of the density matrix shows a strong analogy with the equation of time evolution of operators $A^{(H)}(t)$ in Heisenberg's picture where $\hat{\rho}$ is independent of time:

$$i\hbar\frac{dA^{(H)}(t)}{dt} = -\left[\hat{H}(t), A^{(H)}(t)\right] \ , \tag{3.3}$$

up to a sign difference given by the different picture. In this section and for the rest of this work we will focus on the (simpler) case of pure states.

The solution to Eq. (3.2) defines the time evolution operator $\hat{\mathcal{U}}(t, t_0)$,

$$|\psi(t)\rangle = \hat{\mathcal{U}}(t, t_0)\,|\psi(t_0)\rangle \ , \tag{3.4}$$

where the state vector at time t_0 provides the initial condition for the time evolution of $|\psi(t)\rangle$. Plugging Eq. (3.4) in Eq. (3.2) yields a partial differential equation for $\hat{\mathcal{U}}(t, t_0)$; now, since the initial state is arbitrary, it follows that the propagator must satisfy

$$i\hbar\frac{\partial}{\partial t}\hat{\mathcal{U}}(t, t_0) = \hat{H}(t)\hat{\mathcal{U}}(t, t_0) \tag{3.5}$$

subject to the initial condition $\hat{\mathcal{U}}(t_0, t_0) = \mathbb{1}$. By Taylor expansion, one can derive an expression for $\hat{\mathcal{U}}(t + \delta t, t)$ in the limit $t \to 0$,

$$\hat{\mathcal{U}}(t + \delta t, t) = \exp\left[-i\hat{H}(t)\delta t/\hbar\right] + \mathcal{O}(\delta t^2) \ . \tag{3.6}$$

To obtain an explicit formula for the solution to Eq. (3.4) we discretize time in regular δt intervals,

$$t \in [t_1, t_2, \ldots, t_N] = [0, \delta t, \ldots, T] \qquad t_j = (j-1)\delta t \ , \tag{3.7}$$

with time indices denoted as subscripts. Physical quantities evaluated at these grid points are similarly denoted by $\hat{H}_n = \hat{H}(t_n)$. By recursively applying the fundamental composition law of the time evolution propagator, it follows that

$$\hat{\mathcal{U}}(t, t_0) = \prod_{n=1}^{N} \hat{\mathcal{U}}\left(t_0 + n\delta t, t_0 + (n-1)\delta t\right) \ , \tag{3.8}$$

which can be rewritten as a product of exponential using the expansion in Eq. (3.6). At this point, if one further assumes for the Hamiltonian operator to satisfy $[\hat{H}(t_i), \hat{H}(t_j)] = 0$ for all choices of t_i and t_j, the expression above simplifies to

$$\hat{\mathcal{U}}(t, t_0) = \lim_{\delta t \to 0} \exp\left[-\frac{i\delta t}{\hbar} \sum_{n=0}^{N-1} \hat{H}(t_0 + n\delta t)\right] \tag{3.9}$$

and by finally by taking the limit,

$$\hat{\mathcal{U}}(t, t_0) = \exp\left[-\frac{i}{\hbar} \int_{t_0}^{t} \hat{H}(t)\, dt\right] . \tag{3.10}$$

However, this is not the general case: for $[\hat{H}(t_i), \hat{H}(t_j)] \neq 0$ the evaluation of Eq. (3.8) via the nested commutators of the Baker-Campbell-Hausdorff (BCH) formula becomes prohibitive. A different approach consists in rewriting Eq. (3.5) in integral form,

$$\hat{\mathcal{U}}(t, t_0) = \mathbb{1} - \frac{i}{\hbar} \int_{t_0}^{t} \hat{H}(t')\hat{\mathcal{U}}(t', t_0)\, dt' \tag{3.11}$$

which can be used to obtain a formal expansion of $\hat{\mathcal{U}}(t, t_0)$ in form of a Dyson series,

$$\hat{\mathcal{U}}(t, t_0) = \mathbb{1} + \sum_{n=1}^{\infty} \left(-\frac{i}{\hbar}\right)^n \int_{t_0}^{t} dt_1 \int_{t_0}^{t_1} dt_2 \cdots \int_{t_0}^{t_{n-1}} dt_n \hat{H}(t_1)\hat{H}(t_2) \ldots \hat{H}(t_n) . \tag{3.12}$$

To rewrite the nth term of the previous expression in a more convenient form in which the integration limits are uncoupled, one must introduce the time-ordered product of operators,

$$\mathcal{T}\left[\hat{H}(t_1)\hat{H}(t_2) \ldots \hat{H}(t_n)\right] = \hat{H}(t_{i_1})\hat{H}(t_{i_2}) \ldots \hat{H}(t_{i_n}), \quad t_{i_1} > t_{i_2} > \ldots t_{i_n} \tag{3.13}$$

with n positive integer. That is, the time-ordered product of operators is an instruction to reorder the operators such that the time arguments of the corresponding operators decrease as one moves from left to right. Using this new formalism, one can rewrite Eq. (3.12) as

$$\hat{\mathcal{U}}(t, t_0) = \mathbb{1} + \sum_{n=1}^{\infty} \frac{1}{n!} \left(-\frac{i}{\hbar}\right)^n \int_{t_0}^{t} dt_1 \int_{t_0}^{t} dt_2 \cdots \int_{t_0}^{t} dt_n \mathcal{T}\left[\hat{H}(t_1)\hat{H}(t_2) \ldots \hat{H}(t_n)\right] . \tag{3.14}$$

The right hand side of the previous expression defines the time-ordered exponential, $\mathcal{T} \exp\left[-\frac{i}{\hbar} \int_{t_0}^{t} \hat{H}(t)\, dt\right]$: in light of this new definition, we conclude that the most general solution to Eq. (3.5) with the initial condition

$\hat{\mathcal{U}}(t_0, t_0) = \mathbb{1}$ is given by

$$\hat{\mathcal{U}}(t, t_0) = \mathcal{T} \exp\left[-\frac{i}{\hbar}\int_{t_0}^{t} \hat{H}(t)\, dt\right] . \tag{3.15}$$

As a final remark, we note that if the Hamiltonian operators evaluated at different times commute, then we may simply drop the $\mathcal{T}$ symbol, since it has no effect in this case.

3.2 Frame transformation general theory

The mathematical formulation of the dynamics of a quantum system is not unique. The two orthogonal approaches are considering either the basis states or the wavevectors completely static in time: these are known, as previously mentioned, as the Heisenberg and Schrödinger pictures respectively. We will refer to a basis which does not contain implicit time-evolution of the basis states as bare basis states and as dressed basis states when instead such time-dependence is allowed.
Ultimately, all description must satisfy the Schrödinger equation in Eq. (3.2).

3.2.1 Unitary transformation formula

A unitary transformation (or frame change) can be expressed in terms of a time-dependent Hamiltonian $\hat{H}(t)$ and unitary operator $V(t)$ - or equivalently as a transformation to a new dressed basis $|\phi\rangle = V(t)\,|\psi\rangle$. Under this change, the evolution transforms as

$$\begin{aligned}
-i\hat{H}_\phi\,|\phi\rangle &= \frac{d}{dt}\,|\phi\rangle \\
-i\hat{H}_\phi\,|\phi\rangle &= \left(\frac{d}{dt}V\right)|\psi\rangle + V\left(\frac{d}{dt}\,|\psi\rangle\right) \\
\hat{H}_\phi\,|\phi\rangle &= \left(V\hat{H}_\psi + i\dot{V}\right)|\psi\rangle \\
\hat{H}_\phi\,|\phi\rangle &= \left(V\hat{H}_\psi V^\dagger + i\dot{V}V^\dagger\right)|\phi\rangle \ ,
\end{aligned} \tag{3.16}$$

from which we can distill the Hamiltonian transformation $\hat{H}_\psi \to V\hat{H}_\psi V^\dagger + i\dot{V}V^\dagger =: \hat{H}_\phi$. From this last expression one can see how the total evolution of a system $\hat{H}_\phi$ is given by a combination of the explicit evolution of the wavevectors given by $\hat{H}_\phi$ and the implicit evolution of its basis states given by V. Moreover, if the wave function $\psi(t)$ satisfies the original equation, then $V(t)\psi(t)$ will satisfy the new equation: for the time evolution operator acting on an initial wavevector state $|\phi(0,T)\rangle$ to a final state $|\phi(0,T)\rangle$ it holds instead [14]

$$\hat{\mathcal{U}}_\phi(0,T) = V(T)\hat{\mathcal{U}}_\psi(0,T)V^\dagger(0) \ . \tag{3.17}$$

One of the most important transformation that one can perform is the diagonalization of the Hamiltonian, which is e.g. the primary method for its exponentiation (see also Section 4.4).

3.2.2 Equivalent frames

The additional requirement for the the state of the quantum system to be identical at the start and at the end of gate operations in both frames,

$$V(0) = V(T) = \mathbb{1} \,, \tag{3.18}$$

defines the subclass of transformations that leave the computational states unaltered. It goes without saying that such transformations leave complete freedom in the evolution $\hat{\mathcal{U}}$, which is left untouched.

3.2.3 Interaction frames

Unitary transformations can be seen as a generalization of the interaction (or Dirac) picture. The latter approach is a sort of a middle ground between the Schrödinger picture and the Heisenberg picture as both the quantum states and the operators carry time dependence. We consider an Hamiltonian in the form

$$\hat{H} = \hat{H}_0 + \hat{H}_1, \qquad \left[\hat{H}_1(t), \hat{H}_1(t')\right] = 0 \,, \tag{3.19}$$

containing both a free term and an interaction term. The correspondence to a unitary transformation can be shown by choosing $V(t) = \exp\left(i \int_0^t H_1(t)\, dt\right)$: then, by virtue of Eq. (3.16)

$$\begin{aligned} \hat{H}_{\text{eff}} &= V(\hat{H}_0 + \hat{H}_1)V^\dagger + i\dot{V}V^\dagger \\ &= VH_0V^\dagger + VH_1V^\dagger - H_1VV^\dagger \\ &= \exp\left(i\int_0^t H_1(t)\, dt\right) H_0 \exp\left(-i\int_0^t H_1(t)\, dt\right) \,, \end{aligned} \tag{3.20}$$

i.e. the Hamiltonian in Eq. (3.19) transforms as $VH_0V^\dagger$.
The state vectors in the interaction picture evolve in time according to the interaction term only, while operators only through the free Hamiltonian H_0. The evolution is given by $\hat{\mathcal{U}}_{\text{eff}} = V(T)\hat{\mathcal{U}}$ for a non-cyclic interaction term H_1 - in which case the interaction frame evolution $\hat{\mathcal{U}}_{\text{eff}}$ is equivalent to that of the original frame.

3.3 Phase transformations

If we loosen the boundary condition in Eq. (3.18) to include the indistinguishably by measurement,

$$\mathrm{e}^{i\phi_1} V(0) = \mathrm{e}^{i\phi_2} V(T) = \mathbb{1} \,, \tag{3.21}$$

we include all transformations which are diagonal. In the following we will introduce the case of rotating frame transformations with the example of a qubit subject to a linearly polarized drive [15], which in turn will serve as an excuse to introduce the Rotating Wave Approximation (RWA).

3.3.1 Rotating frames

The system Hamiltonian of a qubit subject to a linearly polarized drive in the laboratory frame reads

$$\hat{H}_{\text{lab}}(t) = \frac{\omega_0}{2}\sigma_z + \frac{H_1(t)}{2}\cos(\omega t + \phi)\sigma_x \ , \tag{3.22}$$

where ω_0 and ω are the qubit and drive frequencies respectively and $H_1(t)$ is the time-dependent amplitude function (envelope) of the applied drive. Furthermore, the drive is assumed to have a constant phase offset ϕ.
A suitable frame of reference for this system is the one rotating around the z axis with the drive frequency ω of the applied field. Transforming the Hamiltonian according to Eq. (3.16) with $V(t) = \mathrm{e}^{-i\omega t\sigma_z/2}$ leads to

$$\begin{aligned}\hat{H}_{\text{rot}}(t) = \frac{H_1(t)}{4}(\ &\cos(\phi)\sigma_x + \cos(2\omega t + \phi)\sigma_x \\ &+ \sin(\phi)\sigma_y - \sin(2\omega t + \phi)\sigma_y) + \frac{\Delta}{2}\sigma_z \ ,\end{aligned} \tag{3.23}$$

where we have introduced the amplitude function $H_1(t)$ and the detuning $\Delta = \omega_0 - \omega$. The period of the counter-rotating field of the drive is instead given by $t_c = \pi/\omega$. We now restrict to drives that are in resonance with the qubit, i.e. $\omega = \omega_0$ or $\Delta = 0$ and that have zero phase offset, $\phi = 0$: for this special case the rotating-frame Hamiltonian reduces to

$$\hat{H}_{\text{rot}}(t) = \frac{H_1(t)}{4}(\sigma_x + \cos(2\omega t)\sigma_x - \sin(2\omega t)\sigma_y) \ . \tag{3.24}$$

Thus, in the standard rotating frame the Hamiltonian contains non-commuting terms (which can be attributed to the counter rotating field of the drive) that oscillate quickly: when fully taken into consideration, these prevent a simple analytic form for the qubit's exact time evolution.
Applying the RWA results in a simple Hamiltonian which is straightforward to integrate for the cases in which the drive is weak ($|H_1(t) \ll \omega$), near resonant ($\omega \approx \omega_0$), and varies only slightly on the time scale dictated by the inverse qubit frequency $1/\omega$. To do this, one neglects the fast-oscillating terms in the rotating-frame Hamiltonian: this results in a significantly simplified expression that depends on time only through $H_1(t)$,

$$\hat{H}_{\text{RWA}}(t) = \frac{H_1(t)}{4}(\cos(\phi)\sigma_x + \sin(\phi)\sigma_y) + \frac{\Delta}{2}\sigma_z \ . \tag{3.25}$$

If $\Delta = 0$ and, for example, the qubit state is initialized to $|\psi(t)\rangle = |0\rangle$, solving the dynamics via RWA results in Rabi oscillations with period of $4\pi/H_1$ for a constant H_1. Finally, we highlight that for field strengths $|H_1(t)| \gtrsim 0.01\omega$ the RWA if often not applicable.

Chapter 4

Numerical simulation and optimisation

4.1 Time-slicing

In Section 3.1 we introduced temporal discretization as aid to finding the solution for the time evolution propagator: in order to numerically integrate the Schrödinger equation, the discretization becomes necessary. Again, the evolution is discretized into piece-wise constant intervals,

$$t \in [t_1, t_2, \ldots, t_{N_t}] = [0, \delta t, \ldots, T] \qquad t_j = (j-1)\delta t \;, \tag{4.1}$$

with time indices denoted as subscripts.
The process of discretization leads to a product of time evolution operators,

$$\hat{\mathcal{U}}(t, t_0) \approx \prod_{j=1}^{N_t - 1} \hat{\mathcal{U}}_j = \hat{\mathcal{U}}_{N_t - 1} \ldots \hat{\mathcal{U}}_2 \hat{\mathcal{U}}_1 \;, \tag{4.2}$$

where

$$\hat{\mathcal{U}}_n = \exp\left(-i \int_{t_n}^{t_n + \delta t} \hat{H}(t, u(t))\, dt\right) \tag{4.3}$$

is the propagator across the time interval $[t_n, t_{n+1}] = (t_n, t_n + \delta t)$. This approximation can be seen as deriving from substituting Eq. (3.10) in Eq. (3.8): indeed, it holds true for $\left[\hat{H}(t_j), \hat{H}(t_i)\right] = 0$ for all t_i, t_j. Instead, the error made with respect to Eq. (3.15) is given by $N_t - 1$ first-order Trotter expansions and vanishes when $\delta t \to 0 \iff N_t \to \infty$.

4.2 Optimization algorithms

4.2.1 Unconstrained optimization

An unconstrained optimization problem [16] takes the form

$$\min_{\boldsymbol{x}\in\mathbb{C}^n} f(\boldsymbol{x}) \tag{4.4}$$

for a (smooth) objective function $f : \mathbb{C}^n \to \mathbb{R}$. In most applications of optimization problems, one is usually interested in a global minimum $\boldsymbol{x}^\star$, which satisfies $f(\boldsymbol{x}^\star) \leq f(\boldsymbol{x})$ for all x in $\mathbb{C}^n$ (or at least for all x in the domain of interest). Optimization algorithms are often not guaranteed to converge to global minima but only yield local minima. Formally, a point $\boldsymbol{x}^\star$ is called a (strict) local minimum if there is a neighborhood $\mathcal{N}$ such that $f(\boldsymbol{x}^\star) \leq f(\boldsymbol{x})$ ($f(\boldsymbol{x}^\star) < f(\boldsymbol{x})$) $\forall \boldsymbol{x} \in \mathcal{N}$ (with $\boldsymbol{x} \neq \boldsymbol{x}^\star$). For the purposes of this discussion we will assume f at least continuously differentiable.

Line search methods

To traverse the optimization landscape, one has to take informed decisions using local information about the landscape topography for every new iteration k. The line search strategy is one of two fundamental approaches to this problem (the other one being the trust region). General line search methods for solving the optimization problem in Eq. (4.4) take the form

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k \ , \tag{4.5}$$

where $\alpha_k > 0$ is called the step length and $\boldsymbol{p}_k$ is called the search direction. Practically, $\boldsymbol{p}_k$ is direction along which the objective function will be reduced while α_k determines how far $\boldsymbol{x}$ should move along that direction.
A natural requirement is for $\boldsymbol{p}$ to be such that the slope of f in the direction $\boldsymbol{p}$ is negative. Since

$$\lim_{t\to 0^+} \frac{f(\boldsymbol{x}+t\boldsymbol{p}) - f(\boldsymbol{x})}{\|t\boldsymbol{p}\|} = \nabla f(\boldsymbol{x})^\intercal \boldsymbol{p} \ , \tag{4.6}$$

a vector $\boldsymbol{p} \neq 0$ is called descent direction of a function f at a point $\boldsymbol{x}$ if $\nabla f(\boldsymbol{x})^\intercal \boldsymbol{p} < 0$. The search direction $\boldsymbol{p}_k$ can be thus calculated from the current local gradient (as is the case for steepest descent but also for conjugate gradient and quasi-Newton directions) and possibly also the Hessian (as e.g. Newton direction). On the other hand, the step length α_k can be determined either exactly or inexactly: its values largely depend on both the chosen algorithm, search direction, problem scaling and how close to a minimum the iterate is (typically is of $\mathcal{O}(1)$ as it approaches convergence).
Each step in the problem of Eq. (4.4) involves (approximately) solving the sub-problem

$$\min_{\alpha} f(\boldsymbol{x}_k + \alpha \boldsymbol{p}_k) \ : \tag{4.7}$$

the minimization algorithm might either exactly solve it for $\alpha_k \in \mathbb{R}_+$, or rather loosely, asking for a sufficient decrease in the objective function. More quantitatively, as a requirement for any guessed α, practical implementations of line search algorithms make use of a specific set of inequalities, referred to as Wolfe conditions: a step length α_k is said to satisfy the Wolfe conditions restricted to the direction $\boldsymbol{p}_k$, if the following two inequalities hold:

$$1)\ f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k) - f(\boldsymbol{x}_k) \leq c_1 \alpha_k \nabla f(\boldsymbol{x}_k)^\intercal \boldsymbol{p}_k \text{ (Armijo rule)} \tag{4.8}$$

$$2)\ \nabla f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k)^\intercal \boldsymbol{p}_k \geq c_2 \nabla f(\boldsymbol{x}_k)^\intercal \boldsymbol{p}_k \text{ (curvature condition)}\ , \tag{4.9}$$

with $0 < c_1 < c_2 < 1$ and $c_1 \ll 1$, $c_1 \ll c_2$. The first condition in Eq. (4.8) ensures that the reduction in f is proportional to the step length and the directional derivative, guaranteeing a sufficient decrease in the objective function, while Eq. (4.9) makes sure that α_k does not become unreasonably small. This latter expression can be modified to force α_k to lie close to a critical point of Eq. (4.7) (restricted to the direction $\boldsymbol{p}_k$) by modifying the curvature condition as

$$|\nabla f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k)^\intercal \boldsymbol{p}_k| \leq -c_2 \nabla f(\boldsymbol{x}_k)^\intercal \boldsymbol{p}_k\ ; \tag{4.10}$$

together, Eq. (4.9) and Eq. (4.10) are known as strong Wolfe conditions.
It can be shown that there always exist step lengths satisfying the Wolfe conditions under reasonable assumptions on f.

Method of steepest descent

It is reasonable to choose $\boldsymbol{p}$ such that the slope of f in the direction $\boldsymbol{p}$ is as negative as possible. This leads to the minimization problem

$$\min_{\|\boldsymbol{p}\|=1} \nabla f(\boldsymbol{x})^\intercal \boldsymbol{p} \tag{4.11}$$

which can be easily solved. Indeed, if $\nabla f(\boldsymbol{x}) \neq 0$, then the above expression has the unique solution

$$\boldsymbol{p} = -\frac{\nabla f(\boldsymbol{x})}{\|\nabla f(\boldsymbol{x})\|} =: \boldsymbol{p}_{\min}\ , \tag{4.12}$$

and any vector of the form $\alpha \cdot \boldsymbol{p}_{\min}$ with $\alpha > 0$ is called direction of steepest descent.
An important result from optimization theory states that, for the case of steepest descent, the line search method converges (globally) to a stationary point. However, this does not imply that the method converges to a local minimum: such a statement requires to make additional information about the Hessian, which will in turn lead to the Newton methods discussed in the following section.

The Newton method

The Newton method offers a faster algorithm for minimizing $f(\boldsymbol{x})$, which is based on obtaining further information about the landscape topology by means of the second order derivative. This amounts to choosing the search direction as

$$\boldsymbol{p}_k = -H(\boldsymbol{x}_k)^{-1}\nabla f(\boldsymbol{x}_k) \,, \tag{4.13}$$

with H being the Hessian matrix. One way to motivate this expression for $\alpha_k = 1$ is to see it as the standard Newton method for solving the nonlinear equation $\nabla f(\boldsymbol{x}) = 0$; moreover, the same form minimize exactly the second order expansion to the objective function. In practice, the Newton method is often combined with the Armijo or the full Wolfe conditions. If the objective function is sufficiently simple near an optimal solution $\boldsymbol{x}^\star$, a nearby initial guess $\boldsymbol{x}_0$ will converge as

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^\star\| \leq C\|\boldsymbol{x}_k - \boldsymbol{x}^\star\|^q \,, \tag{4.14}$$

where $C \geq 0$ for most cases. The Newton method can be shown to converge quadratically ($q = 2$ in Eq. (4.14)).

Quasi-Newton methods

The computation of the Newton direction in Eq. (4.13) is often too expensive, due to both the need of determining the Hessian and the effort of solving a linear system. The general idea of quasi-Newton methods is to approximate $H(\boldsymbol{x}_k)$ by a symmetric positive matrix B_k, leading to a search direction of the form

$$\boldsymbol{p}_k = -B_k^{-1}\nabla f(\boldsymbol{x}_k) \,. \tag{4.15}$$

To ensure faster convergence than the steepest descent method, it is important to quantify the extent to which B_k shall approximate $H(\boldsymbol{x}_k)$: to this end, it is sufficient to require that B_k provides an increasingly accurate approximation of $H(\boldsymbol{x}_k)$ along the search direction $\boldsymbol{p}_k$:

$$\lim_{k\to\infty} \frac{\|(B_k - H(\boldsymbol{x}_k))\boldsymbol{p}_k\|}{\|\boldsymbol{p}_k\|} = 0 \,. \tag{4.16}$$

There is a lot of freedom in choosing B_k to satisfy the relation in Eq. (4.16). Quasi-Newton methods choose a sequence $\{B_0, B_1, B_2, \ldots\}$ satisfying the condition

$$B_{k+1}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) = \nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k) \,, \tag{4.17}$$

starting from an initial symmetric positive definite matrix B_0, preferably chosen to be an approximation of $H(\boldsymbol{x}_0)$. The expression in Eq. (4.17) mimics the approximation of a tangent vector by the secant vector: if we let

$$\boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k = \alpha_k \boldsymbol{p}_k \qquad \boldsymbol{y}_k = \nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k) \,, \tag{4.18}$$

then B_{k+1} satisfies $B_{k+1}\boldsymbol{s}_k = \boldsymbol{y}_k$, formally known as the secant equation. The term quasi-Newton is due to this "secant" condition from Newton's root-finding algorithm.
One usually further restricts the freedom in Eq. (4.16) by requiring the update $B_{k+1} - B_k$ to be a low-rank matrix, which allows for the efficient inversion of B_{k+1}. In particular, when requiring the update to be symmetric and of rank 1, the choice of B_{k+1} becomes unique:

$$B_{k+1} = B_k + \frac{(\boldsymbol{y}_k - B_k\boldsymbol{s}_k)(\boldsymbol{y}_k - B_k\boldsymbol{s}_k)^\intercal}{(\boldsymbol{y}_k - B_k\boldsymbol{s}_k)^\intercal\boldsymbol{s}_k} \,. \tag{4.19}$$

The quasi-Newton method resulting from Eq. (4.18) is called Symmetric Rank-1 (SR1). However, some care needs to be applied when using SR1: the denominator $(\boldsymbol{y}_k - B_k\boldsymbol{s}_k)^\intercal\boldsymbol{s}_k$ may become negative or even zero, preventing positive definiteness.
By far, the most popular quasi-Newton method is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) slgorithm, with update:

$$B_{k+1} = B_k + \frac{\boldsymbol{y}_k\boldsymbol{y}_k^\intercal}{\boldsymbol{y}_k^\intercal\boldsymbol{s}_k} - \frac{(B_k\boldsymbol{s}_k)(B_k\boldsymbol{s}_k)^\intercal}{\boldsymbol{s}_k^\intercal B_k\boldsymbol{s}_k} \,. \tag{4.20}$$

The analysis of BFGS is significantly more complicated than the analysis of the Newton method, due to the evolution of B_k. Most notably, since the updates of the BFGS curvature matrix do not require matrix inversion, its computational complexity is only $\mathcal{O}(n^2)$, compared to $\mathcal{O}(n^3)$ in Newton's method. Under suitable conditions, it can be shown that BFGS satisfies Eq. (4.16) and hence converges super-linearly (i.e. with $1 < q < 2$ in Eq. (4.14)).

Due to its approximate nature, BFGS can yield to sub-optimal solutions with respect to Hessian-based optimizations, since the latter has generally more information on the landscape curvature: second order exact analytical derivatives have been shown to give an improvement compared to the approximated BFGS version in quantum dynamics [11] and to require fewer iterations to reach convergence [17]. A visual representation of this behavior, together with a comparison with a simple Gradient Descent (GD) algorithm, is given in Fig. 4.1. A popular version of BFGS is L-BFGS, that uses a limited amount of computing memory and is thus particularly well suited for optimization problems with many variables. The Hessian approximation B_k, instead of being based on the full history of gradients, relies only on the most recent m updates, where generally $m < 10$: for a problem of dimension n this is a much smaller storage requirement than the $(n \times n)$ elements required to otherwise store the full Hessian estimate. Moreover, B_k is never explicitly formed nor stored: rather, the m are used to do implicit operations requiring matrix-vector products.

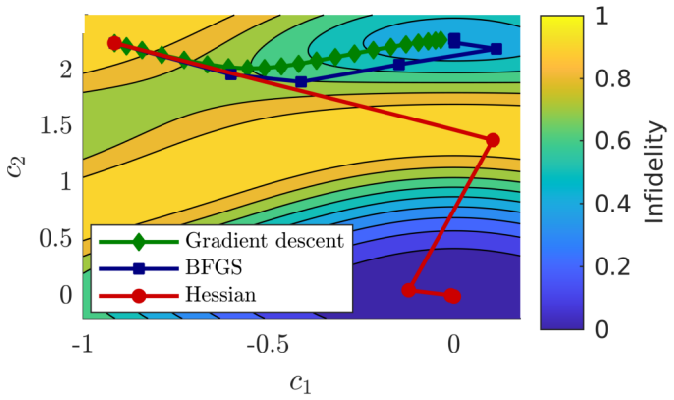

Figure 4.1: Comparison of exact and approximate second-order methods on a two dimensional optimization problem. BFGS converges towards a local solution, while the Hessian-based is able to find the global minimum. Figure from [11].

BFGS is nowadays a standard within optimal control theory [18–21] and has been shown to outperform first-order methods by providing a greater convergence acceleration [22].

4.2.2 Constrained optimization

Constrained optimization is the same problem of Eq. (4.4) in presence of bounds on the variables with respect to the minimizing/maximizing variables,

$$\min_{\boldsymbol{x}\in\mathbb{C}^n} f(\boldsymbol{x}) \text{ subject to}$$
$$g_i(\boldsymbol{x}) = c_i \quad \text{for } i = 1,\dots,n \tag{4.21}$$
$$h_j(\boldsymbol{x}) \geq d_j \text{ for } j = 1,\dots,m \tag{4.22}$$

where $g_i(\boldsymbol{x}) = c_i$ for $i = 1,\dots,n$ and $h_j(\boldsymbol{x}) \geq d_j$ for $j = 1,\dots,m$ are the equality and inequality constraints to be satisfied (in Eq. (4.21), Eq. (4.22) respectively). The paradigmatic example in mathematical optimization of a problem in subject to Eq. (4.21) are the Lagrange multipliers, in which a constrained problem is reformulated in terms of an unconstrained one, to which the usual (derivative) techniques apply.
The methods discussed for the unconstrained case are often transferable to the constrained one, and many algorithms can be adapted to fit the added condition. An example is given by L-BFGS-B, a version of the limited memory BFGS that handles bound constraints on the optimizing variables and that will be discussed in the following section.

L-BFGS-B

L-BFGS-B [23] extends L-BFGS in that it ads the conditions $l_i \leq x_i \leq u_i \; \forall i$ where l_i, u_i are lower and upper bound on each component of $\boldsymbol{x}$. The optimization is performed by distinguishing between fixed and free variables at each step using a basic gradient method; successively L-BFGS is applied exclusively to the free variables to improve accuracy: the process is then repeated.
This algorithm is particularly useful when simulating an optimal control problem, since physically realizable pulses are always confined in amplitude due to either experimental equipment limits or, more typically, to minimize unwanted phenomena like as leakage, ionization, heating, decoherence, and theoretical model breakdown.

For these mentioned reasons, for its particularly favorable convergence properties and its widespread implementation in several software packages, L-BFGS-B is the preferred optimization method in this work and will be used throughout Chapter 7.

Software implementations

Once the gradient has been calculated it can be directly fed into the optimization toolboxes implemented in numerous programming languages, most notably Python [24] and MATLAB [25] or standalone packages like Julia's `Optim.jl` [26] and `OptimKit.jl` [27].
For this work, Fortran's L-BFGS-B version [28, 29] has been found to provide a robust implementation of the algorithm: moreover, the routines are conveniently wrapped for both Python [24] and Julia [30]. Other packages offer a more extensive customization in the line search algorithms [31]: however, no tangible advantage has been observed over the built-in methods of L-BFGS-B which additionally provides low-level access to quantities of interest during run-time.

4.3 Automatic differentiation

As we saw in the previous chapter, computing derivatives is a fundamental operation in the maximization/minimization of an objective function and a common requirement for a large class of numerical methods. As algorithms gain in complexity, computing derivatives becomes a very challenging task: for many problems, we cannot calculate derivatives analytically, but only evaluate them numerically at certain points. In this section we qualitatively describe some fundamental aspects of Automatic Differentiation (AD), a method that will be explored in Section 7.2. The main ideas of this chapter are taken from [32, 33], to which we refer for an in-depth review of this

family of techniques.

Methods for the computation of derivatives in computer programs can be classified into four categories: (1) hand-coded analytical derivatives; (2) numerical differentiation using finite differences approximations; (3) symbolic differentiation using expression manipulation in computer algebra frameworks such as Mathematica; (4) AD, also referred to as algorithmic differentiation.
Manual differentiation is time consuming, error prone and not applicable to problems with implicit solutions. Numerical differentiation is simple to implement but is subject to floating point precision errors and slow, especially in high dimensions; the method requires at least n evaluations, with n being the number of partial derivatives required. Symbolic differentiation addresses the weaknesses of both the manual and numerical methods, but is particularly slow and memory intensive. Furthermore, manual and symbolic methods require a formulation of the task as closed-form expression, which rules out a large portion of problems. In contrast, AD is based on the fact that, independently of the complexity, every computer written function can be broken down to fundamental operations such as summations and multiplications that can be easily differentiated. By iteratively applying the chain rule AD is able to generate numerical derivative evaluations rather than derivative expressions: this allows to obtain derivatives at machine precision with only a small constant factor of overhead and (ideally) asymptotic efficiency.
For what has being said so far, it goes without saying that fundamental to AD is the decomposition of differentials provided by the chain rule. Usually, AD operates in two distinct modes, i.e. forward and reverse accumulation, based on the direction in which the chain rule is applied.
We refer to [32] for an extensive review of this mechanisms: given the purpose of this work we do not further indulge in technicalities.

4.4 Matrix exponentiation

The fundamental step to solve Schrödinger equation is the matrix exponentiation (as in Eq. (3.15)) which is one of the most expensive operations in solving the dynamics of a given system. We will first introduce the problems that exact diagonalization faces and than present the Padé approximation and the Krylov subspace methods for matrix exponentiation. In [34] is presented a thorough survey on matrix exponentiation algorithms, whereas we will use [35, 36] as main reference for the following sections.

4.4.1 Exact diagonalization

The starting point of an exact diagonalization algorithm is a appropriate representation of the computational basis. Basis states can be encoded very efficiently in a computer using the bit representation of integer variables and identifying for e.g. a N spin-1/2 system the eigenvalues of the $\hat{S}_z$ operator as $|\downarrow 0 \equiv 0\rangle$ and $|\uparrow\rangle \equiv 1$. In this manner we can relate any physical representation to an index, simplifying operations with the basis list.
Once a basis list $\{b_i\}_{i,\dots,N}$ is set up the matrix elements $\hat{H}_{ij} = \langle b_i|\hat{H}|b_j\rangle$ can be computed by considering the action of the constituents of the Hamiltonian on each basis element: this operation returns the Hamiltonian matrix representation. The resulting matrix can be diagonalized by standard numerical algorithms yielding the eigenbasis $\{|\mathcal{E}_i\rangle\}_{i,\dots,N}$ with corresponding eigenenergies $\{\mathcal{E}_i\}_{i,\dots,N}$. It is then straightforward to compute the time evolution of an initial state $|\psi_0\rangle$ as

$$|\psi(t)\rangle = \mathrm{e}^{-i\hat{H}t}\,|\psi_0\rangle = \sum_{i=1}^{N} \mathrm{e}^{-i\mathcal{E}_i t}\,\langle\mathcal{E}_i|\psi_0\rangle\,|\mathcal{E}_i\rangle \ , \tag{4.23}$$

which is exact for all times.
A first approach to exponentiating the Hamiltonian, would consider its eigenvector decomposition

$$\hat{H} = \mathcal{R} H_{\mathrm{diag}} \mathcal{R}^{\dagger} \ , \tag{4.24}$$

which exponentiate trivially to

$$\hat{\mathcal{U}} = \mathrm{e}^{-i\hat{H}t} = \mathcal{R}\mathrm{e}^{-i\hat{H}_{\mathrm{diag}t}}\mathcal{R}^{\dagger} \ ; \tag{4.25}$$

the exponential can then be obtained by exponentiating each entry on the main diagonal. Clearly, the maximal system size that can be treated in this way is limited by the available computational resources. The dimension of the Hilbert space to describe a system of size N is d^N, where d is the dimension of the local Hilbert space. Hence, writing a matrix representation of a Hamiltonian means storing d^{2N} numbers in memory: assuming the matrix elements are all real and stored with double precision an Hamiltonian matrix of a spin-1/2 system with $N = 14$ lattice sites already requires almost 2.2 GB of memory. Moreover, the computational cost to diagonalize the matrix grows exponentially, because it is polynomial in N. This obstacle can be mitigated by taking into account certain symmetries of the system but the size of system that are tractable with Exact Diagonalization (ED) remains unavoidably limited. Approximations are thus necessary to obtain numerically implementable solutions.

4.4.2 Padé approximation method

In mathematics a Padé approximant is the approximation of a function by a rational function of given order, i.e. the power series of the approximant agrees with that of the function it is approximating. The (p, q)-Padé approximation to e^A is defined by the relation

$$R_{pq}(A) = [D_{pq}(A)]^{-1} N_{pq}(A) \ , \tag{4.26}$$

where

$$N_{pq}(A) = \sum_{j=0}^{p} \frac{(p+q+-j)!p!}{(p+q)!j!(p-j)!} A^j \tag{4.27}$$

and

$$D_{pq}(A) = \sum_{j=0}^{q} \frac{(p+q+-j)!p!}{(p+q)!j!(p-j)!} (-A)^j \ . \tag{4.28}$$

Padé approximants are used e.g. in SciPy's `expm` method (see Section 7.4 for a comparison).

4.4.3 Krylov subspace methods

Krylov subspace methods are a family of algorithms based on the exact representation of the many-body Hilbert space that allows for a reduction both in terms of memory and computational complexity. In many application, as for the quantum time evolution, we do not really need the full representation of the matrix $\mathrm{e}^{-i\hat{H}t}$ but rather its action on some given vector [37].
The central idea of the Krylov subspace methods is to approximately project the exponential of the large intractable matrix onto a small Krylov subspace, after which the only effective exponential operation computed is with a much smaller matrix size.
Formally, we are interested in approximations to the matrix exponential operation $\mathrm{e}^A v$ of the form

$$\mathrm{e}^A v \approx p_{m-1}(A) v \ , \tag{4.29}$$

where A is a matrix of dimension N, v is a normalized vector, and p_{m-1} is a polynomial of degree $m-1$; $p_{m-1}(A)v$ is an element of the Krylov subspace defined by

$$\mathcal{K}_m = \mathrm{span}\left\{v, Av, \ldots, A_{m-1}v\right\} \ . \tag{4.30}$$

For the general non-symmetric case, we can apply the Arnoldi algorithm [38] or the non-symmetric Lanczos algorithm [39]. Both reduce to the symmetric Lanczos algorithm when the matrix A becomes symmetric.
This Arnoldi algorithm is applied to a (generally non-symmetric) matrix A and a random vector v. Consequently, $\{v_1, v_2, \ldots, v_m\}$ is built as an orthonormal basis of the Krylov subspace $\mathcal{K}_m$ and $V_m \coloneqq [v_1, v_2, \ldots, v_m]$ is

defined as an orthogonal matrix of dimensions $n \times m$. If we let $H_m \coloneqq [h_{ij}]$ be the $m \times m$ upper Hessenberg matrix[1] then

$$AV_m = V_m H_m + h_{m+1,m} v_{m+1} \mathrm{e}_m^T \, . \tag{4.31}$$

Thanks to the orthogonality of V_m, we have $H_m = V_m^T A V_m$, which is the wanted projection of A onto the Krylov subspace $\mathcal{K}_m$ with respect to the basis V_m. Consequently, the Krylov subspace method for matrix exponentiation consist in the approximation

$$\mathrm{e}^A v \approx V_m \mathrm{e}^{H_m} \mathrm{e}_1 \, . \tag{4.32}$$

Krylov subspace methods are implemented e.g. in the Python package QuSpin [40] and `KrylovKit.jl` [41] for Julia. This latter will be used in Section 7.4 to compare exact and approximate state dynamics calculations.

[1] A square $n \times n$ matrix A is said to be in upper Hessenberg form if $a_{i,j} = 0 \; \forall i, j$ with $i > j + 1$. A lower Hessenberg matrix is the transposed of an upper Hessenberg matrix.

Chapter 5

Quantum optimal control theory

Control is a key component in turning science into technology [42]. In a broad sense, the goal of control is to provide the experimentalist with external parameters to steer a given dynamical system in a determined way rather than passively observing its dynamics. Quantum Optimal Control (QOC) transfers this idea to quantum systems dynamics and thus deals with designing an optimal control field modulation that most precisely implements a desired quantum operation with minimum energy consumption and maximum robustness against hardware imperfections as well as external noise [43]. Practically, the desired quantum dynamics is realized with the help of electric, magnetic, or electromagnetic control fields.
In this chapter we will discuss the main aspects of control theory, describing its salient features: after a general introduction to the problem of QOC we will give some definition to characterize the controllability of a system and the physical time limits to the transformation one can operate. We will then present the cost function used to find the control realizing the unitary (or nonunitary) operations and introduce gradient-based optimal control with two prime examples. Finally, we will describe a method to efficiently compute exact derivatives from the perspective of a QOC problem.

5.1 Applied quantum optimal control

We consider an Hamiltonian of the form

$$\hat{H} = \hat{H}(t, \mathbf{\Omega}(t)) = \hat{H}^d(t) + \sum_k c_k(t)\hat{H}_k^c(t, \mathbf{\Omega}(t))\,, \tag{5.1}$$

$c_k(t)$ is a continuous field parameterized by the control vector Ω_k. The drift Hamiltonian $\hat{H}^d(t)$ represents parts of the system dynamics that is

uncontrollable. The control Hamiltonian $\hat{H}_k^c(t, \mathbf{\Omega}(t))$ on the other hand, depends on the control Ω_k and is used to realize the desired dynamics. To fix the ideas, for a single-particle system the drift could be the kinetic energy $\hat{H}^d(t) = \hat{T}$ and the control Hamiltonian the potential profile $\hat{H}^c = \hat{V}(x, \Omega)$. It is common practice (see [44]) to use the discretization (Section 4.1) as a map between the continuous fields of Eq. (5.1) and the control vector,

$$c_k(t) = \sum_i^{N-1} \Omega_{k,i} \Pi_i(t, \Delta t) \ , \tag{5.2}$$

$$\Pi_i(t, \Delta t) \equiv [\Theta(t - i\Delta t) - \Theta(t - (i+1)\Delta t)] \ , \tag{5.3}$$

Π and Θ being the rectangular and Heaviside function respectively [14].

5.1.1 Controllability

In this section we provide a quick overview of controllability in the quantum context: for a more in-depth description of the central concepts from linear algebra and differential geometry which are here neglected, we refer to [45]. Let $\Gamma = [\gamma_i]_{i=1}^N$ be a (finite) set of configurations for a given quantum system satisfying Eq. (5.1): the system is said to be controllable if for any pair of configurations $\gamma_1, \gamma_2 \in \Gamma$ there exists a set of time-dependent controls $\boldsymbol{u}_n$ that can drive the system from the initial configuration γ_1 to the final one γ_2 in a finite time t. As we will later see for the cost functions in Section 5.1.3, depending on the specific control problem, the notion of configuration refers to either the state of the system, the expectation value of a specific observable or the evolution operator $\hat{\mathcal{U}}(t, t_0)$. For an n-level closed system, the unitary $\hat{\mathcal{U}}(t, t_0)$ with initial conditions $\hat{\mathcal{U}}(t_0, t_0) = \mathbb{1}$ varies on the Lie group of (special) unitary matrices $U(n)$ ($SU(n)$). In particular, we can distinguish between:

— Operator-Controllability (OC). The system is operator-controllable if every desired unitary operation on the state can be performed using an appropriate control field. Operator controllability in the unitary case is referred to as complete controllability: for this special case every unitary operator $\hat{\mathcal{U}}$ is accessible from the identity operator $\mathbb{1}$ via a (common) path $\gamma(t) = \hat{\mathcal{U}}(t, t_0)$ where $\hat{\mathcal{U}}(t, t_0)$ satisfies Eq. (3.5) [46].

— Pure-State-Controllability (PSC). The system is pure-state-controllable if for every pair of initial and final states, $|\psi_0\rangle$ and $|\psi_f\rangle$ there exist a vector of controls $\boldsymbol{u}_n$ and a time $t > 0$ such that the solution of the time evolution at time t with initial condition $|\psi_0\rangle$ is $|\psi(t)\rangle \equiv |\psi_f\rangle$.

— Equivalent-State-Controllability (ESC). The system is equivalent-state-controllable if, for every pair of initial and final states, $|\psi_0\rangle$ and $|\psi_f\rangle$ there exist a vector of controls Ω_n and a phase factor ϕ such that

the solution of the time evolution with initial condition $|\psi_0\rangle$ satisfies $|\psi(t)\rangle = \mathrm{e}^{i\phi}\,|\psi_f\rangle$ at some $t > 0$.

It is immediate to see how having PSC is as good as having ESC.
With the given definitions, it can been demonstrated [47, 48] that a necessary and sufficient condition for OC is that the the Lie group generated by the system's (traceless) Hamiltonian is $U(N)$ ($SU(N)$). Nevertheless, the controllability criterion is not a guarantee of success: even if a system is controllable, finding the optimal driving pulses can become arbitrarily difficult. On the contrary, it might still be possible to identify a specific transformation after a given time even if a system is not controllable: thus, the controllability criterion can only infer on the theoretical possibility of the existence of all such transformations [49].

5.1.2 Quantum speed limit

The Quantum Speed Limit (QSL) provides fundamental bound on how fast a quantum system can evolve between the initial and the final states, a property that relates to the time-energy uncertainty relation. The notion of QSL naturally arises in time-optimal control theory, where the goal is to achieve quantum operations with both high accuracy and speed. This approach is opposed to the one of the adiabatic theorem [50], for which one leaves enough transition time and only varies dynamical parameters of the Hamiltonian slowly, such that the instantaneous eigenstates are kept well under control. However, this strategy is too slow for practical uses and does not guarantee for fast times for the initial and final state to be eigenstates of the system.
If we want to maintain coherence and the high-speed criterion of driving the quantum system to its target as fast as possible, for every non-trivial quantum control problem a lower bound on the transfer time occurs. This limit is not to be attributed to numerical or experimental imperfections nor our lack in finding a better solution. As theoretical esteem, there exist two independent relations that define this minimal time τ_{QSL}, the first one known as Mandelstam-Tamm bound [51],

$$\tau \geq \tau_{\mathrm{QSL}} = \frac{\pi\hbar}{2\Delta\mathcal{E}}\ , \tag{5.4}$$

where $\Delta\mathcal{E}$ is the energy variance $\sqrt{\langle\psi_0\,|\,(H-\mathcal{E})^2\,|\,\psi_0\rangle}$ and $\mathcal{E} = \langle\psi_0\,|\,H\,|\,\psi_0\rangle$. Alternatively, there is the Margolus-Levitin bound [52],

$$\tau \geq \tau_{\mathrm{QSL}} = \frac{\pi\hbar}{2\mathcal{E}}\ ; \tag{5.5}$$

in general, the two speed limits are independent and the minimal evolution time is therefore given by

$$\tau \geq \tau_{\mathrm{QSL}} = \max\left\{\frac{\pi\hbar}{2\Delta\mathcal{E}}, \frac{\pi\hbar}{2\mathcal{E}}\right\} = \frac{\pi\hbar}{\mathcal{E} + \Delta\mathcal{E} - |\mathcal{E} - \Delta\mathcal{E}|} \,. \tag{5.6}$$

As a final remark, a more direct approach consists in considering a cost functional resulting from the optimization process as a function of the time evolution window T: the threshold is then found for times $T \geq \tau_{\mathrm{QSL}}$ above which the objective functional reaches values arbitrarily close to the desired ones [53].

5.1.3 Cost functions

Upon temporal discretization of the propagators (see Section 4.1), the optimal control task consists in finding appropriate control vector(s) $\boldsymbol{\Omega}$ that correspond to local (and hopefully global) minima in the control landscape: the latter is in its most general form defined by the cost

$$J(\boldsymbol{u}) = J\left(\hat{\mathcal{U}}_{N_t-1} \ldots \hat{\mathcal{U}}_2 \hat{\mathcal{U}}_1\right) \,. \tag{5.7}$$

The expression of $J(\boldsymbol{u})$ depends on the particular unitary task (as e.g. gate synthesis, state transfer, or maximization of a given observable, see [54]) and whether a pure state or density matrix description is considered. In the following we will give a short overview of the different expressions depending on each specific case.

Average error

As shown in [55] the state fidelity of a unitary (or anti-unitary) map $\hat{\mathcal{U}}$ and a general linear, trace-preserving, transformation $\mathcal{M}$ acting on an initially pure state $|\psi\rangle\langle\psi|$ is given by

$$\mathcal{F}_{|\psi\rangle\langle\psi|} = \int d\psi \, \mathrm{Tr}\left[\hat{\mathcal{U}}_{\mathrm{ideal}} |\psi\rangle\langle\psi| \hat{\mathcal{U}}_{\mathrm{ideal}}^{\dagger} \mathcal{M}\left[|\psi\rangle\langle\psi|\right]\right] \,, \tag{5.8}$$

where the average is defined by integrating over all pure input states. We can in turn define the average error as

$$J_{|\psi\rangle\langle\psi|} = 1 - \mathcal{F}_{|\psi\rangle\langle\psi|} \,. \tag{5.9}$$

For single qubits the expression simplifies to

$$E_G = 1 - \frac{1}{6} \sum_{j=\pm x, \pm y, \pm z} \mathrm{Tr}\left[\hat{\mathcal{U}}_{\mathrm{ideal}} \rho_j \hat{\mathcal{U}}_{\mathrm{ideal}}^{\dagger} \mathcal{M}\left[\rho_j\right]\right] \,, \tag{5.10}$$

where x, y, z represent the 6 axial states on the Bloch sphere. From this, one can clearly see how the worst error for the qubit case is less than 6 times the average error.

Overlap fidelity

Let us first consider the average complex overlap of the target states with the forward-propagated states. If we denote with s the dimension of the Hilbert subspace (which coincides with the number of objectives), this is given by

$$f_\tau = \frac{1}{s}\sum_{k=1}^{s} w^{(k)}\tau^{(k)} \ , \quad \tau^{(k)} = \begin{cases} \langle \psi_{\text{tgt}}^{(k)} | \psi^{(k)}(T)\rangle & \text{in Hilbert space} \\ \\ \text{Tr}\left(\hat{\rho}_{\text{tgt}}^{\dagger(k)}\hat{\rho}^{(k)}(T)\right) & \text{in Liouville space} \end{cases} , \tag{5.11}$$

where the superscript (k) labels each objective (e.g. a specific state for the case of a gate transfer). The weight attributes $w^{(k)}$ are not automatically normalized, but rather assumed to have values such that the resulting f_τ lies in the unit circle of the complex plane. This means that the weights should sum to s and are usually considered $w^{(k)} \equiv 1 \ \forall k$.

We can now introduce two measures of "closeness" between quantum states, namely the state-to-state phase-insensitive fidelity $\mathcal{F}_{\text{ss}}$ and the square-modulus fidelity $\mathcal{F}_{\text{sm}}$,

$$\mathcal{F}_{\text{ss}} = \frac{1}{s}\sum_{k=1}^{s} w^{(k)}|\tau^{(k)}|^2 \ , \qquad\qquad \mathcal{F}_{\text{sm}} = |f_\tau|^2 \ , \tag{5.12}$$

both defined $\in [0,1]$. Solving an optimal control problem with $\mathcal{F}_{\text{sm}}$ as a function of merit is simpler in the sense that we are neglecting the phase information. The same ideas can be translated for the purposes of evaluating quantum gates, where a standard measure of how close the two unitary operations are to one other is given by the overlap fidelity [44],

$$\Phi_4 = \frac{1}{s^2}\left|\text{Tr}\left(\hat{\mathcal{U}}^\dagger \hat{\mathcal{U}}_{\text{get}}^\dagger(T)\right)\right|^2 \ ; \tag{5.13}$$

the squaring removes the global phase and gives the interpretation of a probability. The reformulation of the same expression with the overlaps $\tau^{(k)}$ instead of gates leads to $\mathcal{F}_{\text{sm}}$ and can be trivially obtained with a few analytical steps. The expression in Eq. (5.13) is related to the average fidelity in Eq. (5.8) through [56],

$$\mathcal{F}_{|\psi\rangle\langle\psi|} = \frac{1}{d_Q+1}(1 + d_Q\Phi_4) \ . \tag{5.14}$$

Subspace fidelity

For higher dimensional cases where the qubits are embedded in a larger Hilbert space (as e.g. in presence of leakage channels), one can instead use

[57]

$$\Phi_5 = \frac{1}{s^2}\left|\mathrm{Tr}\left(\hat{\mathcal{U}}^\dagger \mathbb{P}_Q \hat{\mathcal{U}}^\dagger_{\mathrm{get}}(T)\mathbb{P}_Q\right)\right|^2 , \tag{5.15}$$

where $\mathbb{P}_Q$ is a projector onto the computational subspace. The notation of the subscripts for Φ is the one adopted in [44], to which we refer for further discussion.

Given a fidelity $\mathcal{F}$, the actual cost function, the so called infidelity, is then simply defined as $1-\mathcal{F}$, with the optional addition of a re-scaling factor $1/2$ in front.
We do not address the more complicated case of a fidelity measure for incoherent dynamics.

5.1.4 Gradient-based quantum optimal control

In this work we focus on derivative-based local optimization methodologies characterized by update rules that rely on local gradient and Hessian calculation of the optimization objective: the search direction $\boldsymbol{p}^{(k)}$ is calculated from the current local gradient (e.g. steepest descent, conjugate gradient, quasi-Newton directions) and possibly also the Hessian (e.g. Newton direction), as seen in Section 4.2.
The Krotov and GRadient Ascent Pulse Engineering (GRAPE) algorithm constitute two fundamental gradient-based approaches to the optimal control problem and will be briefly presented in the following section.

5.1.5 The GRAPE and Krotov algorithm

The key idea of the GRAPE algorithm [44] is to Taylor-expand the yield functional $J_{\mathcal{F}}(\boldsymbol{\Omega})$ to first order in Ω,

$$J_{\mathcal{F}}(\boldsymbol{\Omega}+\Delta\boldsymbol{\Omega}) \approx J_{\mathcal{F}}(\boldsymbol{\Omega}) + \Delta\boldsymbol{\Omega}\frac{dJ_{\mathcal{F}}}{d\boldsymbol{\Omega}} ; \tag{5.16}$$

the update to the controls can be derived to be

$$\Delta\Omega = \Omega^{(i+1)}(t) - \Omega^{(i)}(t) = -\frac{2\alpha}{\hbar}\mathfrak{Im}\left\{\left\langle \chi^{(i)}(t)\left|\frac{\partial\hat{H}}{\partial\Omega^{(i)}}\right|\psi^{(i)}(t)\right\rangle\right\} , \tag{5.17}$$

where i is the iteration number and α the search length: the update $\Delta\boldsymbol{\Omega}$ is then being applied on the previous pulse for all times at once which makes it a concurrent algorithm.
The state trajectory $|\psi^{(i)}(t)\rangle$ ($|\chi^{(i)}(t)\rangle$) are both solutions to the Schrödinger equation and are obtained from a forward (backward) propagation of the system with the previous (next) pulse $\Omega^{(i)}$. The function to be maximized is the overlap squared with the target state $|\psi_{\mathrm{tgt}}\rangle$, $|\langle\psi_{\mathrm{tgt}}|\psi(T)\rangle|^2$ for a time

window T. For higher precision and faster convergence rates, second order information can also be taken into account in Eq. (5.16), which however becomes prohibitively computationally complex for most practical scenarios: instead, quasi-Newton algorithms are thus employed.

The Krotov algorithm has been developed before GRAPE: formally, there are two different algorithms that go under the name "Krotov". The first one [58] is based on a forward propagation in time, updating controls as well as the state trajectory $|\psi(t)\rangle$ simultaneously which makes it a sequential algorithm: the central equation for updating the controls reads

$$\Omega^{(i+1)}(t) = \Omega^{(i)}(t) + \alpha S(t) \mathfrak{Re} \left\{ \left\langle \chi^{(i)}(t) \left| \frac{\partial \hat{H}}{\partial \Omega^{(i)}} \right| \psi^{(i+1)}(t) \right\rangle \right\} , \qquad (5.18)$$

with the same definitions seen for the GRAPE case and $S(t) \in [0, 1]$ being the "update shape" function. In contrast, in the formulation presented in [59], the sweeping procedure is applied also for the backward propagation on the adjoint equation.
An interesting aspect of the Krotov method is that it presents the so called monotonic convergence property for a discretization time $\delta t \to 0$, which gives an exclusively positive increase in the yield functional at every iteration, provided that a proper regularization term R is added to the objective $J_{\mathcal{F}} = J_{\mathcal{F}} + R$. In practice however, due to the stringent condition of small time steps, true convergence cannot be guaranteed and we are left with a sub-optimal solution.

We close this section with a few remarks on the pathologies of gradient-based algorithms, finally proposing a novel approach to the problem.

Numerical issues

As highlighted in [7, 60] current challenges and goals of modern control design are accuracy and computational speed. In the framework of gradient-based methods first-order methods are however prone to the build-up of errors in the Hessian approximation (which build iteratively from gradients in standard quasi-Newton methods, see Section 4.2.1) that eventually lead convergence astray. An alternative circumventing such issues was presented in [61] and will be discussed in the following section.

5.2 Efficient exact derivatives

In this chapter we describe a method to efficiently compute exact derivatives with an approximate dynamics. In short, the basic idea is to derive analytically exact gradients and Hessian for an approximated dynamics using

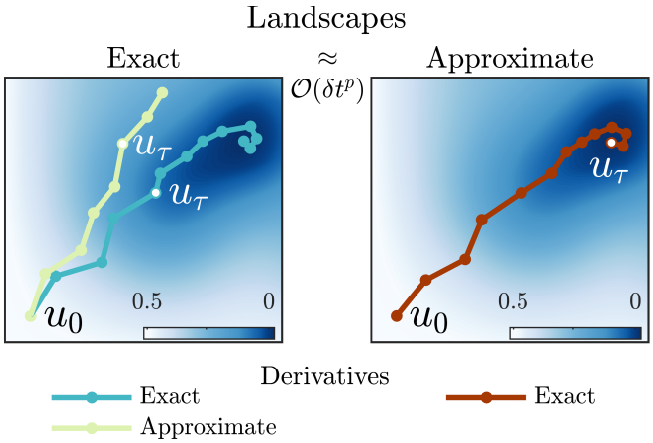

Figure 5.1: The planes are spanned by control functions $u(t)$ with associated functional values given by the color-map, dots denote optimization iterates. Following the identification of optimization landscape with a dynamical scheme, we can derive analytically exact expression for the derivatives in both the exact and approximate scenario. As here abstractly depicted, exact derivatives in an exact landscape (left panel) are expensive but ultimately lead to optimal results: despite being cheaper, approximate derivatives may not. On the other hand, exact derivatives on the approximated landscape (right panel) are significantly cheaper (as the dynamics itself) and yield faithful optimal results under appropriate conditions. Figure from [61].

Suzuki-Trotter propagation schemes: these are interpreted as (not necessarily dynamically exact) optimization landscapes, for which a visual representation is given in Fig. 5.1. If we additionally introduce the assumption of a control-diagonal Hamiltonian, the derivative calculations greatly simplify. At this point, analytically exact derivatives can be computed very efficiently, and are upper bounded only by the time it takes to propagate states which is also particularly cheap to implement due to the dynamical approximation adopted [61]. By using this two-fold assumption, in [61] is obtained a significant reduction in complexity of the analytical exact derivatives and in parallel a reduced propagation time due to the dynamical approximation, which directly address the main challenges of accuracy and speed mentioned in the previous section.
With these new tools, for the same optimization wall-clock time, a common initial guess in the optimization landscape leads to a different solution and hence manifests a performance gap (Fig. 5.1), which becomes particularly appealing as the system size enters the many-body regime, where exact diagonalization, exact propagation, and associated exact derivatives are no more numerically feasible to compute. In the following we will dive deeper into the theoretical aspects of this approach (we will refer to it as Jensen's method, for the first author in [61]), discussing its properties and limits, in particular its generalization to scenarios with more than one control.

General framing

Independently of the particular unitary task, the cost function is in the form of Eq. (5.7), which in turn leads to the same principal expression for the control derivative calculations, specifically

$$(\nabla J)_n \sim \frac{\partial}{\partial u_n}\left(\hat{\mathcal{U}}_{N_t-1}\ldots\hat{\mathcal{U}}_2\hat{\mathcal{U}}_1\right) \tag{5.19}$$

$$(H(J))_{n,m} \sim \frac{\partial^2}{\partial u_n \partial u_m}\left(\hat{\mathcal{U}}_{N_t-1}\ldots\hat{\mathcal{U}}_2\hat{\mathcal{U}}_1\right) . \tag{5.20}$$

For simplicity, we are going to restrict ourselves to the case of a pure single state transfer $|\psi_{\text{ini}}\rangle \rightarrow |\psi_{\text{tgt}}\rangle$, ideally obtaining unit fidelity given by Eq. (5.11) and here rewritten with a slight change of notation,

$$\mathcal{F} = |\langle\psi_{\text{tgt}}|\psi(T)\rangle|^2 \equiv \left|\left\langle\psi_{\text{tgt}}\right|\hat{\mathcal{U}}(T;0)\left|\psi_{\text{ini}}\right\rangle\right|^2 , \tag{5.21}$$

where $|\psi_{\text{ini}}\rangle$, $|\psi_{\text{tgt}}\rangle$ are the initial and final states respectively. Considering the state-transfer case can be done without loss of generality since the largest numerical effort is dictate by Eq. (5.19), thus any other unitary control task can be seen as a straight generalization of this simple case.
The state $|\psi(T)\rangle \equiv |\psi_{N_t}\rangle = \hat{\mathcal{U}}(T;0)|\psi_{\text{ini}}\rangle$ at final time T is produced by step-wise evolution according to $|\psi_{n+1}\rangle = \hat{\mathcal{U}}_n|\psi_n\rangle$. The associated fidelity cost is $J_{\mathcal{F}} = (1-\mathcal{F})/2$. We re-write the fidelity as $\mathcal{F} = o^\star o$, with o being the overlap (or transfer amplitude) defined as

$$o = \langle\chi_{N_t}|\psi_{N_t}\rangle = \langle\chi_{N_t}|\hat{\mathcal{U}}_{N_t-1}\ldots\hat{\mathcal{U}}_n\hat{\mathcal{U}}_1|\psi_1\rangle , \tag{5.22}$$

where $|\chi_{N_t}\rangle \equiv |\psi_{\text{tgt}}\rangle$ is an auxiliary state with $|\chi_n\rangle = \hat{\mathcal{U}}_n^\dagger|\chi_{n+1}\rangle$. This rewriting allows for the derivatives of the cost function to be expressed as

$$\begin{aligned}\frac{\partial J_{\mathcal{F}}}{\partial u_n} &= -\mathfrak{Re}\left(o^\star\frac{\partial o}{\partial u_n}\right) \\ \frac{\partial^2 J_{\mathcal{F}}}{\partial u_n \partial u_m} &= -\mathfrak{Re}\left(\left(\frac{\partial o}{\partial u_m}^\star\frac{\partial o}{\partial u_n} + o^\star\frac{\partial^2 o}{\partial u_n\partial u_m}\right)\right) ,\end{aligned} \tag{5.23}$$

while for the overlap derivatives it holds

$$\begin{aligned}\frac{\partial o}{\partial u_n} &= \left\langle\chi_{N_t}\right|\frac{\partial}{\partial u_n}\left(\hat{\mathcal{U}}_{N_t-1}\ldots\hat{\mathcal{U}}_n\hat{\mathcal{U}}_1\right)\left|\psi_1\right\rangle \\ \frac{\partial^2 o}{\partial u_n\partial u_m} &= \left\langle\chi_{N_t}\right|\frac{\partial^2}{\partial u_n\partial u_m}\left(\hat{\mathcal{U}}_{N_t-1}\ldots\hat{\mathcal{U}}_n\hat{\mathcal{U}}_1\right)\left|\psi_1\right\rangle .\end{aligned} \tag{5.24}$$

Since Eq. (5.23) depend on Eq. (5.24), the problem has now moved to analytically calculate the latter, will clearly depend on the details of the propagator

$\hat{\mathcal{U}}_n$.

For the purpose of evaluating Eq. (4.3), we consider numerical quadratures based on the left-point rectangle rule and the trapezoidal rule,

$$\int_{t_n}^{t_n+\delta t} \hat{H}(t,u(t))\,dt \approx \delta t \begin{cases} \hat{H}_n^{(1)} \equiv \frac{1}{2}\left(\hat{H}_{n+1}+\hat{H}_n\right) & \text{(trapezoidal)} \\ \hat{H}_n^{(2)} \equiv \hat{H}\left(t_n,u_n\right) & \text{(rectangle)} \end{cases}, \quad (5.25)$$

with integration errors $\mathcal{O}\left(\delta t^3\right)$ and $\mathcal{O}\left(\delta t^2\right)$, respectively if the underlying time dependence is assumed continuous and exactly known; for a piece-wise constant time dependence the rectangular approximation becomes exact. We refer to the exact propagators as

$$\hat{\mathcal{U}}_n^{\mathrm{EX}_1} = \exp\left(-i\hat{H}_n^{(1)}\delta t\right) \qquad \hat{\mathcal{U}}_n^{\mathrm{EX}_2} = \exp\left(-i\hat{H}_n^{(2)}\delta t\right), \quad (5.26)$$

and their corresponding Suzuki-Trotter (or trotterized) propagators as, respectively

$$\hat{\mathcal{U}}_n^{\mathrm{ST}_1} = \hat{\mathcal{U}}_{n+1}^{c/2}\hat{\mathcal{U}}_n^{d}\hat{\mathcal{U}}_n^{c/2} \approx \hat{\mathcal{U}}_n^{\mathrm{EX}_1}, \qquad \hat{\mathcal{U}}_n^{\mathrm{ST}_2} = \hat{\mathcal{U}}_n^{c/2}\hat{\mathcal{U}}_n^{d}\hat{\mathcal{U}}_n^{c/2} \approx \hat{\mathcal{U}}_n^{\mathrm{EX}_2}, \quad (5.27)$$

with definitions for the control and drift exponentials as

$$\hat{\mathcal{U}}_n^{c/2} \equiv \exp\left(-i\frac{\delta t}{2}\hat{H}_n^{(2)c}\right), \hat{\mathcal{U}}_n^{d} \equiv \begin{cases} \exp\left(-i\hat{H}_n^{(1)d}\delta t\right) \text{ for } \mathrm{ST}_1 \\ \exp\left(-i\hat{H}_n^{(2)d}\delta t\right) \text{ for } \mathrm{ST}_2 \end{cases}. \quad (5.28)$$

The separation of the operators is due to the Trotter product formula (see e.g. also [62]) which gives a limit where the factorization $\mathrm{e}^{ST} = \mathrm{e}^{S}\mathrm{e}^{T}$ holds in a weak sense, even when S and T are non-commutative. In more formal terms, if S and T are bounded self-adjoint operators on $\mathcal{H}$ with domains $\mathcal{D}(S)$ and $\mathcal{D}(T)$ respectively, then for every $t \in \mathbb{R}$ and for all $\xi \in \mathcal{D}(S+T) = \mathcal{D}(S) \cap \mathcal{D}(T)$,

$$\lim_{n\to\infty}\left\|\mathrm{e}^{-i\delta t(S+T)}\xi - \left(\mathrm{e}^{-i\frac{\delta t}{n}S}\mathrm{e}^{-i\frac{\delta t}{n}T}\right)\xi\right\| = 0\,. \quad (5.29)$$

This types of factorization are often referred to as Suzuki-Trotter expansions (hence the name and notation of Eq. (5.27)) and have Trotterization errors in the order $\mathcal{O}(t^p)$, where $p = 2, 3$ depending on the approximation scheme employed. We do not elaborate on the matter and refer to [63] for a more in-depth analysis.
The local $\mathcal{O}\left(\delta t^p\right)$ Trotterization errors accumulate throughout the $N_t - 1$ evolutions of Eq. (4.2), yielding a total error of

$$\left(N_t - 1\right)\delta t^p \approx (T/\delta t)\delta t^p = T\delta t^{p-1} \sim \mathcal{O}\left(\delta t^{p-1}\right). \quad (5.30)$$

It is convenient to interpret this as an approximation error with respect to the exact landscape [61],

$$J_{\mathcal{F}}^{\mathrm{ST}}(\boldsymbol{u}) = J_{\mathcal{F}}^{\mathrm{EX}}(\boldsymbol{u}) + \mathcal{O}_{J_{\mathcal{F}}^{\mathrm{ST}}}\left(\delta t^{p-1}\right) \tag{5.31}$$

where we dropped the subscripts for notational simplicity.

Analytical results

We can now come back to the problem of computing the expressions in Eq. (5.24) using the propagators in Eq. (5.27); we report in Appendix A the full derivation. We simplify the notation, identifying $\mathrm{ST}_2 \equiv \mathrm{ST}$. First, we consider the recursive commutator,

$$[X, Y]_k = [X, [X, Y]_{k-1}]\,,\ [X, Y]_0 = Y\ . \tag{5.32}$$

The key aspect of this approach is considering the derivative of the exact propagator in the form

$$\frac{\partial \hat{\mathcal{U}}_n^{\mathrm{EX}}}{\partial u_n} = \hat{\mathcal{U}}_n^{\mathrm{EX}}(-i\delta t) \sum_{k=0}^{\infty} \frac{i^k \delta t^k}{(k+1)!} \left[\hat{H}_n, \frac{\partial \hat{H}_n^c}{\partial u_n}\right]_k , \tag{5.33}$$

where $\partial \hat{H}_n^c / \partial u_n = \partial \hat{H}_n / \partial u_n$ is the control derivative Hamiltonian. Since the Hamiltonian and its control derivative generally do not commute, $\left[\hat{H}_n, \frac{\partial \hat{H}_n^c}{\partial u_n}\right] \neq 0$, the recursive commutator is not guaranteed to terminate. This implies that in the most general case the summation has to continue until machine precision is reached or up to a desired accuracy. A similar expression to Eq. (5.33) can be found for the trotterized propagators in Eq. (5.27), with the substitution $\hat{H}_n \to \hat{H}_n^c$ in the first argument of the recursive commutator. If we now introduce the additional hypothesis of a diagonal control Hamiltonian, Eq. (5.33) simplifies dramatically as two diagonal matrices always commute, $\left[\hat{H}_n^c, \frac{\partial \hat{H}_n^c}{\partial u_n}\right]_k = \frac{\partial \hat{H}_n^c}{\partial u_n} \cdot \delta_{0,k}$:

$$\frac{\partial}{\partial u_n}\left(\hat{\mathcal{U}}_n^{\mathrm{ST}_1} \hat{\mathcal{U}}_{n-1}^{\mathrm{ST}_1}\right) = (-i\delta t) \cdot \hat{\mathcal{U}}_n^{\mathrm{ST}_1} \frac{\partial \hat{H}_n^c}{\partial u_n} \hat{\mathcal{U}}_{n-1}^{\mathrm{ST}_1}, \tag{5.34}$$

$$\frac{\partial \hat{\mathcal{U}}_n^{\mathrm{ST}_2}}{\partial u_n} = -\frac{i\delta t}{2}\left(\frac{\partial \hat{H}_n^c}{\partial u_n} \hat{\mathcal{U}}_n^{\mathrm{ST}_2} + \hat{\mathcal{U}}_n^{\mathrm{ST}_2} \frac{\partial \hat{H}_n^c}{\partial u_n}\right) . \tag{5.35}$$

We thus obtain the wanted gradient elements,

$$\frac{\partial J_{\mathcal{F}}^{\mathrm{EX}}}{\partial u_n} = \mathfrak{Re}\left(io^\star \left\langle \chi_n \left| \frac{\partial \hat{H}_n^c}{\partial u_n} \right| \psi_n \right\rangle \right) \delta t + \mathcal{O}_{\nabla J_{\mathcal{F}}^{\mathrm{EX}}}\left(\delta t^2\right) \tag{5.36}$$

$$\frac{\partial J_{\mathcal{F}}^{\mathrm{ST}_1}}{\partial u_n} = \mathfrak{Re}\left(io^\star \left\langle \chi_n \left| \frac{\partial \hat{H}_n^c}{\partial u_n} \right| \psi_n \right\rangle \right) \delta t \tag{5.37}$$

$$\frac{\partial J_{\mathcal{F}}^{\mathrm{ST}_2}}{\partial u_n} = \mathfrak{Re}\left(\frac{io^\star}{2} \left\{ \left\langle \chi_{n+1} \left| \frac{\partial \hat{H}_n^c}{\partial u_n} \right| \psi_{n+1} \right\rangle + \left\langle \chi_n \left| \frac{\partial \hat{H}_n^c}{\partial u_n} \right| \psi_n \right\rangle \right\} \right) \delta t \ , \tag{5.38}$$

where the remainder term for the exact propagator inherits the infinite series of Eq. (5.33)

$$\mathcal{O}_{\nabla J_{\mathcal{F}}^{\mathrm{EX}}}\left(\delta t^2\right) \propto \left\langle \chi_n \left| \left(\sum_{k=1}^{\infty} \frac{i^k \delta t^k}{(k+1)!} \left[\hat{H}_n, \frac{\partial \hat{H}_n^c}{\partial u_n} \right]_k \right) \right| \psi_n \right\rangle \delta t \ . \tag{5.39}$$

An interesting thing to notice is that the gradient for the trapezoidal rule corresponds to the first-order approximation $k = 0$ of the exact expression. Looking back at Eq. (5.31), we can distill a central argument of Jensen's method: the derivatives should match the dynamics. Exact derivatives for an approximate landscape offer an increased accuracy at a lower computational cost.

Generalization to multiple controls

We now give the extension of the discussion beyond the case of a single, generically parameterized control Hamiltonian $\hat{H}^c(t, \boldsymbol{u}(t))$. We start from defining a set of K controls and their corresponding control Hamiltonians by

$$C(t) = \left\{ \hat{H}_{(k)}^c \left(t, u_{(k)}(t) \right) \right\}_{k=1}^{K} = \cup_{q=1}^{Q} C_q(t) \ , \tag{5.40}$$

sorted into a number Q of sets C_q of mutually commuting elements with $K = \sum_q |C_q(t)|$. We denote with $\hat{\mathcal{R}}_q(t)$ the unitary basis change operator that simultaneously diagonalizes the elements of $C_q(t)$ from a chosen reference basis.

The core idea here is that Jensen's method depends on the computational feasibility of maintaining the diagonality criteria for the control Hamiltonians. For this to be an effective approach for $Q > 1$, performing the basis change $\hat{\mathcal{R}}_{n,q}$ must be significantly cheaper than the original dense exponentiation. As stated in [61], Sec. V, "The one exceptional instance where this condition is not met occurs when $\hat{\mathcal{R}}_{n,q}$ depends on the control value and, simultaneously, no closed analytical solution to the transformations are known. This implies that $\hat{\mathcal{R}}_{n,q}$ must be obtained anew in each iteration

by numerical diagonalization, which is as expensive as dense exponentiation. Otherwise, $\hat{\mathcal{R}}_{n,q}$ and products involving these need only be calculated maximally once and can be stored on the disk and be loaded into memory at run-time."
If we further impose for the controls to be bilinear, i.e. simply a scaling factor to the matrix diagonalization, $\hat{H}^c_{(k)}(t, \boldsymbol{u}(t)) \to \boldsymbol{u}_{(k)}(t)\hat{H}^c_{(k)}(t)$, this exceptional case can be directly ruled out. In particular, this is valid (but not exclusively) for a transmon Hamiltonian, which constitutes the one physical case considered in this work.

Further considerations to address these specificities will be discussed in Chapter 7, where we will also present some progresses towards an increased generalization of Jensen's method.

Chapter 6

Tensor networks

The exponential growth of the Hilbert space dimension poses a fundamental limit to the simulation of many-body quantum systems: the exact treatment based on a full uncompressed representation of the wave function becomes prohibitively expensive already at small system sizes. This so-called curse of dimensionality is in reality (at least in part) misleading: what bounds the simulation capabilities of classical resources is not so much the size of the system as the amount of entanglement.
In this regard, Tensor Network (TN) ansätze have proven to be an incredibly successful technique, with far reaching applications in condensed matter, statistical models and machine learning [64]. TNs allow to target the (small) "corner" of the full Hilbert space occupied by the physically relevant states, usually characterized by low entanglement, with sub-exponential resources [65]. In such approaches, the wave function is encoded as a tensor contraction of a network of individual tensors: the power of such representations is owed to the fact that the size of the corner is governed by favorable so-called area scaling laws [66] for the entanglement entropy [67].
In the numerical regime, TNs provide variational classes of states which can be efficiently described, most notably Matrix Product States (MPS), projected entangled pair states (PEPS), and multiscale entanglement renormalisation ansatz (MERA).
In the following we will give a brief overview of the formalism of TNs and subsequently focus on the class of MPS, presenting some of its key properties, and introduce a variant of the Time Evolution Block-Decimation (TEBD), a standard tool in numerical many-body physics. Images in this chapter are taken from the open source project in [68].

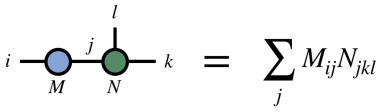

Figure 6.1: Example of TNN for a tensor contraction in the form $\sum_j M_{ij} N_{jkl}$.

6.1 Tensor diagram notation and tensor operations

Tensors are the generalization of vectors in any dimension. In this context, an N-rank tensor is an object with N indices in the form $T_{i_1,i_2,\ldots,i_N}$, where i_j labels each index.
Tensor Network Notation (TNN) is a graphical notation used to represent tensors and tensor operations and first introduced by R. Penrose in the early 1970s [69]. The widespread success of this representation lies in its graphical and intuitive nature that does not sacrifice rigorousness: many general properties of the quantum states can be identified directly from the structure of the network needed to describe them.
TNN can be seen as a generalization of Einstein summation notation: tensors are notated by shapes, tensor indices are notated by lines and a contraction (or summation) over indices is implied by connecting two index lines (see Fig. 6.2). Representation conventions than varies from author and context: for example, shapes or shadings can be used to mark properties of tensors and arrows on lines to distinguish contravariant and covariant indices. An example is given in Fig. 6.1.
TNN has various benefits, the main feature of which is the ability to represent various operations, such as a trace, tensor product, or tensor contraction in a straightforward manner. For instance, the tensor product between two tensors $\boldsymbol{A}$ and $\boldsymbol{B}$ is the element-wise product of all the components,

$$[\boldsymbol{A} \otimes \boldsymbol{B}]_{i_1,\ldots,i_{N_A},j_1,\ldots,j_{N_B}} \coloneqq A_{i_1,\ldots,i_{N_A}} \cdot B_{j_1,\ldots,j_{N_B}} \ , \tag{6.1}$$

with $\mathrm{rank}(\boldsymbol{A}) = N_A$ and $\mathrm{rank}(\boldsymbol{B}) = N_B$ and is obtained in diagrammatic notation by placing 2 tensors next to one another (see Fig. 6.3).

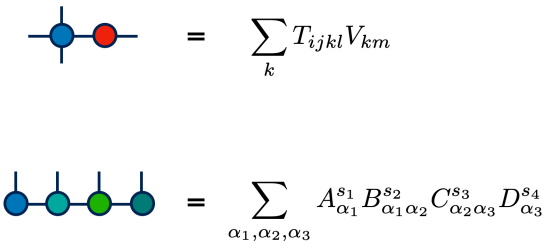

Figure 6.2: Example of tensor contraction, which can be seen as a tensor product followed by a summation over the connected indexes

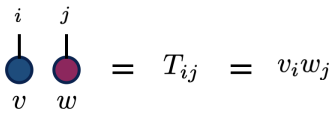

Figure 6.3: Outer product between 2 tensors in diagrammatic form.

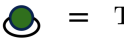

Figure 6.4: Trace of a tensor in diagrammatic form.

Next, from a tensor $\boldsymbol{A}$ having two indexes m, n with identical dimension ($d_m = d_n$), the partial trace over these indexes is the joint summation

$$\mathrm{Tr}_{m,n}(\boldsymbol{A}) \sum_{\alpha=1}^{N} A_{i_1,\ldots i_{n-1},\alpha,i_{n+1},\ldots,i_{m-1}\alpha i_{m+1},\ldots i_N} \ , \tag{6.2}$$

which is simply obtained graphically by connecting two index lines of the same tensor (see Fig. 6.4).

We can now give a single definition of a tensor network: a tensor network is a collection of tensors connected via contracted links (indices). Using a TN as many-body wave function ansatz for a system with local Hilbert space dimension p and N components, the quantum many body state can be written as

$$|\Psi\rangle = \sum_{i_1,\ldots,i_N=1}^{p} \boldsymbol{T}^{i_1\ldots i_N} |i_1\rangle \otimes \cdots \otimes |i_N\rangle \ , \tag{6.3}$$

with $\{|i\rangle\}_{i=1}^{p}$ a canonical basis of the i-th subsystem. The complex wave function coefficients determine the state uniquely: the TN expresses the many-body state as the contraction of a set of smaller rank tensors $\left\{T^{(q)}\right\}_{q=1,\ldots Q}$ over virtual (contracted) indices $\{\alpha_i\}_{i=1,\ldots L}$, namely

$$\boldsymbol{T}^{i_1\ldots i_N} = \sum_{\alpha_1,\ldots,\alpha_N} \boldsymbol{T}^{(1)}_{\{i\}_1\{\alpha\}_1} \boldsymbol{T}^{(2)}_{\{i\}_2\{\alpha\}_2} \ldots \boldsymbol{T}^{(Q)}_{\{i\}_Q\{\alpha\}_Q} \ , \tag{6.4}$$

with Q and L the number of nodes and links respectively.

With this new formalism at hand, there are several quantities of interest obtainable from a TN representations: in this discussion we will focus only on the state overlap and the Von Neumann entanglement entropy.
For any two states $|\Psi\rangle$ and $|\Psi'\rangle$, their overlap can be computed with the same numerical cost as for the state norm, i.e. in polynomial computational time (a property which is referred to as efficient contractibility). This is always the case for loop-free TN (i.e. without closed cycles in the network

geometry) as MPS to which we will limit our discussion.
Given then a loop-free TN, we can consider its bipartition A and B, which will in turn correspond to the bipartition of the system: by contracting the two sub-networks we can compute the Von Neumann entropy (or bipartition entanglement, entanglement entropy) between the two.
Formally, for a bipartition of the TN $|\Psi\rangle$ into partitions A and B, we first have to introduce the reduced density matrix with respect to system A as

$$\rho_A = \mathrm{Tr}_B \, |\Psi\rangle \langle\Psi| = \sum_\alpha p_\alpha \, |\psi_\alpha^A\rangle \langle\psi_\alpha^A| \ , \tag{6.5}$$

with p_α being the eigenvalues of ρ_α. The entropy S_{VN} of the sub-network A is then defined as

$$S_{\mathrm{VN}}(A) = -\,\mathrm{Tr}\, \rho_A \log \rho_A = -\sum_\alpha p_\alpha \log p_\alpha \ , \tag{6.6}$$

and is a measure of the amount of entanglement between the partitions A and B. Naturally, the entanglement entropy can be computed for all the possible bipartition of the network, that is for every link connecting two sites.

We will now introduce MPS, a particular class of TNs that will be used to study a many-body transmon chain in Chapter 7.

6.2 Matrix product state and matrix product operator

The MPS (also referred to as Tensor Train (TT)) TN is a factorization of a tensor with N indices into a chain-like product of three-index tensors. An MPS of a tensor $\boldsymbol{T}$ can be expressed in traditional notation as

$$\boldsymbol{T}^{s_1 \dots s_N} = \sum_{\{\alpha\}} \boldsymbol{A}^{s_1}_{\alpha_1} \boldsymbol{A}^{s_2}_{\alpha_1 \alpha_2} \dots \boldsymbol{A}^{s_N}_{\alpha_{N-1}} \ , \tag{6.7}$$

where the indices α are summed over; this is equivalent in diagrammatic notation to Fig. 6.5.

The dimension of the indices α (which can vary from bond to bond) is referred to as bond dimension m (or TT-rank, virtual dimension) and is a fundamental parameter of an MPS, controlling its expressivity. Via its bond dimension, an MPS can interpolate between a fully separate state and its correct representation: if we consider a tensor $\boldsymbol{T}^{s_1,\dots,s_N}$ having N indices

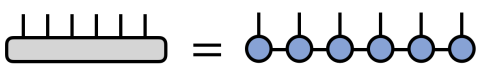

Figure 6.5: Diagrammatic representation of an MPS for 6 sites.

all of dimension d, then this tensor can always be represented exactly as an MPS with bond dimension $m = d^{N/2}$. In most applications the MPS form is used as an approximation and m is either fixed at a moderate size, or determined adaptively.
If a generic tensor necessitates of d^N parameters (exponential in N) to be completely determined, representing the same tensor via MPS requires Ndm^2 (linear in N) parameters, a number that can be reduced even further by imposing or exploiting certain constraints on the factor tensors. Typically, a maximal discarded weight is introduced as control parameter, based on which the bond dimension is adjusted dynamically such that it is minimal with the constraint that the allowed discarded weight is not exceeded.

Leaving for a moment formalisms aside, if one thinks of an MPS as parameterizing a large vector in a high-dimensional space, then an Matrix Product Operator (MPO) is the generalization to the case of a matrix acting in the same space. More rigorously, in analogy to an MPS, an MPO is a factorization of a tensor with N covariant and N contravariant indices into a contracted product of smaller tensors, each with one of the original contravariant and covariant indices and bond indices connecting to the neighboring tensors. This can be expressed in traditional notation as

$$\boldsymbol{M}^{s_1\ldots s_N}_{s'_1\ldots s'_N} = \sum_{\{\alpha\}} \boldsymbol{A}^{s_1\alpha_1}_{s'_1} \boldsymbol{A}^{s_2\alpha_2}_{\alpha_1 s'_2} \ldots \boldsymbol{A}^{s_N}_{\alpha_{N-1} s'_N} , \tag{6.8}$$

which is equivalent in diagrammatic notation to Fig. 6.6.
One particularly useful application of MPOs is representing sums of local terms in a compressed manner, which becomes especially convenient for algorithms involving MPSs [70].

6.3 Time-evolution block-decimation

Time evolution is the fundamental operation for quantum optimal control. In this section we present a variant used in [67] and inspired from [71] which is be more tailored to the structure of our problem, thus allowing to speed up computations. For a complete derivation of the traditional TEBD algorithm we refer to [72].

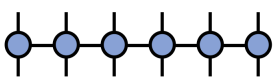

Figure 6.6: Diagrammatic representation of an MPO for 6 sites.

Time evolution block decimation for trotterized propagators

The TEBD (or Trotter decomposition) algorithm is based on a Trotter-Suzuki [62] decomposition and successive approximation of the time-evolution operator $\hat{\mathcal{U}}^{\text{exact}}(\delta t)$. We report here the main calculation as shown in [67] to derive a TEBD scheme for the trotterized propagators in Eq. (5.27) and the Hamiltonian in the form (see Section 7.1.2)

$$\hat{H}_n = \sum_{j=1}^{N} \left[\Delta_j \hat{b}_j^\dagger \hat{b}_j + \frac{1}{2}\delta_j \hat{b}_j^\dagger \hat{b}_j \left(\hat{b}_j^\dagger \hat{b}_j - 1 \right) + \sum_{\langle i,j \rangle} J_{ij} \left(\hat{b}_j^\dagger \hat{b}_i + \hat{b}_i \hat{b}_j^\dagger \right) \right] \tag{6.9}$$

$$+ \sum_{j=1}^{N} \Omega_n^{(j)} \left(\hat{b}_j^\dagger + \hat{b}_j \right) = \hat{H}^d + \hat{H}_n^c \, . \tag{6.10}$$

We denote with $\hat{h}^d_{[j,j+1]}$ and $\hat{h}^c_{[j]}$ the arguments of the sums in Eq. (6.9) and Eq. (6.10) respectively. From now on, we focus on the trapezoidal quadrature (ST_1 in Eq. (5.27)), identifying $\text{ST}_1 \equiv \text{ST}$. Each term in the diagonal control Hamiltonian commutes and we may thus write exactly

$$\hat{\mathcal{U}}_n^{c/2} = \exp\left(-i \left(\sum_{j=1}^{N} \left[u_{n,[j]} (\hat{b}_j^\dagger + \hat{b}_j) \right] \right) \frac{\delta t}{2} \right) = \prod_{j}^{N} \hat{\mathcal{U}}^c_{n,[j]} \, , \tag{6.11}$$

where $\hat{\mathcal{U}}^c_{n,[j]} = \exp\left(-i \hat{h}^c_{[j]} \delta t/2 \right)$. For the drift Hamiltonian we can obtain a first-order Suzuki-Trotter expansion by applying the same technique as in standard TEBD for nearest-neighbor Hamiltonians,

$$\begin{aligned} \hat{\mathcal{U}}^d &\overset{\text{(a)}}{=} \mathrm{e}^{-i\left(\hat{H}^d_{\text{even}} + \hat{H}^d_{\text{odd}}\right)\delta t} \approx \exp\left(-i \hat{H}^d_{\text{even}} \delta t \right) \exp\left(-i \hat{H}^d_{\text{odd}} \delta t \right) \\ &= \left(\prod_{j \text{ even}}^{N-1} \hat{\mathcal{U}}^d_{[j,j+1]} \right) \left(\prod_{j \text{ odd}}^{N-1} \hat{\mathcal{U}}^d_{[j,j+1]} \right) \, , \end{aligned} \tag{6.12}$$

where $\hat{\mathcal{U}}^D_{[j,j+1]} = \exp\left(-i \hat{h}^d_{[j,j+1]} \delta t/2 \right)$ and in (a) we grouped even and odd terms,

$$\hat{H}^d = \hat{H}^d_{\text{even}} + \hat{H}^d_{\text{odd}} = \sum_{j \text{ even}}^{N} \hat{h}^d_{[j,j+1]} + \sum_{j \text{ odd}}^{N} \hat{h}^d_{[j,j+1]} \, . \tag{6.13}$$

Since $\left[\hat{H}^d_{\text{even}}, \hat{H}^d_{\text{odd}} \right] \neq 0$ the approximation introduces a $\mathcal{O}(\delta t^2)$ error: each term has then total internal-commutativity, which allows the subsequent exact product form in Eq. (6.12).
If now combine the above expressions and move each individual even (odd) $\hat{h}^d_{j,j+1}$ to the left (right) until they meet a non-commutative operator, we

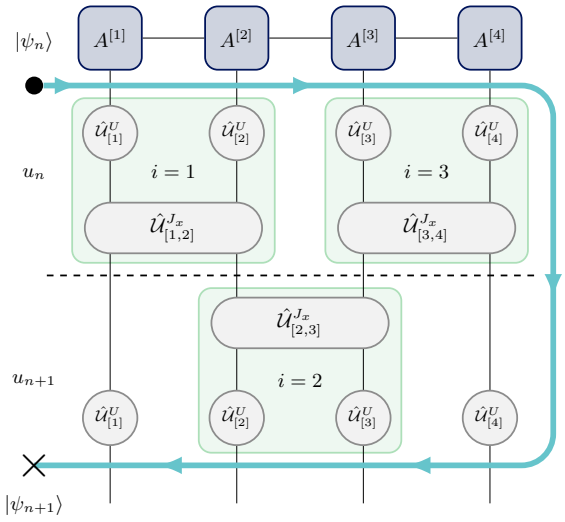

Figure 6.7: TN diagram showing the full calculation $|\psi_{n+1}\rangle = \hat{\mathcal{U}}_n^{\mathrm{ST}} |\psi_n\rangle = \hat{\mathcal{U}}_{n+1}^c \hat{\mathcal{U}}_n^d \hat{\mathcal{U}}_n^c |\psi_n\rangle$ for $N = 4$. The notation slightly differs from the one used in this chapter since it refers to the problem in [67].

obtain (N even)

$$\begin{aligned}
\hat{\mathcal{U}}_n^{\mathrm{ST}} &\approx \prod_j^N \hat{\mathcal{U}}_{n+1,[j]}^c \prod_{j\ \text{even}}^{N-1} \hat{\mathcal{U}}_{[j,j+1]}^d \prod_{j\ \text{odd}}^{N-1} \hat{\mathcal{U}}_{[j,j+1]}^d \prod_j^N \hat{\mathcal{U}}_{n,[j]}^c = \\
&= \hat{\mathcal{U}}_{n+1,[1]}^c \left(\prod_{N-1}^{j\ \text{even}} \hat{\mathcal{U}}_{n+1,[j]}^c \hat{\mathcal{U}}_{n+1,[j+1]}^c \hat{\mathcal{U}}_{[j,j+1]}^d \right) \times \\
&\times \hat{\mathcal{U}}_{n+1,[N]}^c \left(\prod_{j\ \text{odd}}^{N-1} \hat{\mathcal{U}}_{[j,j+1]}^d \hat{\mathcal{U}}_{n,[j+1]}^c \hat{\mathcal{U}}_{n,[j]}^c \right) \equiv \\
&\equiv \hat{\mathcal{U}}_{n+1,[1]}^c \left(\prod_{N-1}^{j\ \text{even}} \hat{\mathcal{U}}_{n+1,[j,j+1]}^{ccd} \right) \times \qquad (6.14) \\
&\times \hat{\mathcal{U}}_{n+1,[N]}^c \left(\prod_{j\ \text{odd}}^{N-1} \hat{\mathcal{U}}_{n,[j,j+1]}^{dcc} \right) \qquad (6.15) \\
&\equiv \hat{\mathcal{U}}_{n+1,[1]}^c \hat{\mathcal{U}}_{\mathrm{backsweep}} \hat{\mathcal{U}}_{n+1,[N]}^c \hat{\mathcal{U}}_{\mathrm{forwardsweep}} \,. \qquad (6.16)
\end{aligned}$$

For odd N it is sufficient to substitute in the last expression $\hat{\mathcal{U}}_{n+1,[N]}^c \to \hat{\mathcal{U}}_{n,[N]}^c$.
Since the grouping of product triples $\hat{\mathcal{U}}_{n,[j,j+1]}^{dcc}$ act only on nearest-neighbor pairs of indices $[j, j+1]$, the overhead in the tensor network contraction

$\hat{\mathcal{U}}_n^{\mathrm{ST}} |\psi\rangle$ is reduced. The central site (gauge[1]) of the matrix product state is advanced after each contraction: the product of triples is applied over odd j in a "forward sweep" (Eq. (6.15)) and over even j in a "backward sweep" (Eq. (6.14)), as shown in Fig. 6.7.

[1]Gauge transformations (a term borrowed from field theories) are a class of linear transformations which leave the physical quantum many-body state (and thus all the physical quantities) unchanged. In many architectures (most notably in MPS) adapting the gauge during run-time is a central operation to achieve speed-up and an increased precision. [73]

Chapter 7

Simulations and results

In this chapter we report the analytical results and simulations of this work. The discussion can be seen as tripartite: first we apply Jensen's method in different scenarios, in particular to the case not dealt with in the literature of a gate transfer, focusing on some relevant points of discussion. We will then enter the many body regime, tackling the same problem on a longer chain of qubits via MPS: in this context, the use of automatic differentiation will be explored, with the gained advantage of a greater generalisability. However, no relevant result will be (yet) obtained due to the long compilation times that make it unsuitable for optimisation routines.
Finally, we will tackle the case of the time evolution using a Krylov-Lanczos propagation scheme. We verify the quality of the solutions found via Jensen's method with the exact (physical) dynamics, showing that an optimal solution in the Trotterized landscape ceases to be such for an exact landscape, thus highlighting some fundamental weaknesses of this approach.

Most of the results of this chapter have been generated on a 2017 laptop with 16 GB RAM using two 6-core 2.20 GHz Intel Core i7 processors. For the most computational demanding tasks we used the resources made available by the PGI-8 cluster at Forschungszentrum Jülich, using four 16-core 2.60 GHz Intel Xeon Gold processors.

7.1 Approximate dynamics

In this first section we apply the methodologies discussed in Section 5.2 to the case of both a two-level and a transmon system, similarly to what has been shown in [61]. The numerical simulations are written in Python and we make extensive use of the NumPy [74] and SciPy [24] libraries.

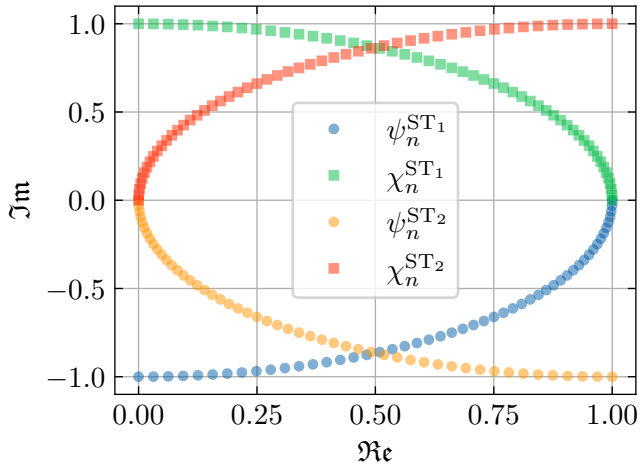

Figure 7.1: Real and imaginary part for the (back-) evolved wave functions for the known solution to the Landau-Zener (LZ) transition problem. The states for the rectangular rule ST_2 are mirrored around the $x = 1/2$ axis for better visibility. By reading the plot from right to left (e.g. following the blue line), one can see that the target state $(0, 1)^\intercal$ is reached from the initial state $(1, 0)^\intercal$ with an additional phase of $e^{i\pi}$ which is irrelevant to the optimization cost function.

7.1.1 Two-level system

In this section we study the dynamics of one of the simplest non-trivial quantum problem in which two states are optically coupled by a LZ type Hamiltonian of the form [75]

$$\hat{H} = -\Omega(t)\sigma_z + \omega(t)\sigma_x \,, \tag{7.1}$$

where Ω is the detuning and ω the Rabi frequency. From the perspective of an optimal control problem, Ω is the control function that we shall optimize: furthermore, we assume $\omega = 1/2$ time-independent.
We consider the state transfer

$$|\psi_{\mathrm{ini}}\rangle \equiv |\uparrow\rangle \rightarrow |\psi_{\mathrm{tgt}}\rangle \equiv |\downarrow\rangle \ . \tag{7.2}$$

The problem is of interest also because it constitutes a first basic step for the control of complex many-body systems, whose evolution is in many cases a cascade of LZ transitions [53].
The advantage of such a model is that for a time $T^{\mathcal{F}}_{\mathrm{min}} = \pi$ it has well-understood solutions of a single, analytical π-pulse $\Omega_n = 0$. By formulating the problem in the basis $\{|\uparrow\rangle, |\downarrow\rangle\}$ the problem is naturally set in the control-diagonal basis. We set $T = T^{\mathcal{F}}_{\mathrm{min}}$ in units of π and discretize the problem in 50 time steps. We first test the time evolution on the know solution: in Fig. 7.1 we show on the two axis the real and imaginary part of the forward- and back-evolved wave function which, by virtue the particular choice of

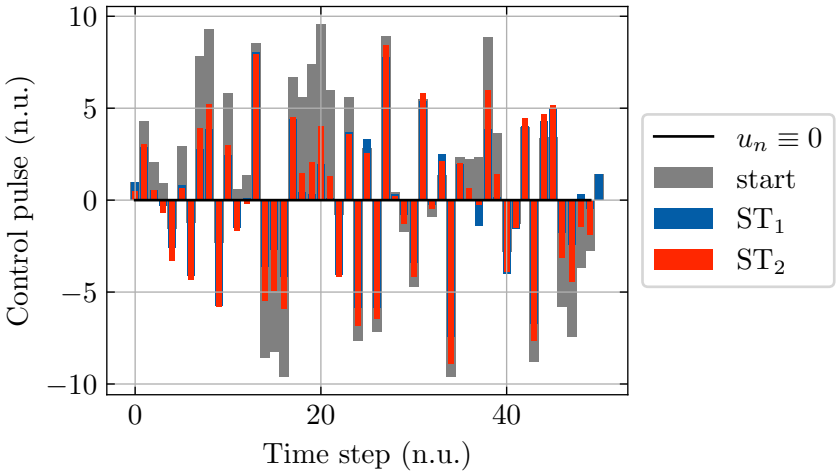

Figure 7.2: Example of a piece-wise constant control pulse for the Landau-Zener transition for both the trapezoidal and rectangular rule of integration. Axes are in numerical (simulation) units, $T = T^{\mathcal{F}}_{\text{min}}$.

initial conditions, coincide with the first and second component of the wave vector respectively.
As an additional sanity check, one can notice that regardless of the problem addressed, due to the properties of the evolved $|\psi_n\rangle$ and auxiliary state $|\chi_n\rangle$ it holds for the overlap $\langle\chi_j|\psi_j\rangle = c \in \mathbb{R}\ \forall j \in [1, \dots, N_t]$

$$\langle\chi_j|\psi_j\rangle = \langle\chi_{N_t}|\hat{\mathrm{U}}_{N_t-1}\hat{\mathrm{U}}_{N_t-2}\dots\hat{\mathrm{U}}_j\hat{\mathrm{U}}_{j-1}\hat{\mathrm{U}}_2\hat{\mathrm{U}}_1|\psi_1\rangle = \langle\chi_1|\psi_1\rangle\ , \tag{7.3}$$

which is constant. This fact will be often used as a test, considering the modulus squared of the overlap to avoid complex values.

In the trapezoidal setting, the first and last element of the gradient are zero; in fact, recalling Eq. (5.37),

$$\begin{aligned}
\frac{\partial\mathcal{F}}{\partial\Omega_j} &= \delta_t(2-\delta_{j,0}-\delta_{j,N_t})\cdot\mathfrak{Im}\left[\langle\psi_j\,|\,\chi_j\rangle\left\langle\chi_j\left|\frac{\partial H_c(t_j)}{\partial\Omega_j}\right|\psi_j\right\rangle\right] \\
&= \delta_t(2-\delta_{j,0}-\delta_{j,N_t})\cdot\mathfrak{Im}\left[\mathrm{Tr}\left(|\chi_j\rangle\left\langle\chi_j\left|\frac{\partial H_c(t_j)}{\partial\Omega_j}\right|\psi_j\right\rangle\langle\psi_j|\right)\right] \\
&= \delta_t(2-\delta_{j,0}-\delta_{j,N_t})\cdot\mathfrak{Im}\left[\mathrm{Tr}\left(\hat{\rho}_j^{(\chi)}\frac{\partial H_c(t_j)}{\partial\Omega_j}\hat{\rho}_j^{(\psi)}\right)\right] \\
&\overset{(a)}{=} \frac{\delta}{4}(2-\delta_{j,0}-\delta_{j,N_t})\sin(\theta_j^{(\chi)})\sin(\theta_j^{(\psi)})\sin(\varphi_j^{(\chi)}-\varphi_j^{(\psi)})\ ,
\end{aligned} \tag{7.4}$$

where in (a) we used

$$\frac{\partial H_c(t_j)}{\partial\Omega_j} = \frac{1}{2}\hat{\sigma}_z, \qquad\qquad \hat{\rho} = \frac{1}{2}(\mathbb{1} + \vec{r}\cdot\vec{\hat{\sigma}})\ . \tag{7.5}$$

Thus, whenever one of two points is at the pole of the Bloch sphere, $\frac{\partial\mathcal{F}}{\partial\Omega_j} = 0$. This is exactly what happens for $j = 0$, $j = N_t$ with the initial and final

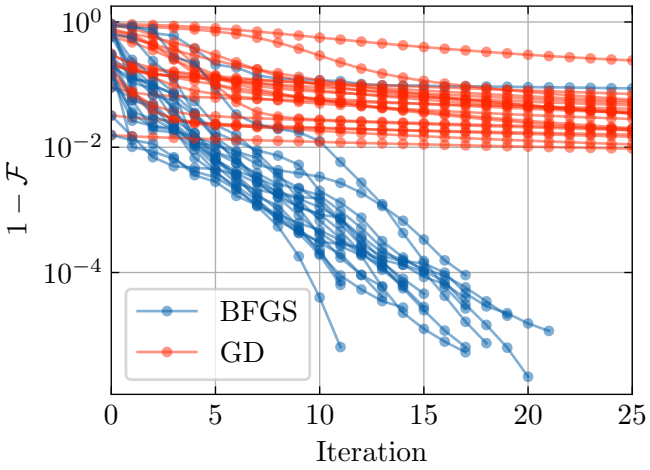

Figure 7.3: Comparison for 20 different seeds for GD and L-BFGS-B (GD and BFGS respectively in the legend box) using the trapezoidal rule.

conditions in Eq. (7.2).
This example shows that the gradient can indicate 0 in a lot of cases where the overlap is not maximal. Indeed, the fidelity in terms of the same parametrization reads:

$$\mathcal{F} = \frac{1}{2}\left(1 + \cos(\theta^{(\chi)})\cos(\theta^{(\psi)}) + \sin(\theta^{(\chi)})\sin(\theta^{(\psi)})\cos(\varphi^{(\chi)} - \varphi^{(\psi)})\right) . \tag{7.6}$$

Both the results for the overlap and gradient are numerically tested and found true up to machine precision.

We now come to the proper optimal control search. We generate piece-wise constant controls in the interval $[-10; 10]$ (as previously done in [61]) by uniform sampling: as a side note, due to the expression of the gradient in Eq. (5.38) (specifically, due to the $n+1$ subscript in the first braket), the rectangular rule has Ω_n with $n = 1, \dots, N_t - 1$ instead of the more natural $n = 1, \dots, N_t$ that holds for the trapezoidal rule.
We optimize each initial condition via both a simple GD routine and using the L-BFGS-B search direction implemented in SciPy [24] for both trapezoidal and rectangular rule (Fig. 7.3 and Fig. 7.4 respectively: lower is better). We find that while GD gets easily stuck after < 5 iterations, BFGS is able to reach 4 to 5 figure fidelity (i.e. $10^{-4/-5}$ infidelity), qualitatively proving the convergence properties mentioned in Section 4.2. Performing the optimization for shorter time steps allows for a smaller number of required iterations to reach convergence: empirically, is due to the fact that the control landscape approaches (i.e. better approximates) the exact one. For this cases the optimized fidelities differ only slightly, meaning that different seeds converge to the same optimal solution.

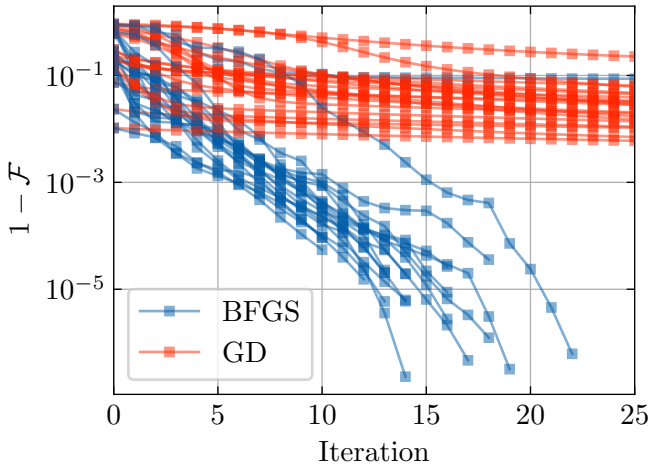

Figure 7.4: Comparison for 20 different seeds for GD and L-BFGS-B (GD and BFGS respectively in the legend box) using the rectangular rule.

7.1.2 Transmon system

In this section we study the higher-dimensional problem of a superconducting transmon system discussed in Section 2.2.1. This setup is a common test-bed for gradient and machine learning based optimal control, as e.g. in [76, 77].
For the $N = 2$ sites, the computational basis is composed by the set of vectors $\{|00\rangle, |10\rangle, \ldots, |12\rangle, |22\rangle\}$ and the system described by the Hamiltonian

$$\hat{H}_n = \left[\Delta \hat{b}_1^\dagger \hat{b}_1 + \frac{1}{2}\sum_{j=1,2} \delta_j \hat{b}_j^\dagger \hat{b}_j \left(\hat{b}_j^\dagger \hat{b}_j - 1\right) + J\left(\hat{b}_1^\dagger \hat{b}_2 + \hat{b}_1 \hat{b}_2^\dagger\right)\right] + \Omega_n \left(\hat{b}_1^\dagger + \hat{b}_1\right) = \hat{H}^d + \hat{H}_n^c \,. \tag{7.7}$$

For a more in-depth traction of the form for the Hamiltonian we refer to Section 2.2.1. As for the parameters, initially set as in [11], we make use of the calibration data (as of 16th September, 2022) of the 127-qubit IBM Washington processor and reported in Appendix B and available through the IBM Quantum Compute Resources cloud [78].

State transfer

We consider the state transfer

$$|\psi_{\text{ini}}\rangle = |10\rangle \to |\psi_{\text{tgt}}\rangle = |11\rangle \ , \tag{7.8}$$

i.e. a single state mapping of a CNOT gate in the qubit subspace $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. Note that the control Hamiltonian is not diagonal in the natural computational basis and to obtain a proper representation for the Trotter exact

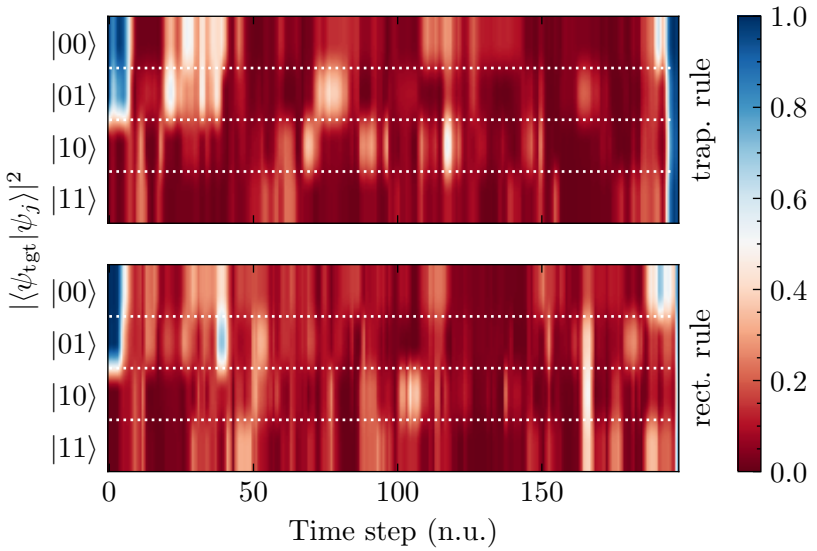

Figure 7.5: Square modulus of the overlap for the evolved wave function with the target state for each basis state at each time step: the upper (lower) panel show the results for the trapezoidal (rectangular) rule. The first 2 states have unit overlap also in the first few iterations, since they are left untouched by a CNOT gate.

derivatives we therefore numerically diagonalize $\hat{H}_n^c = \Omega_n(\hat{b}_1^\dagger + \hat{b}_1)$. The controls Ω_n constitute only a scaling factor of the control Hamiltonian $\hat{H}_n^c$, whose eigenvectors are identical $\forall\ \Omega_n \neq 0$: we thus consider $\Omega_n = 1$ for simplicity. We rotate both the Hamiltonian and the states via the transformations

$$\begin{aligned} \hat{H}_n^c &\leftarrow \hat{\mathcal{R}}^\dagger \hat{H}_n^c \hat{\mathcal{R}}, & \hat{H}^d &\leftarrow \hat{\mathcal{R}}^\dagger \hat{H}^d \hat{\mathcal{R}} \\ |\psi_{\text{ini}}\rangle &\leftarrow \hat{\mathcal{R}}^\dagger |\psi_{\text{ini}}\rangle\,, & |\psi_{\text{tgt}}\rangle &\leftarrow \hat{\mathcal{R}}^\dagger |\psi_{\text{tgt}}\rangle\ , \end{aligned} \tag{7.9}$$

where $\leftarrow$ indicate assignment and $\hat{\mathcal{R}}$ is the basis transformation operator whose columns are the eigenvectors of $\hat{H}^c$. Where possible, we use a sparse matrix representation to aid computation time; for the matrix diagonalization we fall back to the dense representation, since the available sparse algorithms (as e.g. the implicitly restarted Arnoldi method implemented in `scipy.sparse.linalg.eigs`) are capable of computing only a few eigenvalues with specific features such as largest real part or largest magnitude.

Quantities are given in non-dimensionalized numerical units: setting $\hbar = 1$, frequencies are in units of GHz, energies in units of the coupling parameter $\mathcal{E}_{\text{unit}} \coloneqq J \approx 5.654 \cdot 10^{-2}$ n.u. and time in units of $1/|J|$. For the time evolution, we find Eq. (7.3) true up to $100 \cdot \varepsilon$ where $\varepsilon = 2.22 \cdot 10^{-16}$ is the machine precision for the double-precision floating-point format.

We limit our controls to be in the range of $\Omega/2\pi = \pm 200\,\text{MHz}$ and uniformly draw piece-wise constant sequences. This constrained amplitude, which will be enforced by the optimization algorithm, determines in part the QSL, for

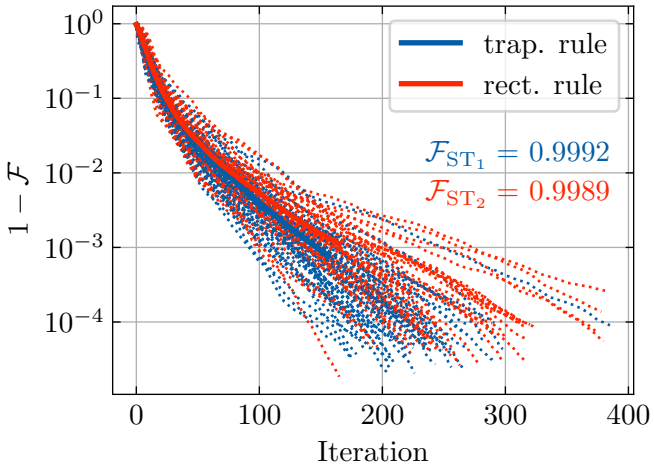

Figure 7.6: Infidelity for 50 different initial conditions for the transmon gate transfer: means are superimposed (solid lines). Overall, the trapezoidal rule shows to have slightly better convergence properties in terms of number of iterations and final fidelity. Due to the different number of iterations for each run, the mean is computed truncating the trajectories to the length of the shortest one.

which the control problem becomes exactly solvable [11, 53]. We evolve the state for a duration $T = 50\,\mathrm{ns}$ and $N_t = 100$; having fixed the time interval, the effective gate duration is reduced by the fraction needed for it to be a multiple of it, $T_{\mathrm{eff}} \coloneqq T - [(T\delta t)\%1]\delta t$ (% is the remainder operator).
After optimizing with L-BFGS-B we find in both cases fidelities $> 99.99\%$.

This prototypical state-to-state transfer can best serve as a starting point for more complicated cases rather than a standalone problem. In fact, a more interesting task is given by realizing a specific transformation for the whole set of the basis states, namely a gate transfer.

Gate transfer

In this section we synthesize a CNOT gate using piece-wise constant pulse $\Omega(t)$. The dynamics is again described by an Hamiltonian in the form of Eq. (7.7),

$$\hat{H}_n = \sum_{j=1}^{N} \left[\Delta_j \hat{b}_j^\dagger \hat{b}_j + \frac{1}{2}\delta_j \hat{b}_j^\dagger \hat{b}_j \left(\hat{b}_j^\dagger \hat{b}_j - 1 \right) + \sum_{\langle i,j \rangle} J_{ij} \left(\hat{b}_j^\dagger \hat{b}_i + \hat{b}_i \hat{b}_j^\dagger \right) \right] \tag{7.10}$$

$$+ \sum_{j=1}^{N} \Omega_n^{(j)} \left(\hat{b}_j^\dagger + \hat{b}_j \right) = \hat{H}^d + \hat{H}_n^c \,, \tag{7.11}$$

where N is the number of sites and $\langle \cdot \rangle$ indicates nearest neighbors indices. As an important caveat, the detuning is 0 for the target qubit involved in the

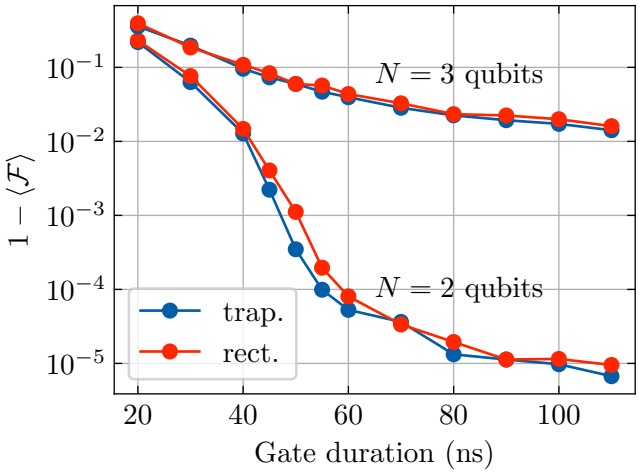

Figure 7.7: Average infidelity as a function of the gate duration for a fixed discretization time step $\delta t = 2.5 \cdot 10^{-2}$. For 2 qubits a 4-figure fidelity is reached at approximately 60 ns: an added third spectator qubit prevents a fidelity under 2 figures also for much longer time windows.

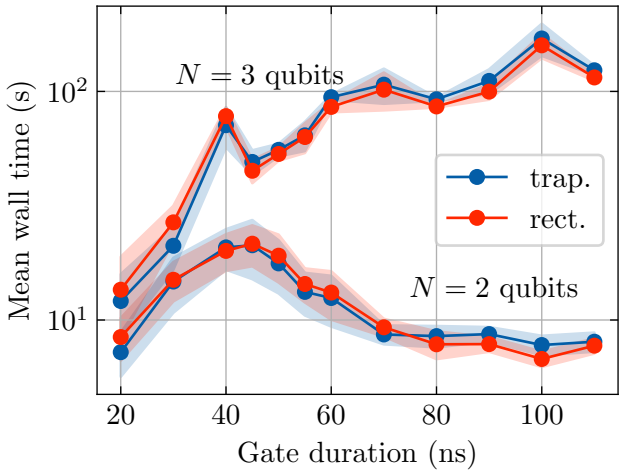

Figure 7.8: L-BFGS-B mean wall time as a function of the gate duration for a fixed discretization time step $\delta t = 2.5 \cdot 10^{-2}$: shaded areas are $\pm$ one standard deviation around the mean. Interestingly, for 2 qubits the time it takes for the algorithm to converge stays roughly constant above 60 ns, a threshold that is also observed in Fig. 7.7 to reach acceptable values of fidelity: the convergence time is thus fixed by the QSL independently of the gate duration.

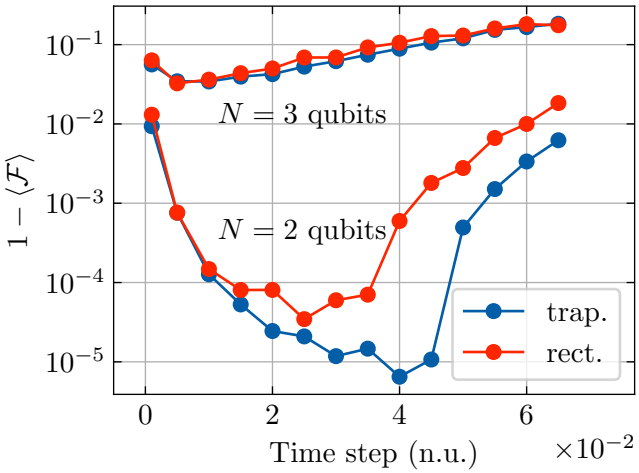

Figure 7.9: Mean infidelity as a function of the discretization time step δt for a fixed physical time of $T = 60\,\text{ns}$. Remarkably, arbitrarily decreasing δt does not lead to a higher fidelity: a possible issue with shorter time steps is that the larger frequency of the system is responsible for setting the time scale. With shorter time steps, the system picks up the initial set of random controls as white noise, since it is not able to generate the lower frequencies: shorter time steps imply a larger frequency window. A possible workaround would be to instead generate ad-hoc controls with characteristic frequencies as a first informed initial guess. Values around $4 \cdot 10^{-2} \approx 1\,\text{ns}$ are within the sampling frequencies of standard microwave pulse generators.

gate transformation as a consequence of the rotating frame transformation (see Section 3.3.1).
Practically, the problem consists in simultaneously evolving all basis states and consider the fidelity in the form Eq. (5.15), which represents a cumulative figure of merit of the composite problem. The derivative have therefore to be changed accordingly: retracing the steps done in Appendix A one can recover for e.g. the trapezoidal rule in the formalism of Section 5.2 the gradient form

$$\nabla J_{\mathcal{F}_{\text{sm}}} = \frac{1}{s^2} \left| \sum_{k=1}^{s} \langle \chi_{N_t}^{(k)} | \psi_{N_t}^{(k)} \rangle \right|^2 , \tag{7.12}$$

which is the reformulation of Eq. (5.15) in terms of overlaps and allows to take into account only the dynamics in the computational subspace of interest, tracing out the remaining transmon levels. The case of $\mathcal{F}_{\text{ss}}$ can be trivially obtained by taking the modulus squared under the summation sign.
Both the cases of local ($\Omega_n^{(j)}$) and global ($\Omega_n^{(j)} \equiv \Omega_n \, \forall j$) control are implemented. We choose a gate duration of 60 ns for a time step of $\delta t = 1.5 \cdot 10^{-2}$ n.u.. The normalization parameter for the energy is in this case chosen to be the absolute value of the minimum of the coupling parameters

J_{ij}. The time evolution over the entire range of gate duration is rendered visually in Fig. 7.5 for both rules.
We optimize 50 different seeds and the optimization trajectories are shown in Fig. 7.6: on average with Jensen's derivative method we reach fidelities $> 99\%$.

Scaling with the number of qubits

With this tools we can now start asking more complex questions. In Fig. 7.7 and Fig. 7.8 we show the mean infidelity and the mean wall time consumption for each gate duration averaged over 10 different seeds. In the former we see that when the gate duration increases above 60 ns the control problem becomes exactly solvable for 2 qubits in the sense that the infidelities become insignificantly low: moreover, a larger time window does not contribute to better results. The gate time reported by IBM [78] are inferior to this result by at least an order of magnitude. This is due to the fact that the simulated system is more controllable, in the sense that I can resolve frequencies above 1 GHz and drive both qubit and leakage transitions; moreover, the Hamiltonian describing the dynamics is a theoretical model which does not include (most notably) leakage and other experimentally present effects.
For a certain set of initial conditions the optimization landscape becomes more difficult to traverse: over the (small) statistics collected, for e.g. 2 qubit at $T = 20$ roughly 60% of the initial seeds failed. Similarly, for too big time steps (see Fig. 7.9) starting form $\delta t = 5.0 \cdot 10^{-2}$ to $\delta t = 6.5 \cdot 10^{-2}$ the seeds failure rate grows from 9% to more than 97%. Importantly, both rules perform equally on average even at a shorter gate duration.
For 3 qubits no significant improvement in the fidelity is observed at the price of an exponential trend in the computational time.

In a real-life scenario, the number of controls is limited by experimental constraints: here, since we are not constrained by any practical engineering issues, we run the optimization tasks for system with and without an individual set of controls for each qubit. Results are shown in Fig. 7.10 for the average fidelity over 10 different seeds: fundamentally, more controls bring a relevant advantage only for a low system size and the inclusion of spectator qubits is detrimental to the fidelity. In Fig. 7.11 is evident the exponential scaling (note the log-scale) with the number of qubits of the mean wall time: both rules perfectly overlap with one another.
Being the two integration rules comparable by all means, for the sake of clarity we will restrict our analysis from this section onward to the sole case of the trapezoidal rule.

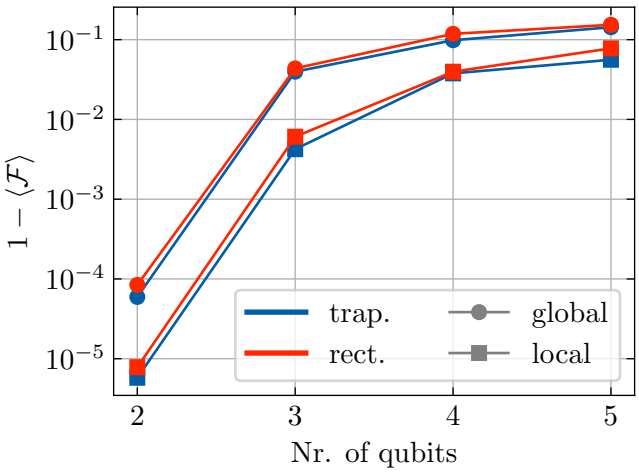

Figure 7.10: Mean infidelity as a function of the number of qubits. While performances are almost identical for both integration rules, local controls provide a gain of one more significant figure in fidelity. This trend does not scale with the number of sites: for 5 qubits the fidelities become comparable with the global control case.

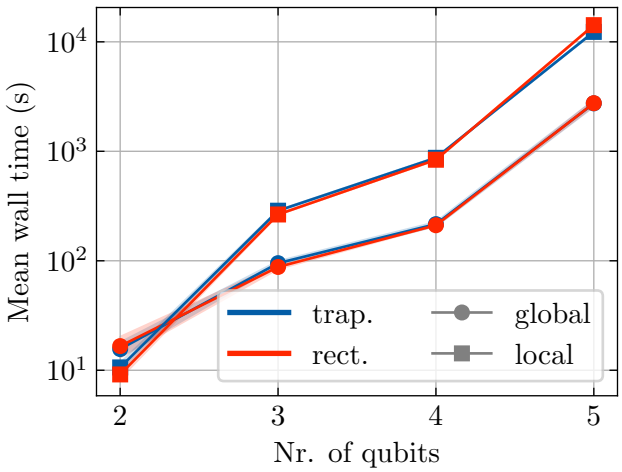

Figure 7.11: Mean wall time as a function of the number of qubits. For both local and global controls the trend is linear (in log-scale): curves become separated by almost an order of magnitude for a number of qubit ≥ 4.

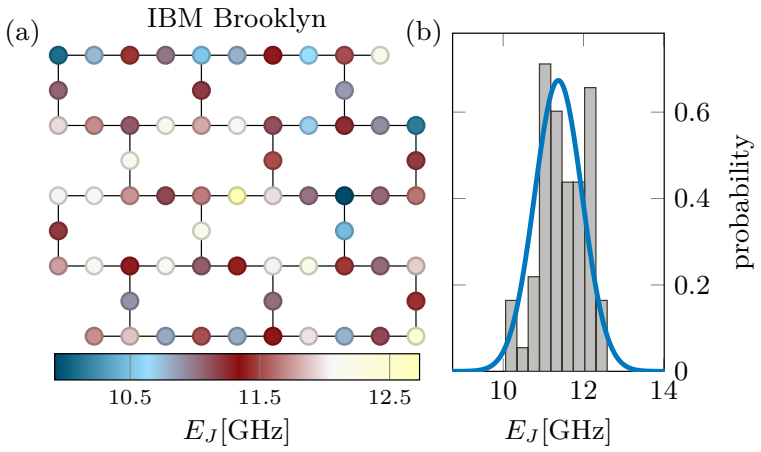

Figure 7.12: (a) Layout of the 65-qubit transmon array "Brooklyn": note the variation of the Josephson energies E_J which is largely uncorrelated in space. (b) Spread of the E_J plotted for the "Brooklyn" chip, consistent with a Gaussian distribution (solid line). Similar levels of disorder and distributions are found in all IBM's transmon devices. Figure and caption from [79].

7.2 Many-body systems

We now enter the many-body regime with the theoretical tools introduced in Chapter 6. Our goal is to demonstrate the feasibility of the methods in Section 5.2 also for many-body systems using TNs methods. For this, we analyze a state transfer from the ground $|\psi_{\text{ini}}\rangle = |0, 0, \ldots, 0\rangle$ to the excited state $|\psi_{\text{tgt}}\rangle = |1, 1, \ldots, 1\rangle$ of a $N = 10$ transmon chain.
The code in this section has been developed in Julia using ITensors [80]; we make use of the `LBFGSB.jl` [30] previously mentioned, fixing the maximum number of iterations to 1800 and to 10^{-10} the tolerance for the the infinity norm of the projected gradient. In contrast with Section 7.1.2, the simulation parameters are chosen by random sampling from a gaussian distribution with a 10% standard deviation around the values of [11]. This type of "noisy" approach is actually enforced by recent results: indeed, "a certain amount of intentional frequency detuning ('disorder') is crucially required to protect individual qubit states against the destabilizing effects of nonlinear resonator coupling" [79]. To report a concrete example, Fig. 7.12(b) shows that the spread of Josephson energies in IBM devices is consistent with a Gaussian distribution, a common feature for all IBM current devices whose parameters are documented publicly [78].
After the initialization of the initial and final state via MPS, we do not set a maximal bond dimension but instead fix a cutoff parameter that allows the new bond dimension after each step to be determined adaptively as long as the resulting truncation error remains below the value provided. In Fig. 7.13 we show a color-map of the squared overlap of the evolved wave function

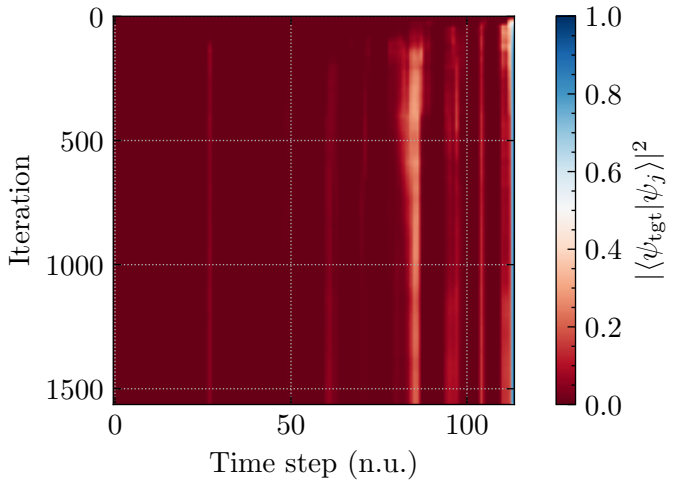

Figure 7.13: Color-map for each iteration and time step of the fidelity, indicated by the color dimension. After the first couple of hundreds iteration, the fidelity follows the same evolution pattern (horizontal sections): we interpret this as the controls effectively "learning" the correct time evolution.

with the target state (i.e. the fidelity) for each time step, over all the optimization iterations. The final fidelity is around 91.93% (see Fig. 7.14, with the red dashed line), a result that could be further improved by relaxing the fixed convergence constraints. Moreover, the representation in TN allows to extract some additional information: we can observe the evolution of the Von Neumann entropy of Eq. (6.6) for each iteration, which is reported in Fig. 7.14.
The results here shown demonstrate the possibility of extending Jensen's argument to the many-body case, with the added possibility offered by the TN framework to compute physical quantities of interest.

7.2.1 Automatic differentiation for Tensor Networks

The hypotheses set by Jensen's method pose a restriction to the size of the class of problems we can tackle. The possibility to free the gradient form from additional assumptions is given by AD (see Section 4.3), which is able to numerically evaluate derivatives independently of the problem's nature. AD has just begun to attract the attention of the machine learning community and has so far seen few applications to the TN sphere (see e.g. [81]).
We analyze this possibility by first tackling the more simple problem of finding the energies of a 10-sites 1/2-spin chain in the transverse field Ising model.

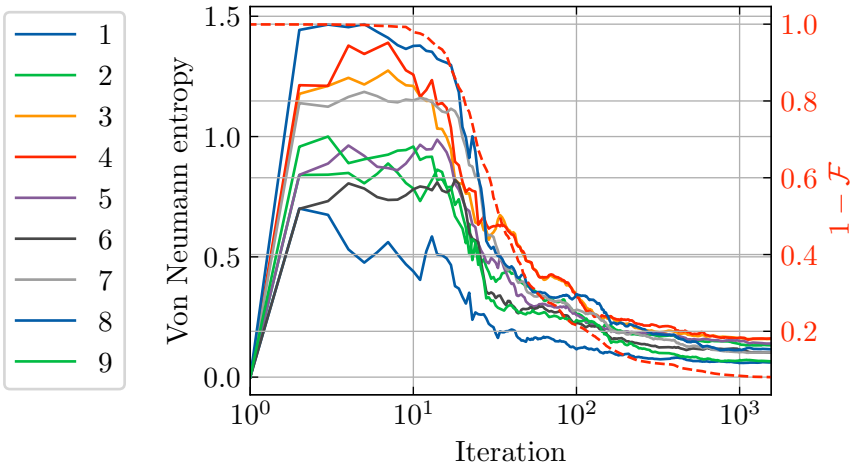

Figure 7.14: Von Neumann entropies for each bipartition of a 10-sites transmon chain. As expected for pure state, the initial value is identically null for all bipartitions and approaches 0 from above in the last few iterations: its value of $\mathcal{O}(10^{-2})$, together with the fidelity, gives another measure (more general since it is not state specific) of the goodness of the optimized set of controls.

1−D Ising chain in transverse field[1]

The system dynamics follows the Hamiltonian

$$\hat{H} = -\frac{J}{4}\sum_{\langle i,j\rangle}\sigma_i^z\sigma_j^z - \frac{h}{2}\sum_{i=1}^{N}\sigma_i^x \,, \tag{7.13}$$

where $\sigma_i^{x/z}$ denote Pauli operators acting on site i and the first sum runs over neighboring lattice sites. As the computational basis we choose the spin basis states $|\vec{s}\rangle = |s_1, \ldots, s_N\rangle$ with $s_i = \uparrow, \downarrow$.
The transverse field Ising model can exactly be solved by mapping it to a non interacting fermion model via Jordan-Wigner transformations: this Hamiltonian can then be solved by finding the eigenstates of the single particle Bogoliubov-de Gennes Hamiltonian of dimensions $(2N \times 2N)$ [82]. This allows us to have the exact solutions at hand to confront with the numerically obtained ones. The optimization is performed by a GD routine using, in a similar fashion to variational methods, the energy of the k-th level as a cost function. The gradient is computed using `Zygote.jl` [83], an AD library for the Julia language.
We introduce a penalty, i.e. an extra term to the cost function, to enforce the separation of states: this term is chosen to be the projection of the current state onto all previous ones up to a multiplicative factor w fixed at

[1]The code for this section was developed by Niklas Tausendpfund, a PhD student in the host group at Forschungszentrum Jülich.

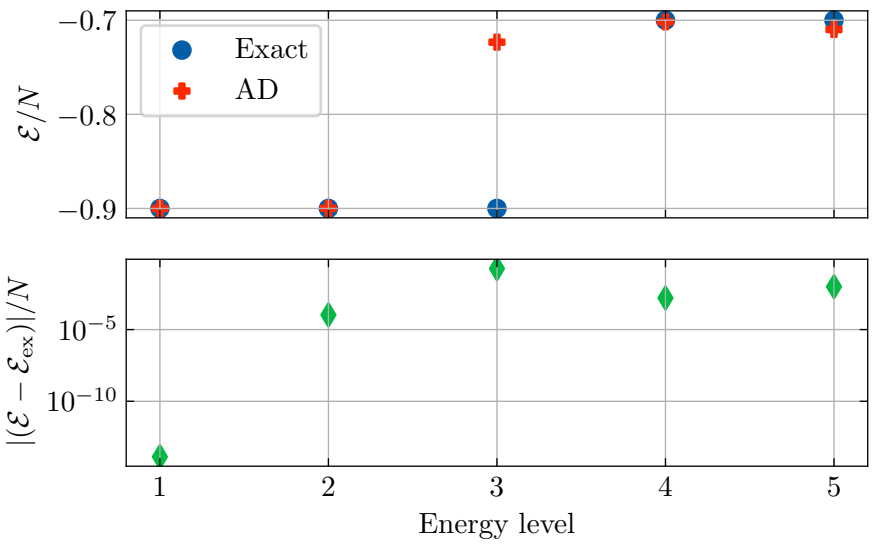

Figure 7.15: Upper panel: theoretical and numerical normalized expectation values of the energies for site for the transverse field Ising model. Bottom panel: modulus of the difference between theoretical and numerical solution. We attribute the discrepancy obtained for the third energy level to the fact that the addition of a penalty term often brings with it poor convergence properties of the optimization algorithm, as it makes many small adjustments to ensure the parameters satisfy the constraints.

100. Thus, for the k-th energy level, the cost function $J_k(\psi_x)$ has the form

$$J_k(\psi_x)\,\langle\psi_x|\psi_x\rangle = \left\langle \psi_x \middle| \hat{H} \middle| \psi_x \right\rangle + (1-\delta_{k,0}) w \sum_{i=1}^{k-1} |\,\langle\psi_x|\psi_k\rangle\,| \,. \tag{7.14}$$

Results are shown in Fig. 7.15: the numerical solutions show a good accordance with the expected theoretical values, with the exception made of the third energy level (see caption to Fig. 7.15).

7.2.2 Transmon chain

The transmon chain is much more challenging than the spin chain case due to the greater complexity of the Hamiltonian and the additional computational difficulty due to the calculation of the time evolution. Moreover, integrating libraries with each other is a highly non-trivial task. The (apparent) immediacy of Zygote comes at the price of many practical issues, the main one being mutating functions. AD systems like Zygote are built on top of basic principle of calculus where a function $f(x)$ does not modify x and only produces the output y based on x. AD systems are built to programmatically apply the chain rule to a series of function calls, which is often not how typical programs behave. Variable or array mutations as a result of

```
# benchmark for Jensen's cost function and gradient evaluation
Range (min … max):  855.982 ms …    1.151 s  ┊ GC (min … max): 7.32% … 8.74%
Time  (median):     961.111 ms               ┊ GC (median):    8.03%
Time  (mean ± σ):   982.266 ms ± 107.560 ms  ┊ GC (mean ± σ):  8.09% ± 0.45%

Memory estimate: 749.09 MiB, allocs estimate: 2722431.
```

Figure 7.16: Time and memory benchmark to compute the analytical value of the cost function and gradient with Jensen's derivative method. The median, as a robust measure of central tendency, should be relatively unaffected by outliers and can be used as representative value.

the application of a function are difficult for AD systems to handle, because they must track the changes and store older versions of the variable. Thus, most AD frameworks (including Zygote) do not support e.g. in-place linear algebra operations, which can greatly speed up the simulation of quantum dynamics. Circumventing this limitations requires to manually derive a gradient (where possible) and writing a custom differentiation rule. Without going into the technicalities of AD theory, we refer to such custom rules as "adjoints".

As a reference point, we benchmark [84] the time needed both to evaluate the cost function and to compute the gradient, here shown in Fig. 7.16. Our first naive approach consists in simply using the AD engine as a black box. A benchmarks of the time and memory usage is reported in the bottom panel of Fig. 7.17: clearly the amount of resources required sets this first attempt outside of any practical use.

A more informed approach requires computing the adjoints. For this, one has to define a "pullback" (in the differential geometry sense) $\mathcal{B}_y$ that given

```
# benchmark for AD with custom rules
Range (min … max):  1.372 s …    1.707 s  ┊ GC (min … max): 10.47% … 9.55%
Time  (median):     1.455 s               ┊ GC (median):    10.26%
Time  (mean ± σ):   1.497 s ± 147.867 ms  ┊ GC (mean ± σ):   9.77% ± 0.85%

Memory estimate: 1.14 GiB, allocs estimate: 4247737.
```

```
# benchmark for AD withouot custom rules
Single result which took 20.874 s (11.50% GC) to evaluate,
 with a memory estimate of 2.26 GiB, over 15534040 allocations.
```

Figure 7.17: Time and memory benchmark to compute the analytical value of the cost function and gradient with AD for the TN transmon chain with (upper panel) and without (bottom panel) custom differentiation rules.

$y = f(x)$ and the gradient $\bar{x} \coloneqq \frac{d\ell}{dx}$ implements the vector-Jacobian product

$$\bar{x} = \frac{d\ell}{dx} = \frac{d\ell}{dy}\frac{dy}{dx} = \mathcal{B}_y(\bar{y}) \ . \tag{7.15}$$

Let's make some examples. The scaling function $f(\alpha, A) = \alpha A$ of a complex variable has a pullback according to

$$f^*_{\alpha,A}(B^b)^\# = \mathfrak{Re}\left(\mathrm{Tr}(A^\dagger B)\right) e_n + \alpha B v_\perp \ , \tag{7.16}$$

where e_n denotes the direction along α and $v_\perp$ stands symbolically for the orthogonal directions, i.e. along A. The pullback of the norm function $f(A) = \left(\sqrt{\mathrm{Tr}(A^\dagger A)}\right)^{-1}$ is given by

$$f^*_A(\bar{c})^\# = -\frac{\bar{c}}{f(A)^3} A \ , \tag{7.17}$$

while the normalizing function $f(A) = A/\|A\|$ has the pullback

$$f^*_A(B^b)^\# = -\frac{\mathfrak{Re}\left(\mathrm{Tr}[A^\dagger B]\right)}{\|A\|^3} A + \frac{1}{\|A\|} B \ . \tag{7.18}$$

As one can see, things get easily very complicated, very fast. Implementing Eq. (7.18) and a custom rule to compute the exponential of a matrix used to compute the Trotter gates does bring some advantage both in terms of time and memory (see the upper panel in Fig. 7.17). The full expressions are not here reported both for brevity and for the fact that ultimately the effort did not pay off to any tangible result. Moreover, such computations are tedious, error prone and not easily implementable: several obstacles have to be overcome of inter-library compatibilities.
As pointed out in [85], "The more high-level the elemental functions are, the less the numerical overhead of AD. However, this comes at the cost of having to define more and more analytical adjoints. This is why many AD frameworks have been slow to adopt operations that are outside of the narrow scope of machine learning, which only requires real-valued dense matrix-vector operations. In contrast, quantum dynamics is inherently described with complex-valued state vectors, and operators are usually sparse. Defining AD adjoints for complex linear algebra operations is possible, but has only recently seen adoption."

Despite the great advantage that AD would potentially bring in terms of accuracy and generaizability, we conclude that such methods are still immature for practical use when applied to TN methods.

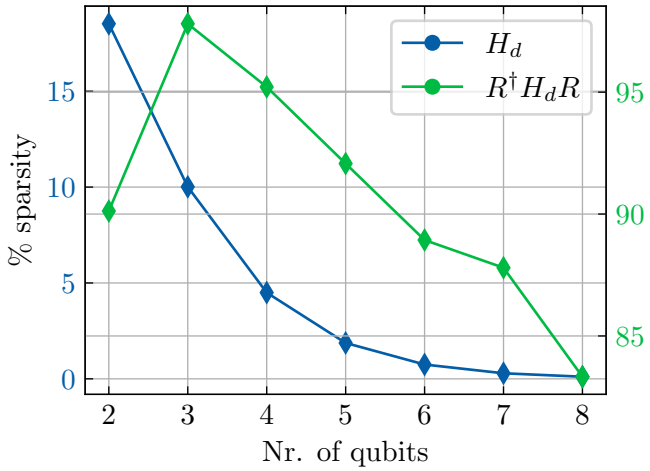

Figure 7.18: Percentage sparsity (note the different axis scales) of the drift Hamiltonian H_d as a function of the number of qubits. The change of basis heavily effects the number of non-zero elements present, preventing to exploit the computational advantage that derives by using a sparse representation formalism. The sparsity for H_d follows a $\mathcal{O}(N/d^N)$ trend, with N the number of qubit and d the local dimension.

7.3 General comments on Jensen's derivative method

In this section we analyze some specificities of Jensen's method, discussing some of its limitations and ultimately aiming to extend its applicability to more general scenarios.

As a first step, we note that the change of basis in Eq. (7.9) involves a decisive increase in the percentage of non-zero elements in the matrix representation of the control Hamiltonian, as depicted in Fig. 7.18. This potentially poses a threat to the use of ED methods, but can be circumvented (as the authors in [61] mention for separate reasons) by caching the matrix calculation and loading it into memory at run time.

One important detail that is not explicitly stressed in [61] is that in order to have the simplified gradient formula, the diagonality condition for the control Hamiltonian is just one special case of the required hypotheses of commutativity between the control Hamiltonian and its derivative with respect to the controls. If we want to strictly adhere to the formalism of Eq. (5.33), remembering the definition of the recursive commutator in Eq. (5.32), the effective condition is

$$\left[\hat{H}_n^c, \frac{\partial \hat{H}_n^c}{\partial u_n}\right] \stackrel{!}{=} 0 \,, \tag{7.19}$$

which is valid in particular for the (less general) case in which the Hamiltonian is bilinear in the controls, i.e. such that $H_n^c = f(\Omega_n)H^c$, which incidentally corresponds to the class of problems we are considering.
If we focus on the trapezoidal rule for a moment, substituting the rule for unitary transformation Eq. (3.17) in the definition of the trotterized propagator $\hat{\mathcal{U}}_n^{\mathrm{ST}_1}$ in Eq. (5.27) we get

$$\begin{aligned}\tilde{\hat{\mathcal{U}}}_n^{\mathrm{ST}_1} &= \tilde{\hat{\mathcal{U}}}_{n+1}^{c/2}\tilde{\hat{\mathcal{U}}}_n^{d}\tilde{\hat{\mathcal{U}}}_n^{c/2} \\ &= \hat{\mathcal{R}}_n^{\dagger}\hat{\mathcal{U}}_n^{c/2}\hat{\mathcal{R}}_n\hat{\mathcal{R}}_{n-1}^{\dagger}\hat{\mathcal{U}}_{n-1}^{d}\hat{\mathcal{R}}_{n-1}\hat{\mathcal{R}}_{n-1}^{\dagger}\hat{\mathcal{U}}_{n-1}^{c/2}\hat{\mathcal{R}}_{n-1} \\ &\neq \hat{\mathcal{R}}_n^{\dagger}\hat{\mathcal{U}}_n^{\mathrm{ST}_1}\hat{\mathcal{R}}_{n-1}\ , \end{aligned} \tag{7.20}$$

where $\tilde{\cdot}$ indicates the transformed unitary. The last inequality is due to the fact that

$$\hat{\mathcal{R}}_n^{\dagger}\hat{\mathcal{R}}_{n+1} = \mathrm{e}^{-i\dot{\hat{\mathcal{R}}}_n^{\dagger}\hat{\mathcal{R}}_n\,dt}\ , \tag{7.21}$$

a relation that can be simply proven by choosing e.g.

$$\hat{\mathcal{R}} = \exp\left(-if(t)\theta\right) \qquad f'(t) = \frac{f(t+dt)-f(t)}{dt}\ , \tag{7.22}$$

where θ is some constant operator.
Given that for a single time instant it holds $\hat{\mathcal{R}}^{\dagger}\hat{\mathcal{R}} = \mathbb{1}$, we have a series of lucky cancellations happening in Eq. (7.20) only (a) in the time-independent case and (b) for the rectangular rule also in presence of time dependence (the subscripts are identical between next-nearest rotations). In both cases, the formulae for the gradients Eq. (5.37), Eq. (5.38) become basis-independent, which constitutes an important improvement of the method. Note that this also applies to the overlap formula in Eq. (5.22) by construction.

The authors in [61] state that the only case in which there is no advantage in performing the basis change over the dense exponentiation for $Q > 1$ (see Section 5.2 for the notation) is that for which (1) the rotation matrix depends on the controls (and hence on the time) and simultaneously (2) there is no analytic closed expression. In fact, if (1) is true but not (2) the matrix can always be expressed as a function of the controls and successively simply evaluated for each instant (which is cheaper than constructing $\hat{\mathcal{R}}$ by diagonalization). On the contrary, if (2) is also true, the only way is diagonalisation at each step.
This confirms what we previously stated. Indeed, if only (1) is true, there is no computational disadvantage: what does increase, however, is the difficulty of implementing the solution, which in this case requires many more rotations - one for each time step. The same holds for the $Q = 1$ case.
As already stressed, the fact that the rotations are to be calculated at each time instant in the most general case is a central aspect of the method which

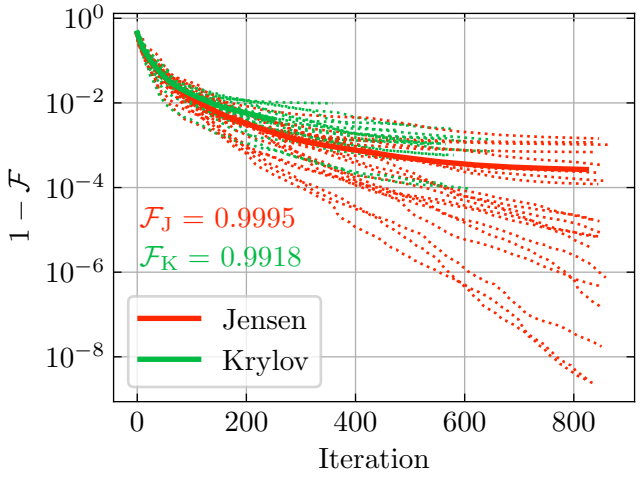

Figure 7.19: Infidelity for 20 different initial conditions for the transmon gate transfer for both the exact (Krylov) and approximate (Jensen) dynamics: means are superimposed (solid lines). Jensen's method is capable of reaching very high fidelities for certain initial conditions, whereas the exact dynamics cannot get under 4 significant figures. However, the overall performances remain comparable.

does not emerge in the examples treated in the previous sections, where the matrix H_c was already diagonal or $H_n^c = f(\Omega_n)H^c$. This difficulty has to be directly addressed in further studies for a complete benchmarking of the method also in more general scenarios.

To (partially) demonstrate this observations, since condition (a) holds for the transmon case, the simulations of the next section are implemented without the rotations listed in Eq. (7.9).

7.4 Krylov methods

In this last section we compare Jensen's method (from here on: Jensen) to the exact propagation: the goal is to verify the claim that an exact gradient with approximate dynamics yields better results compared to an approximate gradient with exact dynamics. To do so, we abandon the use of the trotterized propagators and instead adopt an ED approach where the matrix exponentiation is computed via Krylov subspace methods introduced in Section 4.4.3 (from here on: Krylov), where exponential operator applications are performed without explicit construction. The gradient is instead computed as in Eq. (5.37), which we interpret in this context as first order approximation to the analytical form of the gradient, i.e. $k = 0$ in Eq. (5.36). As a first test, we implement the algorithm in Python and confront its performances with SciPy's common matrix exponential functions. In detail,

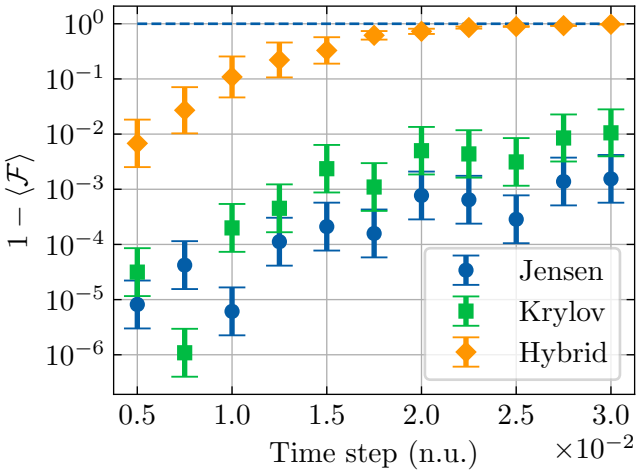

Figure 7.20: Optimal solutions as found by Jensen's approximate dynamics (blue), Krylov's full exponentiation (green) and Jensen's optimal set of controls in Krylov evolution (yellow). Error bars are ± one standard deviation around the average for 10 runs. Remarkably, the error of Jensen's dynamics accumulates in non-trivial ways, making a solution found with the approximate dynamics impractical for the real experimental case that instead follows an exact evolution.

`expm` implements the matrix exponential described in [86], which is essentially a Padé approximation; `exp_multiply` is based on [87] and implements the action of the exponential of matrix on a matrix/vector. Finally the `krylov` method shown is a Python wrapper to the implementation of `expm` from Expokit build for Fortran and C++ [88]. As an important note, all three methods produced the same results up to a relative tolerance fixed at 10^{-7}.
Results are reported in Fig. 7.21 and Fig. 7.22 respectively for constant 10% and 90% sparsity (here intended as fraction of non-zero elements) and show encouraging results. The case of a sparsity following an exponentially decreasing trend in the form $\mathcal{O}(N/d^N)$ represents the physical scenario and is thus of central interest.

We now address the problem of the dynamics. For the sake of comparison, we use the same system size and simulation parameters of [61] for the case of gate transfer analyzed previously. In Fig. 7.19 we show the comparison of different convergence trajectories in the optimization landscape. We highlight that for the majority of cases, Krylov's iteration ends due to satisfying the constrain on the gradient projection whereas Jensen's iterations saturate the number of function calls.

To evaluate the error committed by approximating the dynamics we plug

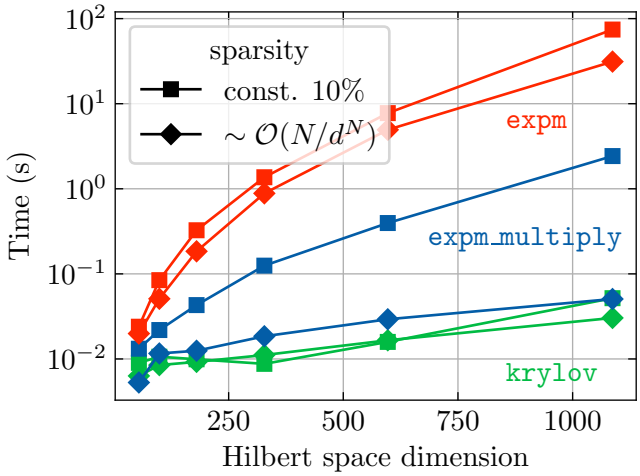

Figure 7.21: Benchmark for 3 different exponential methods as a function of the Hilbert space dimension. See the main text for their precise description. Krylov subspace projection techniques allow for a dramatic speed-up with respect to the known SciPy methods. Interestingly for the case of an exponentially decreasing sparsity, the time remains approximately constant over all matrix sizes. Noticeably, for a dimension < 100 feeding the matrices to the `expm_multiply` method is faster: this is due to the fact that Krylov approximations rely on repeated matrix-vector multiplications, which are not advantageous for such small Hilbert space dimensions.

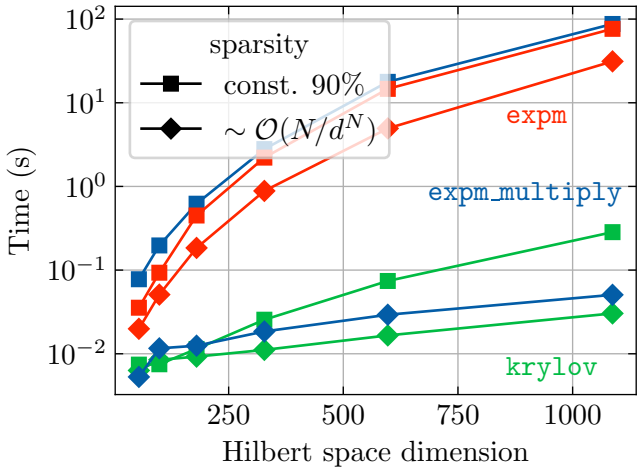

Figure 7.22: Benchmark as in Fig. 7.21 with constant sparsity $10\% \to 90\%$. The increased sparsity worsens performances noticeably: the `expm_multiply` method becomes comparable with `expm` for the densly populated matrix cases. Still, Krylov's method maintain the computational advantage.

the optimized set of controls via Jensen into the exact dynamics. Results are show in Fig. 7.20.
As a first remark, we note that the average distance between Jensen and Krylov is around an order of magnitude, which coincides with the observations of the averages in Fig. 7.19 and confirms the exceptionally good fidelity solutions as outliers. Moreover, this distanc ecould be further reduced by using the exact derivatives for the Krylov optimization.
Clearly, an optimal solution for Jensen is not an optimal solution for Krylov. This trend is steady across different values of the time discretization and only mitigates for very small intervals (in particular, smaller the one used in [61]) when the approximated landscape comes closer to the exact one. Under this light the trajectories in Fig. 7.19 appear as too optimistic and not leading to real, experimentally obtainable solution. In fact, the time evolution obtained with Krylov is, up to numerical errors, what happens in a real-life scenario: the only context in which this is does not hold is the special case of a stroboscopic evolution protocol (or Floquet dynamics).
Ultimately, although we observe by empirical benchmarking roughly a factor of 10 separating the performances of exact and approximate time evolution methods, sacrificing gradient exactness in favour of a gained computational advantage proves detrimental for the quality of the minima of the control landscape. This evidence exposes a fundamental flaw of Jensen's method: the need for in-depth studies to further explore this aspect sets the natural conclusion to this section.

Chapter 8

Conclusions and outlook

In this thesis we explored gradient-based optimal control algorithms for superconducting qubits in the few- and many-body regime.
We examined a recently developed method (Jensen's method) to efficiently derive analytically exact control derivatives assuming a diagonal control Hamiltonian and adopting two trotterized propagators to describe the (approximate) dynamics, which was interpreted in terms of optimization landscape. We demonstrated the main ideas by considering a state transfer for two problems of varying Hilbert space size, namely a Landau-Zener transition and the higher dimensional case of a superconducting transmon system, achieving for both a fidelity of over 99.99%.
We expanded this treatment to the case of a gate transfer and showed its application in the many-body scenario using matrix product states combined with a variant of the time evolution block decimation algorithm tailored to the structure of the problem. The results validate the approach to the class of very high-dimensional, many-body state transfer problems: an unsatisfactory fidelity value was not considered of central importance since (1) the aim was to show the extensibility of the method to many-body problems for which (2) better results are only a matter of fine tuning of the optimization algorithm.
Successively, we explored the application of automatic differentiation to tensor network methods, first considering the energy levels of an Ising model in transverse field and then the state transfer from ground to first excited of the transmon chain. Particular attention was posed in the fruitful interplay of scientific computing libraries and the efficiency of the written code. Despite their appealing advantages, time and memory constraints ruled out the adoption of automatic differentiation techniques in the optimal control context.
We extended Jensen's argument of exact derivatives, freeing it from the constraint of formulating the problem in a control-diagonal basis. Finally, we

compared this approach to the exact dynamics via Krylov subspace methods to compute the matrix exponential. The results obtained show that an optimal solution of the approximate landscape found via approximate derivatives does not lead to the same minima when applied to the exact evolution. This last point sets the stage for further discussion on the topic.

Future prospects can be developed in many directions. Starting with the method at the heart of the discussion in this work, accurate time benchmarks and study on the quality of the solutions are needed to assess the validity of such methodologies for the search of experimentally realizable solutions. Within the context of tensor networks, an interesting line of research could explore the use of semi-implicit methods, generalizing these techniques to long range interactions, an approach that has still not being done in the literature. In this context gradients would also easier to write analytically, being the ratio of two polynomials. As for automatic differentiation, hybrid approaches could be a winning strategy, as shown in some some very recent works (see e.g. [85]).

In conclusion, this thesis poses a first stepping-stone to the study of gradient-based optimal control within many-body systems. Future work based on the study of efficient numerical techniques able to simulate large quantum systems may open new pathways to solve new problems in the field of quantum optimal control.

Appendix A

Exact derivatives calculation

In this section we report the calculation calculations leading to the exact gradient expressions for the two Trotterized propagators with a diagonal control Hamiltonian defined in Eq. (5.27). We make use of the same notation employed in Section 5.2, considering however a slightly different derivation as the one shown in [61], Appendix A.
Our goal is thus to evaluate the derivatives in Eq. (5.23) for the propagators in Eq. (5.27). Remembering the time discretization, we recall

$$|\psi_n\rangle = \hat{\mathcal{U}}_n \dots \hat{\mathcal{U}}_2 \hat{\mathcal{U}}_1 |\psi_0\rangle \tag{A.1}$$
$$|\chi_n\rangle = \hat{\mathcal{U}}^\dagger_{N_t-n+1} \dots \hat{\mathcal{U}}^\dagger_{N_t-1} \hat{\mathcal{U}}^\dagger_{N_t} |\chi_0\rangle \ , \tag{A.2}$$

and the objective function to be maximized is as in Eq. (5.21).

Trapezoidal rule, ST_1

In the specific control-drift setting, with the trapezoidal rule for the Trotter-splitting we have

$$\hat{\mathcal{U}}_n = \hat{\mathcal{U}}_n^{c/2} \hat{\mathcal{U}}_{n-1}^{d} \hat{\mathcal{U}}_{n-1}^{c/2}, \text{ with } \begin{cases} \hat{\mathcal{U}}_n^{c/2} = \exp\left(-i\frac{\delta t}{2} \hat{H}_n^{(2)c}\right) \\ \hat{\mathcal{U}}_n^{d} = \exp\left(-i\delta t \hat{H}_n^{(1)d}\right) \end{cases} , \tag{A.3}$$

where again $\hat{H}_n^{(1)d}$ is computed as two-point average. For the derivatives of the propagator it holds (δ_{ij} being the Kronecker delta)

$$\frac{\partial \hat{\mathfrak{U}}_n}{\partial \Omega_j} = \delta_{j,n-1}\hat{\mathfrak{U}}_{j+1}^{c/2}\hat{\mathfrak{U}}_j^d\frac{\partial \hat{\mathfrak{U}}_j^{c/2}}{\partial \Omega_j} + \delta_{j,n}\frac{\partial U_j^{c/2}}{\partial \Omega_j}\hat{\mathfrak{U}}_{j-1}^d\hat{\mathfrak{U}}_{j-1}^{c/2} \tag{A.4}$$

$$\begin{aligned}
&\overset{(a)}{=} -i\frac{\delta t}{2}\Bigg(\delta_{j,n-1}\hat{\mathfrak{U}}_{j+1}^{c/2}\hat{\mathfrak{U}}_j^d\hat{\mathfrak{U}}_j^{c/2}\left(\frac{\partial H_c(t_j)}{\partial \Omega_j}\right) \\
&\qquad + \delta_{j,n}\left(\frac{\partial H_c(t_j)}{\partial \Omega_j}\right)\hat{\mathfrak{U}}_j^{c/2}\hat{\mathfrak{U}}_{j-1}^d\hat{\mathfrak{U}}_{j-1}^{c/2}\Bigg) \\
&= -i\frac{\delta t}{2}\left(\delta_{j,n-1}\hat{\mathfrak{U}}_{\underbrace{j+1}_{=n}}\left(\frac{\partial H_c(t_j)}{\partial \Omega_j}\right) + \delta_{j,n}\left(\frac{\partial H_c(t_j)}{\partial \Omega_j}\right)\hat{\mathfrak{U}}_{\underbrace{j}_{=n}}\right) ,
\end{aligned} \tag{A.5}$$

where in (a) we used the fact that

$$\begin{aligned}
\frac{\partial \exp(-iA)}{\partial u} &= \frac{\partial}{\partial u}\left(\sum_{\alpha=0}^{\infty}\frac{(-i)^\alpha}{\alpha!}A^\alpha\right) \\
&= \sum_{\alpha=1}^{\infty}\frac{(-i)^\alpha}{\alpha!}\left(\sum_{\beta}^{\alpha-1}A^\beta\frac{\partial A}{\partial u}A^{\alpha-\beta-1}\right) \\
&\overset{(b)}{=} -i\frac{\partial A}{\partial u}\exp(-iA) = -i\exp(-iA)\frac{\partial A}{\partial u} .
\end{aligned} \tag{A.6}$$

The simplification happening in (b) is possible if A is diagonal and therefore commuting with its derivatives, which is one of the two fundamental hypothesis of Jensen's method. Coming to the gradient computation we can write

$$\begin{aligned}
\frac{\partial \mathcal{F}}{\partial \Omega_j} &= o^\star\frac{\partial o}{\partial \Omega_j} + \frac{\partial o^\star}{\partial \Omega_j} = 2\mathfrak{Re}\left(o^\star\frac{\partial o}{\partial \Omega_j}\right) \\
&= 2\mathfrak{Re}\left(o^\star\sum_{n=1}^{N}\left\langle\chi_0\middle|\hat{\mathfrak{U}}_{N_t}\dots\hat{\mathfrak{U}}_{n+1}\frac{\partial \hat{\mathfrak{U}}_n}{\partial \Omega_j}\hat{\mathfrak{U}}_{n-1}\dots\hat{\mathfrak{U}}_1\middle|\psi_0\right\rangle\right) \\
&\overset{(c)}{=} -\delta t\mathfrak{Re}\Bigg(io^\star\sum_{n=1}^{N}\Bigg(\delta_{j,n-1}\left\langle\chi_{N_t-n-1}\middle|\frac{\partial H_c(t_j)}{\partial \Omega_j}\middle|\psi_{n-1}\right\rangle + \\
&\qquad + \delta_{j,n}\left\langle\chi_{N_t-n}\middle|\frac{\partial H_c(t_j)}{\partial \Omega_j}\middle|\psi_n\right\rangle\Bigg)\Bigg) \\
&= -\delta(2-\delta_{j,0}-\delta_{j,N_t})\mathfrak{Re}\left(io^\star\left\langle\chi_{N_t-j}\middle|\frac{\partial H_c(t_j)}{\partial \Omega_j}\middle|\psi_j\right\rangle\right) \\
&= \quad \delta t(2-\delta_{j,0}-\delta_{j,N_t})\mathfrak{Im}\left(o^\star\left\langle\chi_{N_t-j}\middle|\frac{\partial H_c(t_j)}{\partial \Omega_j}\middle|\psi_j\right\rangle\right) ,
\end{aligned} \tag{A.7}$$

where we recognize in Eq. (A.7) the expression of Eq. (5.37). In (c) we substituted the results of Eq. (A.4), taking care of the fact that the first (second) addend in the braket in Eq. (A.5) has meaning only for $j < N_t$ ($j > 0$), first to evaluate the derivative of the propagator and successively to recombine the initial propagators, since $\left[\frac{\partial \hat{H}_n^c}{\partial \Omega_n}, \hat{\mathcal{U}}_n^{c/2}\right] = 0$. From Eq. (A.7) one can see that the gradient on Ω_0, Ω_{N_t} is just $1/2$ of the same expression as for Ω_j with $j = 1, \ldots, N_t - 1$. Finally, we notice that the expression for Jensen's cost function carries an additional factor $-1/2$ which has to be further taken into consideration.

Rectangular rule, ST_2

The Suzuki-Trotter expansion now reads $\hat{\mathcal{U}}_n^{\mathrm{ST}_2} = \hat{\mathcal{U}}_n^{c/2}\hat{\mathcal{U}}_n^{d}\hat{\mathcal{U}}_n^{c/2}$: for the overlap derivatives it holds

$$\frac{\partial o}{\partial \Omega_n} = \frac{\partial}{\partial \Omega_n}\left(\hat{\mathcal{U}}_{N_t-1}^{\mathrm{ST}_2} \ldots \hat{\mathcal{U}}_n^{\mathrm{ST}_2} \ldots \hat{\mathcal{U}}_1^{\mathrm{ST}_2}\right) = \left\langle \chi_{n+1} \middle| \frac{\partial \hat{\mathcal{U}}_n^{\mathrm{ST}_2}}{\partial \Omega_n} \middle| \psi_n \right\rangle . \tag{A.8}$$

Again, thanks to the imposed diagonality criterion on the control Hamiltonian, we can use the exponential expansion in Eq. (A.6) (or equivalently the form with the recursive commutator in Eq. (5.33)), to compute the derivatives of $\hat{\mathcal{U}}_n^{c/2}$ as

$$\begin{aligned}
\frac{\partial \hat{\mathcal{U}}_n^{\mathrm{ST}_2}}{\partial \Omega_n} &= \frac{\partial \hat{\mathcal{U}}_n^{c/2}}{\partial \Omega_n}\hat{\mathcal{U}}_n^{d}\hat{\mathcal{U}}_n^{c/2} + \hat{\mathcal{U}}_n^{c/2}\hat{\mathcal{U}}_n^{d}\frac{\partial \hat{\mathcal{U}}_n^{c/2}}{\partial \Omega_n} \\
&= -\frac{i\delta t}{2}\left(\hat{\mathcal{U}}_n^{c/2}\frac{\partial \hat{H}_n^c}{\partial \Omega_n}\hat{\mathcal{U}}_n^{d}\hat{\mathcal{U}}_n^{c/2} + \hat{\mathcal{U}}_n^{c/2}\hat{\mathcal{U}}_n^{d}\hat{\mathcal{U}}_n^{c/2}\frac{\partial \hat{H}_n^c}{\partial \Omega_n}\right) \\
&= -\frac{i\delta t}{2}\left(\frac{\partial \hat{H}_n^c}{\partial \Omega_n}\hat{\mathcal{U}}_n^{\mathrm{ST}_2} + \hat{\mathcal{U}}_n^{\mathrm{ST}_2}\frac{\partial \hat{H}_n^c}{\partial \Omega_n}\right) .
\end{aligned} \tag{A.9}$$

If we substitute back into Eqs. (5.21) and (A.8) we obtain

$$\begin{aligned}
\frac{\partial o}{\partial \Omega_n} &= -\frac{i\delta t}{2}\left\langle \chi_{n+1} \middle| \left(\frac{\partial \hat{H}_n^c}{\partial \Omega_n}\hat{\mathcal{U}}_n^{\mathrm{ST}_2} + \hat{\mathcal{U}}_n^{\mathrm{ST}_2}\frac{\partial \hat{H}_n^c}{\partial \Omega_n}\right) \middle| \psi_n \right\rangle \\
&= -\frac{i\delta t}{2}\left\{\left\langle \chi_{n+1} \middle| \frac{\partial \hat{H}_n^c}{\partial \Omega_n} \middle| \psi_{n+1} \right\rangle + \left\langle \chi_n \middle| \frac{\partial \hat{H}_n^c}{\partial \Omega_n} \middle| \psi_n \right\rangle\right\} \\
&= -\frac{i\delta t}{2}\sum_{p=n}^{n+1}\left\langle \chi_p \middle| \frac{\partial \hat{H}_n^c}{\partial \Omega_n} \middle| \psi_p \right\rangle \\
&\Rightarrow \frac{\partial J_{\mathcal{F}}^{\mathrm{ST}_2}}{\partial \Omega_n} = \mathfrak{Re}\left(\frac{io^\star}{2}\sum_{p=n}^{n+1}\left\langle \chi_p \middle| \frac{\partial H_n^c}{\partial \Omega_n} \middle| \psi_p \right\rangle\right)\delta t ,
\end{aligned} \tag{A.10}$$

for all $n = 1, \ldots, N_t - 1$ which is the expression in Eq. (5.38).

Appendix B

IBM "Washington" processor experimental parameters

We report in Table B.1 the experimental parameters for the 127-qubit IBM "Washington" chip whose map view is shown in Fig. B.1.
$T_{1,2}$ are colloquially known as decoherence times, but slightly more precisely also as the (qubit) relaxation time (T_1) and the (qubit) dephasing time (T_2); f is the frequency and Δ the anharmonicity. R^{err} and R^{len} are the readout error and length respectively. The first is obtained by preparing the qubit in either the 0 or the 1 state and then immediately measuring the qubit state: at this point, two types of readout errors can occur, namely $\mathcal{P}(1|0)$ and $\mathcal{P}(0|1)$ (often referred to as SPAM, state-preparation-and-measurement, errors). R^{err} is the average of the rate of those two errors. R^{len} is instead the time it takes to perform a measurement and constitutes one of the main limiting factors of current quantum hardware, as these times are often considerably longer than T_2, which makes intermediate measurements (i.e. measurements that are not at the end of a circuit) practically unfeasible.
As for the single-qubit σ_x, the idea is to use randomized benchmarking [89], in which sequences of gates (specifically Clifford gates) are applied with the goal of taking the qubit on a random walk among certain points on the Bloch sphere and returning it to the 0 state it started in. As the number of gates in the sequence is increased, the chance of returning to zero drops exponentially and eventually saturates near 50%. The gate error rate is extracted from the fit to this exponential decay. Likewise, the ID error is measure of the error induced by having the qubit idle for a typical gate time, while the $\sqrt{x}$ error is a measure of error induced by applying the gate in question and averaging over all different qubits.

The reason there is no error shown for the σ_z is because there is (virtually) none. This is because the IBM machines don't really implement z rotations - they keep track of the z rotations through software, and update the x (and y) gates accordingly. Effectively, they rotate the x and y axes of the Bloch sphere along the z axis: the σ_x gate is thus actually any rotation along an axis in the $x-y$ plane of the Bloch sphere. By changing the phase of the induced wave on the resonator, one can pick the angle the axis makes with a predetermined reference point, and thus implement both x and y gates.

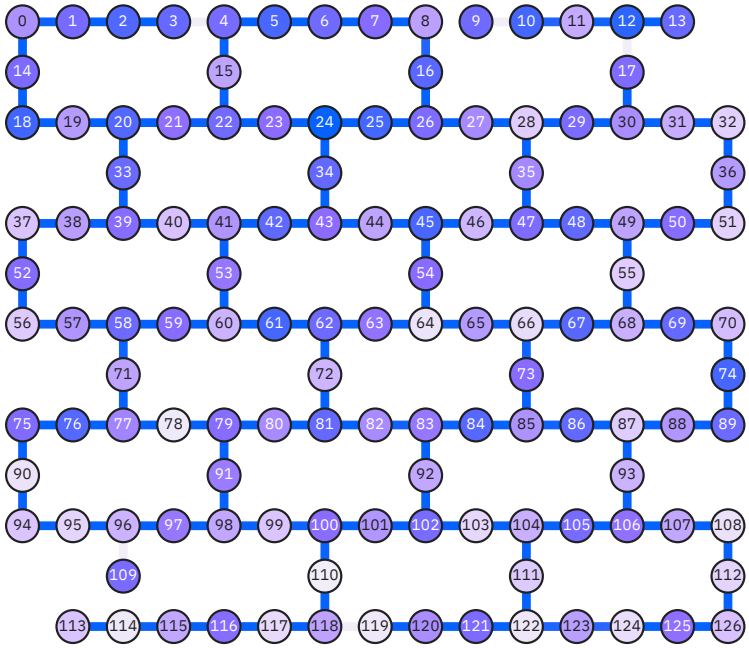

Figure B.1: Layout of the 127-qubit transmon array "Washington". The color code from blue to white indicate the frequency (min. 4.767 GHz, max. 5.292 GHz) for the qubits and the CNOT error (min. $5.721 \cdot 10^{-3}$, max. 1.000) for the bonds. The parameters used for the simulations in this work are taken from the row starting with the 18th to the 32nd qubit.

Qubit	T_1 (µs)	T_2 (µs)	f (GHz)	Δ (GHz)	R^{err}	$\mathcal{P}(0\|1)$	$\mathcal{P}(1\|0)$	R^{len} (ns)	ID error	$\sqrt{x}$ err.	1-qubit σ_x err.
Q18	106.54	138.24	4.866	-0.31125	0.0709	0.0632	0.0786	864	0.0002096	0.0002096	0.0002096
Q19	99.56	127.71	5.104	-0.30768	0.0086	0.0052	0.012	864	0.0001973	0.0001973	0.0001973
Q20	92.51	68.07	4.902	-0.30965	0.0045	0.0084	0.0006	864	0.0003562	0.0003562	0.0003562
Q21	110.19	111.65	5.04	-0.30739	0.0059	0.0098	0.002	864	0.001323	0.001323	0.001323
Q22	104.54	48.49	4.973	-0.30076	0.0149	0.0216	0.0082	864	0.003358	0.003358	0.003358
Q23	96.17	154.48	5.033	-0.30851	0.0591	0.0628	0.0554	864	0.0005689	0.0005689	0.0005689
Q24	119.46	173.91	4.767	-0.30398	0.0574	0.0654	0.0494	864	0.002263	0.002263	0.002263
Q25	100.58	201.45	4.87	-0.31089	0.0078	0.0112	0.0044	864	0.000212	0.000212	0.000212
Q26	85.27	195.63	4.999	-0.30886	0.003	0.0038	0.0022	864	0.0002112	0.0002112	0.0002112
Q27	112.77	51.51	5.077	-0.30693	0.095	0.1014	0.0886	864	0.0002968	0.0002968	0.0002968
Q28	109.43	87.2	5.198	-0.30578	0.0026	0.0034	0.0018	864	0.0002251	0.0002251	0.0002251
Q29	69.2	20.89	4.994	-0.30757	0.0053	0.0078	0.0028	864	0.000246	0.000246	0.000246
Q30	54.34	89.35	5.083	-0.30708	0.0035	0.0066	0.0004	864	0.0002544	0.0002544	0.0002544
Q31	82.56	69.28	5.129	-0.30656	0.0023	0.003	0.0016	864	0.0002657	0.0002657	0.0002657
Q32	110.33	25.97	5.204	-0.30413	0.0088	0.0124	0.0052	864	0.0002015	0.0002015	0.0002015

Table B.1: Experimental parameters as of calibration at the 16th September, 2022. More specifics are available at [78].

Bibliography

[1] R. P. Feynman. "Simulating physics with computers". In: *International Journal of Theoretical Physics* 21.6-7 (June 1982), pp. 467–488. DOI: 10.1007/bf02650179 (cit. on p. 1).

[2] J. Preskill. "The Physics of Quantum Information". In: (Aug. 2022). arXiv: 2208.08064 [quant-ph] (cit. on p. 3).

[3] S. Kwon et al. "Tutorial: Gate-based superconducting quantum computing". In: *Journal of Applied Physics 129, 041102 (2021)* (Sept. 2020). DOI: 10.1063/5.0029735. arXiv: 2009.08021 [quant-ph] (cit. on pp. 3, 8).

[4] L. Hardy. "Quantum Theory From Five Reasonable Axioms". In: (Jan. 2001). arXiv: quant-ph/0101012 [quant-ph] (cit. on p. 5).

[5] M. A. Nielsen et al. *Quantum computation and quantum information - 10. ed.* Cambridge University Press, 2010 (cit. on p. 6).

[6] D. P. DiVincenzo and IBM. "The Physical Implementation of Quantum Computation". In: (Feb. 2000). DOI: 10.1002/1521-3978(200009)48:9/11<771::AID-PROP771>3.0.CO;2-E. arXiv: quant-ph/0002077 [quant-ph] (cit. on p. 6).

[7] A. Acín et al. "The European Quantum Technologies Roadmap". In: *New J. Phys. 20 (2018) 080201* (Dec. 2017). DOI: 10.1088/1367-2630/aad1ea. arXiv: 1712.03773 [quant-ph] (cit. on pp. 6, 37).

[8] D. Castelvecchi. "Quantum computers ready to leap out of the lab in 2017". In: *Nature* 541.7635 (Jan. 2017), pp. 9–10. DOI: 10.1038/541009a (cit. on p. 6).

[9] https://newsroom.ibm.com/2021-11-16-IBM-Unveils-Breakthrough-127-Qubit-Quantum-Processor (cit. on p. 6).

[10] P. Krantz et al. "A Quantum Engineer's Guide to Superconducting Qubits". In: *Applied Physics Reviews 6, 021318 (2019)* (Apr. 2019). DOI: 10.1063/1.5089550. arXiv: 1904.06560 [quant-ph] (cit. on p. 8).

[11] M. Dalgaard et al. "Hessian-based optimization of constrained quantum control". In: *Phys. Rev. A 102, 042612 (2020)* (June 2020). DOI: 10.1103/PhysRevA.102.042612. arXiv: 2006.00935 [quant-ph] (cit. on pp. 8, 23, 24, 57, 59, 64).

[12] E. Magesan and J. M. Gambetta. "Effective Hamiltonian models of the cross-resonance gate". In: *Physical Review A* 101.5 (May 2020), p. 052308. DOI: 10.1103/physreva.101.052308 (cit. on pp. 8, 9).

[13] J. R. Schrieffer and P. A. Wolff. "Relation between the Anderson and Kondo Hamiltonians". In: *Physical Review* 149.2 (Sept. 1966), pp. 491–492. DOI: 10.1103/physrev.149.491 (cit. on p. 9).

[14] Motzoi, Felix. "Controlling Quantum Information Devices". PhD thesis. 2012 (cit. on pp. 11, 14, 32).

[15] D. Zeuch et al. "Exact Rotating Wave Approximation". In: *Annals of Physics 423, 168327 (2020)* (July 2018). DOI: 10.1016/j.aop.2020.168327. arXiv: 1807.02858 [quant-ph] (cit. on p. 16).

[16] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, Ltd, May 2000. DOI: 10.1002/9781118723203 (cit. on p. 20).

[17] J. Nocedal and S. Wright. *Numerical Optimization (Springer Series in Operations Research and Financial Engineering)*. Springer, 2006, p. 664 (cit. on p. 23).

[18] S. Machnes et al. "Comparing, optimizing, and benchmarking quantum-control algorithms in a unifying programming framework". In: *Physical Review A* 84.2 (Aug. 2011), p. 022305. DOI: 10.1103/physreva.84.022305 (cit. on p. 24).

[19] J. R. Johansson, P. D. Nation, and F. Nori. "QuTiP 2: A Python framework for the dynamics of open quantum systems". In: *Comp. Phys. Comm. 184, 1234-1240 (2013)* (Nov. 2012). DOI: 10.1016/j.cpc.2012.11.019. arXiv: 1211.6518 [quant-ph] (cit. on p. 24).

[20] J. Sørensen et al. "QEngine: A C++ library for quantum optimal control of ultracold atoms". In: *Computer Physics Communications* 243 (Oct. 2019), pp. 135–150. DOI: 10.1016/j.cpc.2019.04.020 (cit. on p. 24).

[21] M. H. Goerz and Contributors. *QuantumControl.jl*. Version 0.3.0. Sept. 9, 2022 (cit. on p. 24).

[22] P. de Fouquieres et al. "Second order gradient ascent pulse engineering". In: *Journal of Magnetic Resonance, 212 (2011) 412-417* (Feb. 2011). DOI: 10.1016/j.jmr.2011.07.023. arXiv: 1102.4096 [quant-ph] (cit. on p. 24).

[23] R. H. Byrd et al. "A Limited Memory Algorithm for Bound Constrained Optimization". In: *SIAM Journal on Scientific Computing* 16.5 (Sept. 1995), pp. 1190–1208. DOI: 10.1137/0916069 (cit. on p. 25).

[24] P. Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2 (cit. on pp. 25, 53, 56).

[25] I. The MathWorks. *Symbolic Math Toolbox*. Natick, Massachusetts, United State, 2019 (cit. on p. 25).

[26] P. K. Mogensen and A. N. Riseth. "Optim: A mathematical optimization package for Julia". In: *Journal of Open Source Software* 3.24 (2018), p. 615. DOI: 10.21105/joss.00615 (cit. on p. 25).

[27] J. Haegeman and Contributors. *OptimKit.jl*. Version 0.3.1. Sept. 11, 2020 (cit. on p. 25).

[28] C. Zhu et al. "Algorithm 778: L-BFGS-B". In: *ACM Transactions on Mathematical Software* 23.4 (Dec. 1997), pp. 550–560. DOI: 10.1145/279232.279236 (cit. on p. 25).

[29] J. L. Morales and J. Nocedal. "Remark on "algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization"". In: *ACM Transactions on Mathematical Software* 38.1 (Nov. 2011), pp. 1–4. DOI: 10.1145/2049662.2049669 (cit. on p. 25).

[30] Y. Qi and Contributors. *LBFGSB.jl*. Version 0.4.1. Feb. 2011 (cit. on pp. 25, 64).

[31] A. N. Riseth and Contributors. *LineSearches.jl*. Version 7.2.0. Aug. 24, 2022 (cit. on p. 25).

[32] C. C. Margossian. "A Review of automatic differentiation and its efficient implementation". In: (Nov. 2018). DOI: 10.1002/WIDM.1305. arXiv: 1811.05031 [cs.MS] (cit. on pp. 25, 26).

[33] A. G. Baydin et al. "Automatic differentiation in machine learning: a survey". In: *Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. The Journal of Machine Learning Research, 18(153):1–43, 2018* (Feb. 2015). arXiv: 1502.05767 [cs.SC] (cit. on p. 25).

[34] C. Moler and C. V. Loan. "Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later". In: *SIAM Review* 45.1 (Jan. 2003), pp. 3–49. DOI: 10.1137/s00361445024180 (cit. on p. 26).

[35] M. Schmitt. "Dynamics of isolated quantum many-body systems far from equilibrium". PhD thesis. DOI: 10.53846/goediss-6693 (cit. on p. 26).

[36] W. Hao. "The Krylov Subspace Methods for the Computation of Matrix Exponentials". PhD thesis. 2015 (cit. on p. 26).

[37] M. Hochbruck and C. Lubich. "On Krylov Subspace Approximations to the Matrix Exponential Operator". In: *SIAM Journal on Numerical Analysis* 34.5 (Oct. 1997), pp. 1911–1925. DOI: 10.1137/s0036142995280572 (cit. on p. 28).

[38] W. E. Arnoldi. "The principle of minimized iterations in the solution of the matrix eigenvalue problem". In: *Quarterly of Applied Mathematics* 9.1 (1951), pp. 17–29. DOI: 10.1090/qam/42792 (cit. on p. 28).

[39] C. Lanczos. "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators". In: *Journal of*

Research of the National Bureau of Standards 45.4 (Oct. 1950), p. 255. DOI: 10.6028/jres.045.026 (cit. on p. 28).

[40] https://weinbe58.github.io/QuSpin/ (cit. on p. 29).

[41] J. Haegeman and Contributors. *KrylovKit.jl*. Version 0.5.4. Apr. 2012 (cit. on p. 29).

[42] F. K. Wilhelm et al. "An introduction into optimal control for quantum technologies". In: (Mar. 2020). arXiv: 2003.10132 [quant-ph] (cit. on p. 31).

[43] T. S. Mahesh, P. Batra, and M. H. Ram. "Quantum Optimal Control: Practical Aspects and Diverse Methods". In: (May 2022). arXiv: 2205.15574 [quant-ph] (cit. on p. 31).

[44] N. Khaneja et al. "Optimal control of coupled spin dynamics: design of NMR pulse sequences by gradient ascent algorithms". In: *Journal of Magnetic Resonance* 172.2 (Feb. 2005), pp. 296–305. DOI: 10.1016/j.jmr.2004.11.004 (cit. on pp. 32, 35, 36).

[45] D. D'Alessandro. *Introduction to Quantum Control and Dynamics*. Chapman and Hall/CRC, July 2021. DOI: 10.1201/9781003051268 (cit. on p. 32).

[46] S. G. Schirmer, H. Fu, and A. I. Solomon. "Complete controllability of quantum systems". In: *Physical Review A* 63.6 (May 2001), p. 063410. DOI: 10.1103/physreva.63.063410 (cit. on p. 32).

[47] S. G. Schirmer, A. I. Solomon, and J. V. Leahy. "Degrees of controllability for quantum systems and application to atomic systems". In: *Journal of Physics A: Mathematical and General* 35.18 (Apr. 2002), pp. 4125–4141. DOI: 10.1088/0305-4470/35/18/309 (cit. on p. 33).

[48] C. Brif, R. Chakrabarti, and H. Rabitz. "Control of quantum phenomena: past, present and future". In: *New Journal of Physics* 12.7 (July 2010), p. 075008. DOI: 10.1088/1367-2630/12/7/075008 (cit. on p. 33).

[49] J. Zoller. "Optimal quantum engineering". en. PhD thesis. 2018. DOI: 10.18725/OPARU-10199 (cit. on p. 33).

[50] T. Kato. "On the Adiabatic Theorem of Quantum Mechanics". In: *Journal of the Physical Society of Japan* 5.6 (Nov. 1950), pp. 435–439. DOI: 10.1143/jpsj.5.435 (cit. on p. 33).

[51] L. Mandelstam and I. Tamm. "The Uncertainty Relation Between Energy and Time in Non-relativistic Quantum Mechanics". In: *Selected Papers*. Springer Berlin Heidelberg, 1991, pp. 115–123. DOI: 10.1007/978-3-642-74626-0_8 (cit. on p. 33).

[52] N. Margolus and L. B. Levitin. "The maximum speed of dynamical evolution". In: *Physica D: Nonlinear Phenomena* 120.1-2 (Sept. 1998), pp. 188–195. DOI: 10.1016/s0167-2789(98)00054-2 (cit. on p. 33).

[53] T. Caneva et al. "Optimal Control at the Quantum Speed Limit". In: *Phys. Rev. Lett. 103, 240501 (2009)* (Feb. 2009). DOI: 10.1103/

PhysRevLett.103.240501. arXiv: 0902.4193 [quant-ph] (cit. on pp. 34, 54, 59).

[54] S. G. Schirmer and P. de Fouquieres. "Efficient Algorithms for Optimal Control of Quantum Dynamics: The "Krotov" Method unencumbered". In: *New Journal of Physics 13 (7), 073029 (2011)* (Mar. 2011). DOI: 10.1088/1367-2630/13/7/073029. arXiv: 1103.5435 [quant-ph] (cit. on p. 34).

[55] M. D. Bowdrey et al. "Fidelity of single qubit maps". In: *Physics Letters A* 294.5-6 (Mar. 2002), pp. 258–260. DOI: 10.1016/s0375-9601(02)00069-5 (cit. on p. 34).

[56] L. H. Pedersen, K. Molmer, and N. M. Moller. "Fidelity of quantum operations". In: *Phys. Lett. A 367 (2007), no. 1-2, 47–51.* (Jan. 2007). DOI: 10.1016/j.physleta.2007.02.069. arXiv: quant-ph/0701138 [quant-ph] (cit. on p. 35).

[57] P. Rebentrost and F. K. Wilhelm. "Optimal control of a leaking qubit". In: *Phys. Rev. B 79, 060507(R) (2009)* (Aug. 2008). DOI: 10.1103/PhysRevB.79.060507. arXiv: 0808.2680 [quant-ph] (cit. on p. 36).

[58] D. J. Tannor, V. Kazakov, and V. Orlov. "Control of Photochemical Branching: Novel Procedures for Finding Optimal Pulses and Global Upper Bounds". In: *Nato ASI Series*. Springer US, 1992, pp. 347–360. DOI: 10.1007/978-1-4899-2326-4_24 (cit. on p. 37).

[59] W. Zhu and H. Rabitz. "A rapid monotonically convergent iteration algorithm for quantum optimal control over the expectation value of a positive definite operator". In: *The Journal of Chemical Physics* 109.2 (July 1998), pp. 385–391. DOI: 10.1063/1.476575 (cit. on p. 37).

[60] S. J. Glaser et al. "Training Schrödinger's cat: quantum optimal control". In: *Eur. Phys. J. D 69, 279 (2015)* (Aug. 2015). DOI: 10.1140/epjd/e2015-60464-1. arXiv: 1508.00442 [quant-ph] (cit. on p. 37).

[61] J. H. M. Jensen et al. "Approximate dynamics leading to more optimal control: Efficient exact derivatives". In: *Physical Review A* 103.6 (June 2021), p. 062612. DOI: 10.1103/physreva.103.062612 (cit. on pp. 37, 38, 41, 42, 53, 56, 70, 71, 73, 75, 79).

[62] M. Suzuki. "Generalized Trotter's formula and systematic approximants of exponential operators and inner derivations with applications to many-body problems". In: *Communications in Mathematical Physics* 51.2 (June 1976), pp. 183–190. DOI: 10.1007/bf01609348 (cit. on pp. 40, 50).

[63] N. Hatano and M. Suzuki. "Finding Exponential Product Formulas of Higher Orders". In: *"Quantum Annealing and Other Optimization Methods," Eds. A. Das and B.K. Chakrabarti (Springer, Berlin, 2005) pp. 37-68* (June 2005). DOI: 10.1007/11526216_2. arXiv: math-ph/0506007 [math-ph] (cit. on p. 40).

[64] E. M. Stoudenmire and D. J. Schwab. "Supervised Learning with Quantum-Inspired Tensor Networks". In: *Advances in Neural Infor-*

mation Processing Systems 29, 4799 (2016) (May 2016). arXiv: 1605.05775 [stat.ML] (cit. on p. 45).

[65] S. Montangero. *Introduction to Tensor Network Methods*. Springer International Publishing, 2018. DOI: 10.1007/978-3-030-01409-4 (cit. on p. 45).

[66] D. Perez-Garcia et al. "Matrix Product State Representations". In: *Quantum Inf. Comput. 7, 401 (2007)* (Aug. 2006). arXiv: quant-ph/0608197 [quant-ph] (cit. on p. 45).

[67] J. H. M. Jensen et al. "Achieving fast high-fidelity optimal control of many-body quantum dynamics". In: *Physical Review A* 104.5 (Nov. 2021), p. 052210. DOI: 10.1103/physreva.104.052210 (cit. on pp. 45, 49–51).

[68] https://tensornetwork.org/ (cit. on p. 45).

[69] R. Penrose. "Applications of negative dimensional tensors". In: *Combinatorial mathematics and its applications* 1 (1971), pp. 221–244 (cit. on p. 46).

[70] C. Hubig, I. P. McCulloch, and U. Schollwöck. "Generic construction of efficient matrix product operators". In: *Physical Review B* 95.3 (Jan. 2017), p. 035129. DOI: 10.1103/physrevb.95.035129 (cit. on p. 49).

[71] A. J. Daley et al. "Time-dependent density-matrix renormalization-group using adaptive effective Hilbert spaces". In: *J. Stat. Mech.: Theor. Exp. (2004) P04005* (Mar. 2004). DOI: 10.1088/1742-5468/2004/04/P04005. arXiv: cond-mat/0403313 [cond-mat.str-el] (cit. on p. 49).

[72] S. Paeckel et al. "Time-evolution methods for matrix-product states". In: *Annals of Physics 411, 167998 (2019)* (Jan. 2019). DOI: 10.1016/j.aop.2019.167998. arXiv: 1901.05824 [cond-mat.str-el] (cit. on p. 49).

[73] P. Silvi et al. "The Tensor Networks Anthology: Simulation techniques for many-body quantum lattice systems". In: *SciPost Phys. Lect. Notes 8 (2019)* (Oct. 2017). DOI: 10.21468/SciPostPhysLectNotes.8. arXiv: 1710.03733 [quant-ph] (cit. on p. 52).

[74] C. R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2 (cit. on p. 53).

[75] G. C. Hegerfeldt. "Driving at the quantum speed limit: Optimal control of a two-level system". In: *Physical Review Letters 111, 260501 (2013)* (May 2013). DOI: 10.1103/PhysRevLett.111.260501. arXiv: 1305.6403 [quant-ph] (cit. on p. 54).

[76] S. Kirchhoff et al. "Optimized cross-resonance gate for coupled transmon systems". In: *Physical Review A* 97.4 (Apr. 2018), p. 042348. DOI: 10.1103/physreva.97.042348 (cit. on p. 57).

[77] M. Dalgaard et al. "Global optimization of quantum dynamics with AlphaZero deep exploration". In: *npj Quantum Information* 6.1 (Jan. 2020). DOI: 10.1038/s41534-019-0241-0 (cit. on p. 57).

[78] https://quantum-computing.ibm.com/services/resources (cit. on pp. 57, 62, 64, 85).

[79] C. Berke et al. "Transmon platform for quantum computing challenged by chaotic fluctuations". In: *Nature Communications 13, 2495 (2022)* (Dec. 2020). DOI: 10.1038/s41467-022-29940-y. arXiv: 2012.05923 [quant-ph] (cit. on p. 64).

[80] M. Fishman, S. R. White, and E. M. Stoudenmire. *The ITensor Software Library for Tensor Network Calculations*. 2020. arXiv: 2007.14822 (cit. on p. 64).

[81] H.-J. Liao et al. "Differentiable Programming Tensor Networks". In: *Phys. Rev. X 9, 031041 (2019)* (Mar. 2019). DOI: 10.1103/PhysRevX.9.031041. arXiv: 1903.09650 [cond-mat.str-el] (cit. on p. 65).

[82] G. B. Mbeng, A. Russomanno, and G. E. Santoro. "The quantum Ising chain for beginners". In: (Sept. 2020). arXiv: 2009.09208 [quant-ph] (cit. on p. 66).

[83] M. J. Innes and Contributors. *Zygote.jl*. Version 0.6.49. Sept. 21, 2022 (cit. on p. 66).

[84] J. Revels and Contributors. *BenchmarkTools.jl*. Version 1.3.1. Feb. 12, 2022 (cit. on p. 68).

[85] M. H. Goerz, S. C. Carrasco, and V. S. Malinovsky. "Quantum Optimal Control via Semi-Automatic Differentiation". In: (May 2022). arXiv: 2205.15044 [quant-ph] (cit. on pp. 69, 78).

[86] A. H. Al-Mohy and N. J. Higham. "A New Scaling and Squaring Algorithm for the Matrix Exponential". In: *SIAM Journal on Matrix Analysis and Applications* 31.3 (Jan. 2010), pp. 970–989. DOI: 10.1137/09074721x (cit. on p. 73).

[87] http://eprints.ma.man.ac.uk/1591/ (cit. on p. 73).

[88] https://www.maths.uq.edu.au/expokit/ (cit. on p. 73).

[89] E. Magesan, J. M. Gambetta, and J. Emerson. "Characterizing Quantum Gates via Randomized Benchmarking". In: *Phys. Rev. A 85, 042311 (2012)* (Sept. 2011). DOI: 10.1103/PhysRevA.85.042311. arXiv: 1109.6887 [quant-ph] (cit. on p. 83).