THE UNIVERSITY OF CHICAGO

# QUANTUM OPTIMAL CONTROL USING AUTOMATIC DIFFERENTIATION

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF PHYSICS

BY

MOHAMED RAGAB ABDELHAFEZ

CHICAGO, ILLINOIS
AUGUST 2019

To Sara, Laila and all future additions to the family.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ACKNOWLEDGMENTS

My PhD journey at the University of Chicago has been very rich, fruitful but also challenging. I have been very lucky to have many people who supported me emotionally and intellectually. I am thankful for getting the chance to interact and learn from many knowledgeable and kind figures who shaped my thinking and my future.

First and foremost, I am forever grateful for my advisor Jens Koch. I have been struggling to find a matching advisor for almost two years then was very humbled to meet him. I knew immediately that I would be lucky to work with him. Accepting to supervise and guide a PhD student from afar is certainly challenging, but Jens was very welcoming to supervise my work. Jens is one of the founders and pioneers of superconducting qubits and he is always willing to explain some pieces of his knowledge every time we meet. I come out of every meeting with him with more solid understanding of the field. He was always very patient and understanding to all the challenges I am facing. The number of things I learned from him is uncountable. To mention a few, I learned how to organize my thoughts in a structured way, how to identify the assumptions I am unconsciously making and how to present my work in a clear and interesting format. His attention to detail is second to none, and I have been very blessed to have him guiding, motivating and editing my ideas and writing.

In addition, I would like to thank my co-advisor, David Schuster. Like Jens, Dave is one of the pioneers in the field of circuit QED. His passion to invent new techniques and applications in the lab always inspired me. He always has new brilliant ideas that push the field forward and I was lucky to work on some of his ideas.

I would like to also thank my fellow PhD students in Schuster Lab and Koch research group. Nelson Leung has been an inspiration to me with how much coding experience he has and how many interesting ideas he can work on simultaneously. He is also always willing to help and explain all the details I need in depth. In addition, working with Brian Baker has been really fun. We think together, implement together, struggle together and then finally

celebrate the results together. He was also always patient and dedicated in periods when I had to travel a lot or focus on thesis writing.

Special thanks goes to Peter Groszkowski, Andy Li, Ziwen Huang, Danny Weiss, Xinyuan You and Nick Irons for the numerous conversations that helped me throughout my PhD. It has been a real pleasure working and discussing physics with you all!

Finally, I would like to thank my family for the continuous support. My father Ragab Abdelhafez, mother Nadia Hamdy, brother Ahmed Ragab and sister Bassant Ragab always believed in me and pushed me forward all the way from Egypt. The real heroes of my PhD are my wife Sara Ali and daughter Laila who endured all the hard times with me and never left my side. Everything that I do in my life is powered by their love, and everything I do in my life is purely aiming for their happiness.

# ABSTRACT

We implement quantum optimal control algorithms for closed and open quantum systems based on automatic differentiation. Automatic differentiation allows us to specify advanced optimization criteria and incorporate them in the optimization process with ease. We show that the use of GPUs can speed up calculations by more than an order of magnitude. Our strategy facilitates efficient numerical simulations on affordable desktop computers, and exploration of a host of optimization constraints and system parameters relevant to real-life experiments. We demonstrate optimization of closed quantum evolution based on fine-grained evaluation of performance at each intermediate time step, thus enabling more intricate control on the evolution path, suppression of departures from the truncated model subspace, as well as minimization of the physical time needed to perform high-fidelity state preparation and unitary gates. The optimizer is also used to find a universal set of gates for the protected superconducting qubits of fluxonium and 0-$\pi$. The optimization algorithm for open quantum systems utilizes quantum trajectories. Using trajectories allows for optimizing open systems with less computational cost than the regular density-matrix approaches in most realistic optimization problems. We introduce an improved-sampling algorithm which minimizes the number of trajectories needed per optimization iteration. Together with employing stochastic gradient descent techniques, this reduces the complexity of optimizing many realistic open quantum systems to the complexity encountered with closed systems. We utilize the optimizer in a variety of applications to demonstrate how the use of quantum trajectories significantly reduces the memory requirements while achieving a multitude of simultaneous optimization targets. Demonstrated targets include high state-transfer fidelities despite dissipation, faster gate times and maximization of qubit-readout fidelity while maintaining the quantum non-demolition nature of the measurement and allowing for subsequent fast resonator reset.

# CHAPTER 1

# INTRODUCTION

In this chapter, we give a brief overview of the main promises and challenges of quantum computing. While modern technology has been successfully progressing using the principles of classical computing, there is a strong need for a new way of processing information. Quantum computing is the main promising alternative which offers significant speed up over classical computing for some computing problems.

## 1.1 Classical computing

One major foundation of computing is representing all information in binary format. This means that the fundamental unit of information (a bit) can only have one of two values, usually called 0 and 1. Different computing schemes differ in how the 0's and 1's are represented. A classical computer uses voltages flowing through circuits as the bit of information, where no voltage means a logical value of 0, and some fixed non-zero voltage means a logical value of 1. Circuit gates are then used to change the stored information from 0 to 1 and vice versa. These circuits can be fully analyzed using the rules of classical physics.

The advancement in technology using classical computing has been relying on reducing the size of the used circuits such that one can fit more memory and processing power on smaller chips. This gave rise to Moore's law in 1965 [90] which is the observation/prediction that the number of transistors on a processor chip will double every 18 months. This exponential reduction in transistor dimensions has held true for decades as shown in Fig. 1.1.

The current semiconductors used in classical computers are starting to shrink to dimensions under 10 nanometers, which means the transistors may only be a dozen atoms thick. As the size of circuit elements is further decreased, quantum mechanical effects can not

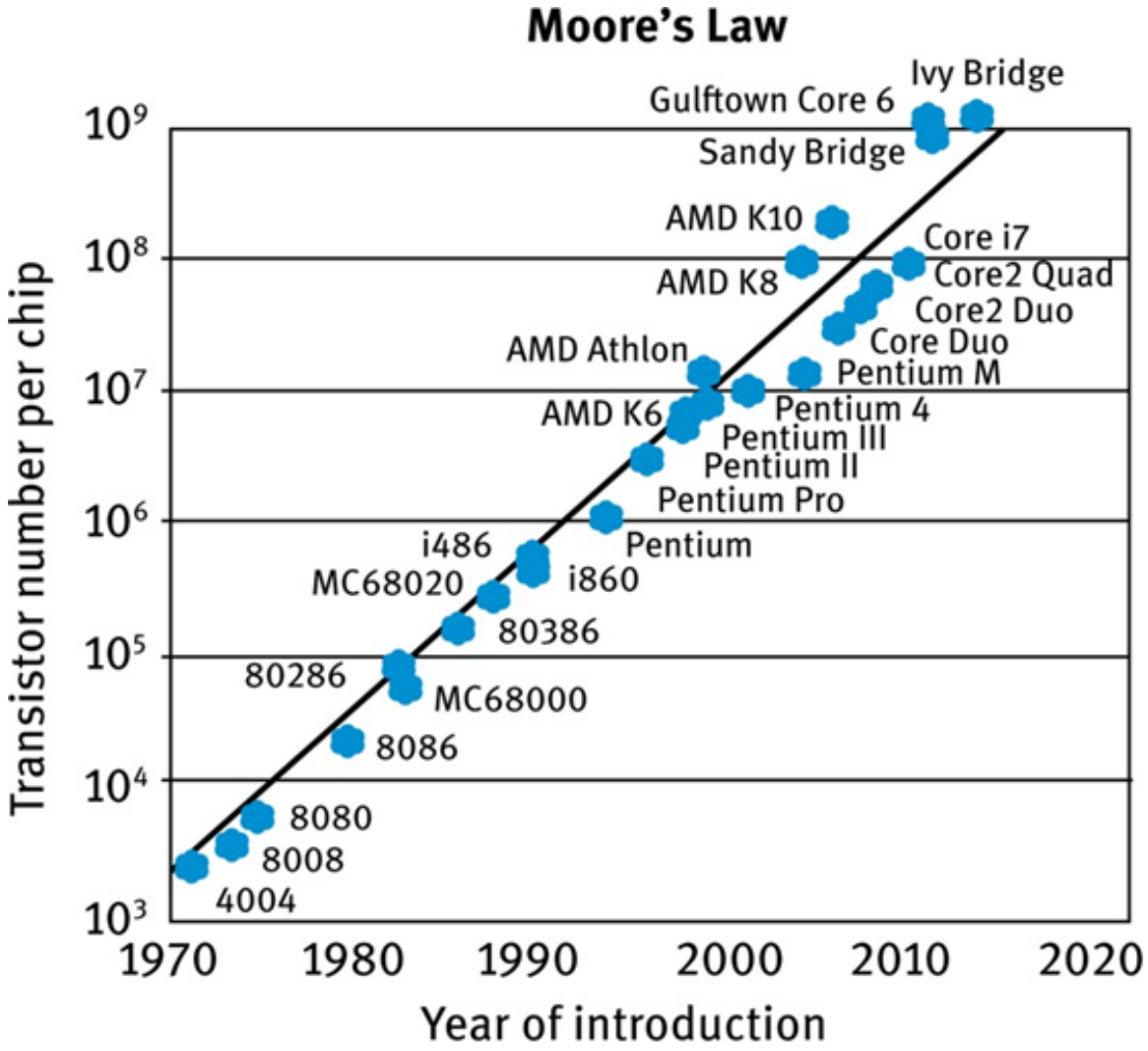


Figure 1.1: The advancement of classical computing processors following Moore's law. The number of transistors on a chip roughly doubles every two years. This is correlated to the exponential reduction of the dimensions of transistors which causes quantum phenomena to affect classical assumptions. Figure is from ref [72].

be ignored. Quantum processes (e.g. tunneling) are essential to consider in that regime and hence the classical treatment of processors will not be valid. Therefore, an alternative strategy, that needs to accommodate the rules of quantum mechanics, must be followed.

## 1.2 Quantum computing

Quantum mechanics is the set of physics laws that describes the behavior of small objects. In general, any system that has small dimensions (comparable to the atomic dimensions) is observed to behave in a very different way from all classical rules. These different characteristics of quantum system are not only the reason why classical computers fail to perform at small dimensions but they also can be harnessed to achieve a great level of computing parallelism.

### *1.2.1 Quantum bits*

In quantum mechanics, there are a set of orthonormal states (eigenstates of some measurement operator) $\{|i\rangle\}$ that the quantum system can occupy, with quantized energy values $E_i$. These eigenstate constitute a complete basis set for all possible states of the system with orthonormality defined as $\langle i|\, j\rangle = \delta_{ij}$. The generic state of the system before measurement is mathematically represented by a ket $|\psi\rangle$ that could be a linear superposition of all the eigenstates.

$$|\psi\rangle = \sum_i \alpha_i \, |i\rangle \tag{1.1}$$

The superposition coefficients $\alpha_i$ are complex numbers that satisfy the normalization condition

$$\sum_i |\alpha_i|^2 = 1 \tag{1.2}$$

This condition is a result of $|\alpha_i|^2$ being the probability of finding the system in eigenstate $|i\rangle$, so the probabilities must sum to one. When the quantum system is measured, its state

collapses to one of the eigenstates of the measurement operator $|i\rangle$ with probabilities $|\alpha_i|^2$.

Quantum computing utilizes two of the eigenstates of a quantum system to represent the computational $|0\rangle$ and $|1\rangle$, calling the quantum system a quantum bit (qubit). A qubit is then fundamentally different from a classical bit since one qubit can be in a superposition of both 0 and 1 at the same time, while a classical bit is either a 0 or 1. Similarly, for n qubits, the general state of the system can be a superposition of $2^n$ computational states at the same time. Meanwhile, n classical bits store only 1 of those $2^n$ choices. This ability to include exponentially more information in the state of a quantum computer is harnessed by several quantum algorithms to perform certain computing tasks much faster than any classical computer.

It is important to note that a superposition of eigenstates will always collapse to a single eigenstate upon measurement, i.e. the parallel inclusion of information returns back to a definite classical-like state. Therefore, it is important for quantum algorithms to perform operations on the state of the quantum system before measuring it. These operations are represented mathematically by unitary transformations (gates) that transform the quantum state while keeping the normalization condition valid,

$$U\,|\psi\rangle = U\big[\sum_i \alpha_i\,|i\rangle\,\big] = \sum_i \alpha_i'\,|i\rangle\,, \quad \sum_i |\alpha_i'|^2 = 1. \tag{1.3}$$

To illustrate the concept of unitary transformations needed by a quantum algorithm, we next present a brief overview of Grover's algorithm for searching.

### *1.2.2 Grover's search algorithm*

Suppose we have $N = 2^n$ entries in a database and we need to search for a certain entry $w$ in it. This entry is defined by the function, $f(i) = \delta_{iw}$ that highlights whether we found the entry or not, where $i$ and $w$ are binary sequences of $n$ bits each. One first unitary gate

that is needed is a way of querying this function. The corresponding unitary transformation, known as the oracle, is represented by the operation

$$U_w |i\rangle = (-1)^{f(i)} |i\rangle . \tag{1.4}$$

If represented as an $N \times N$ matrix, $U_w$ is a diagonal matrix with $-1$ as the $w$-th entry and $+1$ elsewhere. Therefore, it can also be written as

$$U_w = \mathbb{1} - 2 |w\rangle \langle w| . \tag{1.5}$$

The second gate needed in Grover's algorithm is known as Grover's diffusion gate. It is defined as

$$U_s = 2 |s\rangle \langle s| - \mathbb{1}, \tag{1.6}$$

where $|s\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle$ is the state of equal superposition of all $N$ entries.

The algorithm uses both gates in the following steps [52]:

- Prepare the initial equal superposition state $|s\rangle$. With all $n$ qubits usually starting in their ground states, the Hadamard gate is used to create the equal superposition. The Hadamard gate is defined by its matrix elements $H_{ij} = \frac{1}{\sqrt{2^n}}(-1)^{i.j}$ where $i.j$ is the bit-wise dot product of binary sequences $i$ and $j$.
- Perform $\mathcal{O}(\sqrt{N})$ iterations of:
  - Apply $U_w$.
  - Apply $U_s$.
- Measure the final state.

The effect of every iteration of applying $U_s U_w$ is called amplitude amplification where the amplitude of state $|w\rangle$ is amplified in the superposition and all other amplitudes are

weakened. To see that, consider the effect of the first iteration

$$\begin{aligned} U_w |s\rangle &= \left[\mathbb{1} - 2 |w\rangle \langle w| \right] |s\rangle = |s\rangle - \frac{2}{\sqrt{N}} |w\rangle \\ U_s U_w |s\rangle &= \left[2 |s\rangle \langle s| - \mathbb{1}\right] \left(|s\rangle - \frac{2}{\sqrt{N}} |w\rangle\right) = 2 |s\rangle - |s\rangle - \frac{4}{N} |s\rangle + \frac{2}{\sqrt{N}} |w\rangle \\ &= \frac{N-4}{N} |s\rangle + \frac{2}{\sqrt{N}} |w\rangle . \end{aligned} \tag{1.7}$$

Therefore, the coefficients of all states are weakened by a factor of $\frac{N-4}{N}$ while only the coefficient of $|w\rangle$ has an increase of $\frac{2}{\sqrt{N}}$. Therefore, after the first iteration, the system has an increased chance to be measured in the correct $|w\rangle$ state.

Careful analysis of the amplitudes after every iteration [93] show that only $\mathcal{O}(\sqrt{N})$ iterations are needed for optimal results. This gives a quadratic speed up over classical search algorithms which need $\mathcal{O}(N)$ queries. Note that for the case where $n = 2$, $N = 4$, the amplitude weakening factor $\frac{N-4}{N}$ after the first iteration is exactly 0 meaning that we find the solution with just one query.

### *1.2.3 Shor's algorithm*

Another very important quantum algorithm is Shor's algorithm [112] for factoring large integers. The algorithm achieves exponential speed-up over all classical factoring algorithms. For an integer $N$, Shor's algorithm finds the prime factors with a time-complexity $\mathcal{O}(\text{poly}(\log N))$. On the other hand, the best classical factoring algorithm (the general number field sieve) has time-complexity of $\mathcal{O}(\exp(\text{poly}(\log N)))$. The widely used public-key cryptography scheme RSA is based on the practical impossibility of obtaining factors of large integers (typically integers with over 200 digits). Therefore, quantum computing promises to break the current RSA cryptography code.

As evident in Fig. 1.2, the number of operations needed by a classical computer to break the RSA scheme with the record of 230 digits is of order $10^{25}$. Meanwhile, a quantum

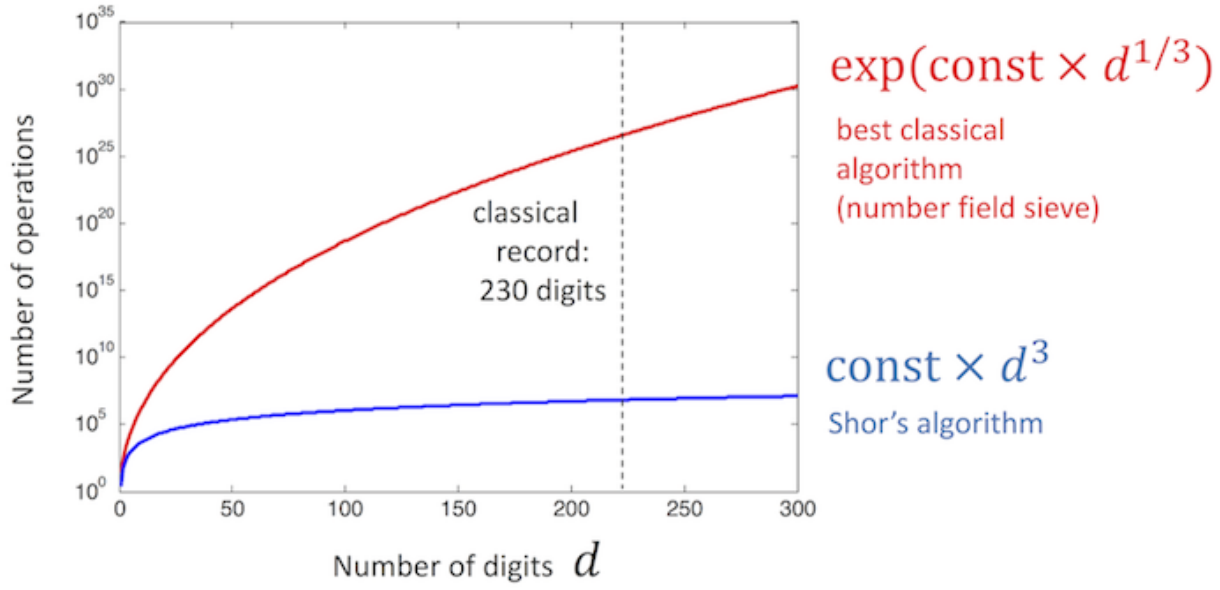


Figure 1.2: The number of operations required by classical and quantum computers to factor an integer of $d$ digits. Shor's algorithm reduces the exponential complexity to polynomial in the number of digits. The figure is taken from ref. [108].

computer reduces that significantly to order $10^5$ operations with the help of Shor's algorithm [108].

The full details of the algorithm are well presented in ref. [93]. The main point to emphasize is that again a sequence of unitary gates act on the initial equal superposition of the $N$ eigenstates. One crucial unitary gate for Shor's algorithm is the quantum Fourier transform (QFT). QFT is defined as

$$\text{QFT}\,|j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j k/N}\,|k\rangle\,. \tag{1.8}$$

In summary, quantum algorithms rely on using unitary gates to initialize and modify the state of the qubits. Hence, all speed-up promises of quantum computing depend heavily on the ability to perform gates reliably.

### *1.2.4 Universality of quantum gates*

We have encountered several different unitary gates on $n$ qubits in the implementation of quantum algorithms. One way to construct these gates is by using many gates on smaller number of qubits as building blocks. As shown in ref. [93], all quantum gates on any number of qubits can be approximated up to arbitrary precision using a finite set of gates. One choice of the building blocks includes two single-qubit gates: T and H and one two-qubit gate: CNOT. We define the gates with their matrix elements in the qubit(s) space as:

1. Single-qubit Hadamard gate:

$$\mathrm{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{1.9}$$

   The Hadamard gate is usually used to transform the qubit from $|0\rangle$ or $|1\rangle$ to the equal superposition of both and vice versa.

2. Single-qubit T gate:

$$\mathrm{T} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}. \tag{1.10}$$

   The T gate is used to create a local phase shift of $\frac{\pi}{4}$ between the $|0\rangle$ and $|1\rangle$ coefficients in a superposition.

3. Two-qubit CNOT gate

$$\mathrm{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{1.11}$$

The CNOT gate (also known as controlled-NOT or controlled-X gate) is one way of coupling/entangling two qubits. Based on the value of the first (control) qubit, the second qubit will either remain the same or flips from 0 to 1 and vice versa.

This thesis focuses on finding practical ways for implementing these gates in a certain implementation of qubits. While it is sufficient to implement the universal set of gates described above, in most practical cases, one can find faster ways to construct the full gate at once. Therefore, we will look at a variety of gates in this thesis.

## 1.3 Thesis Overview

In the first chapter, we have summarized the main differences between classical and quantum computing. We have also highlighted the main unitary gates needed for quantum algorithms. Chapter 2 describes the theory of implementing a quantum bit using superconducting circuits. Next, Chapter 3 presents the details of optimal control theory for closed quantum systems and the main targets of optimization. Chapter 4 introduces the concept of automatic differentiation that will be used to implement a flexible and fast optimal-control interface. The implementation details of this interface are presented in Chapter 5. Chapter 6 discusses several examples of the use of our implementation to perform different gates and state transfers in superconducting qubits. After that, Chapter 7 discusses the effects of the environment on open quantum systems and how to model them efficiently. Next, Chapter 8 gives an overview of existing optimal control theory techniques for open systems. Chapter 9 discusses the implementation of an automatic-differentiation based optimizer for open quantum systems. Finally, Chapter 10 discusses practical applications where the relaxation and dephasing effects on superconducting qubits are suppressed using our optimizer.

Sections 3.4, 3.5, 5.1-4 and 6.1 are closely based on our paper, ref. [77]. In addition, chapters 9 and 10 are closely based on our work in ref. [2].

# CHAPTER 2

# SUPERCONDUCTING QUBITS

In this chapter, we discuss the theory behind using superconducting circuits as quantum bits. While there are many quantum systems that are good candidates for realization of qubits, superconducting circuits are one of the most popular and promising architectures. There are many industrial efforts to realize quantum computing via superconducting circuits including those by IBM, Intel, Google and many other companies and research groups. While some other architectures use natural atoms, superconducting qubits can be engineered to have designed properties. Therefore, they are called artificial atoms. The fabrication of such circuits is also very similar to existing semiconductor fabrication techniques. We will focus in this thesis on applications of superconducting qubits since they are fabricated in our lab, Schuster Lab. However, optimal control theory and the optimizer we built can be used to control a broad variety of quantum systems.

To describe the different superconducting qubits, we first review a simpler basic quantum system: the quantum harmonic oscillator.

## 2.1 Quantum harmonic oscillator

In classical mechanics, a point mass that resides in a region where the potential energy is quadratic in position undergoes simple harmonic motion. The Hamiltonian of that classical 1D system can be written in terms of two conjugate variables: position $x$ and momentum $p$ as

$$H = \frac{p^2}{2m} + \frac{1}{2}kx^2 \tag{2.1}$$

where $m$ is the mass of the particle and $k$ is the spring constant. Following the classical equations of motion, we find that $a = -\omega^2 x$, where $\omega = \sqrt{\frac{k}{m}}$ is the angular frequency of the oscillation.

To treat such system quantum mechanically, we must promote the conjugate variables $x$ and $p$ to operators that obey the commutation relation

$$[x, p] = i\hbar. \tag{2.2}$$

We define the operator $a$ and its adjoint operator $a^\dagger$ as

$$\begin{aligned} a &= \sqrt{\frac{m\omega}{2\hbar}}\big(x + \frac{ip}{m\omega}\big) \\ a^\dagger &= \sqrt{\frac{m\omega}{2\hbar}}\big(x - \frac{ip}{m\omega}\big). \end{aligned} \tag{2.3}$$

The Hamiltonian can then be expressed in terms of $a$ and $a^\dagger$ as

$$H = \hbar\omega(a^\dagger a + \frac{1}{2}), \tag{2.4}$$

where the commutation relation between the two operators $a$ and $a^\dagger$ becomes

$$[a, a^\dagger] = 2 \times \frac{m\omega}{2\hbar}\big[x, \frac{-ip}{m\omega}\big] = -i \times i = 1. \tag{2.5}$$

So, if we label the eigenstates of the Hamiltonian (and hence of the operator $N = a^\dagger a$) as $|n\rangle$ where $n$ can be any real number for now, with energies $E_n = \hbar\omega(n + \frac{1}{2})$, we find that

$$\begin{aligned} (a^\dagger a)a\,|n\rangle &= \big(a^\dagger aa + aa^\dagger a - aa^\dagger a\big)\,|n\rangle \\ &= \big([a^\dagger, a]a + aa^\dagger a\big)\,|n\rangle = (n-1)a\,|n\rangle \end{aligned} \tag{2.6}$$

Similarly, $(a^\dagger a)a^\dagger\,|n\rangle = (n+1)a^\dagger\,|n\rangle$.

Therefore, we see that both $a\,|n\rangle$ and $a^\dagger\,|n\rangle$ are still eigenstates of the number operator

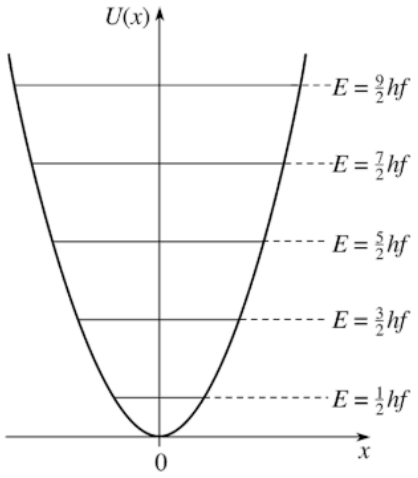


Figure 2.1: The quadratic potential energy function of a quantum harmonic oscillator with energy levels equally spaced. The energy spacing between consecutive levels is $\hbar\omega = hf$.

$a^\dagger a$ with eigenvalues $n-1$ and $n+1$ respectively. This leads to the realization that

$$a\left|n\right\rangle = c_1\left|n-1\right\rangle, \qquad a^\dagger\left|n\right\rangle = c_2\left|n+1\right\rangle \tag{2.7}$$

where $c_1$ and $c_2$ are constants that can be evaluated using the normalization condition to be $c_1 = \sqrt{n-1}$ and $c_2 = \sqrt{n}$. Therefore the operators $a$ and $a^\dagger$ are usually called the lowering and raising operators, respectively.

Finally, to ensure that the system is physical, one can not keep applying the lowering operator indefinitely (leading to energies approaching $-\infty$). This adds the restriction that $n$ must be an integer with lowest values of $n = 0$ for the ground state. In summary, a quantum harmonic oscillator has discrete energy levels labeled by $|n\rangle$ and corresponding energies of $\hbar\omega(n+\frac{1}{2})$ where $n = 0, 1, 2, \ldots$ We note that the energy levels are separated by a constant energy difference of $\hbar\omega$ between consecutive levels as shown in Fig. 2.1. Therefore, we call the quantum harmonic oscillator a linear system. To be able to use the harmonic oscillator as a starting point for our qubits, we need to artificially create a system in the lab that behaves like a harmonic oscillator. The quantum LC circuit described next serves that purpose.

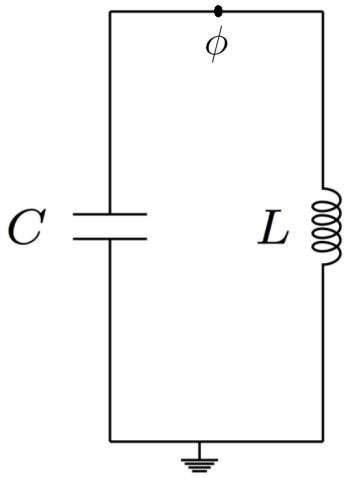


Figure 2.2: The LC oscillator is a circuit realization of a harmonic oscillator. The only degree of freedom in this circuit is in the node flux $\phi$.

## 2.2 LC oscillator

Consider a circuit that has a capacitor of capacitance $C$ connected to an inductor of inductance $L$ as in Fig. 2.2. The only degree of freedom in this circuit is carried by the node flux $\phi$ which plays the role of position $x$ in a quantum harmonic oscillator problem. The Lagrangian of the system can then be expressed in terms of $\phi$ and $\dot{\phi}$ as

$$\mathcal{L} = \frac{C\dot{\phi}}{2} - \frac{\phi^2}{2L}. \tag{2.8}$$

To define the conjugate variable to $\phi$, we take the derivative of $\mathcal{L}$ with respect to $\dot{\phi}$

$$\frac{\partial \mathcal{L}}{\partial \dot{\phi}} = C\dot{\phi} \equiv q, \tag{2.9}$$

where the charge on the capacitor $q$ is the analogue of momentum in the harmonic oscillator. The Hamiltonian of this circuit is then obtained via a Legendre transformation:

$$H = \frac{q^2}{2C} + \frac{\phi^2}{2L}, \tag{2.10}$$

which looks exactly like the harmonic oscillator Hamiltonian with ($q$, $\phi$, $C$, $L$) replacing ($p$, $x$, $m$, $k^{-1}$) respectively. Hence the system analysis follows like in the previous section, starting with the commutation relation

$$[\phi, q] = i\hbar \tag{2.11}$$

Then, the lowering and raising operators become

$$\begin{aligned} a &= \sqrt{\frac{1}{2\hbar Z_0}}\big(\phi + iZ_0 q\big) \\ a^\dagger &= \sqrt{\frac{1}{2\hbar Z_0}}\big(\phi - iZ_0 q\big) \end{aligned} \tag{2.12}$$

where $Z_0 = \sqrt{\frac{L}{C}}$. Finally, we obtain the quantized energy levels $|n\rangle$ with energies $\hbar\omega(n + \frac{1}{2})$ where $\omega = \sqrt{\frac{1}{LC}}$. Therefore, it is possible to engineer the energy spacing by tuning the values of $L$ and $C$ in the lab resulting in an artificial particle in a harmonic field.

However, using harmonic oscillators as qubits is problematic because it is difficult to isolate two levels of it as our computational subspace. This is due to the fact that the frequency of a single excitation is on resonance with subsequent excitations as they are all equally spaced. Hence, it is desirable to add some non-linearity to the harmonic oscillator to change the energy spacing of the energy levels. The Josephson junction is the only known circuit element that is strongly nonlinear and also non-dissipative at low temperatures [74]. That is why all superconducting qubits rely on the use of Josephson junctions which we will discuss next.

## 2.3 Josephson junctions

A Josephson junction consists of two superconductors coupled by a weak, typically insulating, link. We will label the superconducting phase difference of the wavefunction across the

Josephson junction $\phi$. This is the only degree of freedom representing the Josephson junction. The current and voltage across the junction are given by the Josephson relations

$$I = I_c \sin(\phi(t)) \tag{2.13}$$

and

$$V = \frac{\hbar}{2e}\frac{\partial \phi}{\partial t}, \tag{2.14}$$

where $I_c$ is the critical current of the junction, the current above which the junction behaves like a normal resistive junction and $e$ is the charge of the electron. We also define the magnetic flux quantum $\Phi_0 = \frac{h}{2e}$. If we take the first derivative of the superconducting current with respect to time, we get

$$\frac{\partial I}{\partial t} = I_c \frac{\partial \phi}{\partial t}\cos(\phi). \tag{2.15}$$

Plugging into equation 2.14, we find

$$V = \frac{\Phi_0}{2\pi I_c \cos(\phi)}\frac{\partial I}{\partial t} \tag{2.16}$$

which resembles the voltage-current relation of a regular inductor $V = L\frac{\partial I}{\partial t}$. Therefore, a Josephson junction can be thought of as a non-linear inductor with inductance

$$L_J = \frac{\Phi_0}{2\pi I_c \cos(\phi)}. \tag{2.17}$$

Hence, one approach to engineering an artificial atom is to replace the inductor in an $LC$ circuit by a Josephson junction. To discuss the exact effect of such change on the energy spectrum, we need to find the Hamiltonian of the system. In particular, we need to identify the potential energy stored in the junction as a function of time. Initializing the phase to

be $\pi/2$ at $t = 0$, the potential energy becomes

$$U(t) = \int_0^t IV dt = \int_{\pi/2}^{\phi(t)} I_c \sin(\phi'(t)) \frac{\Phi_0}{2\pi} d\phi' = \frac{I_c \Phi_0}{2\pi} \big(0 - \cos(\phi(t))\big). \tag{2.18}$$

Grouping the constants together, we define the Josephson energy $E_J$ as

$$E_J = \frac{I_c \Phi_0}{2\pi}, \tag{2.19}$$

which leads to

$$U = -E_J \cos(\phi). \tag{2.20}$$

This sinusoidal behavior of the junction potential energy is in contrast to the usual inductor quadratic potential energy of $\frac{\phi^2}{2L}$. This change in potential is crucial to achieve the desired anharmonicity of energy levels.

## 2.4 The Cooper-pair box (CPB) and transmon qubits

Replacing the inductor of an LC circuit with a Josephson junction results in the first family of superconducting qubits. The circuit is generally called the Cooper-pair box (CPB) which can be operated in different regimes. In the CPB, an additional gate voltage $V_g$ is capacitively coupled to the circuit that results in an extra gate charge $q_g = C_g V_g$ as shown in Fig. 2.3. The Hamiltonian of a CPB is then

$$H = \frac{(q - q_g)^2}{2C} - E_J \cos(\phi) = 4E_C (n - n_g)^2 - E_J \cos(\phi), \tag{2.21}$$

where $E_C = \frac{e^2}{2C}$ is the charging energy of the capacitor, $C = C_g + C_J$, $n = \frac{q}{2e}$ is the number of Cooper-pairs on one side of the capacitor and $n_g$ is the dimensionless offset charge. Choosing different values for $E_J$ and $E_C$ operate the CPB in different modes as evident in Fig. 2.4. It

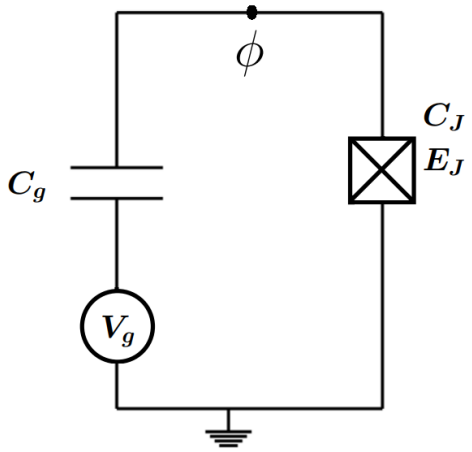


Figure 2.3: The Cooper-pair box (CPB) consists of a Josephson junction with energy $E_J$ and capacticance $C_J$ coupled through a capacitance $C_g$ to a voltage gate $V_g$. The circuit is grounded at one end of the junction and the only degree of freedom is in the superconducting phase $\phi$.

is clear that in the limit $E_J \gg E_C$, the energy levels are insensitive to the offset charge. The CPB operated in this limit is called the transmon qubit [71] which is protected from charge noise. This improves dephasing times compared to other CPB regimes. In the transmon regime, the lowest wavefunctions are localized in one well as shown in Fig. 2.5. This allows us to use an expansion in $\phi$ to gain more insight into the physics of a transmon qubit.

Working at $n_g = 0$ and expanding in $\phi$, we get

$$H = 4E_C n^2 - E_J + \frac{1}{2}E_J\phi^2 - \frac{1}{24}E_J\phi^4 + O(\phi^6). \tag{2.22}$$

The leading order $O(\phi^2)$ is a simple harmonic oscillator potential but the corrections from higher orders are the reason for achieving the desired anharmonicity. Rewriting the Hamiltonian in terms of the harmonic oscillator raising and lowering operators $a^\dagger$, $a$ results in [71]

$$H = \sqrt{8E_J E_C}(a^\dagger a + \frac{1}{2}) - E_J - \frac{E_C}{12}(a + a^\dagger)^4. \tag{2.23}$$

The first term is an effective harmonic oscillator with frequency $\omega = \sqrt{8E_J E_C}$. The second

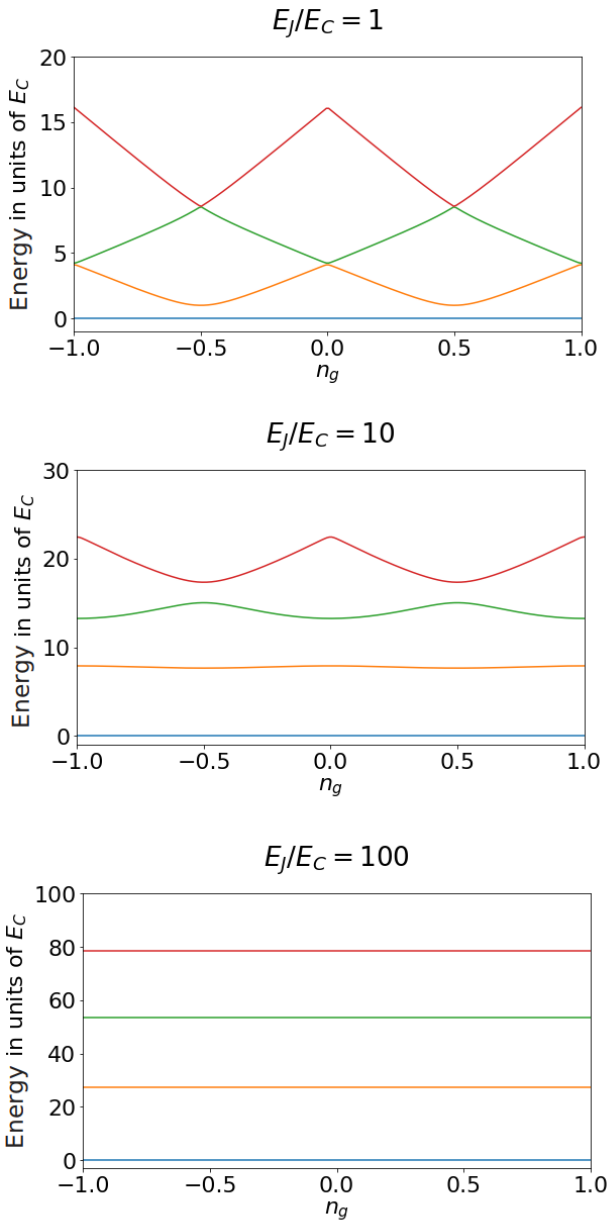


Figure 2.4: The four lowest energy values of the Cooper-pair box in different limits of $E_J/E_C$ vs. the offset charge $n_g$. We see that when $E_J \gg E_C$, the energies are independent of $n_g$.

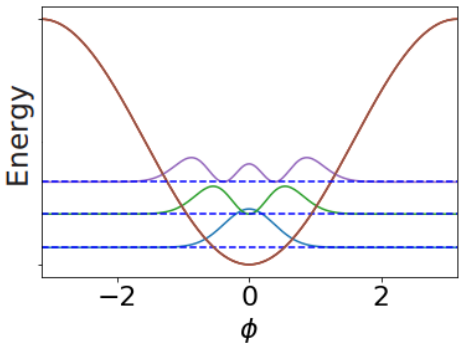


Figure 2.5: A qualitative sketch of the probability density functions of the lowest three levels in a transmon qubit, deeply localized in one well of the cosine potential.

term is a constant shift in energies, and the third is the correction introducing anharmonicity. The third term can be treated as a perturbation to the quantum harmonic oscillator Hamiltonian (with shifted energies) since it is proportional to $E_C \ll E_J$. The leading-order correction to the eigenenergies $E_k$ is then given by

$$E_k^{(1)} = -\frac{E_C}{12} \langle k| (a + a^\dagger)^4 |k\rangle = -\frac{E_C}{12}(6k^2 + 6k + 3) \tag{2.24}$$

If we define the anharmoncitiy $\alpha$ of a transmon as $\alpha = E_{21} - E_{10}$, we find that

$$\begin{aligned} \alpha &= E_2 - E_1 - E_1 + E_0 = E_2^{(1)} - 2E_1^{(1)} + E_0^{(1)} \\ &= -\frac{E_C}{12}(24 + 12 + 3 - 12 - 12 - 6 + 3) = -E_C \end{aligned} \tag{2.25}$$

This anharmonicity allows modeling a transmon as an effective two-level system with Hamiltonian

$$H = \frac{E_{10}}{2}\, \sigma_z, \tag{2.26}$$

if driving conditions do not occupy higher levels.

More recent superconducting circuits have been introduced that have more complex energy spectrum. Such qubits are called protected qubits, where the $|0\rangle$ and $|1\rangle$ wavefunctions are localized in two separate potential wells to decrease the effect of qubit relaxation. In

the following sections, we give an overview of two protected qubits, the fluxonium and 0-$\pi$ qubits.

## 2.5 The fluxonium qubit

The fluxonium qubit [85] is a promising superconducting circuit that may, in its most recent variants as "heavy fluxonium" [85, 35], outperform the widely used transmon qubit [71]. In contrast to the transmon, heavy fluxonium combines strong Josephson non-linearity with qubit relaxation protection due to disjoint support of its lowest-lying localized wave functions. Heavy fluxonium devices utilize a decreased capacitive energy $E_C$, which emphasizes the localization of states [85, 35]. Moreover, fluxonium eigenenergies are intrinsically insensitive to slow offset charge variations [70].

The fluxonium qubit is realized via adding a parallel large superinductor to the CPB circuit as shown in Fig. 2.7. The large superinductor is usually realized as an array of many large-area Josephson junctions to achieve an equivalent inductance of the junction array of $(L_{JA})$. The circuit is subject to external flux $\Phi_{\text{ext}}$ which allows for further tuning of the qubit parameters.

The Hamiltonian of the circuit is

$$H = 4E_C n^2 - E_J \cos(\phi - 2\pi\Phi_{\text{ext}}/\Phi_0) + \frac{1}{2}E_L\phi^2, \tag{2.27}$$

where $E_L = \Phi_0^2/2L_{JA}$ is the inductive energy of the superconducting junction array. The fluxonium wavefunctions structure can be engineered through designing the manufactured values of $E_J$, $E_C$ and $E_L$.

Ref. [36] explains in detail the effects of changing such parameters and how to realize the qubit experimentally. We discuss the potential and wavefunction changes due to changing the energy parameters $E_J$, $E_C$ and $E_L$ in Fig. 2.6.

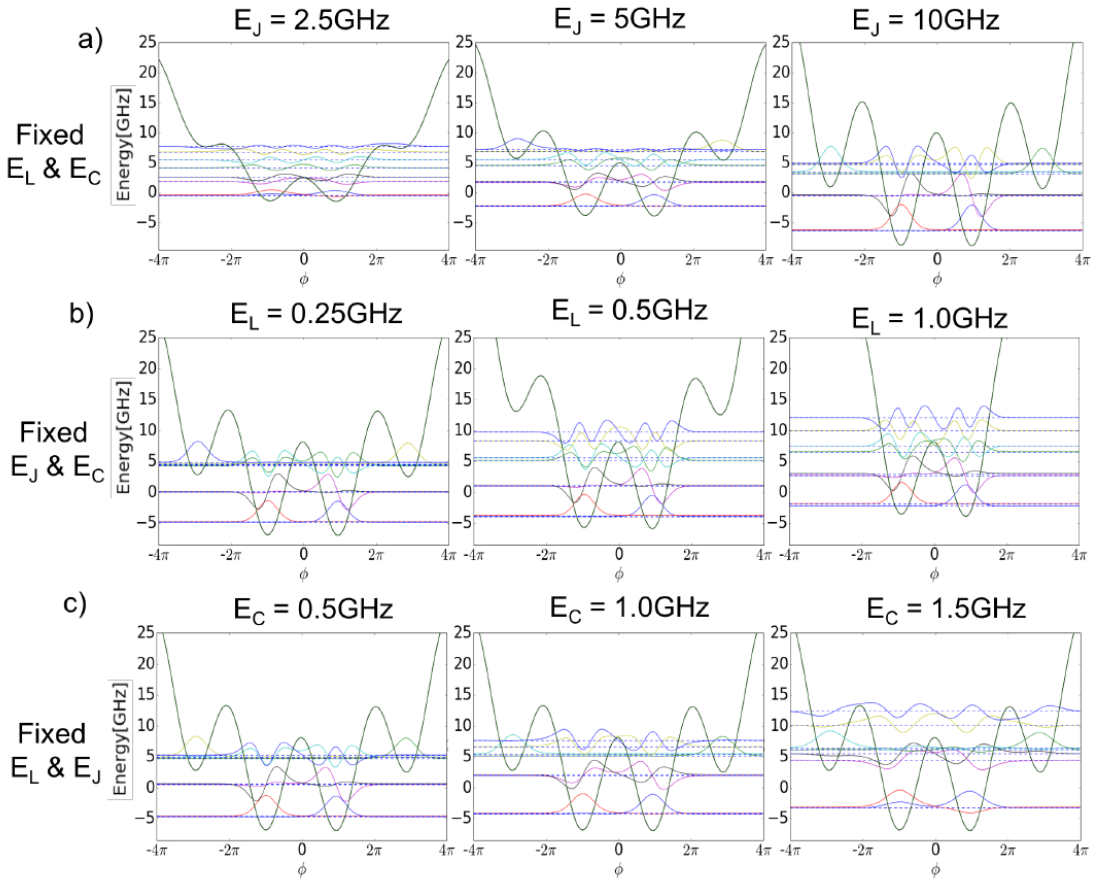


Figure 2.6: Ref. [36] plots the potential energy and wavefunctions of the fluxonium qubit in different regimes. The values of fixed parameters in the plot are $E_J$=8.11 GHz, $E_C$ =0.43 GHz and $E_L$= 0.24 GHz. Panel (a) shows that the effect of increasing $E_J$ is increasing the barrier height between the two deepest wells. This in turn suppresses the tunneling matrix element of the wavefunctions. Panel (b) shows that as $E_L$ is increased, the number of potential wells is decreased as the quadratic potential becomes more dominant. Finally, panel (c) shows that increasing $E_C$ increases the tunneling between wells and the wavefunctions start to lose their localization.

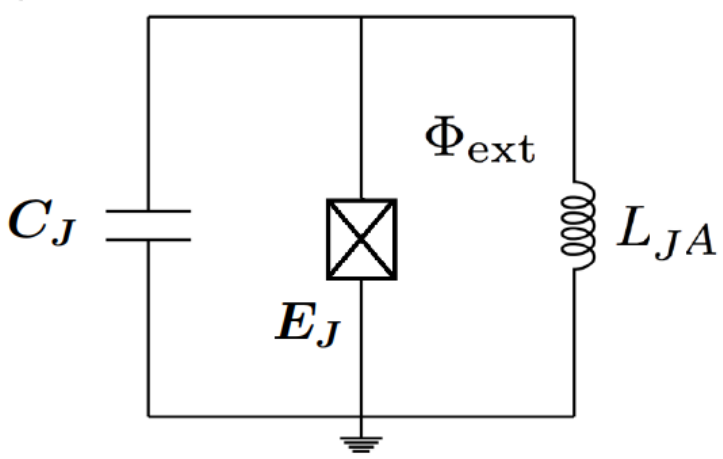


Figure 2.7: The fluxonium qubit consists of a small Josephson junction (of energy $E_J$) with junction capacitance $C_J$ connected to an array of Josephson junctions generating a superinductance of $L_{JA}$. The circuit is also subject to external flux $\Phi_{\text{ext}}$.

In summary, the fluxonium qubit is intrinsically protected from relaxation processes. It also has a high degree of flexibility in engineering its parameters. In the following section, we discuss another superconducting qubit that has similar protection and flexibility with additional advantages and challenges.

## 2.6 The 0-$\pi$ qubit

The 0-$\pi$ qubit initially proposed by Brooks et al. in 2013 [19] is another protected superconducting qubit. Unlike fluxonium, the 0-$\pi$ device can combine *both* exponential suppression of dephasing and relaxation due to wave function localization [51]. The ideal symmetric 0-$\pi$ device consists of two identical Josephson junctions with energies $E_J$ and junction capacitances $C_J$ and two large identical superinductors with inductance $L$ forming a loop as in Fig. 2.8. Opposite nodes of the loop are connected via large capacitors of capacitance $C$. The fluxes of the four nodes can be chosen to be the circuit degrees of freedom. However, because of the symmetry of the circuit, the following four variables give more insight into

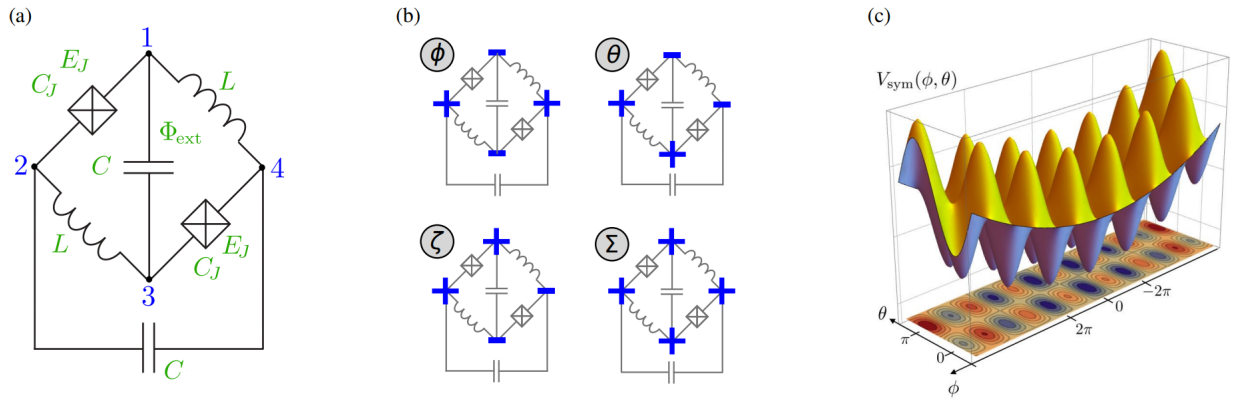


Figure 2.8: Plots from ref. [51] for the ideal symmetric 0-$\pi$ device. Panel (a) shows the circuit diagram involving two identical Josephson junctions, two identical inductors and two identical capacitances in a loop. Panel (b) shows the four modes $\theta$, $\phi$, $\zeta$ and $\Sigma$. Panel (c) shows the 3D potential energy of the symmetric circuit in $\theta$ and $\phi$.

the circuit behavior. The four modes are shown in Fig. 2.8.b.

$$\begin{aligned}
\theta &= \frac{1}{2}(\phi_2 - \phi_1 + \phi_3 - \phi_4) \\
\phi &= \frac{1}{2}(\phi_2 - \phi_3 + \phi_4 - \phi_1) \\
\zeta &= \frac{1}{2}(\phi_2 - \phi_3 - \phi_4 + \phi_1) \\
\Sigma &= \frac{1}{2}(\phi_1 + \phi_2 + \phi_3 + \phi_4).
\end{aligned} \tag{2.28}$$

The $\zeta$ mode involves phase differences across the inductors and does not bias the junctions, hence it remains a pure harmonic mode (in the absence of any circuit disorder). In addition, the $\Sigma$ mode is cyclic and can be omitted.

Therefore, the ideal symmetric 0-$\pi$ Hamiltonian has two degrees of freedom $\theta$ and $\phi$ that store quantum information and reads

$$H_{0-\pi} = \frac{q_\theta^2}{2C_\theta} + \frac{q_\phi^2}{2C_\phi} + E_L\phi^2 - 2E_J\cos(\theta)\cos(\phi - \phi_{\text{ext}}/2). \tag{2.29}$$

In this equation, $q_\theta = 2en_\theta$ and $q_\phi = 2en_\phi$ are the conjugate charge operators corresponding

to $\theta$ and $\phi$ where $[\theta, n_\theta] = i$ and $[\phi, n_\phi] = i$. The capacitances $C_\theta$ and $C_\phi$ are modifications of the circuits' large capacitor $C$ and Josephson capacitance $C_J$, $C_\theta = 2(C + C_J) + C_g$ and $C_\phi = 2C_J + C_g$, where $C_g$ is a small capacitance due to coupling to external circuitry. $E_L$ and $E_J$ denote the inductor energy and Josephson junction energy respectively, and $\phi_{\text{ext}} = \frac{2\pi\Phi_{\text{ext}}}{\Phi_0}$ denotes the external bias flux in dimensionless units.

While the 0-$\pi$ qubit exhibits intrinsic protection from decoherence, like heavy fluxonium this protection inadvertently prevents us from driving direct transitions between the two qubit states ($|0\rangle$ and $|\pi\rangle$). Therefore, there is a pressing need for finding optimal paths to control the qubit and perform required transitions. In the next chapter, we discuss the theory of quantum optimal control which will be used to find optimal solutions for driving transitions in superconducting qubits.

# CHAPTER 3

# CLOSED-SYSTEM OPTIMAL CONTROL

## 3.1 Introduction

Optimal control is a versatile concept which can be applied to a vast variety of quantum systems. Typically there is a primary goal (e.g. maximizing fidelity to a target state/unitary), as well as additional constraints/costs associated with specific experimental systems. Examples of such constraints include fixed maximum amplitudes of control pulses [114, 68], maximum overall power of control signals [69], and limited time resolution of arbitrary waveform generators [91]. Further, finite coherence of quantum systems motivates minimizing the overall time needed for reaching the intended state or unitary (time-optimal control) [23]. In certain cases, steering the quantum system along an optimal path (time-dependent target) may be desired [109].

To achieve these targets, there have been numerous theoretical developments of numerical and analytical methods for quantum optimal control of closed quantum systems (see ref. [44] for a recent review). The algorithms involved are predominantly based on gradient methods, such as realized in gradient ascent pulse engineering (GRAPE) [66, 30], Krotov algorithms [115, 39, 49, 94, 107, 99, 120] or rapid monotonically convergent algorithms [136, 137, 97, 83, 96, 13, 33], and are available in several open-source packages, including QuTiP [61, 62], DYNAMO [81], Spinach [57], and SIMPSON [122]. Quantum optimal control has been remarkably successful in determining optimized pulse sequences [12], designing high-fidelity quantum gates [116, 106, 94, 92, 63, 82, 38, 78, 105, 37, 73], and preparing entangled states [34, 103, 130, 46, 48].

In this chapter, we discuss the theory behind quantum optimal control and especially the closed-system GRAPE technique. In addition, we investigate ways to analytically incorporate different optimization targets with GRAPE.

## 3.2 Theory

We briefly review the essential idea of quantum optimal control and introduce the notation used throughout the thesis. We consider the general setting of a quantum system with intrinsic Hamiltonian $\mathcal{H}_0$ and a set of external control fields $\{u_1(t), \ldots, u_m(t)\}$ acting on the system via control operators $\{\mathcal{H}_1, \ldots, \mathcal{H}_m\}$. The resulting system Hamiltonian is given by $H(t) = \mathcal{H}_0 + \sum_{k=1}^{m} u_k(t)\mathcal{H}_k$. Optimal control theory aims to minimize deviations from a target state or target unitary by appropriate adjustments of the control fields $u_k(t)$. To implement this optimization, the time interval of interest is discretized into a large number $n$ of sufficiently small time steps $\delta t$. Denoting intermediate times by $t_j = t_0 + j\,\delta t$, the Hamiltonian at time $t_j$ takes on the form

$$H_j = \mathcal{H}_0 + \sum_{k=1}^{m} u_{k,j}\mathcal{H}_k. \tag{3.1}$$

The control fields subject to optimization now form a set $\{u_{k,j}\}$ of $z = m \cdot n$ real numbers.

The quantum evolution from the initial time $t = t_0$ to time $t_j$ is described by a propagator $U_j$, decomposed according to

$$U_j = M_j M_{j-1} M_{j-2} \ldots M_1 \tag{3.2}$$

where

$$M_j = \exp(-iH_j\delta t) \tag{3.3}$$

is the propagator for the short time interval $[t_j, t_j + \delta t]$. (Here and in the following, we set $\hbar = 1$.) Evolution of a select initial state $|\Psi_0\rangle$ from $t = t_0$ to $t = t_j$ then takes the usual form

$$|\Psi_j\rangle = U_j|\Psi_0\rangle. \tag{3.4}$$

In the decomposition of $U_j$, each short-time propagator $M_i$ can be evaluated exactly by matrix exponentiation or approximated by an appropriate series expansion. Propagation methods which go beyond the piecewise-constant approximation for the propagation, can further improve speed and accuracy [82]. Optimization of the discretized control fields $\mathbf{u} \in \mathbb{R}^z$ can be formulated as the minimization of a cost function $C(\mathbf{u})$ where $C : \mathbb{R}^z \to \mathbb{R}^+$. Next, we consider the important cost functions and their gradients, starting with the closed-system GRAPE algorithm for the target gate infidelity cost function.

## 3.3 Closed-system GRAPE algorithm

The closed-system GRAPE algorithm [66] focuses on maximizing the target gate fidelity using gradient ascent. The gradients are analytically calculated in GRAPE. We define the target gate fidelity as $F = |\operatorname{Tr}(U_t^\dagger U_f)/d|^2$ where $U_t$ is the target gate unitary, $U_f = U_n$ is the final achieved unitary at the final ($n$-th) time-step and $d$ is the Hilbert space dimension. GRAPE treats the optimization as a minimization problem with the cost (error) function to minimize being $C = 1 - F = 1 - |\operatorname{Tr}(U_t^\dagger U_f)/d|^2$. Gradients of this cost function with respect to the control parameters are calculated analytically. The cost function can be rewritten as

$$C = 1 - \frac{1}{d^2}|\operatorname{Tr}(U_t^\dagger U_f)|^2 = 1 - \langle U_t|U_f\rangle\langle U_f|U_t\rangle, \tag{3.5}$$

where $\langle U_t|U_f\rangle \equiv \frac{1}{d}\operatorname{Tr}(U_t^\dagger U_f)$.

Then, defining the forward and backward propagators at time-step $j$ as

$$\begin{aligned} X_j &= M_j M_{j-1} \dots M_1 \\ P_j &= M_{j+1}^\dagger \dots M_n^\dagger U_t \end{aligned} \tag{3.6}$$

we can write the cost function as

$$C = 1 - \langle P_j|X_j\rangle\langle X_j|P_j\rangle = 1 - \langle P_j|M_j X_{j-1}\rangle\langle M_j X_{j-1}|P_j\rangle. \tag{3.7}$$

Therefore, taking the gradient of this cost function with respect to some control parameter $u_{k,j}$ will only act on $M_j$ as it is the only matrix that depends on that control parameter. This simplifies the gradient calculation to

$$\frac{\partial C}{\partial u_{k,j}} = -\langle P_j|\frac{\partial M_j}{\partial u_{k,j}} X_{j-1}\rangle\langle X_j|P_j\rangle + \text{c.c.} \tag{3.8}$$

Finally, remembering that $M_j = \exp\big[-i\delta t(\mathcal{H}_0 + \sum_{k=1}^{m} u_{k,j}\mathcal{H}_k)\big]$, the derivative is approximated to first order in $\delta t$ as

$$\frac{\partial M_j}{\partial u_{k,j}} \approx -i\delta t \mathcal{H}_k M_j. \tag{3.9}$$

This leads to the final form for the analytical gradients of the target gate infedility cost function

$$\frac{\partial C}{\partial u_{k,j}} = -\delta t\, \mathrm{Im}\{\langle P_j|\mathcal{H}_k X_j\rangle\langle X_j|P_j\rangle\} \tag{3.10}$$

This expression allows for memory-efficient techniques for calculating the gradients as will be discussed in section 3.5.1.

After calculating the gradients, first-order gradient-based algorithms minimize the cost function $C(\mathbf{u})$ by the method of steepest descent, updating the controls $\mathbf{u}$ in the opposite direction of the local cost-function gradient $\nabla_{\mathbf{u}} C(\mathbf{u})$:

$$\mathbf{u}' = \mathbf{u} - \eta\, \nabla_{\mathbf{u}} C(\mathbf{u}). \tag{3.11}$$

The choice of the update step size $\eta$ for the control field parameters $\mathbf{u}$, plays an important role for the convergence properties of the algorithm. A number of schemes exist which adaptively

| $\mu$ | **Cost function contribution** | $C_\mu(\mathbf{u})$ |
|---|---|---|
| 1 | Target gate infidelity | $1 - \lvert \operatorname{Tr}(U_t^\dagger U_f)/d \rvert^2$ |
| 2 | Target state infidelity | $1 - \lvert \langle \Psi_t \lvert \Psi_f \rangle \rvert^2$ |
| 3 | Control amplitudes | $\lvert \mathbf{u} \rvert^2$ |
| 4 | Control variations | $\sum_{j,k} \lvert u_{k,j} - u_{k,j-1} \rvert^2$ |
| 5 | Occupation of forbidden state | $\sum_j \lvert \langle \Psi_F \lvert \Psi_j \rangle \rvert^2$ |
| 6 | Evolution time (target gate) | $1 - \frac{1}{n} \sum_j \lvert \operatorname{Tr}(U_t^\dagger U_j)/d \rvert^2$ |
| 7 | Evolution time (target state) | $1 - \frac{1}{n} \sum_j \lvert \langle \Psi_t \lvert \Psi_j \rangle \rvert^2$ |

Table 3.1: Relevant contributions to cost functions for quantum optimal control. Names of contributions indicate the quantity to be *minimized*.

determine an appropriate step size $\eta$ in each iteration of the minimization algorithm.

## 3.4 Important types of cost function contributions

While GRAPE only deals with the gate infidelity cost function, many other optimization targets are usually desired as well. Table 3.1 shows some of the most important cost function contributions used for closed-system optimal control. The total cost function is a linear combination of these cost functions, $C = \sum_\mu \alpha_\mu C_\mu$. The weight factors $\alpha_\mu$ must be determined empirically, and depend on the specific problem and experimental realization at hand.

### *3.4.1 Unitary gate infidelity*

The first cost contribution, $C_1(\mathbf{u})$, is the primary tool for realizing a target unitary $U_t$, such as a single or multi-qubit gate. Cost is incurred for deviations between the target unitary and the realized unitary $U_f$ at a given final time $t_n$. For a system with Hilbert space dimension $d$, its expression $1 - \lvert \operatorname{Tr}(U_t^\dagger U_f)/d \rvert^2$ [66] represents the infidelity obtained from the trace distance between the target unitary and the realized unitary. Minimizing this cost function is the principle goal of the quantum control problem.

### *3.4.2 State transfer infidelity*

The second cost function, $C_2(\mathbf{u}) = 1 - |\langle\Psi_t|\Psi_f\rangle|^2$ measures the distance between a desired target state $|\Psi_t\rangle$ and the state $|\Psi_f\rangle$ realized at the final time $t_n$, as obtained from evolution of a given initial state $|\Psi_0\rangle$. In addition, generalizing $C_2$ to multiple initial and target states is useful for performing a unitary $U_t$ which is only defined on some subspace $H_{\mathcal{S}}$ of the modeled Hilbert space. Such restriction to a selected subspace is of practical importance whenever a desired unitary is to be implemented within some computational subspace only, as is common for quantum computation applications. There, the evolution of higher excited states or auxiliary systems outside the computational subspace is immaterial. Optimal control, then, can be achieved by simultaneous evolution of a set of initial states $\{|\Psi_0^s\rangle\}$ $(s = 1, 2, \ldots, S)$ that forms a basis of $H_{\mathcal{S}}$. Optimal control fields are obtained from minimizing the composite state infidelity $C_{2\Sigma}(\mathbf{u}) = 1 - |\frac{1}{S}\sum_s \langle\Psi_t^s|P_{\mathcal{S}}|\Psi_f^s\rangle|^2$ relative to the desired target states $|\Psi_t^s\rangle = U_t|\Psi_0^s\rangle$. (Here, $P_{\mathcal{S}}$ is the projector onto subspace $H_{\mathcal{S}}$.)

This composite state-transfer cost function when used over a complete basis is equivalent to the gate fidelity, but has several advantages. Most importantly it is more memory efficient requiring only the current state to be stored rather than the whole unitary. In addition, it is very amenable to distributed computing approaches. However, when the unitary transfer matrix can be stored in memory, propagating the full unitary can take advantage of the parallelism of the GPU for smaller problems (see Fig. 5.2).

In order to obtain control fields that are consistent with specific experimental capabilities and limitations, it is often crucial to add further constraints on the optimization. Control fields must be realizable in the lab, should be robust to noise, and avoid large control amplitudes and rapid variations based on signal output specifications of instruments employed in experiments. Exceedingly strong control fields may also be problematic due to heat dissipation which may, for instance, raise the temperature inside a dilution refrigerator. These points motivate the consideration of additional cost function contributions in the following.

### 3.4.3 Pulse power

One such contribution, $C_3(\mathbf{u}) = |\mathbf{u}|^2$ suppresses large control-field amplitudes globally, and is commonly employed in quantum optimal control studies [66, 114, 68, 56]. (The generalization to more fine-grained suppression of individual control fields is straightforward to implement as well.) Penalizing the $L^2$ norm of the control fields favors solutions with low amplitudes. It also tends to spread relevant control fields over the entire allowed time window. While $C_3$ constitutes a "soft" penalty on control-field amplitudes, one may also apply a trigonometric mapping to the amplitudes to effect a hard constraint strictly enforcing fixed maximum amplitudes [40]. For example, the tanh mapping converts tunable optimization weights $w_{jk}$ to pulse amplitudes $u_{jk}$ through the relationship $u_{jk} = A_{\max} \tanh(w_{jk})$, where $A_{\max}$ is the maximum amplitude of the pulse.

### 3.4.4 First and second derivatives of the pulse

The fourth type of contribution to the cost function, $C_4(\mathbf{u}) = \sum_{j,k} |u_{k,j} - u_{k,j-1}|^2$, penalizes rapid variations of control fields by suppressing their (discretized) time derivatives [56]. The resulting smoothening of signals is of paramount practical importance, since any instrument generating a control field has a finite impulse response. If needed, contributions analogous to $C_4$ which suppress higher derivatives or other aspects of the time dependence of fields can be constructed. Together, limiting the control amplitudes and their time variation filters out high-frequency "noise" from control fields, which is an otherwise common result of less-constrained optimization. Smoother control fields also have the advantage that essential control patterns can potentially be recognized and given a meaningful interpretation.

### 3.4.5 Forbidden state occupation

The contribution $C_5(\mathbf{u}) = \sum_j |\langle \Psi_F | \Psi_j \rangle|^2$ to the cost function has the effect of suppressing occupation of a select "forbidden" state $|\Psi_F\rangle$ (or a set of such states, upon summation)

throughout the evolution. The inclusion of this contribution addresses an important issue ubiquitous for systems with Hilbert spaces of large or infinite dimension. In this situation, truncation of Hilbert space is needed or inevitable due to computer memory limitations. (Note that this need even arises for a single harmonic oscillator.) Whenever the evolution generated by optimal control algorithms explores highly excited states, truncation introduces a false non-linearity which can misguide the optimization. Including additional states can, in principle, mitigate this problem, but is generally computationally very expensive. An independent physics motivation for avoiding occupation of highly-excited states consists of spontaneous relaxation in realistic systems: high-energy states are often more lossy (as is usually the case, e.g., for superconducting qubits), and possibly more difficult to model. Active penalization of such states therefore has the two-fold benefit of keeping Hilbert space size at bay, and reducing unwanted fidelity loss from increased relaxation. To address these challenges, we employ an intermediate-time cost function [109, 100]: the cost function $C_5$ limits leakage to higher states during the entire evolution, and at the same time prevents optimization to be misinformed by artificial non-linearity due to truncation. We note that the efficacy of this strategy is system dependent: it works well, for example, for harmonic oscillators or transmon qubits [71] which have strong selection rules against direct transitions to more distant states, but may be less effective in systems such as the fluxonium circuit [85] where low-lying states have direct matrix elements to many higher states.

### *3.4.6 Gate time*

Customarily, algorithms minimizing the cost function $C = \sum_\mu \alpha_\mu C_\mu$ for a given evolution time interval $[t_0, t_n]$ aim to match the desired target unitary or target state at the very end of this time interval. To avoid detrimental effects from decoherence processes during the evolution, it is often beneficial to additionally minimize the gate duration (or state preparation) time $\Delta t = t_n - t_0$ itself. Instead of running the algorithms multiple times

for a set of different $\Delta t$, we employ cost function contributions of the form $C_6(\mathbf{u}) = 1 - \frac{1}{n}\sum_j |\operatorname{Tr}(U_t^\dagger U_j)/d|^2$ for a target unitary, or $C_7(\mathbf{u}) = 1 - \frac{1}{n}\sum_j |\langle\Psi_t|\Psi_j\rangle|^2$ for a target state, respectively. These expressions penalize deviations from the target gate or target state not only at the final time $t_n$, but *at every time step*. This contribution to the overall cost function therefore guides the evolution towards a desired unitary or state in as short a time as possible under the conditions set by the other constraints.

## 3.5 Analytical gradients of cost functions

In the following, we outline the analytical calculation of gradients for cost functions such as those summarized in Table 3.1. We stress that our automatic-differentiation implementation evaluates these gradients autonomously, without the need of these analytical derivations or hard-coding any new gradients. The following derivations are thus merely intended as illustrations for a better mathematical understanding (and appreciation) of the gradients calculated without user input by means of automatic differentiation.

For a systematic treatment of the different types of cost functions, we note that most cost functions involve the absolute-value squared of an inner product between target and final states or target and final unitaries (Hilbert-Schmidt inner product). To obtain the gradients of expressions such as $C_1(\mathbf{u}) = 1 - |\operatorname{Tr}(U_t^\dagger U_f)|^2$ with respect to the control parameters, we note that control parameters enter via the final states or unitaries through the evolution operators, $U_f = M_n(\mathbf{u})M_{n-1}(\mathbf{u})\cdots M_1(\mathbf{u})$. To streamline our exposition, we first summarize multiple matrix-calculus relations of relevance.

Consider two complex-valued matrices $A$ and $B$, compatible in row/column format such that the matrix product $AB$ is defined. Then, one finds

$$\frac{\partial \operatorname{Tr}(AB)}{\partial B_{ji}} = \frac{\partial (A_{nm}B_{mn})}{\partial B_{ji}} = A_{ij}. \tag{3.12}$$

Throughout this appendix, we use Einstein convention for summation, and follow the Jacobian formulation (also known as numerator layout) for derivatives with respect to matrices. We will further encounter expressions of the following form, involving a third matrix $C$ of the same dimensions as $B^t$:

$$\begin{aligned}\operatorname{Tr}\left[\frac{\partial[|\operatorname{Tr}(AB)|^2]}{\partial B}C\right] &= \frac{\partial[\operatorname{Tr}(AB)\operatorname{Tr}(AB)^*]}{\partial B_{ji}}C_{ji} \\ &= \frac{\partial \operatorname{Tr}(AB)}{\partial B_{ji}}\operatorname{Tr}(AB)^* C_{ji} = A_{ij}\operatorname{Tr}(AB)^* C_{ji} \\ &= \operatorname{Tr}(AC)\operatorname{Tr}(AB)^*. \end{aligned} \tag{3.13}$$

In the framework of Wirtinger derivatives in complex analysis, derivatives treat quantities $X$ and $X^*$ as independent variables, and Eq. (3.12) is used in the step from line 1 to line 2.

The evaluation of cost-function gradients requires the application of the chain rule to expressions of the type $\frac{\partial}{\partial u_i}c(k(\mathbf{u}))$. Here, $c$ maps a complex-valued $\ell \times \ell$ matrix $K$ (e.g., the propagator $U_f$ with $\ell$ denoting the Hilbert space dimension) to a real number (the cost). The matrix $K = (K_{mn})$ itself depends on the real-valued control parameters $\mathbf{u} \in \mathbb{R}^z$. The subscript in $u_i$ is understood as a multi-index $i = (k, j)$ encoding the control-field label $k$ and discretized-time index $j$. The matrix-calculus result

$$\begin{aligned}\frac{\partial}{\partial u_i}c(K(\mathbf{u})) &= \frac{\partial c}{\partial K_{mn}}\frac{\partial K_{mn}}{\partial u_i} + \frac{\partial c}{\partial K^*_{mn}}\frac{\partial K^*_{mn}}{\partial u_i} \\ &= \operatorname{Tr}\left(\frac{\partial c}{\partial K}\frac{\partial K}{\partial u_i}\right) + \text{c.c.}\end{aligned} \tag{3.14}$$

is straightforward to derive with the "regular" chain rule by re-interpreting the functions involved as $c\colon \mathbb{C}^{\ell^2} \to \mathbb{R}$ and $K\colon \mathbb{R}^z \to \mathbb{C}^{\ell^2}$. In the following, Eqs. (3.13) and (3.14) are used to obtain the analytical expressions for several examples of cost-function gradients.

### 3.5.1 Gradient for $C_1$: target-gate infidelity

The cost function $C_1 = 1 - |\operatorname{Tr}[U_t^\dagger U_f(\mathbf{u})]/d|^2$, penalizes the infidelity of the realized unitary $U_f = M_n M_{n-1} \ldots M_1$ with respect to the target propagator $U_t$. In the following, we omit the constant factor $d$ since it affects all the gradients only by a constant factor. The cost function then has the gradient

$$\begin{aligned}
\frac{\partial C_1}{\partial u_{k,j}} &\overset{(3.14)}{=} \operatorname{Tr} \frac{\partial C_1}{\partial U_f} \frac{\partial U_f}{\partial u_{k,j}} + \text{c.c.} = -\operatorname{Tr}\left[\frac{\partial[|\operatorname{Tr}(U_t^\dagger U_f)|^2]}{\partial U_f} \frac{\partial U_f}{\partial u_{k,j}}\right] + \text{c.c.} \\
&\overset{(3.13)}{=} -\operatorname{Tr}\left(U_t^\dagger \frac{\partial U_f}{\partial u_{k,j}}\right) \operatorname{Tr}(U_t^\dagger U_f)^* + \text{c.c.} = \operatorname{Tr}\left(U_t^\dagger \Big[\prod_{j'>j} M_{j'}\Big] i\,\delta t\, \mathcal{H}_k U_j\right) \operatorname{Tr}(U_t^\dagger U_f)^* + \text{c.c.} \\
&= -2\,\delta t \operatorname{Im}\left\{\operatorname{Tr}\left(U_t^\dagger \Big[\prod_{j'>j} M_{j'}\Big] \mathcal{H}_k U_j\right) \operatorname{Tr}(U_t^\dagger U_f)^*\right\}
\end{aligned} \tag{3.15}$$

where $\prod$ is understood to produce a time-ordered product.

*Memory-efficient algorithm.*— We note that storage of $\{U_j\}$ can be avoided by applying the strategy introduced in the original GRAPE paper [66]: since the evolution is unitary, one may time-reverse the evolution step by step, and re-calculate the intermediate propagator via $U_j = M_{j+1}^\dagger U_{j+1}$. Here, each short-time propagator $M_j$ is re-generated locally in time, using only the control fields at time $t_j$. Such a backwards-propagation algorithm leads to an increase in computation time by roughly a factor of 2 (each $M_j$ is then calculated twice), but has a memory demand of only $O(\ell^2)$ – which does not scale with $n$, the number of time steps. This memory-efficient algorithm is given by

**Algorithm 1** $C_1$ gradient via backwards propagation

$P = \mathrm{Tr}(U_t^\dagger U_f)^* U_t^\dagger$
$X = U_f$
**for** $j = n$ to $0$ **do**
    **for** all $k$ **do**
        $\partial C_1 / \partial u_{k,j} = -2\delta t \,\mathrm{Im}[\mathrm{Tr}(P\,\mathcal{H}_k\,X)]$
    **end for**
    $X = M_j^\dagger X$
    $P = P\,M_j$
**end for**
return $\partial C_1 / \partial \mathbf{u}$

### *3.5.2 Gradient for $C_2$: target-state infidelity*

For state preparation or unitaries specified only in a subspace, it is sufficient to optimize the evolution for only a few initial states, rather than for the complete basis. This is achieved by minimizing a cost function based on $C_2(\mathbf{u}) = 1 - |\langle\Psi_t|\Psi_f\rangle|^2$, where the realized final state $|\Psi_f\rangle$ depends on the control parameters $\mathbf{u}$. Again applying equations (3.14) followed by (3.13) (and using that the trace of a number results in that number), we obtain

$$\frac{\partial C_2}{\partial u_{k,j}} = -2\,\delta t\,\mathrm{Im}\left[\langle\Psi_t|\Big[\prod_{j'>j} M_{j'}\Big]\mathcal{H}_k|\Psi_j\rangle\langle\Psi_t|\Psi_f\rangle^*\right]$$

*Memory-efficient algorithm.*— By using the same backward propagation strategy as above, a memory-efficient algorithm with memory requirement $O(\ell)$ independent of the time-step number is possible:

**Algorithm 2** $C_2$ gradient via backwards propagation

$P = \langle\Psi_t|\Psi_f\rangle^* \langle\Psi_t|$
$X = |\Psi_f\rangle$
**for** $j = n$ to $0$ **do**
    **for** all $k$ **do**
        $\partial C_2/\partial u_{k,j} = -2\delta t \,\mathrm{Im}[P\,\mathcal{H}_k\,X]$
    **end for**
    $X = M_j^\dagger X$
    $P = P\,M_j$
**end for**
return $\partial C_2/\partial \mathbf{u}$

### *3.5.3 Gradient for $C_5$: occupation of forbidden state*

Occupation of a "forbidden" state is discouraged by the cost function $C_5 = \sum_j |\,\mathrm{Tr}(\Psi_F^\dagger \Psi_j)|^2$. This cost function differs qualitatively from the gate and state infidelity cost functions: the latter are evaluated based on the result at the final time, while forbidden-state occupation involves intermediate states at every time step. Accordingly, the corresponding gradient takes a different form. First, Eq. (3.14) is replaced by

$$\frac{\partial}{\partial u_i} c\big(\Psi_0(\mathbf{u}), \Psi_1(\mathbf{u}), \ldots, \Psi_n(\mathbf{u})\big) = \mathrm{Tr}\,\frac{\partial c}{\partial \Psi_j}\frac{\partial \Psi_j}{\partial u_i} + \text{c.c.} \tag{3.16}$$

where introduction of the trace of a $c$-number is convenient for direct application of Eq. (3.13). We then obtain

$$\frac{\partial C_5}{\partial u_{k,j}} \overset{(3.16)}{=} \sum_J \mathrm{Tr}\,\frac{\partial C_5}{\partial \Psi_J}\frac{\partial \Psi_J}{\partial u_{k,j}} + \text{c.c.} = \sum_{J\geq j}\sum_{j'} \mathrm{Tr}\left[\frac{\partial[|\,\mathrm{Tr}(\Psi_F^\dagger \Psi_{j'})|^2]}{\partial \Psi_J}\frac{\partial \Psi_J}{\partial u_{k,j}}\right] + \text{c.c.} \quad (3.17)$$

$$\overset{(3.13)}{=} \sum_{J\geq j} \mathrm{Tr}\left(\Psi_F^\dagger \frac{\partial \Psi_J}{\partial u_{k,j}}\right)\mathrm{Tr}(\Psi_F^\dagger \Psi_J)^* + \text{c.c.}$$

$$= 2\,\delta t \sum_{J\geq j} \mathrm{Im}\left[\big\langle \Psi_F\big|\big[\textstyle\prod_{j'=j+1}^{J} M_{j'}\big]\mathcal{H}_k\big|\Psi_j\big\rangle\big\langle \Psi_J\big|\Psi_F\big\rangle\right]$$

The double sum of eq (3.17), after simplification, could be calculated in $O(N)$ time. The corresponding backward propagation algorithm then takes the following form:

**Algorithm 3** $C_5$ gradient via backwards propagation

$P = \langle \Psi_n|\Psi_F\rangle\langle\Psi_F|$
$X = |\Psi_n\rangle$
**for** $j = n$ to $0$ **do**
  **for** all $k$ **do**
    $\partial C_5/\partial u_{k,j} = 2\,\delta t\,\mathrm{Im}[P\mathcal{H}_k X]$
  **end for**
  $X = M_j^\dagger X$
  $P = P\,M_j + \langle X|\Psi_F\rangle\langle\Psi_F|$
**end for**
return $\partial C_5/\partial \mathbf{u}$

This cost function and gradient are also used as the time-optimal award function. Rather than penalizing the occupation of a forbidden state at intermediate time-steps, we reward the occupation of the target state at all time-steps. Therefore, the analytical calculation of time-optimal gradients is similar.

While we carefully analyzed how to calculate analytical gradients for many cost functions

in this chapter, a general approach for automatizing the calculation of gradients for any other cost function is desirable. Therefore, we move to the concept of automatic differentiation in the next chapter which is an automated process for calculating the necessary gradients for any optimization target.

# CHAPTER 4

# AUTOMATIC DIFFERENTIATION

## 4.1 Introduction

Incorporating new constraints in the optimization process often requires the analytical derivation and implementation of additional contributions to the gradient calculation. This issue can greatly impede the ability to quickly develop control strategies for new problems.

To overcome this issue, we note that the techniques and algorithms used to optimize the control of quantum systems [44, 66, 30, 115, 39, 49, 94, 107, 99, 120, 136, 137, 97, 83, 96, 13, 33] and those underlying the field of deep neural networks [54, 55] share a number of common elements. Both areas heavily use linear algebra operations combined with gradient descent optimization. Thus, advanced hardware and software technology recently emerging from the rapid development of machine learning also paves the way for a significant boost of optimal quantum control techniques.

In particular, automatic differentiation (AD) [4, 131] has become a central tool in machine learning [5] to generate the gradients of complex neural networks at runtime. It equally applies to the problem of optimal control of quantum systems. In this approach, the gradients of a set of elementary operations are defined and more complex functions are built as computational graphs of these operations. The value of the function is computed by traversing the graph from inputs to the output, while the gradient is computed by traversing the graph in reverse to concatenate the needed gradients in a chain-rule manner. This methodology gives the same numerical accuracy and stability of analytic gradients without requiring one to derive and implement analytical gradients specific to each new trial cost function.

## 4.2 AD elements

An automatic differentiator allows users to define mathematical relationships between an output function and its inputs, then can calculate the corresponding gradients by concatenating a series of pre-defined gradients. The full procedure of automatic differentiation is described in the following steps:

1. The differentiator uses a basic set of $m$ pre-defined operations $\{f_i(x_1, x_2, \ldots, x_{n_i})\}$ where $i = 1, 2, \ldots, m$. For each operation $f_i$, the gradients $\frac{\partial f_i}{\partial x_1}, \ldots, \frac{\partial f_i}{\partial x_{n_i}}$ with respect to all $n_i$ inputs must be defined.

2. The user sets up the optimization problem by defining a cost function that is expressed solely by the operations $f_i$. This step creates the so-called computational graph.

3. The algorithm runs through the computational graph in forward direction, starting from certain values of the inputs and computing the resulting cost-function value. The corresponding gradients are calculated as every operation is executed, and are saved for processing.

4. After completion of the forward path, reverse-mode automatic differentiation is utilized to trace backwards all paths relating the cost function to the inputs. This involves properly summing the stored partial derivatives of each path, generated by a recursive chain rule.

## 4.3 Example: Scalar cost functions

In the following, we discuss a simple example where all operations involved act on scalars. Consider automatic differentiation of the cost function $C(x_1, x_2) = 2x_1^2 + \exp(x_1 x_2)$ with respect to its inputs $x_1$ and $x_2$. To build an automatic differentiator, a set of building block operations needs to be predefined, one example of this set is shown in Table 4.1.

Table 4.1: Example set of scalar operations for automatic differentiation

| Name | Definition | Gradients |
|---|---|---|
| MUL | $f(x_1, x_2) = x_1 x_2$ | $\frac{\partial f}{\partial x_1} = x_2$ and $\frac{\partial f}{\partial x_2} = x_1$ |
| DIV | $f(x_1, x_2) = \frac{x_1}{x_2}$ | $\frac{\partial f}{\partial x_1} = \frac{1}{x_2}$ and $\frac{\partial f}{\partial x_2} = \frac{-x_1}{x_2^2}$ |
| ADD | $f(x_1, x_2) = x_1 + x_2$ | $\frac{\partial f}{\partial x_1} = 1$ and $\frac{\partial f}{\partial x_2} = 1$ |
| SCALE | $f(a, x_1) = a\, x_1$ | $\frac{\partial f}{\partial x_1} = a$ |
| EXP | $f(x_1) = e^{x_1}$ | $\frac{\partial f}{\partial x_1} = e^{x_1}$ |

Note that the exponential operation does not strictly have to be included, since an approximation for it could be represented in a Taylor series involving only MUL, ADD and SCALE operations. The same applies to other functions like sine and cosine, as well as taking the integer power of an input, etc. In practice though, it is convenient to include special functions with compact analytical derivative in the basic set to give the user more flexibility in defining the computational graph.

Using Table 4.1 as the basic-operation set, we next define the computational graph, see Fig. 4.1. After the computational graph is run in forward direction (from $x_1$ and $x_2$ towards $C$), reverse-mode automatic differentiation identifies paths between $C$ and each of $x_1$ and $x_2$. In this example, there is one path (the orange path) relating C to $x_2$ while there are 3 paths (red paths) between $C$ and $x_1$. Gradients are automatically calculated in a backward direction by recursively multiplying the gradients in each path from bottom to top, then summing over all paths contributing to the same variable. Following this scheme, automatic

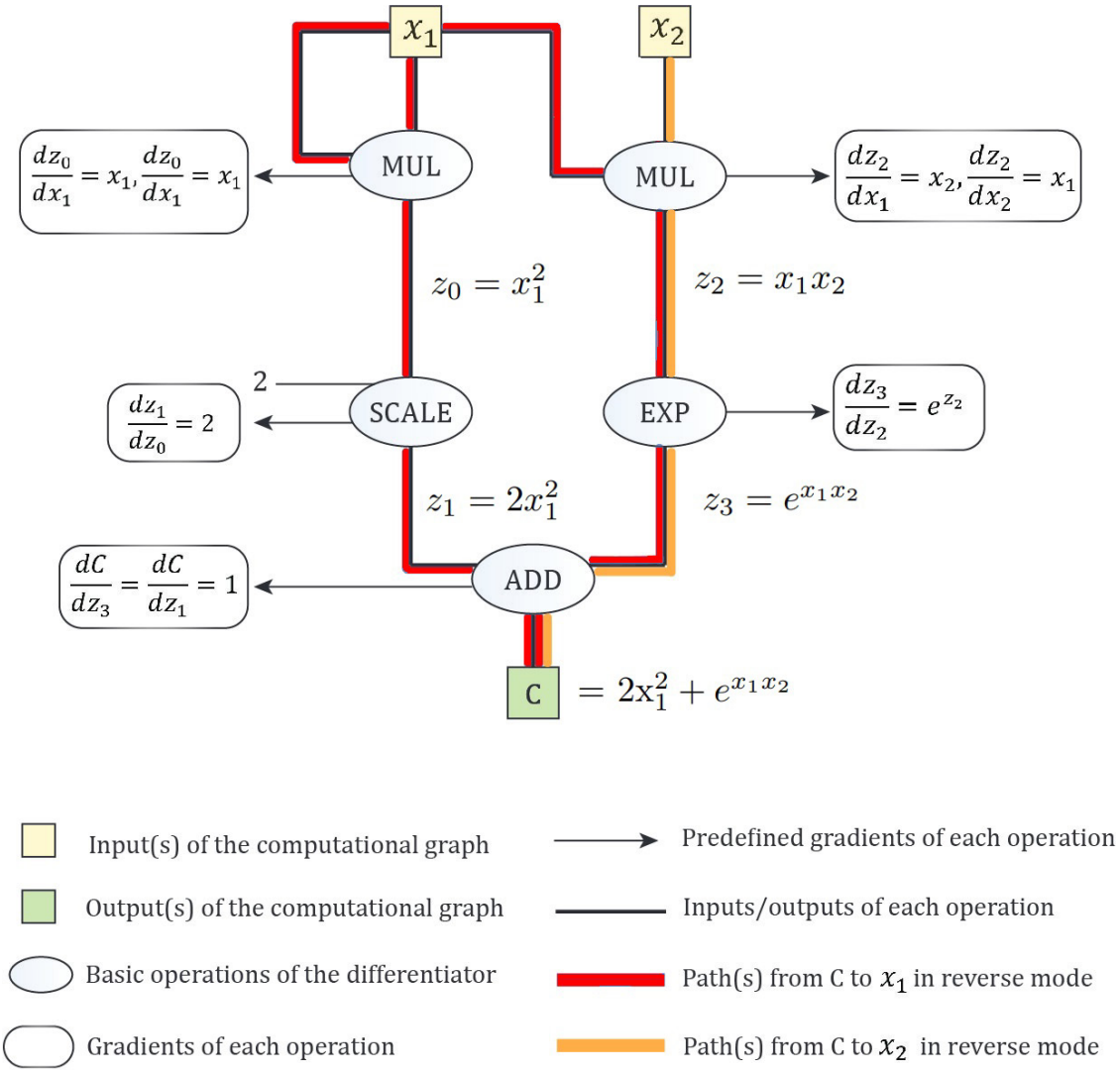


Figure 4.1: Computational graph for the example cost function $C(x_1, x_2) = 2x_1^2 + \exp(x_1 x_2)$, created for automatic differentiation to calculate $\frac{\partial C}{\partial x_1}$ and $\frac{\partial C}{\partial x_2}$. Every ellipse represents one basic operation whose gradients are known and symbolically given by the expressions in rounded rectangles, attached by arrows. The inputs to each operation are represented by lines entering the ellipse from above or from the left/right. The output of each operation is given a name $z_i$ written on the output line emerging from the bottom. After the computational graph run in forward direction (from $x_1$, $x_2$ towards $C$), reverse-mode automatic differentiation identifies paths between $C$ and each of $x_1$ and $x_2$. In this example, there is one path (orange color) relating $C$ to $x_2$ while there are 3 paths (red paths) between $C$ and $x_1$. Gradients are automatically calculated in a backward fashion by recursively multiplying the gradients in each path from bottom to top, then summing over all paths contributing to the same variable.

differentiation calculates the gradients in the reverse mode, yielding

$$\begin{aligned}\frac{\partial C}{\partial x_1} &= \frac{\partial C}{\partial z_1}\frac{\partial z_1}{\partial z_0}\frac{\partial z_0}{\partial x_1} + \frac{\partial C}{\partial z_1}\frac{\partial z_1}{\partial z_0}\frac{\partial z_0}{\partial x_1} + \frac{\partial C}{\partial z_3}\frac{\partial z_3}{\partial z_2}\frac{\partial z_2}{\partial x_1} \\ &= 1 \cdot 2 \cdot x_1 + 1 \cdot 2 \cdot x_1 + e^{z_2}x_2 = 4x_1 + e^{x_1 x_2}x_2, \\ \frac{\partial C}{\partial x_2} &= \frac{\partial C}{\partial z_3}\frac{\partial z_3}{\partial z_2}\frac{\partial z_2}{\partial x_2} = 1 \cdot e^{z_2}x_1 = e^{x_1 x_2}x_1,\end{aligned}$$

which are indeed the correct gradients. In this way, automatic differentiation allows for calculating gradients for any computational graph that is defined in terms of the building blocks of the differentiator.

## 4.4 Linear-algebra-based AD

To utilize automatic differentiation in the optimization of quantum systems, the set of basic operations must include matrix operations since the quantum dynamics is propagated through matrix operations. The definition of gradients then needs to account for the non-commutativity of matrix multiplication. To be able to generate gradients in the backward paths, the differentiator must be able to express the gradients with respect to the outputs in terms of the gradients with respect to the inputs of the operation at hand. Thus, for every basic matrix operation $\boldsymbol{f}(\boldsymbol{M_i})$, we need to specify how the gradients with respect to the inputs of the operation, $\frac{\partial C}{\partial \boldsymbol{M_i}}$, can be calculated given gradients with respect to the output $\frac{\partial C}{\partial \boldsymbol{f}}$, where bold letters indicate matrices. Table 4.2 shows some of the basic matrix operations and their gradient relations needed for automatic differentiation.

Because of the asymmetry in the MATMUL-gradient rule, it is not as simple to express the gradient relations for most special functions as it was in the scalar case. Most of the functions not defined in Table 4.2 (including the matrix exponential) must be expressed in terms of the basic operations like MATMUL, ADD and SCALE so that the gradient calculation is accurate.

Table 4.2: Examples of basic matrix operations needed for automatic differentiation. Arguments of MATMUL must be compatible under matrix multiplication; arguments of ADD must have the same dimensions, and DET is only defined for square matrices. Bold letters represent matrices and regular letters represent scalars.

| Name | Definition | Gradients |
|---|---|---|
| MATMUL | $\boldsymbol{f}(\boldsymbol{M_1}, \boldsymbol{M_2}) = \boldsymbol{M_1}\boldsymbol{M_2}$ | $\frac{\partial C}{\partial \boldsymbol{M_1}} = \boldsymbol{M_2}\frac{\partial C}{\partial \boldsymbol{f}}$ $\frac{\partial C}{\partial \boldsymbol{M_2}} = \frac{\partial C}{\partial f}\boldsymbol{M_1}$ |
| ADD | $\boldsymbol{f}(\boldsymbol{M_1}, \boldsymbol{M_2}) = \boldsymbol{M_1} + \boldsymbol{M_2}$ | $\frac{\partial C}{\partial \boldsymbol{M_1}} = \frac{\partial C}{\partial \boldsymbol{f}}$ $\frac{\partial C}{\partial \boldsymbol{M_2}} = \frac{\partial C}{\partial \boldsymbol{f}}$ |
| SCALE | $\boldsymbol{f}(a, \boldsymbol{M_1}) = a\boldsymbol{M_1}$ | $\frac{\partial C}{\partial \boldsymbol{M_1}} = a\frac{\partial C}{\partial \boldsymbol{f}}$ |
| TRACE | $f(\boldsymbol{M_1}) = \mathrm{Tr}(\boldsymbol{M_1})$ | $\frac{\partial C}{\partial \boldsymbol{M_1}} = \mathbb{1}\frac{\partial C}{\partial f}$ |
| DET | $f(\boldsymbol{M_1}) = \det(\boldsymbol{M_1})$ | $\frac{\partial C}{\partial \boldsymbol{M_1}} = \det(\boldsymbol{M_1})\boldsymbol{M_1}^{-T}\frac{\partial C}{\partial f}$ |
| TRANSPOSE | $\boldsymbol{f}(\boldsymbol{M_1}) = \boldsymbol{M_1}^T$ | $\frac{\partial C}{\partial \boldsymbol{M_1}} = \frac{\partial C}{\partial \boldsymbol{f}}$ |
| CONJUGATE | $\boldsymbol{f}(\boldsymbol{M_1}) = \boldsymbol{M_1}^*$ | $\frac{\partial C}{\partial \boldsymbol{M_1}} = \frac{\partial C}{\partial \boldsymbol{f}}$ |

All cost functions summarized in table 3.1 can be conveniently expressed in terms of common linear-algebra operations. Therefore, it is possible to utilize a linear-algebra-based automatic differentiator to implement a gradient-based quantum optimizer. We describe the details of this implementation in the next chapter.

# CHAPTER 5

# CLOSED-SYSTEM OPTIMAL CONTROL AD IMPLEMENTATION

## 5.1 Introduction

Building on the advantages of automatic differentiation in quantum optimal control, we have implemented a scheme that incorporates constraints via automatic differentiation and utilizes GPUs for boosting computational efficiency. Our quantum optimal control implementation utilizes the TensorFlow library developed by Google's machine intelligence research group [1]. This library is open source, and is being extended and improved upon by an active development community. The simple interface to Python allows non-software professionals to implement high-performance machine learning and optimization applications without excessive overhead. TensorFlow supports GPU and large-scale parallel learning, critical for high-performance optimization.

A crucial factor for recent impressive progress in machine learning has been the leveraging of massive parallelism native to graphics processing units (GPUs) [95, 22, 111, 104, 117]. Similarly, GPUs have been used to accelerate computations in many areas of quantum physics and chemistry [9, 25, 132, 123, 126, 98, 58]. Specifically, GPUs are extremely efficient in multiplying very large matrices [28, 41]. Such multiplications also form a central step in the simulation and optimal control of quantum systems. Exploiting this advantageous feature of GPUs, we achieve significant speed improvements in optimizing control schemes for systems at the current frontiers of experimental quantum computation. As the number of qubits in these experiments is increasing [27, 31, 64], it becomes increasingly important to take advantage of optimal control techniques. Moreover, recent advances in commercially available electronics enabling base-band synthesis of the entire microwave spectrum, afford new capabilities which quantum optimal control is uniquely well-suited to harness.

## 5.2 Implementation

Typical machine-learning applications require most of the same building blocks needed for quantum optimal control. Predefined operations, along with corresponding gradients, include matrix addition, multiplication, matrix traces and vector dot products. We have implemented an efficient kernel for approximate evaluation of the matrix exponential and its gradient. Using these building blocks, we have developed a fast and accurate implementation of quantum optimal control, well-suited to deal with a broad range of engineered quantum systems and realistic treatment of capabilities and limitations of control fields.

In common applications of quantum optimal control, time-evolving the system under the Schrödinger equation – more specifically, approximating the matrix exponential for the propagators $M_j$ at each time step $t_j$ – requires the biggest chunk of computational time. Within our matrix-exponentiation kernel, we approximate $e^{-iH_j\delta t}$ by series expansion, taking into account that the order of the expansion plays a crucial role in maintaining accuracy and unitarity. The required order of the matrix-exponential expansion generally depends on the magnitude of the matrix eigenvalues relative to the size of the time step. General-purpose algorithms such as `expm()` in Python's SciPy framework accept arbitrary matrices $M$ as input, so that the estimation of the spectral radius or matrix norm of $X$, needed for choosing the appropriate order in the expansion, often costs more computational time than the final evaluation of the series approximation itself. Direct series expansion with only a few terms is sufficient for $H_j\delta t$ with spectral radius smaller than 1. In the presence of large eigenvalues, series convergence is slow, and it is more efficient to employ an appropriate form of the "scaling and squaring" strategy, based on the identity

$$\exp X = \left[\exp\left(\frac{X}{2^n}\right)\right]^{2^n}, \tag{5.1}$$

which reduces the spectral range by a factor of $2^n$ at the cost of recursively squaring the

matrix $n$ times [89]. Overall, this strategy leads to an approximation of the short-time propagator of the form

$$M_j \approx \left[\sum_{k=0}^{p} \frac{(-iH_j\delta t/2^n)^k}{k!}\right]^{2^n}, \tag{5.2}$$

based on a Taylor expansion truncated at order $p$. Computational performance could be further improved by employing more sophisticated series expansions [3, 26] and integration methods [59].

As opposed to the challenges of general-purpose matrix exponentiation, matrices involved in a specific quantum-control application with bounded control-field strength ($iH_j\delta t$), will typically exhibit similar spectral radii. Thus, rather than attempting to determine individual truncation levels $p_j$, and performing scaling-and-squaring at level $n_j$ in each time step $t_j$, we make a conservative choice for global $p$ and $n$ at the beginning and employ them throughout. This simple heuristic speeds up matrix exponentiation over the default SciPy implementation significantly, primarily due to leaving out the step of spectral radius estimation.

By default, automatic differentiation computes the gradient of the approximated matrix exponential via backpropagation through the series expansion. However, for sufficiently small spectral radius of $M$, we may approximate [66]

$$\frac{d}{dx}e^{M(x)} \approx M'(x)\, e^{M(x)}, \tag{5.3}$$

neglecting higher-order corrections reflecting that $M'(x)$ and $M(x)$ may not commute. (Higher-order schemes taking into account such additional corrections are discussed in ref. [30].) Equation (5.3) simplifies automatic differentiation: within this approximation, only the same matrix exponential is needed for the evaluation of the gradient. We make use of this in a custom routine for matrix exponentiation and gradient-operator evaluation, further improving the speed and memory performance.

The TensorFlow library currently has one limitation relevant to our implementation of a

quantum optimal control algorithm. Operators and states in Hilbert space have natural resprepresentations as matrices and vectors which are generically complex-valued. TensorFlow, designed primarily for neural network problems, has currently only limited support for complex matrices. For now, we circumvent this obstacle by mapping complex-valued matrices to real matrices via the isomorphism $H \overset{\cong}{\longmapsto} \mathbb{1} \otimes H_{\mathrm{re}} - i\sigma_y \otimes H_{\mathrm{im}}$, and state vectors $\vec{\Psi} \overset{\cong}{\longmapsto} (\vec{\Psi}_{\mathrm{re}}, \vec{\Psi}_{\mathrm{im}})^t$. Here, $\mathbb{1}$ is the $2\times2$ unit matrix and $\sigma_y$ one of the Pauli matrices. Real and imaginary parts of the matrix $H$ are denoted by $H_{\mathrm{re}} = H$ and $H_{\mathrm{im}} = \mathrm{Im}\, H$, respectively; similarly, real and imaginary parts of the state vectors are $\vec{\Psi}_{\mathrm{re}} = \vec{\Psi}$ and $\vec{\Psi}_{\mathrm{im}} = \mathrm{Im}\, \vec{\Psi}$. Written out in explicit block matrix form, this isomorphism results in

$$H\vec{\Psi} \overset{\cong}{\longmapsto} \begin{pmatrix} H_{\mathrm{re}} & -H_{\mathrm{im}} \\ H_{\mathrm{im}} & H_{\mathrm{re}} \end{pmatrix} \begin{pmatrix} \vec{\Psi}_{\mathrm{re}} \\ \vec{\Psi}_{\mathrm{im}} \end{pmatrix}, \tag{5.4}$$

rendering all matrices and vectors real-valued. There are promising indications that future TensorFlow releases may improve complex-number support and eliminate the need for a mapping to real-valued matrices and vectors.

In addition, the optimizer allows for constantly changing the drift and control Hamiltonians. In certain applications, including the 0-$\pi$ qubit, there can be some system parameters that fluctuate due to noise and hence cannot be fixed in the optimization. Instead, we allow the optimizer to simulate the system with a different value for that parameter in every iteration. This in turn changes the drift and control Hamiltonians used in every iteration. Directly applying the gradients from each iteration results in a stochastic gradient descent (SGD) [65, 138] process converging to an average solution over all values of that parameter. Careful tuning of cost-function weights can result in convergence to an almost constant value of the fidelity for all values of the variable parameter.

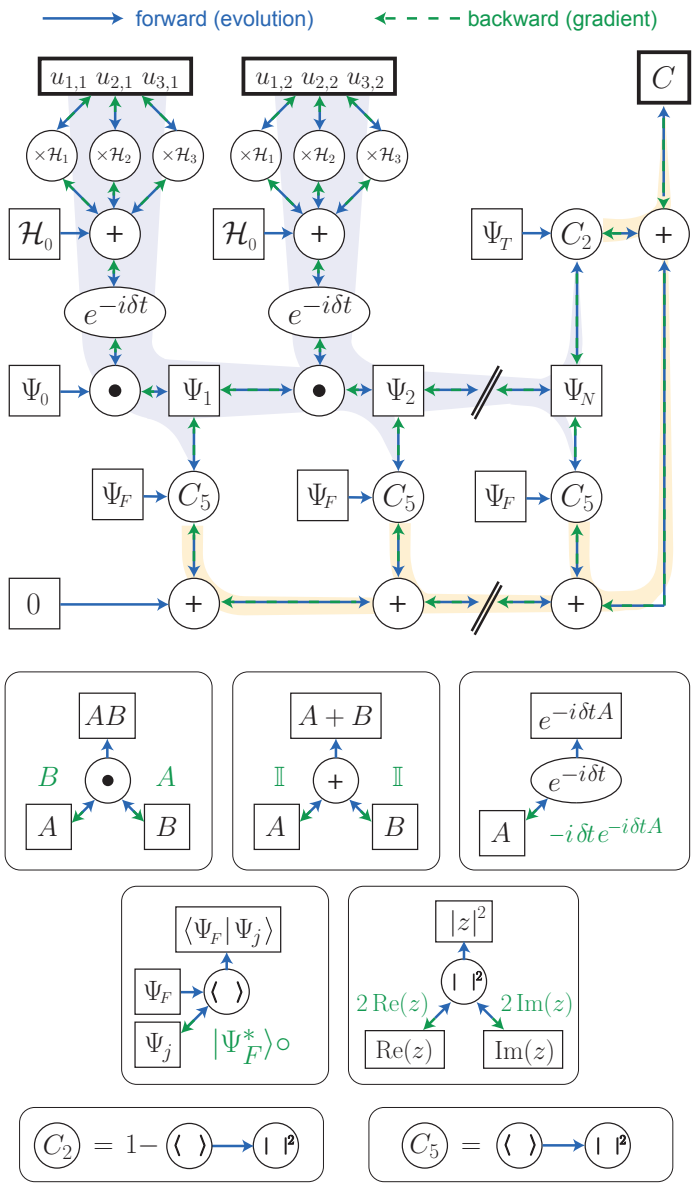


Figure 5.1: Computational network graph for quantum optimal control. Circular nodes in the graph depict elementary operations with known derivatives (matrix multiplication, addition, matrix exponential, trace, inner product, and squared absolute value). Backward propagation for matrices proceeds by matrix multiplication, or where specified, by the Hadamard product $\circ$. In the forward direction, starting from a set of control parameters $u_{k,j}$, the computational graph effects time evolution of a quantum state or unitary, and the simultaneous computation of the cost function $C$. The subsequent "backward propagation" extracts the gradient $\nabla_{\mathbf{u}} C(\mathbf{u})$ with respect to all control fields by reverse-mode automatic differentiation. This algorithm is directly supported by TensorFlow [1], once such a computational network is specified.

## 5.3 Computational graph

Figure 5.1 shows the network graph of operations in our software implementation, realizing quantum optimal control with reverse-mode automatic differentiation. For simplicity, the graph only shows the calculation of the cost functions $C_2$ and $C_5$. The cost function contributions $C_1, C_6$, and $C_7$ are treated in a similar manner. The suppression of large control amplitudes or rapid variations, achieved by $C_3$ and $C_4$, is simple to include, since the calculation of these cost function contributions is based on the control signals themselves and does not involve the time-evolved state or unitary. The host of steps for gradient evaluation is based on basic matrix operations like summation and multiplication.

Reverse-mode automatic differentiation [55] provides an efficient way to carry out time evolution and cost function evaluation by one forward sweep through the computational graph, and calculation of the full gradient by one backward sweep. In contrast to forward accumulation, each derivative is evaluated only once, thus enhancing computational efficiency. The idea of backward propagation is directly related to the GRAPE algorithm for quantum optimal control pioneered by Khaneja and co-workers [66]. While the original GRAPE algorithm bases minimization exclusively on the fidelity of the final evolved unitary or state, advanced cost functions (such as $C_5$ through $C_7$) require the summation of cost contributions from each intermediate step during time evolution of the system. Such cost functions go beyond the usual GRAPE algorithm, but can be included in the more general backward propagation scheme described above.

## 5.4 Performance benchmarking

Obtaining a fair comparison between CPU-based and GPU-based computational performance is notoriously difficult [76]. We attempt to provide a specific comparison under a unified computation framework. TensorFlow allows for straightforward switching from run-

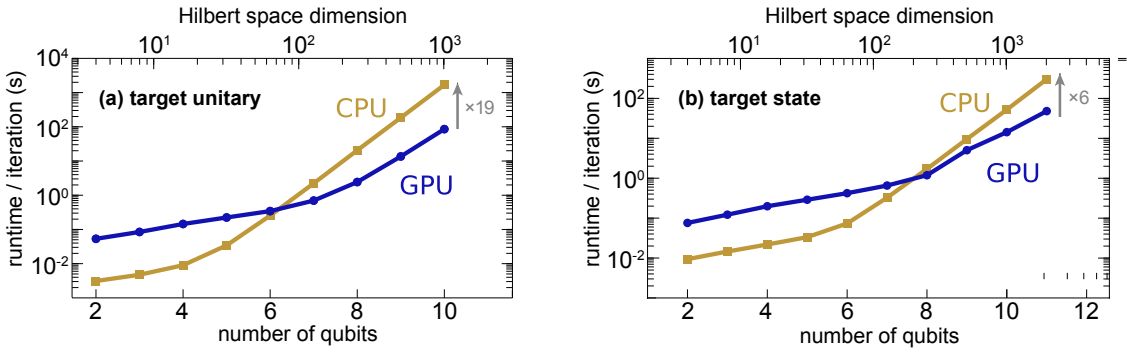


Figure 5.2: Benchmarking comparison between GPU and CPU for (a) a unitary gate (Hadamard transform), and (b) state transfer (GHZ state preparation). Total runtime per iteration scales linearly with the number of time steps. For unitary-gate optimization, the GPU outperforms the CPU for Hilbert space dimensions above $\sim$ 100. For state transfer, GPU benefits set in slightly later, outperforming the CPU-based implementation for Hilbert space dimensions above $\sim$ 300. The physical system we consider, in this case, is an open chain of $N$ spin-1/2 systems with nearest neighbor $\sigma_z\sigma_z$ coupling, and each qubit is controlled via fields $\Omega_x$ and $\Omega_y$.

ning code on a CPU to a GPU. For each operation (matrix multiplication, trace, etc.), we use the default CPU/GPU kernel offered by TensorFlow. Note that properly configured, TensorFlow automatically utilizes all threads available for a given CPU, and GPU utilization is found to be near 100%. Not surprisingly, we observe that the intrinsic parallelism of GPU-based matrix operations allows much more efficient computation beyond a certain Hilbert space size, see Fig. 5.2.

In this example, we specifically inspect how the computational speed scales with the Hilbert space dimension when optimizing an $n$-spin Hadamard transform gate and $n$-spin GHZ state preparation for a coupled chain of spin-1/2 systems presented in Section 6.1.4. (Details of system parameters are described in the same section.) We benchmark the average runtime for a single iteration for various spin-chain sizes and, hence, Hilbert space dimensions. We find that the GPU quickly outperforms the CPU in the unitary gate problem, even for a moderate system size of $\sim$ 100 basis states. For optimization of state transfer, we observe that speedup from GPU usage, relative CPU performance, sets in for slightly larger system sizes of approximately $\sim$ 300 basis states.

The distinct thresholds for the GPU/CPU performance gain stem from the different computational complexities of gate vs. state-transfer optimization. Namely, optimizing unitary gates requires the propagation of a unitary operator (a matrix), involving matrix-matrix multiplications, while optimizing state transfer only requires the propagation of a state (a vector), involving only matrix-vector multiplications:

$$M_j|\Psi\rangle \approx \sum_{k=0}^{p} \frac{(-i\delta t)^k}{k!}(H_j \dots (H_j(H_j|\Psi\rangle))), \tag{5.5}$$

Computing the matrix-vector multiplication is generally much faster than computing the matrix exponential itself [113]. For an $n$-dimensional matrix, the computation of the matrix exponential involves matrix-matrix multiplication, which scales as $O(n^3)$. The computation of state transfer only involves matrix-vector multiplication, which scales as $O(n^2)$ [or even $O(n)$ for sufficiently sparse matrices].

For optimization of the Hadamard transform as well as the GHZ state preparation, we observe a 19-fold GPU speedup for a 10-qubit system (Hilbert space dimension of 1,024) in the former case, and a 6-fold GPU speedup for an 11-qubit system (Hilbert space dimension of 2,048) in the latter case. Since matrix operations are the most computationally intensive task in our software, this speedup is comparable to other GPU application studies that heavily use matrix operations [95, 22, 111, 104, 117, 121, 76]. We emphasize that these numbers are indicative of overall performance trends, but detailed numbers will certainly differ according to the specific system architecture in place. The CPU model we used was an Intel® Core™ i7-6700K CPU @ 4.00 GHz, and the GPU model was an NVIDIA® Tesla® K40c. In this study, all computations are based on dense matrices. Since most physically relevant Hamiltonians are sparse (evolution generally affects sparsity, though), future incorporation of sparse matrices may further improve computation speed for both CPU and GPU [6, 80].

## 5.5 User manual

Our software implementation can be downloaded at: github.com/SchusterLab/quantum-optimal-control. In this section, we highlight the main parameters that a user should supply to the software along with their recommended values. To call the optimizer, the most comprehensive format is

`uks, U_final = Grape(H0,Hops,Hnames,U,total_time,steps,states_concerned_list, convergence, U0, reg_coeffs,dressed_info, maxA, use_gpu, draw, initial_guess, H_time_scales, show_plots, unitary_error, method,state_transfer, no_scaling, freq_unit, file_name, save, data_path)` . If convergence is achieved, the optimizer returns the optimized control pulses, `uks`, and the final unitary acheived `U_final`. We will explain each of the arguments and the values suggested for them in the next section.

### *5.5.1 Mandatory arguments*

The arguments that must be given to the optimizer are

1. `H0`: Drift Hamiltonian (a square $d \times d$ matrix).
2. `Hops`: A list of control Hamiltonians (each is $d \times d$).
3. `Hnames`: A list of control Hamiltonian labels (strings), with the same length as `Hops`.
4. `U`: Target Unitary ($d \times d$) if `state_transfer == False`, or $k$ vectors ($d \times k$) where $k$ is the number of initial states desired if `state_transfer == True`.
5. `total_time`: Total Time of the gate (float).
6. `steps`: Number of time steps (int).
7. `states_concerned_list`: A list of the indices of initial states (e.g [0,1]).

### *5.5.2 Optional arguments*

1. `U0`: Initial Unitary ($d \times d$), default is the identity.

2. `convergence`: A dictionary with convergence conditions. It includes the following parameters with default values as indicated: `convergence = {'rate':0.01, 'update_step':100, 'max_iterations':5000, 'conv_target':1e-8, 'learning_rate_decay':2500,'min_grad': 1e-25}` where

   - `rate`: Learning rate, the size of the step $\eta$ the optimizer applies to control pulses, $\mathbf{u}' = \mathbf{u} - \eta \nabla_{\mathbf{u}} C(\mathbf{u})$.
   - `update_step`: the number of iterations without updating the user with plots or text.
   - `max_iterations`: maximum number of iterations to perform in total.
   - `conv_target`: the target infidelity error.
   - `learning_rate_decay`: A parameter that specifies the exponential decay of the learning rate as needed for good convergence. It should be around half the value of maximum iterations.
   - `min_grad`: The minimum value of summed squared gradients below which optimization is halted.

3. `Initial_guess`: A list of $k$ elements, each of them of length `steps`, defining the initial guesses for all control pulses. If not provided, a random pulse drawn from a Gaussian distribution (mean = 0 and std = sqrt(`maxA`)) is used.

4. `reg_coeffs`: A dictionary of coefficients for the different cost functions. Details of the different cost functions and their suggested values are outlined separately in the next section.

5. `dressed_info` : A dictionary including the eigenvalues and eigenstates of dressed states if they are to be used in the simulation.

6. `maxA`: A list of $k$ numbers, the maximum amplitudes of the control pulses.

7. `use_gpu`: A boolean switch between GPU and CPU usage, default is True.

8. `sparse_H, sparse_U`: Booleans specifying whether Hamiltonians and unitary operators are sparse. Speedup is expected if the corresponding sparsity is satisfied. Only available in CPU mode.

9. `use_inter_vecs`: A boolean to enable/disable the involvement of state evolution in graph building. If False, no intermediate level occupations will be calculated or stored.

10. `draw`: A list including the indices and names for the states to be included in state occupation plots. Ex: `states_draw_list = [0,1]`, `states_draw_names = ['|0⟩','|1⟩']` and `draw = [states_draw_list,states_draw_names]`. Default value is to plot states with indices 0-3.

11. `show_plots`: A boolean (default is True) to switch on/off the display of plots for updating the user with optimization progress.

12. `state_transfer`: A boolean (default is False) switching between state transfer and full unitary evolution.

13. `method`: The numerical method for optimization. Options are 'ADAM', 'BFGS', 'L-BFGS-B' or 'EVOLVE'. Default is 'ADAM'. 'EVOLVE' only simulates the time evolution without optimizing.

14. `Unitary_error`: A float indicating the desired maximum error of the Taylor expansion of the exponential. It is used to choose a proper number of expansion terms. Default target unitary error is 1e-4.

15. `no_scaling`: A boolean (default is False) to disable scaling and squaring in the matrix exponential.

16. `Taylor_terms`: A list of integers as [Taylor expansion terms, scaling and squaring terms], to manually choose the number of Taylor terms for matrix exponentials.

17. `freq_unit`: A string to indicate the units of the given Hamiltonians. Default is 'GHz'. Can only take the following values: 'GHz', 'MHz', 'kHz' or 'Hz'.

18. `save`: A boolean (default is True) to indicate whether the user wants optimization data to be saved. Saved data include control pulses, intermediate vectors and final unitary at every update step.

19. `file_name`: A name for the file where optimization data will be saved.

20. `data_path`: Path for saving the simulation. Default is current directory.

### *5.5.3 Cost functions and their weights*

The currently implemented cost functions are

1. The infidelity cost function: It is defined as the overlap between the target unitary/final state and the achieved unitary/final state. It has a fixed weight of 1 in the code, i.e. all the other weights are normalized relative to it.

2. The Gaussian envelope cost function: A penalty is applied if the control pulses do not have a Gaussian envelope. The user supplies a coefficient called `envelope` in the `reg_coeffs` input. A value of 0.01 is found to be a good starting value empirically.

3. The first derivative cost function: To make the control pulses smoother, the optimizer can penalize the summed squares of pulse first derivatives. The user supplies a coefficient called `dwdt` in the `reg_coeffs` input as its relative weight. A value of 0.001 is found to be a good starting value empirically.

4. The second derivative cost function: A penalty on the summed squares of pulse second derivatives can also be activated. The user supplies a coefficient called `d2wdt2` in the `reg_coeffs` input as its relative weight. A value of 0.000001 is found to be a good starting value empirically.

5. The forbidden-state cost function: It is a cost function to forbid the quantum occupation of certain levels throughout the time of the gate. The user supplies a coefficient called `forbidden` (starting value is around 100 empirically) and a list called `states_forbidden_list` inside the `reg_coeffs` dictionary specifying the indices of the levels to forbid.

6. The time-optimal cost function: If the user wants to speed up the gate, they should provide a coefficient called `speed_up` (starting value is around 100) to award the occupation of the target state at all intermediate states, hence, making the gate as fast as possible.

All of these coefficients should be tweaked by the user to emphasize certain aspects of the optimization they want. In practice, the coefficients and learning rate are found to be the most crucial inputs to be adjusted for better convergence. Starting with the default values in the `reg_coeffs` dictionary along with a learning rate that is between 0.001 and 0.01 achieved good results for most of our applications. The user should keep an eye on how the different optimization targets are behaving as the iterations are processed, and should increase or decrease the relative weights accordingly.

# CHAPTER 6

# CLOSED-SYSTEM OPTIMAL CONTROL APPLICATIONS

In this section, we present a set of example applications of experimental relevance for spin and transmon qubits. The first application demonstrates the importance of cost functions suppressing intermediate occupation of higher-lying states during time evolution, as well as cost functions accounting for realistic pulse shaping capabilities. In a second application, we show how the cost function $C_6$ can yield high-fidelity state transfer within a reduced time interval. Third, we discuss the application of Schrödinger-cat state preparation – an example from the context of quantum optics and of significant interest in recent schemes aiming at quantum information processing based on such states [56, 88, 125]. This application combines considerable system size with a large number of time steps, and utilizes most of the cost functions discussed in Section 3.4. In the fourth application, we demonstrate the algorithm performance in finding optimal solutions for GHZ state preparation and implementation of a Hadamard transform gate in a chain of qubits with a variable number of qubits. We use either the Adam [67] or L-BFGS-B optimization algorithm [21] for pulse optimization, and achieve a minimum fidelity of 99.9% in all transmon examples.

## 6.1 Transmon and spin applications

### *6.1.1 CNOT gate for two transmon qubits*

In the first example, we study realization of a 2-qubit CNOT gate in a system of two coupled, weakly anharmonic transmon qubits. For each transmon qubit ($j = 1, 2$) [71], we take into account the lowest two states spanning the qubit computational space, as well as the next three higher levels. The system Hamiltonian, including the control fields

$\{\Omega_{x_1}(t),\, \Omega_{x_2}(t),\, \Omega_{z_2}(t)\}$, then reads

$$
\begin{aligned}
H(t) = & \sum_{j=1,2} \left[ \omega_j b_j^\dagger b_j + \tfrac{1}{2}\alpha_j\, b_j^\dagger b_j (b_j^\dagger b_j - 1) \right] \\
& + J(b_1 + b_1^\dagger)(b_2 + b_2^\dagger) \\
& + \Omega_{x_1}(t)(b_1 + b_1^\dagger) + \Omega_{x_2}(t)(b_2 + b_2^\dagger) + \Omega_{z_2}(t) b_2^\dagger b_2 .
\end{aligned}
\tag{6.1}
$$

Here, the ladder operators $b_j$, and $b_j^\dagger$ are truncated at the appropriate level. (The qubit frequencies $\omega_j/2\pi$ are chosen as 3.5 and 3.9 GHz, respectively; both transmons have an anharmonicity of $\alpha/2\pi = -225\,\mathrm{MHz}$; and the qubit-qubit coupling strength used in the simulation is $J/2\pi = 100\,\mathrm{MHz}$.) Consistent with recent circuit QED experiments utilizing classical drives as well as parametric modulation, we investigate control fields acting on $H_{x_1} = b_1 + b_1^\dagger$, $H_{x_2} = b_2 + b_2^\dagger$, and $H_{z_2} = b_2^\dagger b_2$.

We next optimize control fields for the realization of a CNOT gate, with transmon qubit $j = 1$ acting as the control qubit. Our control-field optimization reaches a prescribed fidelity of 99.9% for a 10 ns gate duration in all cases, as seen in Fig. 6.1. Results shown in Fig. 6.1(a) are obtained with the standard target-gate infidelity cost function ($C_1$) only. It is evident that the solution encounters two issues: the occupation of the 3rd and 4th excited transmon level ("forbidden") is significant, and control fields are polluted by high-frequency components. Including a cost function contribution of type $C_5$ succeeds in strongly suppressing occupation of higher levels, see Fig. 6.1(b). This both reduces exposure to increased relaxation rates and ensures that the evolution is minimally influenced by our numerical truncation of Hilbert space. In the final improvement step, shown in Fig. 6.1(c), our optimization additionally suppresses excessive control amplitudes and derivatives via cost contributions of type $C_3$ and $C_4$. The inclusion of these terms in the overall cost lessens superfluous "noise" in the control signals, and also helps improve convergence of the algorithm – without reducing the achieved target-gate fidelity.

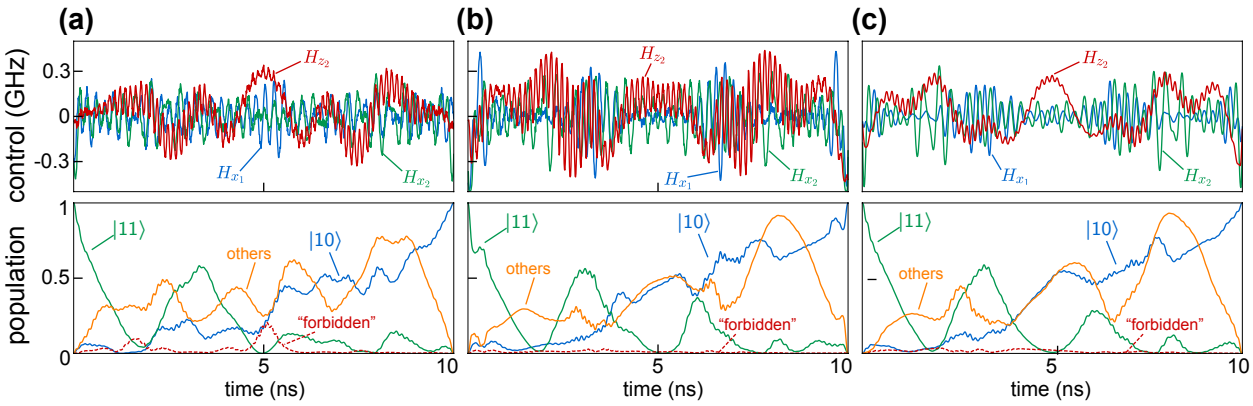


Figure 6.1: Control pulses and evolution of quantum state population for a CNOT gate acting on two transmon qubits, (a) only targeting the desired final unitary, (b) employing an additional cost function suppressing occupation of higher-lying states ($C_5$), and (c) including additional pulse-shape cost functions ($C_3, C_4$). Here, only the evolution of state $|11\rangle$ is shown, as the evolution of state $|11\rangle$ is most susceptible to the occupation of higher level states. In all three cases, the CNOT gate converged to a fidelity of 99.9%. The results differ in important details: in (a), both high-frequency "noise" on the control signals and significant occupation of "forbidden" states (3rd and 4th excited transmon level), shown as dashed red line, are visibile throughout the evolution; in (b), forbidden-state occupation is suppressed at each time step during evolution; in (c), this suppression is maintained and all control signals are smoothened. The maximum occupation of forbidden states is reduced from $\sim 20\%$ in (a) to $\sim 3\%$ in (b) and (c). The population of "others" states (non- $|11\rangle$, $|10\rangle$ or "forbidden") is also shown for completeness. For demonstration purposes, all three examples use the same gate duration of 10 ns, despite being subject to different constraints. In practice, one would typically increase the gate time for a more constrained problem to achieve the best result in maximizing gate fidelity, minimizing forbidden state occupation, and achieving a realistic control signal.

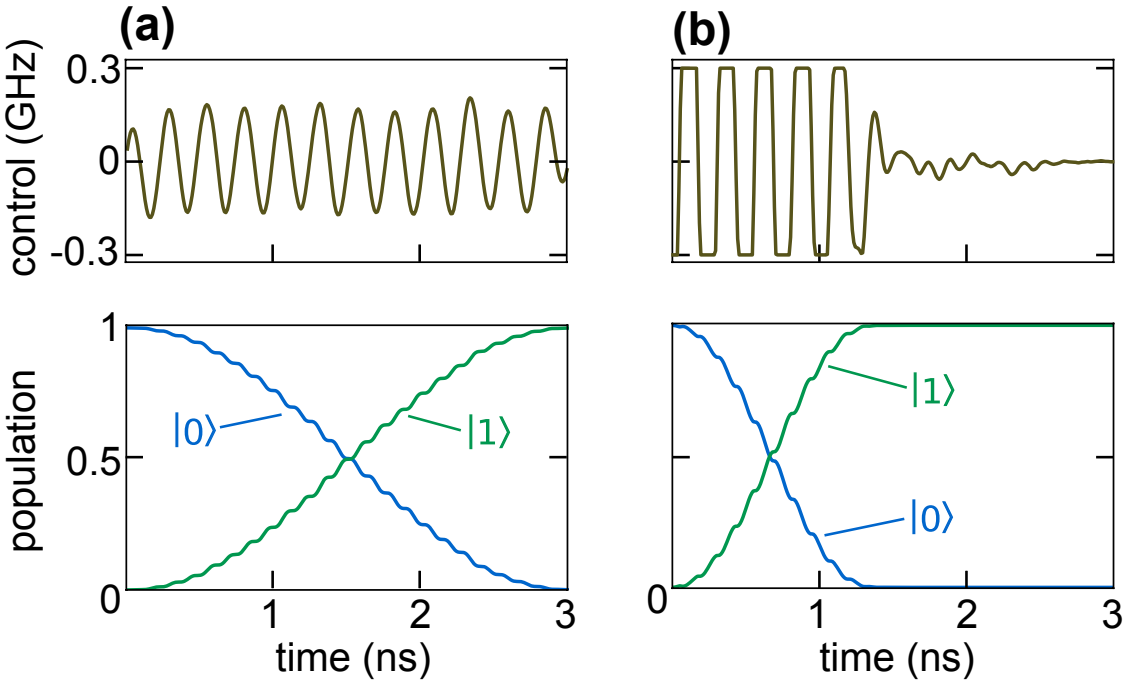


Figure 6.2: Minimizing evolution time needed for a high-fidelity state transfer. (a) No time-optimal award function. (b) With time-optimal award function. (a) Without penalty for the time required for the gate, the control field spreads across the entire given time interval. (b) Once evolution over a longer time duration is penalized with a contribution of type $C_6$ or $C_7$ (see table I), the optimizer achieves target state preparation in a shorter time, without loss of fidelity.

### *6.1.2 Reducing duration of $|0\rangle$ to $|1\rangle$ state transfer*

In this second example, we illustrate the use of cost function contributions (types $C_6$, $C_7$) in minimizing the time needed to perform a specific gate or prepare a desired state. To this end, we consider a two-level spin qubit ($\omega/2\pi$: 3.9 GHz). The system and control Hamiltonians combined are taken to be

$$H = \frac{\omega}{2}\sigma_z + \Omega(t)\,\sigma_x. \tag{6.2}$$

We allow for a control field acting on the qubit $\sigma_x$ degree of freedom, and constrain the maximum control-field strength $\Omega_{\max}/2\pi$ to 300 MHz. When the evolution time needed to perform the state transfer is fixed (rather than subject to optimization itself), we observe that control fields generically spread across the prescribed gate duration time. The desired target

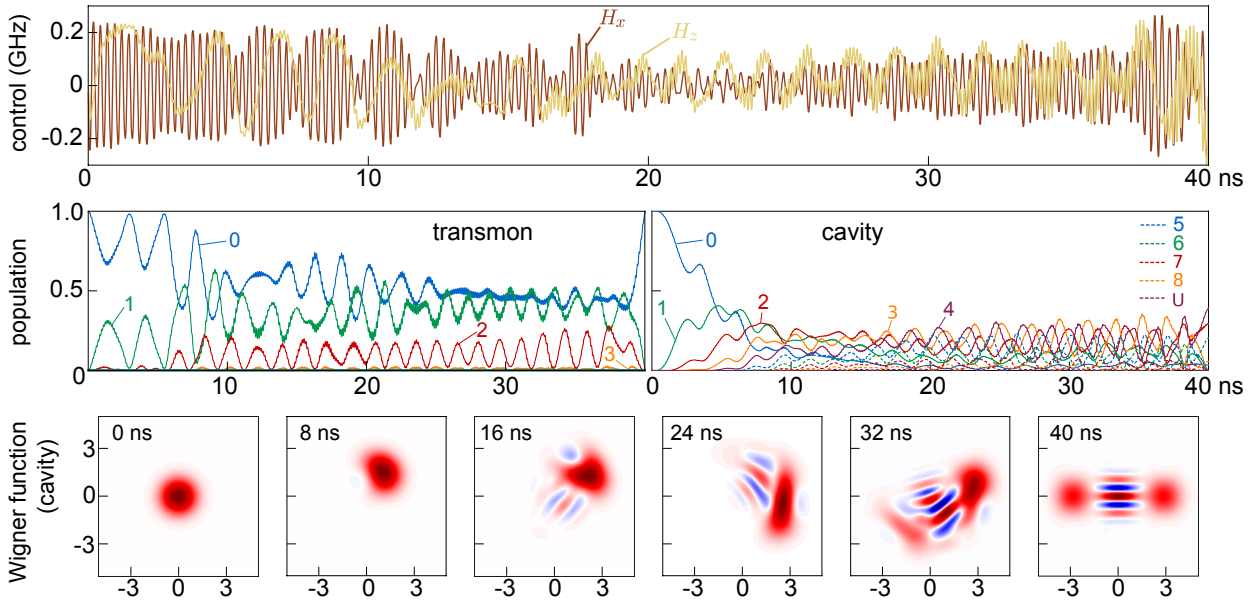


Figure 6.3: Cat state generation. Control pulse, state evolution in Fock basis, and Wigner function tomography of the cavity evolution. Photonic cat state generation is shown as a test of state transfer, challenging the quantum control algorithm with a system of considerable size, large number of required time steps, and inclusion of multiple types of cost function. The desired Schrödinger cat state in the resonator is created indirectly, by applying control fields to a transmon qubit coupled to the resonator, and reached within a prescribed evolution time of 40 ns with a fidelity of 99.9%. (Note that occupation of transmon level 4, 5, 6 remains too small to be visible in the graph.)

state is realized only at the very end of the allowed gate duration. When we incorporate a $C_6$ or $C_7$-type cost contribution, the optimal control algorithm also aims to minimize the overall gate duration, so as to realize the target unitary or state in as short a time as possible, given other active constraints. In our example, this reduces the time for a state transfer from 3 ns to less than 1.5 ns, see Fig. 6.2. We note that it is further possible to adaptively change the overall simulation time during optimization. For instance, if further optimization was desired in the case of Fig. 6.2(b), then the simulation time interval could be adaptively reduced to $\sim 1.5$ ns – resulting in a significant cutback in overall computation time.

### *6.1.3 Generating photonic Schrödinger cat states*

As an example of quantum state transfer, we employ our optimal control algorithm to the task of generating a photonic Schrödinger-cat state. The system we consider to this end is a realistic, and recently studied [88, 125] circuit QED setup, consisting of a transmon qubit capacitively coupled to a three-dimensional microwave cavity. External control fields are restricted to the qubit. Working in a truncated subspace for the transmon (limiting ourselves to levels with energies well below the maximum of the cosine potential), the full Hamiltonian describing the system is

$$\begin{aligned} H(t) = \omega_q b^\dagger b + \tfrac{1}{2}\alpha\, b^\dagger b(b^\dagger b - 1) + \omega_r a^\dagger a \\ + g(a + a^\dagger)(b + b^\dagger) + \Omega_x(t)(b + b^\dagger) + \Omega_z(t) b^\dagger b \end{aligned} \tag{6.3}$$

Here, $a$ and $b$ are the usual lowering operators for photon number and transmon excitation number, respectively. The frequencies $\omega_q/2\pi = 3.5\,\mathrm{GHz}$ and $\alpha/2\pi = -225\,\mathrm{MHz}$ denote the transmon 0-1 splitting and its anharmonicity. The frequency of the relevant cavity mode is taken to be $\omega_r/2\pi = 3.9\,\mathrm{GHz}$. Qubit and cavity are coupled, with a strength parameterized by $g/2\pi = 100\,\mathrm{MHz}$. In our simulation, the overall dimension is $154 =$ (7 transmon levels) $\times$ (22 resonator levels). Note that the rotating wave approximation is not applied in order to reflect the capabilities of modern arbitrary waveform generation.

The state-transfer task at hand, now, is to drive the joint system from the zero-excitation state $|0\rangle_q \otimes |0\rangle_r$ (the ground state if counter-rotating terms in the coupling are neglected) to the photonic cat state $|0\rangle_q \otimes |\mathrm{cat}\rangle_r$. Here, the cat state in the resonator corresponds to a superposition of two diametrically displaced coherent states: $|\mathrm{cat}\rangle_r = \frac{1}{\sqrt{2}}(|\lambda\rangle + |-\lambda\rangle)$. Coherent states are defined in the usual way as normalized eigenstates of the photon annihilation operator $a$, and correspond to displaced vacuum states $|\lambda\rangle = e^{-|\lambda|^2/2} e^{\lambda a^\dagger}|0\rangle$. The cat state $|\mathrm{cat}\rangle_r$ is approximately normalized for sufficiently large $\lambda$. As our concrete target state,

we choose a cat state with amplitude $\lambda = 2$ (normalization error of $\sim 0.03\%$). The state transfer is to be implemented by control fields $\Omega_x(t)$ and $\Omega_z(t)$ acting on the transverse and longitudinal qubit degrees of freedom, $H_x = (b + b^\dagger)$ and $H_z = b^\dagger b$, respectively. Matching experimental realizations of multi-mode cavity QED systems [87], we do not allow for any direct control of the cavity degrees of freedom.

This state-transfer problem provides an excellent test for an optimal control algorithm. It incorporates the simultaneous challenges of a large number of time steps (8,000), a considerable evolution time (40 ns), and the application of most of the cost functions we discussed in Sect. 3.4 and summarized in Table 3.1. Specifically, in addition to minimizing the target state infidelity ($C_2$), we penalize occupation of transmon levels 3 to 6 and cavity levels 20 and 21 ($C_5$) to avoid artifacts from truncation, and penalize control variations ($C_4$) [1]. Results from the optimization are presented in Fig. 6.3, which shows the control-field sequence, as well as the induced state evolution. At the end of the 40 ns time interval, the control fields generate the desired cat state with a fidelity of 99.9%. The maximum populations at the truncation levels of transmon and cavity are $\sim 6 \times 10^{-6}$ and $\sim 7 \times 10^{-10}$, respectively. We independently confirm convergence with respect to truncation by simulating the obtained optimized pulse for enlarged Hilbert space (8 transmon and 23 cavity levels), and find that the evolution continues to reach the target state with 99.9% fidelity.

### *6.1.4 Hadamard transform and GHZ state preparation*

We present a final set of examples illustrating the algorithm performance for increasing system size. To that end, we consider a coupled chain of $N$ qubits, or spin-1/2 systems. We assume that all spins are on-resonance in the multiple-rotating frame. This system is

1. A cost function for reducing evolution time ($C_7$) was not included in this example.

described by the Hamiltonian

$$H(t) = \sum_{n=1}^{N} \left[ \Omega_x^{(n)}(t)\sigma_x^{(n)} + \Omega_y^{(n)}(t)\sigma_y^{(n)} + J\,\sigma_z^{(n)}\sigma_z^{(n+1)} \right], \tag{6.4}$$

where the coupling term is understood to be dropped for the final summand ($n = N$). The qubit-qubit coupling strength is fixed to $J/2\pi = 100\,\mathrm{MHz}$. Each qubit ($n$) is controlled via fields $\Omega_x^{(n)}$ and $\Omega_y^{(n)}$, with a maximum allowed drive strength of $\Omega_{x,y}^{(n)}/2\pi = 500\,\mathrm{MHz}$.

As a first optimization task, we search for control fields to implement the unitary realizing a Hadamard transform, commonly used in various quantum algorithms. The gate time we allow for the Hadamard transform is $(2N)\,\mathrm{ns}$, simulated with $10N$ time steps. Figure 6.4(a) shows the number of iterations and wall-clock time required to converge to the desired 99.9% process fidelity. For the same spin-chain system, we have also employed our code to optimize control fields for transferring the system ground state to a maximally entangled GHZ state. The overall time and time steps we allow for the GHZ state preparation is identical to that used for the Hadamard transform gate. Figure 6.4(b) shows the number of iterations necessary and the total wall-clock time spent for reaching convergence to a result with 99.9% state fidelity. For both examples, we employed computation on either CPU or GPU, depending which one is faster. (This performance benchmarking data was shown in Section 5.4). We note that, when using a modest desktop PC with a graphics card, optimal control problems for small Hilbert space size converge within seconds. For a 10-qubit Hadamard gate (Hilbert space dimension of 1,024) or 11-qubit GHZ state (Hilbert space dimension of 2048), it takes $\sim$1 day to obtain a solution meeting the 99.9% fidelity threshold. The total wall-clock time could likely have been reduced significantly by appropriate choice of optimizer and/or initial control fields. In the case of a spin-chain system, like many quantum information systems, as the number of elements increase, not only does the Hilbert space grow exponentially, the number of control fields and the required number of time steps also get larger. This further increases the complexity of the problem.

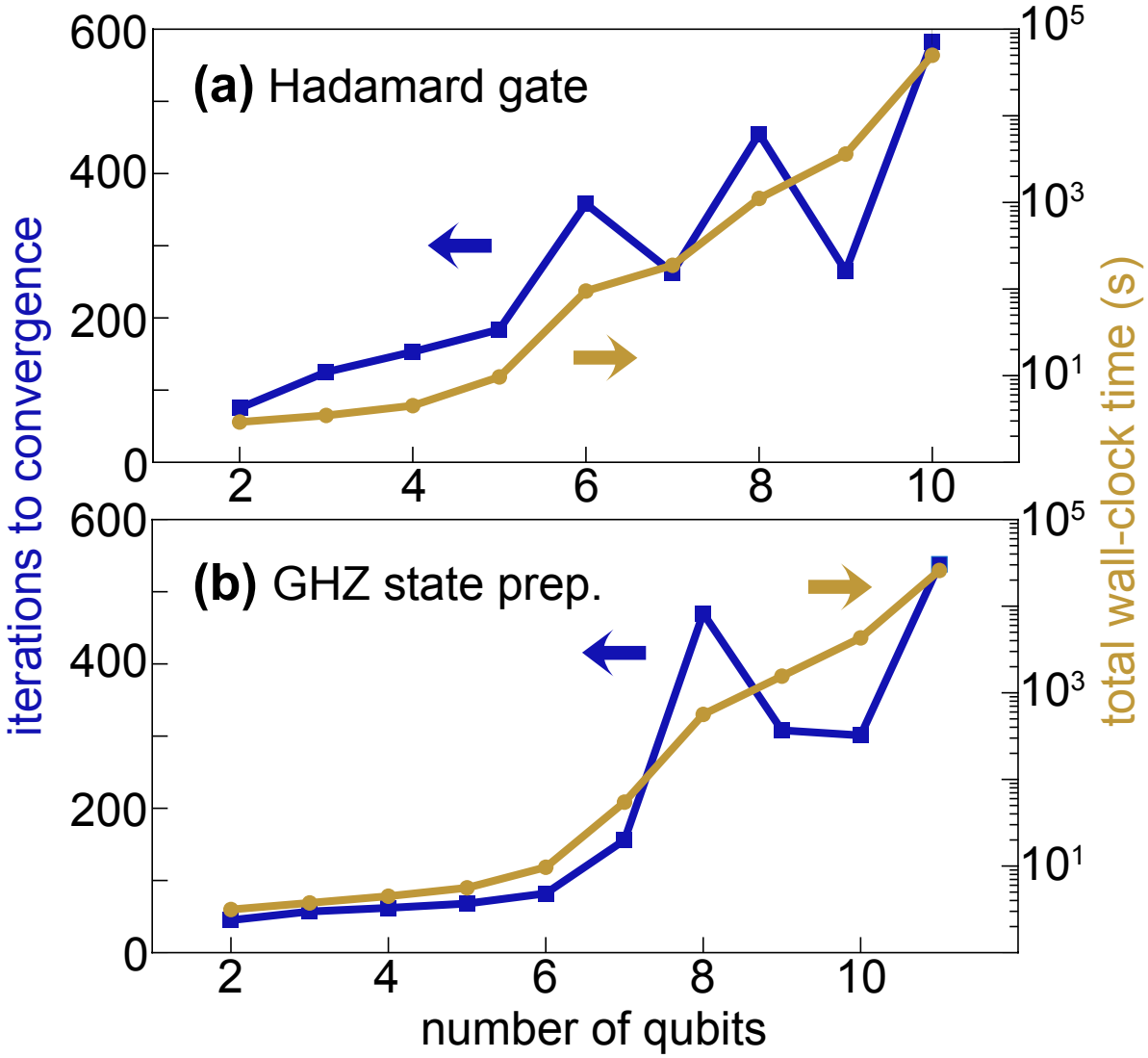


Figure 6.4: Performance of optimal control algorithm as a function of qubit number for (a) a Hadamard transform gate, and (b) GHZ state preparation. As system size increases, total time and number of iterations for the algorithm grow rapidly. The larger number of control parameters and complexity of the target state add to the challenge of quantum optimal control for systems with many degrees of freedom.

## 6.2 Protected qubits applications

### *6.2.1 Fluxonium gates*

As discussed in section 2.5, the heavy fluxonium combines strong Josephson non-linearity with $T_1$ protection due to disjoint support of its lowest-lying localized wave functions. The protection granted by disjoint state support, however, also complicates the realization of universal gate operations by means of external microwave pulses: matrix elements for direct transitions between disjoint-support states remain exponentially suppressed. In this section, we show that optimal control algorithms can nevertheless yield efficient protocols for a universal gate set. Such protocols necessitate involvement of higher qubit levels, and we carefully evaluate fidelity limitations arising from temporary occupation of these states.

Experimentally, gates for heavy fluxonium have been realized by driving Raman transitions [127, 35], which utilize intermediary higher-energy states to assist indirect transitions between the protected states. We will demonstrate a similar approach, exploiting the availability of intermediary state transitions using optimal control theory. The optimal-control formalism offers greater flexibility in terms of pulse shape, and yields fast, high-fidelity single-qubit gates with gate times below 100 ns and fidelities exceeding 99.9%. We obtain optimized pulse shapes for $X$, $H$, and $T$ gates, thereby establishing a blueprint for realizing arbitrary single-qubit gate operations.

As typical in circuit QED [128, 8], each gate is realized by a microwave pulse applied to a transmission-line resonator which, in turn, is coupled to the qubit. The corresponding Hamiltonian for this driven, generalized Jaynes-Cummings model is

$$H_{JC} = \sum_l \epsilon_l \left|l\right\rangle\!\left\langle l\right| + \omega_r a^\dagger a + \sum_{l,l'} g_{ll'} \left|l\right\rangle\!\left\langle l'\right| (a^\dagger + a) + v(t)(a^\dagger + a), \tag{6.5}$$

where $\epsilon_l$, $|l\rangle$ are the fluxonium eigenenergies and eigenstates labeled by index $l$, and $\omega_r$ is the

resonator frequency. The relative coupling strengths are given by $g_{ll'} = g \langle l | n_\phi | l' \rangle$, where $n_\phi$ is the fluxonium charge operator. Fluxonium eigenenergies and eigenstates are governed by the Hamiltonian [85, 70]

$$H_f = 4E_C n_\phi^2 + \frac{1}{2} E_L \phi^2 - E_J \cos(\phi + 2\pi\Phi_{\text{ext}}/\Phi_0), \tag{6.6}$$

in which $E_C$, $E_L$, and $E_J$ denote the capacitive, inductive, and Josephson energies, respectively. $\Phi_{\text{ext}}$ is the external magnetic flux threading the loop formed by the junction and inductor. For the heavy-fluxonium qubit, we choose realistic device parameters $E_C/h = 0.5\,\text{GHz}$ , $E_L/h = 0.25\,\text{GHz}$, and $E_J/h = 4.0\,\text{GHz}$, and a flux working point slightly away from half-integer flux, $\Phi_{\text{ext}} = 0.45\Phi_0$. This places the system in the protected regime of nearly degenerate states $|0\rangle$ and $|1\rangle$ with disjoint support, see Fig. 6.5. (Operating the qubit away from the half-integer flux sweet spot increases sensitivity to dephasing from $1/f$ flux noise.)

Throughout this work, we focus on dispersive control of the qubit, in which the drive tone $v(t)$ steers dynamics within the qubit subsystem, but leaves the resonator state essentially unchanged. This allows us to exclude the resonator subspace from explicit simulation within the optimal-control algorithm. We verified in a separate simulation that the resonator state is unaffected by the drive tone by checking that the resonator average photon number obeys $\langle a^\dagger a \rangle \ll 1$ during the gate. In the resulting driven-fluxonium Hamiltonian

$$H(t) = H_f + V(t), \tag{6.7}$$

we properly account for the fact that the drive on the qubit is filtered through the resonator. The dispersive coupling between qubit and resonator produces an effective drive on the qubit

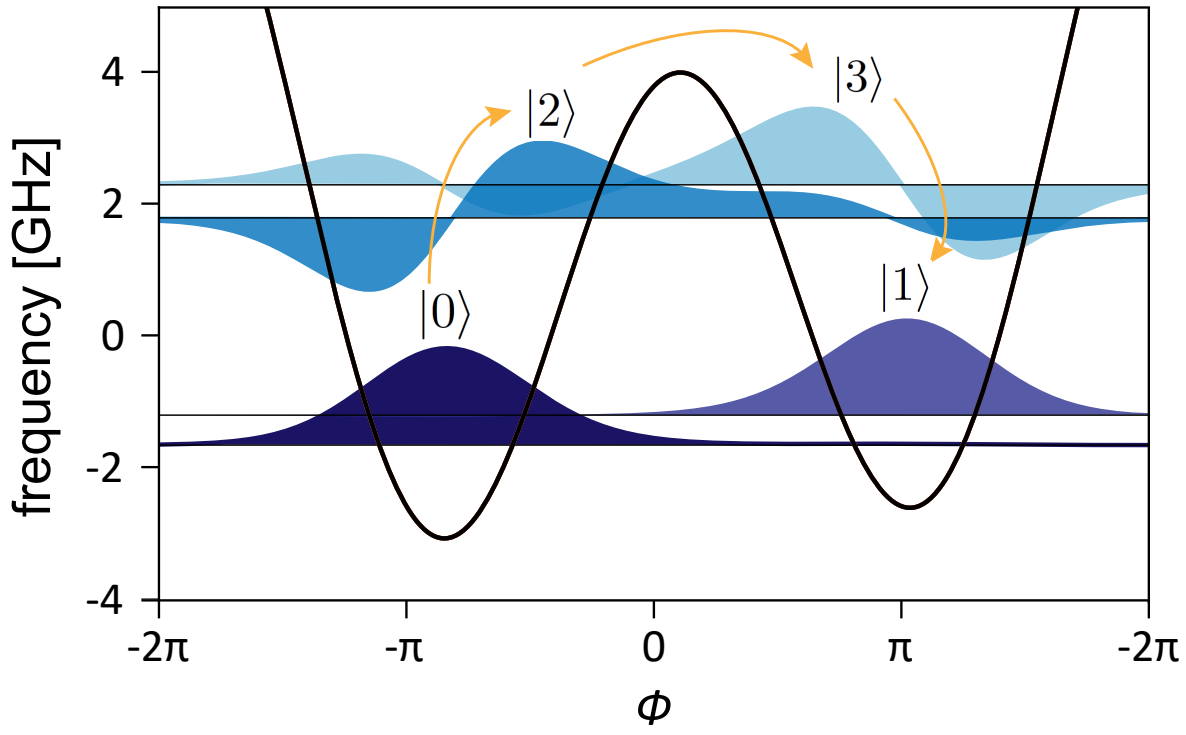


Figure 6.5: First four fluxonium wave functions, slightly away from the flux sweet spot ($\Phi_{\text{ext}} = 0.45\Phi_0$). The two lowest-lying states $|0\rangle$ and $|1\rangle$ are localized and have practically disjoint support. The auxiliary states $|2\rangle$ and $|3\rangle$ delocalize over both potential wells and serve as intermediate states for quantum gates. Gates involving population transfer between $|0\rangle$ and $|1\rangle$ such as X or H gates utilize the delocalized states for transfer across the potential barrier.

of the form

$$\hat{V}(t) = 2g\omega_r v(t) \sum_{l,l'} \frac{\langle l| \hat{n}_\phi |l'\rangle}{(\epsilon_l - \epsilon_{l'})^2 - \omega_r^2} |l\rangle\langle l'| \,, \tag{6.8}$$

For our simulation, we consider a coupling strength and resonator frequency of $g/2\pi = 300$ MHz and $\omega_r/2\pi = 7.5$ GHz, respectively.

## Single qubit gates

Using closed-system optimal control, we optimize the control pulse $v(t)$ to realize three different single-qubit gates: the Pauli-X gate, Hadamard gate, and the T gate,

$$\mathrm{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \mathrm{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad \mathrm{T} = \begin{pmatrix} 1 & 0 \\ 0 & e^{-\pi/4} \end{pmatrix}.$$

The latter two gates are known to form a universal set of single-qubit gates [93]. Optimization must balance two conflicting requirements: gate times $t_g$ should be as short as possible to minimize the influence of dissipation and dephasing; at the same time, the maximum pulse amplitude $\max|v(t)|$ must remain small enough to avoid population of the resonator with unwanted photons. We find that pulses with $t_g$ on the order of a few tens of nanoseconds satisfy these conditions while also producing gates with high fidelities. In addition to the cost-functional contribution, quantifying the target-gate infidelity, we employ additional pulse-shaping cost contributions to limit the time derivatives and maximum amplitude of the pulse $v(t)$. Suppressing the maximum amplitude ensures that occupation of the resonator with spurious photons is minimized. The cost on pulse derivatives helps eliminate unnecessary high-frequency components of $v(t)$ and render the pulses as smooth as possible, which is important for experimental applications, since instruments generating these control fields have a finite impulse response.

The pulses we obtain have a gate duration of $t_g = 60\,\mathrm{ns}$ and closed system fidelities

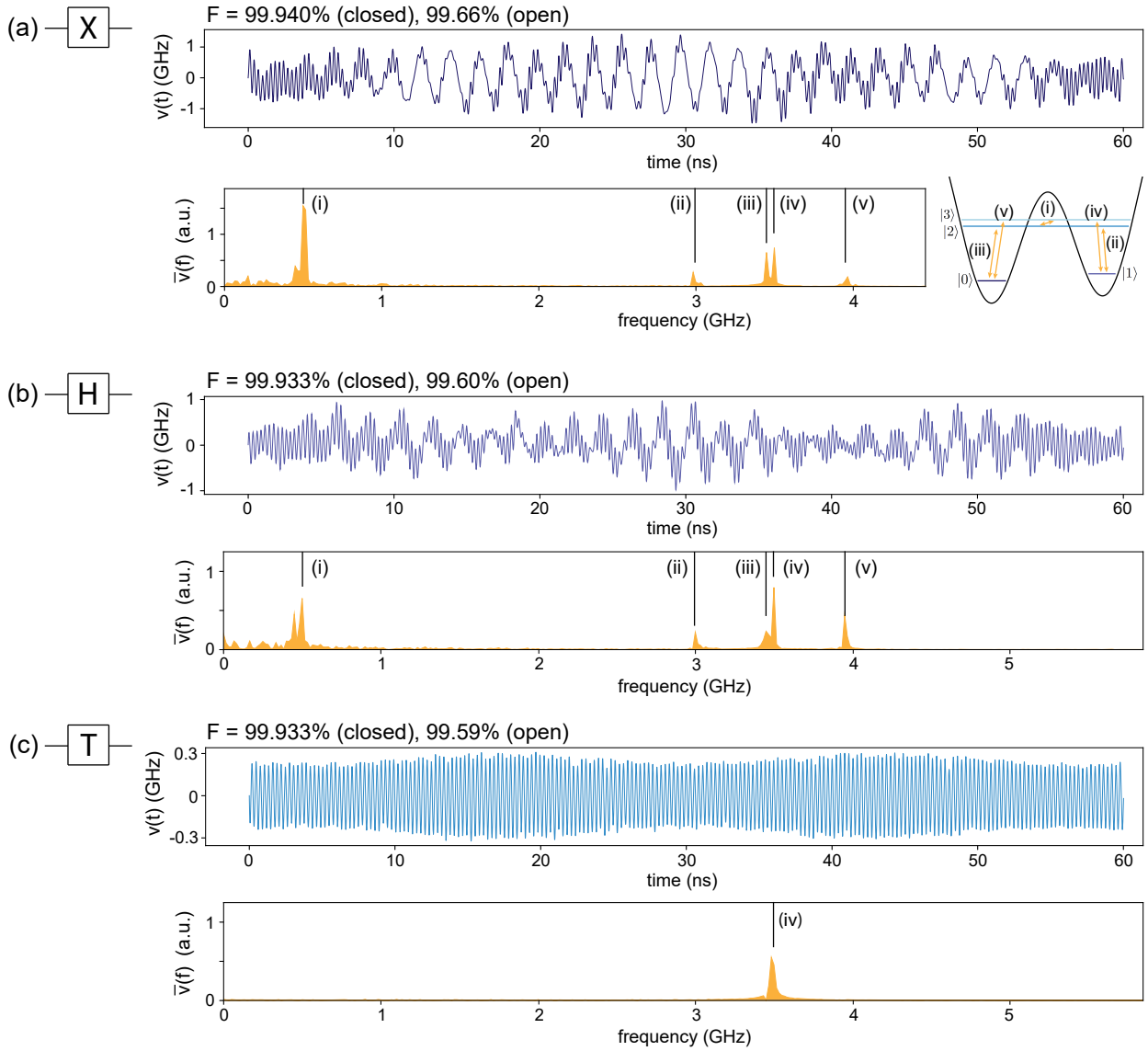


Figure 6.6: High-fidelity single-qubit gates for heavy fluxonium. (a) Optimized pulse shape $v(t)$ and its discrete Fourier transform $\bar{v}(f)$ for the Pauli-X gate, achieving a gate fidelity of 99.94%. The Fourier transform exhibits distinct peaks that align with the transition frequencies among the involved levels (see inset). (b) Corresponding pulse data for the Hadamard gate with a fidelity of 99.933%. (c) Optimized pulse for the T-gate with 99.933% gate fidelity. The Fourier transform shows a single peak centered at the $|1\rangle \leftrightarrow |3\rangle$ transition, serving to induce the required phase shift of $\pi/4$ for state $|1\rangle$.

$> 99.9\%$. The panels of Fig. 6.6(a)–(c) show the pulse $v(t)$ in the time domain and its discrete Fourier transform $\bar{v}(f)$ in the frequency domain. While interpreting GRAPE-optimized pulses is notoriously difficult, we note that general features of the three pulses and their frequency components can be given physical meaning. The Pauli-X and Hadamard gates both exhibit relatively well defined peaks in their Fourier spectra $\bar{v}(f)$ which coincide with the relevant transition frequencies among the lowest four levels primarily involved in the performance of the gate operation, see inset in Fig. 6.6(a). Visual inspection of $v(t)$ further reveals the staggered application of different frequency components. The initial and final $\sim 5\,\mathrm{ns}$ time windows are dominated by high-frequency components related to transferring the system from the $|0\rangle$, $|1\rangle$ subspace to the delocalized states $|2\rangle$, $|3\rangle$ (and back). The central time window between $t = 5\,\mathrm{ns}$ and $55\,\mathrm{ns}$ shows involvement of the low-frequency components associated with the transfer between the intermediary states $|2\rangle$ and $|3\rangle$. The T-gate, by contrast, exhibits a Fourier spectrum $\bar{v}(f)$ with only a single dominant frequency component corresponding to the $|1\rangle \leftrightarrow |3\rangle$ transition. This is plausible, since the T-gate does not necessitate population transfer across the potential well. The transition peak for $|1\rangle \leftrightarrow |3\rangle$ facilitates the needed $e^{i\pi/4}$ phase accumulation for the $|1\rangle$ state.

Further evidence for this interpretation is given by Fig. 6.7, showing the probabilities for occupying the various fluxonium eigenstates as a function of time. For the Pauli-X gate and the Hadamard gate, occupation probabilities $p_l(t) = |\langle l|\psi(t)\rangle|^2$ are obtained for the example of initial qubit state $|\psi(0)\rangle = |0\rangle$. As expected, the X-gate transfers population into the final state $|1\rangle$, while the H-gate takes $|0\rangle$ into an equal superposition of $|0\rangle$ and $|1\rangle$. Both of these gates rely on the auxiliary states $|2\rangle$ and $|3\rangle$ to transfer population between the qubit compuational states. By contrast, the T-gate exhibits qualitatively dissimilar behavior since there is no need for state transfer across the fluxonium potential barrier. Instead, the much weaker control field only causes a small amount of intermediate population transfer from $|1\rangle$ to $|3\rangle$ for phase accumulation.

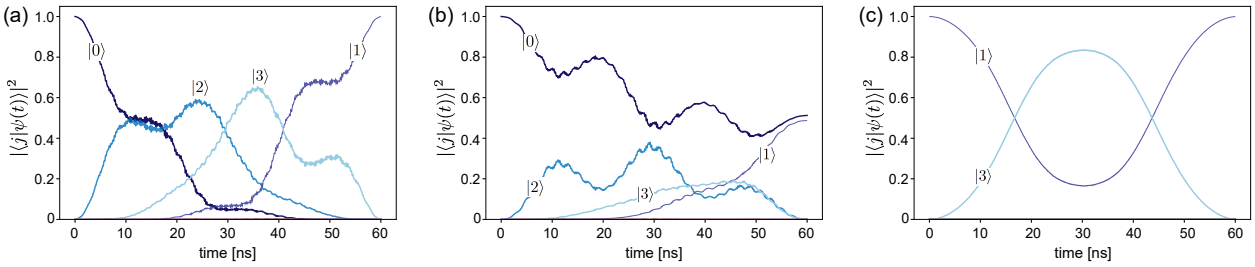


Figure 6.7: Time-evolution of state populations for 60 ns high-fidelity single-qubit gates. (a) The time-evolution of states involved in the X gate shows state transfer between the qubit computational states via the delocalized $|2\rangle$ and $|3\rangle$ states. (b) For the H gate, states are transferred into an (approximately) equal superposition of $|0\rangle$ and $|1\rangle$. (c) In the T gate, the $|1\rangle$ state acquires an additional phase due to temporary state transfer into the $|3\rangle$ state.

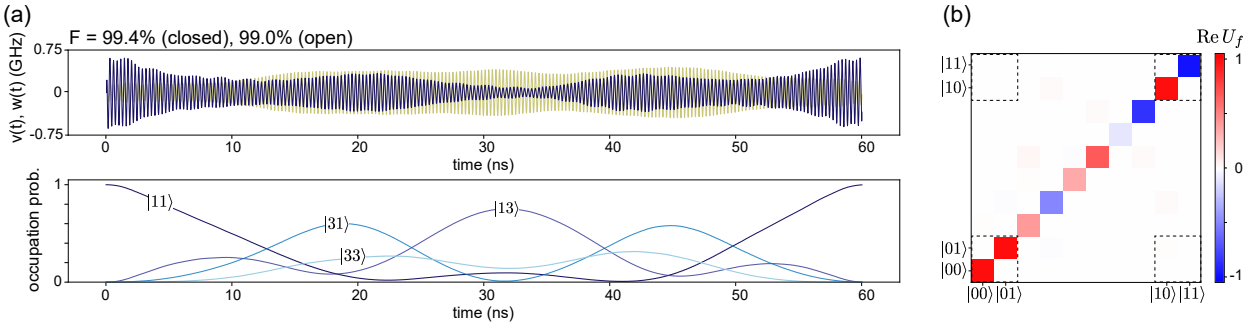


Figure 6.8: Controlled-Z gate for two heavy-fluxonium qubits with a gate time of 60 ns [$\Phi_{\text{ext},1} = 0.45\Phi_0$ (target qubit) and $\Phi_{\text{ext},2} = 0.455\Phi_0$ (control qubit)]. (a) The top panel shows optimized pulses acting on the target and control qubit, $v(t)$ and $w(t)$, respectively, achieving a closed-system fidelity of 99.4% and open-system fidelity of 99.0%. The cost functionals used are $C_1$, $C_3$, and $C_4$. The bottom panel shows occupation probabilities of system eigenstates $|ml\rangle$, with $|11\rangle$ chosen as the initial state. $|11\rangle$ undergoes the significant population to intermediate states so to induce a phase of $e^{i\pi}$, as required for the CZ gate. (b) Real part of the resulting unitary $U_f$ achieved by optimization, showing levels $|0l\rangle$ with $0 \leq l \leq 7$, $|10\rangle$, and $|11\rangle$. Matrix elements between states in the computational subspace are marked by dashed squares.

### Controlled Z gate

A universal set of gates would not be complete without a controlled two-qubit gate. Here, we show a pulse optimized to activate a controlled-Z (CZ) gate using the same gate time $t_g = 60$ ns. Here, we consider two heavy fluxonium qubits that differ by the amount of magnetic flux threading them (all circuit parameters are identical to the ones used for the single qubit case). Each qubit is indirectly coupled to a shared resonator through a small capacitor. For our configuration we choose a target qubit with magnetic flux $\Phi_{\text{ext}} = 0.45\Phi_0$ and a control qubit with magnetic flux $\Phi_{\text{ext}} = 0.475\Phi_0$. The resulting Hamiltonian for such a configuration is given by

$$H(t) = H_f^{(1)} + H_f^{(2)} + H^{(1,2)} + V^{(1)}(t) + V^{(2)}(t). \tag{6.9}$$

This equation is similar to equation (6.7), where $H_f^{(i)}$ is the Hamiltonian for the $i^{th}$ device ($i = 1, 2$) and $V^{(i)}(t)$ is the dispersively filtered drive acting on each, in the form given in equation (6.8). Due to coupling to a shared resonator, there is an effective coupling $H^{(1,2)}$ between the two devices.

The optimization yields a pulse that activates a CZ gate shown in Fig. 6.8(a). We obtain a gate fidelity of 99.4% . We use a truncated Hilbert space dimension of 8 for each qubit. Fig. 6.8(b) depicts the final unitary matrix $U_f$ achieved in the optimization. It is given in the product basis $|mn\rangle$, with $m$ and $n$ labeling control and target qubit levels respectively. The relevant 4x4 computational subspace $\{|00\rangle,|01\rangle,|10\rangle,|11\rangle\}$, marked by dashed rectangles, has the form expected for a controlled-Z unitary.

## *6.2.2 0-$\pi$ gates*

As explained in section 2.6, the 0-$\pi$ device can combine exponential suppression of dephasing and relaxation due to wave function localization and ground state degeneracy. However

unlike heavy fluxonium, there are two major additional complications to consider. The first complication is the presence of disorder in the circuit components. Circuit disorder can lead to a spurious coupling to an additional harmonic degree of freedom [32, 51], the $\zeta$ mode, which leads to unwanted loss of coherence. The second and most pressing complication is the uncontrollable amount of offset charge on the large circuit capacitors faced in experiment. This unpredictable offset charge makes it difficult to realize single qubit gates due to the heavy dependence of higher energy eigenstates on it. Transitions into high energy states are essential to realizing gates. Fig. 6.9 shows the dependence of a selection of matrix elements on the value of the offset charge. We see that some higher-level matrix elements have a significant dependence on $n_g$ and hence we will incorporate that into our optimization.

In running optimal control we choose an "optimistic" parameter set that allows for an appropriate amount of qubit protection but also may be realized in future experiments. The parameter set was obtained from ref. [51] in Table 1. They are $E_L = E_C = 40$ MHz, $E_J = 10$ GHz, and $E_{CJ} = 20$ GHz. The eigenspectrum as a function of external flux is given in Fig. 6.10 along with the device eigenstate wave functions that are used as qubit and auxiliary states. The qubit states are localized along the $\theta = 0$ and $\theta = \pi$ axes so we denote the logical states by $|0\rangle$ and $|\pi\rangle$.

While the 0-$\pi$ qubit exhibits intrinsic protection from decoherence in this parameter regime, like heavy fluxonium this protection inadvertently prevents us from driving direct transitions between the two qubit states ($|0\rangle$ and $|\pi\rangle$). In the work of ref. [101], the authors perform operations between these two states indirectly via a square pulse which drives transitions into intermediate higher excited levels, optimizing over the pulse amplitude and gate time. They achieve different single qubit gates (X, H and Z) by tuning device parameters. However, tuning device parameters in situ is experimentally challenging, so performing universal gates on $|0\rangle$ and $|\pi\rangle$ may be easier met by tuning the shape of an external drive acting on an ancillary resonator. Employing optimal control to find the correct pulse shape for each

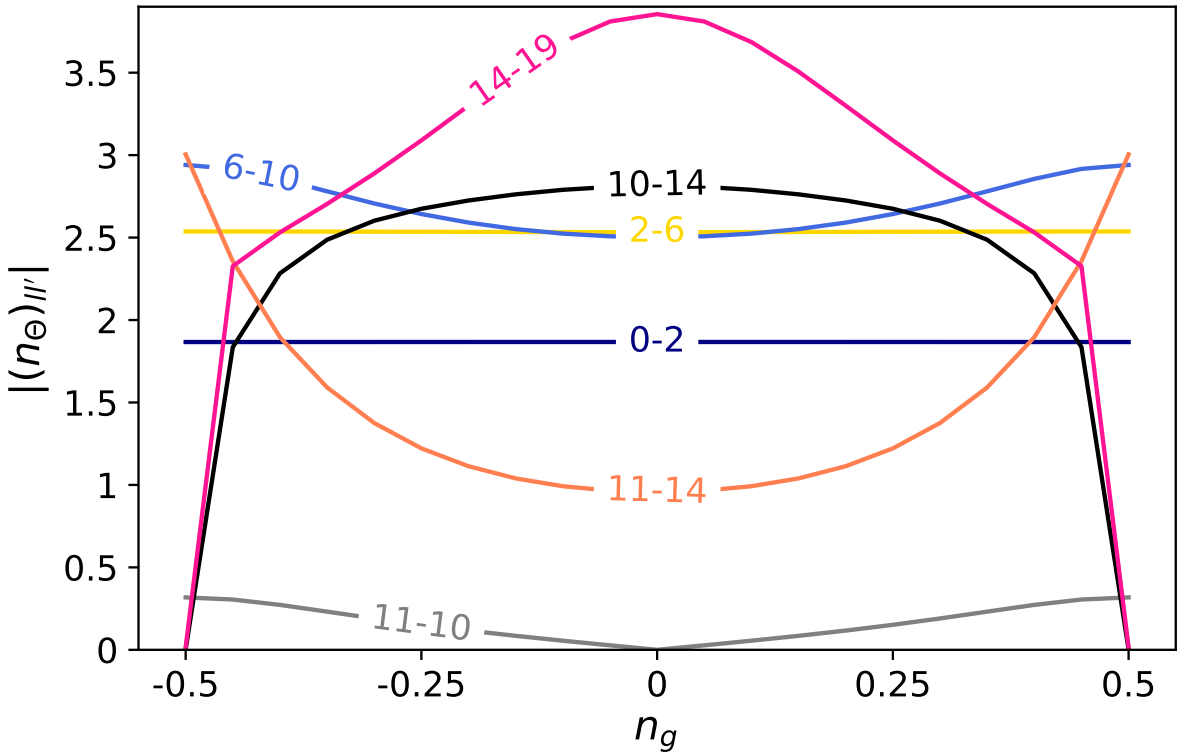


Figure 6.9: 0-$\pi$ charge matrix elements with respect to $\theta$ variable vs. offset charge $n_g$. Transition matrix elements show a much greater dependence on offset charge for transitions between delocalized high energy states. This is a critical issue for optimal control since $n_g$ is not a fixed parameter.

desired gate gives us much more optimization flexibility than optimizing over amplitude and gate time alone. Let us now consider our optimal control scheme where there is no circuit disorder present, focusing solely on the complication of unconstrained offset charge. Offset charge $n_g$ appears in the 0-$\pi$ Hamiltonian as an external parameter in the kinetic term for the periodic degree of freedom $\theta$, $q_\theta^2/2C_\theta \to (q_\theta - n_g)^2/2C_\theta$. Together, the drift and control Hamiltonian for 0-$\pi$ has a similar form to the heavy fluxonium case in the previous section

$$H(t, n_g) = H_{0-\pi}(n_g) + V(t). \tag{6.10}$$

The control Hamiltonian $V(t)$ takes the form of equation (6.8) only with $n_\phi$ replaced by $n_\theta$. This drive is thus filtered through an ancillary resonator as before.

The unpredictable dependence on offset charge $n_g$ requires some modification of closed-system optimal control. Rather than choosing a fixed drift and control Hamiltonian for optimization, we choose a drift and control Hamiltonian randomly from different values of $n_g$ in every iteration. A different fidelity for each value of $n_g$ is obtained, but with careful tuning of cost-function weights, these differences are small and the fidelity is nearly constant for all values. We obtain a pulse that is optimized over all values of offset charge that yields some average fidelity $F_{\text{avg}}$.

In Fig. 6.11 we present pulses optimized using closed-system optimal control for three single qubit 0-$\pi$ gates X, H, and T, using the same gate time $t_g = 60$ ns. Average gate fidelities range from 98.63% for the X gate, to 99.35% for the H gate, and to over 99.9% for the T gate. The lower average fidelities for the X and H gates can be attributed to the heavy dependence on $n_g$ of the transition matrix elements $(n_\theta)_{ll'}$ between high energy unlocalized states (see Fig. 6.10(b)). Pulses for X and H gates require the qubit to occupy these high energy auxiliary states temporarily, causing optimizations for different $n_g$ to compete more. A high fidelity pulse for $n_g = 0.5$ is likely to yield a low fidelity for $n_g = 0$, therefore a lower overall average fidelity would serve as a compromise between the two extremes. Fig. 6.11

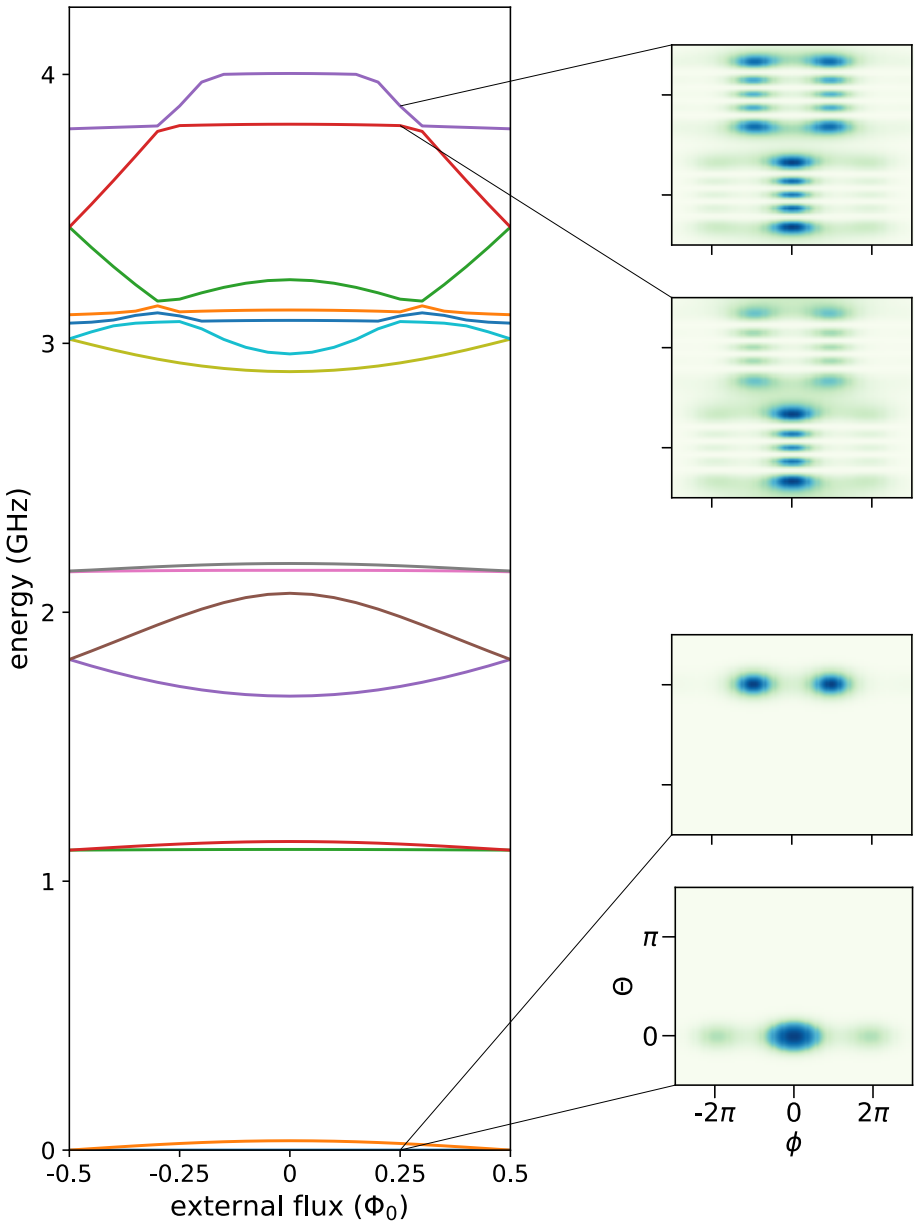


Figure 6.10: 0-$\pi$ energy vs. $\Phi_{\text{ext}}$, the 0-$\pi$ eigenfunctions $|0\rangle$, $|1\rangle$, $|13\rangle$, & $|14\rangle$, and 0-$\pi$ charge matrix elements with respect to $\theta$ variable vs. offset charge $n_g$. The bottom two eigenfunctions shown, used as qubit states, are localized and near degenerate at $\Phi_{\text{ext}} = 0$ with parameters $E_L = E_C = 40$ MHz, $E_J = 10$ GHz, and $E_{CJ} = 20$ GHz. The top two eigenfunctions shown are delocalized high energy states that occupy both potential wells. They serve as two auxiliary states to carry out our gates.

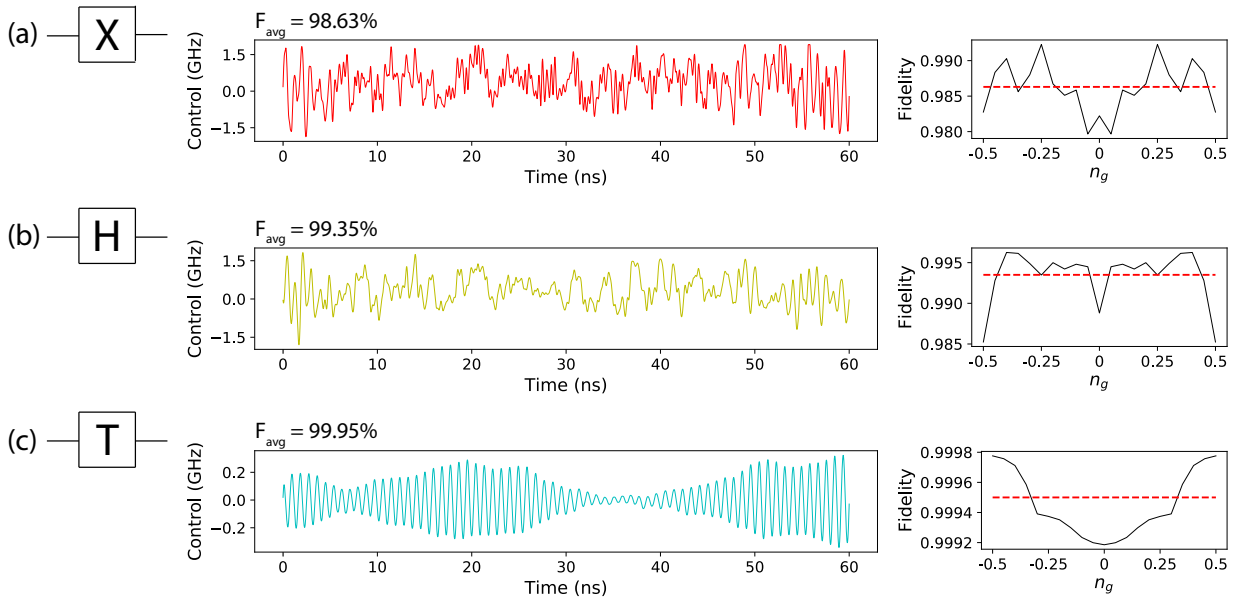


Figure 6.11: Optimized pulses for single qubit 0-$\pi$ gates and pulse fidelities vs. offset charge. a) Control pulse $v(t)$ that activates a qubit flip (X) has three distinct roughly equally spaced regions: 0-5, 5-55, and 55-60 ns. The first and last regions propel the qubit in and out of one of the potential wells and the middle region steers it through the high energy delocalized states. Pulse fidelity variations over offset charge are revealed to be small compared to the average fidelity. This was intended since we want our pulses to be insensitive as possible to changes in offset charge. The red dashed line indicates the average fidelity. b) and c) These pulses are the corresponding ones for the H and T gates.

shows that variations in fidelity across $n_g$ are small relative to the average, which renders the optimized pulse roughly insensitive to offset charge variations. Figure 6.12 (a-c) depict the eigenstate time-evolutions for each gate starting in the ground state. We see in Fig. 6.12 (a-b) for X and H respectively a clear transfer of population from one potential well (containing states $|0\rangle$, $|2\rangle$, & $|6\rangle$) to the other (containing states $|1\rangle$, $|3\rangle$, & $|7\rangle$). High energy auxiliary states are occupied in the middle of the gate which primarily include states $|10\rangle$, $|11\rangle$, $|13\rangle$, and $|14\rangle$. Figure 6.12 (d-f) reveal how high in energy the qubit ventures by showing the average eigenstate number and the fluctuations about the average. We plot the evolution of the total 0-$\pi$ wave-function starting in the ground state for the Hadamard gate pulse in Fig. 6.12 (g) to further illustrate the utility of our pulses.

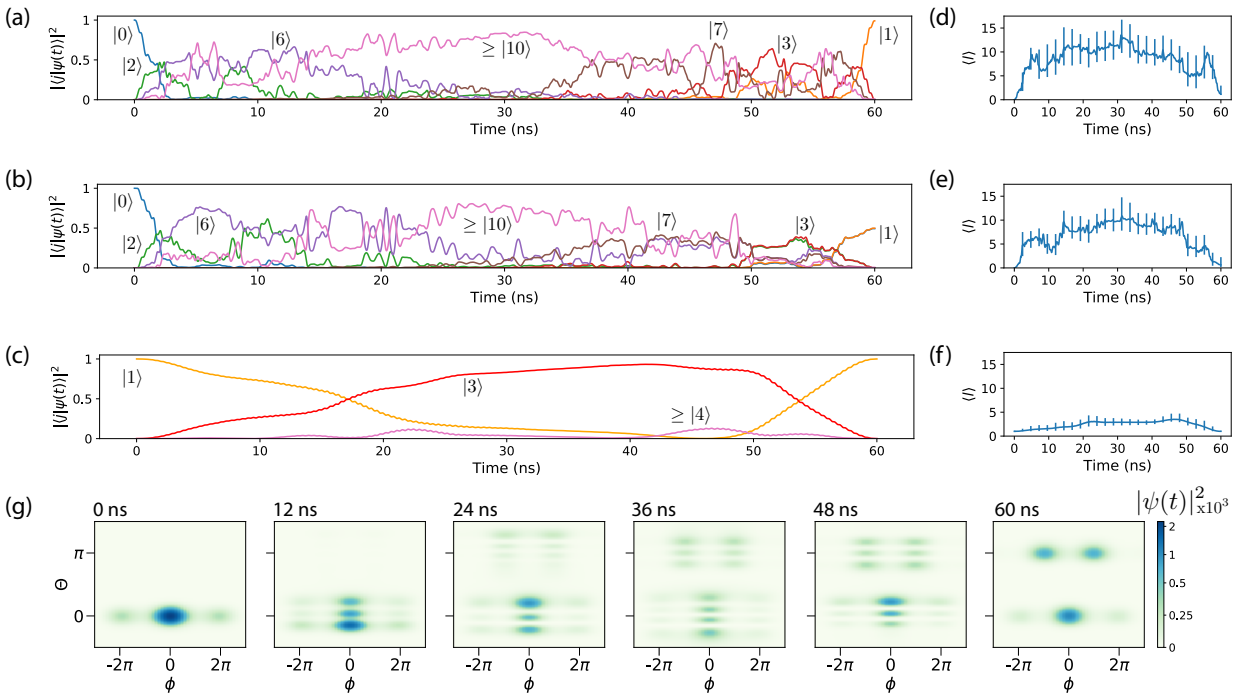


Figure 6.12: 0-$\pi$ eigenstate and average eigenstate number time-evolution at $n_g = 0$ for X, H, and T gates. Equally spaced time snapshots of 0-$\pi$ wave function for our Hadamard gate.

We will now address the concern of the effects of circuit disorder on the device - in particular disorder in the large capacitor $C$ and inductor $L$ which leads to spurious coupling to a harmonic degree of freedom. This degree of freedom $\zeta$ corresponds to a harmonic oscillator mode with mode frequency $\Omega_\zeta/2\pi$ [32]. The resulting Hamiltonian reads

$$H(t) = H_{0-\pi} + \Omega_\zeta a^\dagger a + \sum_{l,l'} (\kappa_{ll'} \left|l\right\rangle\!\left\langle l'\right| a + \text{h.c.}), \tag{6.11}$$

where $|l\rangle$ are 0-$\pi$ eigenstates corresponding to energy $E_l$ and $\kappa_{ll'} = \kappa^\phi_{ll'} + i\kappa^\theta_{ll'}$ are coupling strengths defined by

$$\kappa^\theta_{ll'} = E_{C\Sigma}(dC/2)(32E_L/E_c)^{1/4} \left\langle l|n_\theta|l'\right\rangle, \tag{6.12}$$

$$\kappa^\phi_{ll'} = \frac{1}{2} E_L dE_L (8E_C/E_L)^{1/4} \left\langle l|n_\phi|l'\right\rangle. \tag{6.13}$$

Here, we take the disorder in the inductor and capacitor to be 5%, $dL = dC = 0.05$. The addition of disorder requires us to incorporate an additional component to the drive. A drive on the $\theta$ degree of freedom also spuriously drives the zeta mode. As a result, rather than the drive coupling to just $n_\theta$, the drive couples to the modified operator

$$n_\theta \to n_\theta - \beta n_\zeta, \tag{6.14}$$

in which $n_\zeta$ is the momentum operator of the zeta mode and $\beta = CdC/C_\zeta$ (refer to ref. [101] - appendix A for derivation). We take $C_\zeta \approx 2C$, so $\beta \approx dC/2 = 0.025$ in our simulations.

Utilizing the pulses optimized using our closed-system optimizer shown in Fig. 6.11, we evolved the 0-$\pi$ + $\zeta$-mode system with the Schrödinger equation and calculated the resulting average gate fidelities for X, H, and T. The resulting average fidelities for disordered 0-$\pi$ were 97.40% for X, 98.81% for H, and 99.99% for T. These all constitute reductions in fidelity from the non-disordered case. These reductions are most likely due to the small amount

of 0-$\pi$ state dressing due to $\zeta$-mode coupling and the corresponding changes to the device eigenspectrum. Since $\Omega_\zeta$ is much less than the energy splitting between different 0-$\pi$ states, our pulses did not populate the $\zeta$-mode to any appreciable amount, otherwise we could have expected the average fidelities to be much less. In the future, it may be a viable option to further refine our pulses using optimal control to correct for the loss due to $\zeta$-mode. However this will require a significantly larger amount of computational overhead due to larger Hilbert space.

Lastly, the presence of the $\zeta$-mode may open up an unwanted decoherence channel that is absent in the symmetric 0-$\pi$ device. This channel can include shot-noise dephasing due to thermal excitations of the $\zeta$-mode. However, the authors in ref. [101] proposed a method to cool the $\zeta$-mode to its ground state using an additional frequency-tunable resonator, thereby enhancing its coherence time and eliminating unwanted shot-noise dephasing.

# CHAPTER 7

# OPEN-SYSTEM DYNAMICS

## 7.1 Introduction

Including the effects of dissipation and dephasing on quantum systems requires the simulation of open dynamics. Typically, this can be described by a Markovian Lindblad master equation [17]. A key difference between open and closed dynamics is that the open case propagates the quantum state as a density matrix instead of a state vector. Most of the working closed-system algorithms can be generalized to the open case but would have to propagate density matrices of dimension $d^2$ instead of state vectors of dimension $d$, where $d$ is the system Hilbert space dimension. Open-system optimization algorithms include open versions of GRAPE [66] and Krotov's methods [75]. Algorithms using density matrices to represent the quantum state were successfully used in some applications of small system size [50, 118, 24, 60]. While traditional propagation requires matrix multiplications and exponentials of superoperators of dimension $d^2 \times d^2$, some more recent approaches rely on propagating the density matrix through different integration methods [53, 45, 15]. Notwithstanding the achieved improvements, these techniques are still based on storing and propagating matrices of size at least $d \times d$. In addition, propagation is generally implemented via complex integration techniques. Most of the applications demonstrated with these techniques were limited to moderate Hilbert space sizes.

A useful alternative for simulating the dynamics of open quantum systems is to employ quantum trajectories. Quantum trajectories describe the effect of the environment on the system by a stochastic Schrödinger equation (SSE). This SSE governs how the evolution of the system is conditioned on the measurement processes of the environment [17, 133]. The average over many trajectories reproduces the master equation solution. Trajectories offer a promising memory-efficient approach to open optimal control since generating every

trajectory requires only sparse matrix-vector propagation. Recently, the use of trajectories has been proposed in Krotov-based optimization [47] for a specific choice of optimization target. However, there has been little work on the use of quantum trajectories in direct gradient-based optimal control, since the stochastic nature of trajectories makes it difficult to provide analytical forms of gradients for gradient-based algorithms. In the following chapters, we will present a gradient-based trajectories optimizer that significantly reduces the simulation complexity and is flexible with respect to the choice of optimization targets. The optimizer relies on automatic differentiation to calculate the necessary gradients for optimization. In this chapter, we start by giving an overview of open-system dynamics and the standard open-system optimal control theory.

## 7.2 Lindblad master equation

### *7.2.1 Density matrices*

The state of a quantum system can be represented by a density matrix $\rho$ which is generally a mixture of different initial orthonormal states $\{|\psi_i\rangle\}$ with probabilities $\{p_i\}$. Specifically, the density matrix is a hermitian matrix of dimension $d \times d$, defined as

$$\rho = \sum_i p_i \left|\psi_i\right\rangle \left\langle\psi_i\right| \tag{7.1}$$

The density matrix has the following properties:

1. Normalization: $\mathrm{Tr}(\rho) = \sum_i p_i = 1$.
2. Positivity: $\rho \geq 0$ since $p_i \geq 0$.
3. Hermiticity: $\rho = \rho^\dagger$.

If the density matrix satisfies the condition $\rho^2 = \rho$, this defines a pure density matrix. A pure density matrix can be represented as $\rho = |\psi_j\rangle \langle\psi_j|$, which is equivalent to describing

the system with the ket $|\psi_j\rangle$. In general, however, the system state may not be pure and hence cannot be described by a single ket.

The time evolution of density matrices in closed systems is governed by the von-Neumann equation that generalizes the Schrödinger equation to

$$\frac{\partial \rho(t)}{\partial t} = -i[H, \rho(t)] \tag{7.2}$$

### *7.2.2 Assumptions of the master equation*

To include the effects of the quantum system interacting with its environment, we note that both share a density matrix in the combined Hilbert space of system + environment. The Hamiltonian of such an open system interacting with its environment takes the form

$$H_{\text{tot}} = H + H_{\text{env}} + H_{\text{int}}, \tag{7.3}$$

where $H$ is the Hamiltonian of the system of interest, $H_{\text{env}}$ the Hamiltonian of the environment and $H_{\text{int}}$ the corresponding interaction Hamiltonian. Therefore, one can utilize the von-Neumann equation for evolving the combined density matrix in time. However, as we are only interested in the state of the quantum system $\rho = \text{Tr}_{\text{env}}(\rho_{\text{tot}})$, it is desirable to obtain an approximate equation for the time evolution of just the reduced density matrix $\rho$.

The Lindblad master equation is one such equation that is widely used because its assumptions are reasonable for a variety of systems [17]. The first assumption is the weak coupling of the system to the environment which allows for treating the interaction perturbatively using the Born approximation. Next, the master equation assumes that the correlation times of the environment excitations are much smaller than the characteristic time scales of the quantum system. This allows the approximation of treating the dynamics of the system as Markovian. In Markovian dynamics, the time evolution of the reduced

density matrix only depends on the state at the current time and there is no need to know any past information about the system, i.e. the time evolution has no memory.

### *7.2.3 The master equation*

Performing both approximations yields the final form of the Lindblad master equation

$$\frac{\partial \rho}{\partial t} = -i[H, \rho] - \frac{1}{2} \sum_l \gamma_l \left[ c_l^\dagger c_l \rho + \rho c_l^\dagger c_l - 2 c_l \rho c_l^\dagger \right] = \mathcal{L}\rho. \tag{7.4}$$

Here, $\{c_l\}$ is a set of so-called jump operators, describing relaxation and dephasing emerging from the interaction with the environment, with associated rates $\{\gamma_l\}$. $\mathcal{L}$ denotes the Liouville superoperator and we have set $\hbar = 1$.

Following the evolution of open quantum systems requires propagation of the full $d \times d$ density matrix. Numerically, this may be realized by expressing the density matrix as a vector and propagating it via matrix exponentials of the $d^2 \times d^2$ superoperator. As Hilbert space size increases, this process can require heavy computational resources in comparison to the closed-system dynamics governed by propagators of size $d \times d$. In the next section, we discuss an alternative way for obtaining the open-system dynamics that is more similar to closed-system evolution.

## 7.3 Quantum trajectories

Quantum trajectories are sample paths obtained because of performing measurements on the system by the environment. To discuss the different valid measurement schemes the environment can perform on the system, we first explain what constitutes a valid quantum channel/map.

### 7.3.1 Quantum channels and generalized measurements

A quantum channel is a completely-positive trace-preserving (CPTP) map that preserves the positivity of the density matrix and also its normalization condition. One such CPTP map corresponds to a generalized measurement on the quantum system. A generalized measurement has $k$ possible outcomes with $k$ measurement operators $\{A_i\}$. Depending on the individual measurement outcome, the density matrix is mapped to:

$$\text{outcome } i\text{: } \rho \to \frac{A_i \rho A_i^\dagger}{\text{Tr}(A_i \rho A_i^\dagger)} \tag{7.5}$$

with probability

$$p_i = \text{Tr}(A_i^\dagger A_i \rho). \tag{7.6}$$

For the generalized measurement to qualify as a valid CPTP map, the probabilities must sum to 1 so that the trace of the resulting density matrix is preserved. Hence,

$$\sum_i p_i = 1 \implies \sum_i \text{Tr}(A_i^\dagger A_i \rho) = 1 \implies \sum_i A_i^\dagger A_i = \mathbb{1} \tag{7.7}$$

The operators $\{A_i\}$ are usually called the Kraus operators and they completely define the quantum channel/map $\mathcal{M}$ via

$$\mathcal{M} : \rho \to \sum_i A_i \rho A_i^\dagger \tag{7.8}$$

### 7.3.2 Quantum jump trajectories

A quantum trajectory is realized as the state of the system is conditioned upon measurement results [133]. The simplest kind of quantum trajectories involves jumps, meaning that the evolution is discontinuously conditioned on measurements outcomes. To discuss this discontinuous conditioned evolution, we define a quantum channel (representing the measurement

process) that evolves the system infinitesimally from $\rho(t)$ to $\rho(t+\delta t)$, $\delta t \ll 1$. We know that for closed-system dynamics, the corresponding infinitesimal time evolution for state vectors is obtained by the propagator $U(t; t+\delta t) \approx \mathbb{1} - iH\delta t$. Quantum jump trajectories assume a similar form for the infinitesimal evolution resulting from monitoring the system via the environment [133]. We define a first Kraus operator as

$$A_0 = \mathbb{1} - iH\delta t - \frac{R}{2}\delta t, \tag{7.9}$$

where $R$ is a Hermitian operator. To first order in $\delta t$

$$A_0^\dagger A_0 = \mathbb{1} - R\delta t. \tag{7.10}$$

If $R = 0$, no measurement takes place and we recover the von-Neumann dynamics. However, if $R \neq 0$, there must exist other Kraus operators that satisfy the condition

$$\sum_{i\neq 0} A_i^\dagger A_i = R\delta t. \tag{7.11}$$

One choice for the other Kraus operators is defining $A_i = \sqrt{\gamma_i \delta t}\, c_i$ where $i \neq 0$ and $\{c_i\}$ are the jump operators defined in the Lindblad master equation. Hence, a valid quantum map is defined by the following Kraus measurement operators

$$\begin{aligned} A_i &= \sqrt{\gamma_i \delta t}\; c_i, \qquad i \neq 0 \\ A_0 &= \mathbb{1} - (iH + \frac{\sum_i \gamma_i c_i^\dagger c_i}{2})\delta t. \end{aligned} \tag{7.12}$$

The corresponding evolution of the density matrix follows equation (7.8)

$$\rho(t+\delta t) = \big[\mathbb{1} - (iH + \frac{\sum_l \gamma_l c_l^\dagger c_l}{2})\delta t\big]\rho(t)\big[\mathbb{1} - (-iH + \frac{\sum_l \gamma_l c_l^\dagger c_l}{2})\delta t\big] + \sum_l \gamma_l c_l \rho(t) c_l^\dagger \delta t. \tag{7.13}$$

Keeping only terms of first order in $\delta t$, we recover the Lindblad master equation exactly. Therefore, quantum jump trajectories are equivalent to the master equation evolution.

The probability for any outcome $i \neq 0$ is given by

$$P_i = \delta t \gamma_i \mathrm{Tr}(c_i^\dagger c_i \rho), \tag{7.14}$$

which is infinitesimal if $c_i^\dagger c_i$ is bounded as assumed in the master equation. Therefore, $P_0 = 1 - O(\delta t)$ is the more probable outcome at all infinitesimal time intervals. Hence, the measurement outcome corresponding to $A_i$ is usually referred to as a null result. Measuring this null outcome is equivalent to (unnormalized) propagation with an effective non-Hermitian Hamiltonian $H_{\text{eff}} = H - \frac{i}{2}\sum_l \gamma_l c_l^\dagger c_l$. On the other hand, detecting one of the jump outcomes has an effective (unnormalized) evolution of multiplying the state vector by the jump operator.

Therefore, the open-system quantum dynamics can be obtained via a stochastic-sampling approach which simulates $m$ independent trajectories. Each trajectory undergoes dynamics of a complexity similar to that of a closed system, and thus only requires propagators of size $d \times d$ which helps reduce the computational cost. The generation of these trajectories in a simulation is summarized as follows [29]:

1. Discretize the total evolution time into small time steps $dt$.

2. Propagate the initial state $|\psi_0\rangle$ with the effective non-Hermitian Hamiltonian

$$H_{\text{eff}} = H - \frac{i}{2}\sum_l \gamma_l \; c_l^\dagger c_l. \tag{7.15}$$

   The norm $\||\psi(t)\rangle\|$ of the resulting state will decay over time.

3. Generate a uniformly-distributed random number $r \in [0, 1)$.

4. Keep propagating with $H_{\text{eff}}$ until $\||\psi(t)\rangle\|^2$ reaches $r$. Once reached, randomly select

one jump operator from $\{c_l\}$, according to probabilities $\propto \langle\psi(t)|\,\gamma_l\, c_l^\dagger c_l\,|\psi(t)\rangle$. Apply the jump operator to the state, and normalize the result.

5. Repeat steps 3-4.

The expectation value of any Hermitian operator $A$ can be obtained by averaging over a sufficient number $M$ of trajectories,

$$\langle A\rangle(t) = \mathrm{Tr}[A\rho(t)] \approx \frac{1}{M}\sum_m \langle\psi_m(t)|\,A\,|\psi_m(t)\rangle \tag{7.16}$$

with statistical error $\sigma_A \propto \frac{1}{\sqrt{M}}$ [47].

Using quantum trajectories to simulate the dynamics of open systems has several advantages. First, the complexity of computations is $\mathcal{O}(Md^3)$ instead of $\mathcal{O}(d^6)$ as would be needed to propagate the full density matrix. [Note: matrix multiplication of matrices of size $d\times d$ is $\mathcal{O}(d^3)$.] In addition, we can use the sparsity of matrices that typically arise in most quantum applications to significantly lower the complexity of matrix-vector multiplication to $\mathcal{O}(d^2)$. Therefore, for large Hilbert space dimensions where $M < d^3$ (with $d \propto 2^n$ growing exponentially with n, the number of qubits), the use of quantum trajectories can significantly reduce the complexity. While $M$ must be large enough to overcome the statistical noise in the trajectories outcomes, we will propose schemes to reach statistical convergence with a lower number of trajectories. In many practical cases, quantum jumps turn out to be rare. We propose a protocol that takes advantage of the rarity of jumps, and only requires computation of a smaller set of trajectories, hence reducing the complexity even further. In addition, quantum trajectories reduce the memory requirements for calculating the forward evolution, as superoperators are never stored in memory explicitly; only propagators of size $d\times d$ are calculated instead. Finally, since trajectories are independent of each other, this method is also highly parallelizable. Different trajectories can be run on different computational nodes in parallel, thus reducing overall computation time.

To present how quantum trajectories can be used in optimal control, we first review the relevant gradient-based optimal-control techniques and cost functions in the next chapter.

# CHAPTER 8

# OPEN-SYSTEM OPTIMAL CONTROL THEORY

## 8.1 Open-system fidelity definitions

We first review the different definitions of fidelities in open-system optimal control, since the fidelity constitutes our main maximization target.

### *8.1.1 State transfer fidelities*

One important fidelity metric is concerned with transferring the system from one (or multiple) initial states to specific target state(s). If the target state is $\rho_t$ and the final state is $\rho_f$, the fidelity is defined by [93]

$$F = (\mathrm{Tr}\sqrt{\sqrt{\rho_t}\rho_f\sqrt{\rho_t}})^2. \tag{8.1}$$

Since our target state is usually a pure state $\rho_t = |\psi_t\rangle\langle\psi_t|$, we have $\sqrt{\rho_t} = |\psi_t\rangle\langle\psi_t| = \rho_t$ and the definition reduces to

$$F = (\mathrm{Tr}\sqrt{\rho_t\rho_f\rho_t})^2 = (\mathrm{Tr}\sqrt{\langle\psi_t|\,\rho_f\,|\psi_t\rangle\,\rho_t})^2 = (\sqrt{\langle\psi_t|\,\rho_f\,|\psi_t\rangle}\mathrm{Tr}(\sqrt{\rho_t}))^2 = \mathrm{Tr}(\rho_t\ \rho_f). \tag{8.2}$$

Next, we will discuss definitions for gate-fidelities in open quantum systems.

### *8.1.2 Consistent open-system fidelity metric $F_o$*

It is important to consider the effects of dissipation on optimized gates. Therefore, it is desirable to define an open-system fidelity measure that reduces to the same fidelity expression in the case of treating the system as closed. This allows for a consistent way of comparing fidelities between open and closed system dynamics.

To define our metric, we first introduce a vectorization technique to density matrices. In this technique, we stack the rows of the density matrix $\rho$ ($n$ by $n$) into a vector $|\rho\rangle\rangle$($n^2$ by

1) such that

$$\rho = \sum_{ij} \rho_{ij} |i\rangle \langle j| \longrightarrow |\rho\rangle\rangle = \sum_{ij} \rho_{ij} |i\rangle |j\rangle , \langle\langle\rho| = \sum_{ij} \rho_{ij}^{*} \langle j| \langle i| . \tag{8.3}$$

Then we can write the evolution of the system density matrix $\rho$ in terms of a superoperator $L$ ($n^2$ by $n^2$) as

$$L |\rho(0)\rangle\rangle = |\rho(t)\rangle\rangle. \tag{8.4}$$

We propose the use of the following measure for the open-system gate fidelity

$$F_o = \frac{1}{n^2} \mathrm{Tr}(L_t^{\dagger} L_f), \tag{8.5}$$

where $L_t$ is the target superoperator and $L_f$ is the final achieved superoperator.

This metric reduces to the expression of the closed-system trace fidelity $F_c = |\mathrm{Tr}(U_t^{\dagger} U_f / n)|^2$ if dissipation is ignored. To prove this, we first describe the action on vectorized states equivalent to matrix multiplication from the right and left on $\rho$,

$$A\rho = \sum_{ij} \rho_{ij} A |i\rangle \langle j| \longrightarrow A |\rho\rangle\rangle = \sum_{ij} \rho_{ij} A |i\rangle |j\rangle = A \otimes I |\rho\rangle\rangle, \tag{8.6}$$

$$\rho B = \sum_{ij} \rho_{ij} |i\rangle \langle j| B \longrightarrow B |\rho\rangle\rangle = \sum_{ij} \rho_{ij} |i\rangle B^T |j\rangle = I \otimes B^T |\rho\rangle\rangle. \tag{8.7}$$

In the case of closed dynamics, the system density matrix is a pure state $|\psi\rangle \langle\psi|$ and therefore the action of the superoperator will reduce to

$$L_c \rho = L_c |\psi\rangle \langle\psi| = U |\psi\rangle \langle\psi| U^{\dagger} = U \rho U^{\dagger}, \tag{8.8}$$

where U is the closed dynamics propagator. In the vectorized picture, the action of $L_c$

becomes

$$L_c\rho = U\rho U^\dagger \longrightarrow L_c\left|\rho\right\rangle\!\rangle = U\otimes U^*\left|\rho\right\rangle\!\rangle. \tag{8.9}$$

Therefore, to calculate $F_o$ in that case, we can write

$$F_o = \frac{1}{n^2}\mathrm{Tr}(U_t^\dagger U_f \otimes U_t^T U_f^*) = \frac{1}{n^2}|\mathrm{Tr}(U_t^\dagger U_f)|^2 = F_c, \tag{8.10}$$

which reduces exactly to the closed-system gate fidelity, thus generating consistency of open and closed-system fidelities.

### *8.1.3 Relation between average gate fidelity and $F_o$ for single-qubit gates*

Another common metric for open-system gate fidelities is the average gate fidelity defined by averaging state transfer fidelities over all pure input states

$$\bar{F} = \frac{1}{4\pi}\int F_{|\psi\rangle\langle\psi|}d\Omega, \tag{8.11}$$

where $F_{|\psi\rangle\langle\psi|}$ is the state transfer fidelity defined as

$$F_{|\psi\rangle\langle\psi|} = Tr(U_t\left|\psi\right\rangle\left\langle\psi\right|U_t^\dagger\ \mathcal{M}[\left|\psi\right\rangle\left\langle\psi\right|]), \tag{8.12}$$

and $\mathcal{M}$ is the CPTP map representing the master equation. Here, we investigate the relationship between our definition for the open-system fidelity $F_o$ and the average gate fidelity $\bar{F}$. From this point on, we restrict the discussion to single-qubit gates, $n = 2$.

Ref. [16] shows that $\bar{F}$ can be calculated by expanding pure states into basis of Pauli matrices as

$$\bar{F} = \mathrm{Tr}\left(U_t\frac{\sigma_0}{2}U_t^\dagger\,\mathcal{M}\Big[\frac{\sigma_0}{2}\Big]\right) + \frac{1}{3}\sum_{j=x,y,z}\mathrm{Tr}\left(U_t\frac{\sigma_j}{2}U_t^\dagger\,\mathcal{M}\Big[\frac{\sigma_j}{2}\Big]\right), \tag{8.13}$$

where $\{\sigma_j\}$ are the three Pauli matrices and $\sigma_0$ is the identity matrix. The first term in

equation (8.13) reduces to $\frac{1}{2}$, and the average gate fidelity is finally expressed as

$$\bar{F} = \frac{1}{2} + \frac{1}{12} \sum_{j=x,y,z} \mathrm{Tr}\left( U_t \sigma_j U_t^\dagger \, \mathcal{M}\big[\sigma_j\big] \right). \tag{8.14}$$

This expression shows that the average gate fidelity can be calculated as the sum of three state-transfer fidelities, as opposed to the continuous integral in the original definition. This simplifies the task of calculating $\bar{F}$ as no superoperators are explicitly needed. We claim that $F_o$ can be calculated in a very similar fashion.

Starting from the definition of $F_o$ [equation (8.5)], we calculate the trace in the orthonormal basis of $\{\frac{\sigma_j}{\sqrt{2}}\}$ where $j = 0,\ x,\ y,\ z$. (Note that since $\mathrm{Tr}(\rho_1 \rho_2) = \langle\langle \rho_1 | \, \rho_2 \rangle\rangle$, we have $\langle\langle \sigma_i | \, \sigma_j \rangle\rangle = 2\delta_{ij}$).

$$\begin{aligned} F_o = \frac{1}{4} \sum_{j=0,x,y,z} \frac{1}{2} \langle\langle \sigma_j \, L_t^\dagger L_f | \sigma_j \rangle\rangle &= \frac{1}{8} \sum_{j=0,x,y,z} \mathrm{Tr}\left( U_t \sigma_j U_t^\dagger \mathcal{M}\big[\sigma_j\big] \right) \\ &= \frac{1}{4} + \frac{1}{8} \sum_{j=x,y,z} \mathrm{Tr}\left( U_t \sigma_j U_t^\dagger \, \mathcal{M}\big[\sigma_j\big] \right), \end{aligned} \tag{8.15}$$

which is indeed very similar to the expression for $\bar{F}$. Comparing equations (8.14), (8.15), we conclude that the two fidelity measures can be obtained from one another via

$$F_o = \frac{3}{2}\bar{F} - \frac{1}{2}, \bar{F} = \frac{2}{3}F_o + \frac{1}{3}. \tag{8.16}$$

So, the same techniques and shortcuts used to calculate $\bar{F}$ can be used to calculate $F_o$ while $F_o$ has the advantage of being consistent with the closed-system fidelity metric in the proper limit.

We note that ref. [102] describes the average gate fidelity for a closed-system optimization

and reaches the final expression of

$$\bar{F}_c = \frac{1}{n(n+1)}(|\mathrm{Tr}(U_t^\dagger U_f)|^2 \ + n), \tag{8.17}$$

where $n$ is the Hilbert space dimension. For $n = 2$, we get

$$\bar{F}_c = \frac{1}{6}(|\mathrm{Tr}(U_t^\dagger U_f)|^2 \ + 2) = \frac{2}{3}F_c + \frac{1}{3}. \tag{8.18}$$

This result represents the special case of equation (8.16) when the system is taken to be closed. Our work shows that the result in ref. [102] could be generalized to the case of open-system dynamics.

### *8.1.4 Subspace gate fidelities*

In many realistic scenarios, a single qubit is embedded in a bigger Hilbert space of a qudit ($n > 2$). Both the fluxonium and 0-$\pi$ qubits have higher levels that are ultimately used to transfer the population between $|0\rangle$ and $|1\rangle$. In this section, we revisit definitions of fidelity to accommodate for embedding the qubit in a larger Hilbert space.

The definition of the closed-system trace fidelity can be altered such that the trace is only calculated for the relevant part of the dynamics:

$$\begin{aligned} F_c &= \frac{1}{4}|\mathrm{Tr}(M_{\mathrm{rel}})|^2, \\ M_{\mathrm{rel}} &= PU_t^\dagger U_f P, \end{aligned} \tag{8.19}$$

where $P$ is the projection operator onto the 2-dimensional qubit subspace.

Similarly for the closed-system average gate fidelity, ref. [102] shows that equation (8.17)

should be modified to

$$\bar{F}_c = \frac{1}{n_{\text{rel}}(n_{\text{rel}}+1)}(|\text{Tr}(M_{\text{rel}})|^2 \ + \text{Tr}(M_{\text{rel}}M_{\text{rel}}^\dagger)), \tag{8.20}$$

where $n_{\text{rel}} = 2$ is the qubit subspace dimension.

Focusing on the second term, we get

$$\begin{aligned} \text{Tr}(M_{\text{rel}}M_{\text{rel}}^\dagger) = \text{Tr}(PU_t^\dagger U_f P P U_f^\dagger U_t P) &= \sum_{i=0,1} (\langle i| PU_t^\dagger) U_f P U_f^\dagger (U_t P |i\rangle) \\ &= \sum_{i=0,1} \langle\psi_i| U_f P U_f^\dagger |\psi_i\rangle = \text{Tr}(PU_f P U_f^\dagger P), \end{aligned} \tag{8.21}$$

where in the second line, we used $P^2 = P$. In the third line, we utilized the fact that the target unitary projected onto the qubit space will not leak into higher subspaces. Instead, it will act as a unitary transformation from the orthornormal basis of $\{|0\rangle, |1\rangle\}$ to another complete orthonormal basis $\{|\psi_0\rangle, |\psi_1\rangle\}$. Therefore, the second term in equation (8.20) is not necessarily equal to $n_{\text{rel}}$ as $M_{\text{rel}}$ can be non-unitary, if there is leakage into higher levels at the end of the gate.

Combining the new expressions for both $F_c$ and $\bar{F}_c$, we get

$$\bar{F}_c = \frac{2}{3}F_c + \frac{1}{3}\left(\frac{1}{2}\text{Tr}(PU_f P U_f^\dagger P)\right). \tag{8.22}$$

The extension of this approach to the open-system fidelity $F_o$ is similar where the super-operator product is projected first onto the qubit subspace.

$$F_o = \frac{1}{4}\text{Tr}(PL_t^\dagger L_f P). \tag{8.23}$$

However, expanding $F_o$ into the Pauli matrices basis will not yield the same expression as before due to possible leakage into higher levels. Re-defining the Pauli matrices in the bigger

Hilbert space as

$$\sigma_i' \equiv \sum_{j,k} (\sigma_i)_{jk} \left|j\right\rangle \left\langle k\right|, \tag{8.24}$$

the fidelity takes the form

$$\begin{aligned} F_o &= \frac{1}{8} \sum_{j=0,x,y,z} \mathrm{Tr}\left(U_t \sigma_j' U_t^\dagger \mathcal{M}\left[\sigma_j'\right]\right) \\ &= \frac{1}{4}\mathrm{Tr}\left(U_t \sigma_0' U_t^\dagger \, \mathcal{M}\left[\frac{\sigma_0'}{2}\right]\right) + \frac{1}{8} \sum_{j=x,y,z} \mathrm{Tr}\left(U_t \sigma_j' U_t^\dagger \, \mathcal{M}\left[\sigma_j'\right]\right) \\ &= \frac{1}{4}\mathrm{Tr}\left(\sigma_0' \, \mathcal{M}\left[\frac{\sigma_0'}{2}\right]\right) + \frac{1}{8} \sum_{j=x,y,z} \mathrm{Tr}\left(U_t \sigma_j' U_t^\dagger \, \mathcal{M}\left[\sigma_j'\right]\right). \end{aligned} \tag{8.25}$$

The first term will reduce to $\frac{1}{4}$ only if $\mathcal{M}\left[\frac{\sigma_0'}{2}\right]$ keeps the final density matrix in the qubit subspace.

Finally, the average gate fidelity in the open-system case as in equation (8.13) will be

$$\bar{F} = \frac{1}{2}\mathrm{Tr}\left(\sigma_0' \, \mathcal{M}\left[\frac{\sigma_0'}{2}\right]\right) + \frac{1}{12} \sum_{j=x,y,z} \mathrm{Tr}\left(U_t \sigma_j' U_t^\dagger \, \mathcal{M}\left[\sigma_j'\right]\right), \tag{8.26}$$

which leads to the general relation between $\bar{F}$ and $F_o$:

$$\begin{aligned} F_o &= \frac{3}{2}\bar{F} - \frac{1}{2}\mathrm{Tr}\left(\sigma_0' \, \mathcal{M}\left[\frac{\sigma_0'}{2}\right]\right) \\ \bar{F} &= \frac{2}{3}F_o + \frac{1}{3}\mathrm{Tr}\left(\sigma_0' \, \mathcal{M}\left[\frac{\sigma_0'}{2}\right]\right) \end{aligned} \tag{8.27}$$

This expression reduces to the relationship in equation (8.22) in the case of closed-system dynamics.

### *8.1.5 Calculating gate fidelities with quantum trajectories*

Since the open-system optimizer we are proposing is based on quantum trajectories, the target fidelities need to be expressed in terms of initial pure states. In the case of a 2-dimensional qubit, ref. [16] proves that $\bar{F}$ (and hence $F_o = \frac{3}{2}\bar{F} - \frac{1}{2}$) can indeed be expressed in terms of the average of four state-transfer fidelities of pure states as

$$\bar{F} = \frac{1}{4}\sum_{j=1}^{4} \mathrm{Tr}\left(U_t \rho_j U_t^\dagger \, \mathcal{M}[\rho_j]\right), \tag{8.28}$$

where $\rho_j = \frac{1}{2}\left(\sigma_0 + \vec{r_j}.\,\vec{\sigma}\right)$, $r_1 = \frac{1}{\sqrt{3}}(1,1,1)$, $r_2 = \frac{1}{\sqrt{3}}(-1,-1,1)$, $r_3 = \frac{1}{\sqrt{3}}(-1,1,-1)$ and $r_4 = \frac{1}{\sqrt{3}}(1,-1,-1)$. Therefore, if the qubit is not embedded in a bigger Hilbert space, both $\bar{F}$ and $F_o$ can be calculated by quantum trajectories. This calculation involves comparing the final state achieved in every trajectory with the corresponding target state for the four specified initial states,

$$\bar{F} = \frac{1}{4}\sum_{j=1}^{4} \mathrm{Tr}\left(|\psi_{tj}\rangle\langle\psi_{tj}| \, \mathcal{M}(|\psi_j\rangle\langle\psi_j|)\right) = \frac{1}{4m}\sum_{j=1}^{4}\sum_{m} |\langle\psi_{tj}|\psi_{fj}\rangle|^2, \tag{8.29}$$

where $m$ is the simulated number of trajectories for each initial state, $|\psi_{tj}\rangle$ and $|\psi_{fj}\rangle$ are the target and final states starting with initial state $|\psi_j\rangle$, respectively.

Since the fluxonium and 0-$\pi$ qubits have more than two levels, we need to consider the generalized case represented by equations (8.25) and (8.27). As shown, the expansion in Pauli matrices results in four terms. However, only the identity term differs from the $n = 2$ case. The three other terms can be calculated in a similar manner by averaging over the specific four initial states mentioned above. The full expression becomes

$$\bar{F} = \frac{1}{4}\sum_{j=1}^{4} \mathrm{Tr}\left(U_t \rho_j U_t^\dagger \, \mathcal{M}[\rho_j]\right) + \frac{1}{2}\left[\mathrm{Tr}\left(\sigma_0' \, \mathcal{M}\left[\frac{\sigma_0'}{2}\right]\right) - 1\right], \tag{8.30}$$

and

$$F_o = \frac{3}{8}\sum_{j=1}^{4} \mathrm{Tr}\left(U_t \rho_j U_t^\dagger \, \mathcal{M}[\rho_j]\right) + \frac{1}{4}\left[\mathrm{Tr}\left(\sigma_0' \, \mathcal{M}\left[\frac{\sigma_0'}{2}\right]\right) - 3\right]. \tag{8.31}$$

The first term in both expressions can be calculated with quantum trajectories using equation (8.29). Meanwhile, the second term can be expressed in the trajectories framework with the use of the following identity

$$\begin{aligned} \mathrm{Tr}\left(\sigma_0' \, \mathcal{M}\left[\frac{\sigma_0'}{2}\right]\right) &= \mathrm{Tr}\left(\sigma_0' \, \mathcal{M}\left[\frac{|0\rangle\langle 0| + |1\rangle\langle 1|}{2}\right]\right) \\ &= \frac{1}{2m}\sum_{j=0,1}\sum_{\mathrm{traj}} |\langle \psi_{fj}|P|\psi_{fj}\rangle|^2, \end{aligned} \tag{8.32}$$

where $\{|\psi_{f0}\rangle, |\psi_{f1}\rangle\}$ are the final trajectory states starting with initial states $\{|0\rangle, |1\rangle\}$.

In summary, we have shown that different gate fidelities in open quantum systems can be calculated as a linear superposition of a finite number of state-transfer fidelities. Therefore, we will focus on state-transfer fidelities and the algorithms that exist for calculating their gradients.

## 8.2 Open-system GRAPE

In standard optimal control theory, time evolution is discretized into $N$ time steps of duration $dt$ as discussed in section 3.2. During each time step $j$, the Hamiltonian consists of adjustable control parameters $u_{kj}$, assumed to be constants for each period $dt$. These control parameters multiply a set of control Hamiltonians $H_k$ which are added to the constant system Hamiltonian (the so-called drift Hamiltonian) $H_0$,

$$H_j \equiv H(j\,dt) = H_0 + \sum_k u_{kj} H_k. \tag{8.33}$$

This set of discretized Hamiltonians is used to propagate the initial state(s) at $t = 0$ towards the final state(s) at $t = t_N$. In the case of open quantum dynamics, the standard way to perform such propagation is through Liouville superoperators $\mathcal{L}_j$ which can be separated again into drift (including dissipation) and control superoperators as

$$\mathcal{L}_j = \mathcal{L}_0 + \sum_k u_{kj} \mathcal{L}_k. \tag{8.34}$$

The goal is to determine the optimal control parameters $u_{kj}$ that minimize a certain cost function $C$. Gradient descent algorithms can be utilized as long as gradients of $C$ with respect to the control parameters $u_{kj}$ can be calculated. For instance, an iterative approach may be used in which the control parameters are updated via

$$u_{kj} \to u_{kj} - \epsilon \frac{\partial C}{\partial u_{kj}}, \tag{8.35}$$

where $\epsilon$ is the update step size. Several algorithms exist to change $\epsilon$ adaptively in every iteration for better convergence. More complex gradient descent methods such as ADAM [67] may also be utilized.

Now, we focus on the key cost function in the open case; the distance between the target density matrix $\rho_T$ and the propagated density matrix $\rho_N$ at the final $N$th time step. The corresponding infidelity cost function can be expressed as

$$C = 1 - \mathrm{Tr}[\rho_T \rho_N]. \tag{8.36}$$

The open-system GRAPE method [15] relies on calculating the analytical gradients $\frac{\partial C}{\partial u_{kj}}$. For that purpose, consider the final density matrix $\rho(T)$, obtained after $N$-fold application

of the propagation superoperators

$$\rho_N = \Lambda_N \Lambda_{N-1} \dots \Lambda_1 \rho_0 \tag{8.37}$$

where the propagator at time step $j$ is

$$\Lambda_j = \exp(\mathcal{L}_j \, dt). \tag{8.38}$$

The main approximation in the calculation of gradients in the most basic open-system GRAPE implementation comes from expressing the derivative of every $\Lambda_j$ to first order in $dt$ as

$$\frac{\partial \Lambda_j}{\partial u_{kj}} \approx dt \, \Lambda_j \mathcal{L}_k. \tag{8.39}$$

which ignores higher-order terms that include commutators of $\Lambda_j$ and $\mathcal{L}_k$. Higher-order expansions with better accuracy have been considered [30]. as

$$\frac{\partial \Lambda_j}{\partial u_{kj}} = \Lambda_j \big[ -i dt \mathcal{L}_k + \frac{(dt)^2}{2!}[\Lambda_j, \mathcal{L}_k] + \frac{i(dt)^3}{3!}[\Lambda_j, [\Lambda_j, \mathcal{L}_k]] + \dots \big] \tag{8.40}$$

Working to first order in $dt$, the analytical gradient in this case can be calculated to be

$$\frac{\partial C}{\partial u_{kj}} \approx i \; dt \, \mathrm{Tr}(\lambda_j [H_k, \rho_j]), \tag{8.41}$$

where $\rho_j = \rho(j \, dt) = \Lambda_j \Lambda_{j-1} \dots \Lambda_1 \rho(0)$ is the initial density matrix propagated to time step $j$ and $\lambda_j = \Lambda_{j+1}^\dagger \Lambda_{j+2}^\dagger \dots \Lambda_N^\dagger \rho_T$ is the target density matrix backward-propagated to the same time step $j$.

Therefore, utilizing first-order GRAPE to optimize open quantum systems necessitates the calculation of $\rho_j$ and $\lambda_j$ at every time step $j$. This could be realized by propagating

both the initial and target density matrices using matrix exponentials of superoperators of dimension $d^2 \times d^2$. Hence, the simplest implementation of open-system GRAPE has all the computational and memory drawbacks discussed in section 7.1. Alternatively, direct integration for propagating density matrices [53, 45, 15] has been proposed to circumvent the need for calculating the $d^2 \times d^2$ superoperators. However, these techniques still necessaite the calculation and storing of matrices of size $d \times d$. Therefore, using quantum trajectories is a potential area of improvement for open-system GRAPE when the Hilbert space dimension is large.

However, extending gradient-based algorithms to deal with quantum trajectories is not entirely straightforward due to the randomness inherent in every time step of each trajectory, as we shall discuss next.

## 8.3 Direct gradients in quantum trajectories

Consider the problem of efficiently calculating analytical gradients for quantum trajectories by comparing with the case of closed-system GRAPE. Each quantum trajectory consists of the time evolution of a pure state which is described by a stochastic Schrödinger equation [133]. In this aspect, it resembles the closed-system evolution under the ordinary Schrödinger equation.

However, in the case of quantum trajectories, we note two main differences distinguishing quantum trajectories from the evolution of closed quantum systems. First, the propagation in every time step is no longer unitary due to the non-Hermitian part of $H_{\text{eff}}$, see Eq. (7.15). Second, quantum trajectories involve randomness due to the intermittent occurrence of quantum jumps. The propagation,

$$|\psi_N\rangle = \frac{M_N M_{N-1} \cdots M_1 |\psi(0)\rangle}{\| M_N M_{N-1} \cdots M_1 |\psi(0)\rangle \|} \tag{8.42}$$

is described using the non-unitary propagator

$$M_j = \exp(-iH_j dt - \tfrac{dt}{2} \textstyle\sum_l \gamma_l c_l^\dagger c_l) \tag{8.43}$$

in the absence of a jump, or

$$M_j = c_l, \tag{8.44}$$

if a jump occurs in decoherence channel $l$. Imitation of the cache-free gradient calculation via back-propagation of states will generally fail for quantum trajectories. Back-propagation cannot be achieved by $M_j^\dagger$ anymore due to non-unitarity. Even worse, most realistic jump operators do not even have an inverse as they represent irreversible dynamical changes of the system, e.g., the decay of an excitation. This leads to the necessity of caching intermediate information during the numerical simulation and takes away an important advantage of using fully analytical forms of GRAPE.

Besides the need for caching, obtaining an analytically closed form for gradients is considerably more tedious for quantum trajectories than in the case of closed unitary evolution. In part, this is due to the need for explicit normalization of propagated states, see Eq. (8.42). Different from closed evolution, the final single-trajectory state $|\psi_N\rangle$ now depends on the control parameters not only via the propagators $M_j$ but also via the normalization factor $F_N = \|M_N M_{N-1} \cdots M_1 |\psi(0)\rangle\|$ in the denominator of Eq. (8.42). As a result, gradients with respect to the control parameters thus require both $\partial M_j / \partial u_{kj}$ and $\partial F_N / \partial u_{kj}$. This makes it generally much more cumbersome to obtain analytical gradients and use them in an efficient way. We proceed to showing how the analytical gradients of the cost function

$$C = 1 - |\langle \psi_T | \psi_N \rangle|^2 \tag{8.45}$$

could be calculated in a quantum trajectory.

If we define

$$F_j = \sqrt{\langle\psi(0)|\, M_1^\dagger \cdots M_j^\dagger M_j \cdots M_1\, |\psi(0)\rangle} \tag{8.46}$$

and $Y = F^2$ with $F = F_N$, then the gradient in the case of no jump is

$$\begin{aligned}
&\frac{\partial Y}{\partial u_{kj}} = \frac{\partial}{\partial u_{kj}} \langle\psi(0)|\, M_1^\dagger \dots M_{N-1}^\dagger M_N^\dagger M_N M_{N-1} \dots M_1\, |\psi(0)\rangle = \\
&idt\, \langle\psi(0)|\, M_1^\dagger \dots M_j^\dagger H_k \dots M_N^\dagger M_N M_{N-1} \dots M_1\, |\psi(0)\rangle - \\
&- idt\, \langle\psi(0)|\, M_1^\dagger \dots M_N^\dagger M_N \dots H_k M_j \dots M_1\, |\psi(0)\rangle = \\
&+ 2dtFF_j\, \mathrm{Im}(\langle\psi_N|\, M_N \dots M_{j+1} H_k\, |\psi_j\rangle).
\end{aligned} \tag{8.47}$$

Hence,

$$\frac{\partial(\frac{1}{F})}{\partial u_{kj}} = \frac{\partial(\frac{1}{F})}{\partial Y} \frac{\partial Y}{\partial u_{kj}} = -\frac{dtF_j\, \mathrm{Im}(\langle\psi_N|\, M_N \dots M_{j+1} H_k\, |\psi_j\rangle)}{F^2}. \tag{8.48}$$

Therefore, we can finally write the dependence of the final state $|\psi_N\rangle$ on the control parameters as

$$\frac{\partial\, |\psi_N\rangle}{\partial u_{kj}} = \frac{-idtF_j M_N M_{N-1} \dots M_{j+1} H_k\, |\psi_j\rangle}{F} - \frac{dtF_j\, \mathrm{Im}(\langle\psi_N|\, M_N \dots M_{j+1} H_k\, |\psi_j\rangle)\, |\psi_N\rangle}{F} \tag{8.49}$$

Thus, if no jump happens at time step j, the gradient is

$$\begin{aligned}
\frac{\partial C}{\partial u_{kj}} &= -\langle\psi_T|\frac{\partial|\psi_N\rangle}{\partial u_{kj}}\langle\psi_N|\psi_T\rangle - \langle\psi_T|\psi_N\rangle\frac{\partial\langle\psi_N|}{\partial u_{kj}}|\psi_T\rangle \\
&= -\langle\psi_N|\psi_T\rangle \\
&\langle\psi_T|\Big[\frac{-idtF_jM_NM_{N-1}\dots M_{j+1}H_k|\psi_j\rangle}{F} - \frac{dtF_j\,\mathrm{Im}(\langle\psi_N|M_N\dots M_{j+1}H_k|\psi_j\rangle)|\psi_N\rangle}{F}\Big] \\
&- \langle\psi_T|\psi_N\rangle \\
&\Big[\frac{idtF_j\langle\psi_j|H_kM_{j+1}^\dagger\dots M_{N-1}^\dagger M_N^\dagger}{F} - \frac{dtF_j\,\mathrm{Im}(\langle\psi_N|M_N\dots M_{j+1}H_k|\psi_j\rangle)\langle\psi_N|}{F}\Big]|\psi_T\rangle \\
&= \frac{-2F_jdt}{F}\,\mathrm{Im}\left[\langle\psi_T|\Big[\prod_{j'>j}M_{j'}\Big]H_k|\psi_j\rangle\langle\psi_N|\psi_T\rangle\right] + \\
&(C-1)\frac{dtF_j\,\mathrm{Im}(\langle\psi_N|\left[\prod_{j'>j}M_{j'}\right]H_k|\psi_j\rangle)}{F}.
\end{aligned} \tag{8.50}$$

If we define $f_{jk}(\psi) = \langle\psi|\left[\prod_{j'>j}M_{j'}\right]H_k|\psi_j\rangle$, then the analytical gradients take the form

$$\frac{\partial C}{\partial u_{kj}} = \begin{cases} \frac{-2F_jdt}{F}\,\mathrm{Im}(f_{jk}(\psi_T)\langle\psi_N|\psi_T\rangle) + \frac{(C-1)}{F}dtF_j\,\mathrm{Im}(f_{jk}(\psi_N)) & \text{no jump} \\ 0 & \text{jump} \end{cases} \tag{8.51}$$

Note that the extra gradient term due to the decay of the norm of the state is only negligible at the beginning of optimization when $C \approx 1$ but as the algorithm enhances the fidelity, this term becomes more and more dominant.

This whole analysis also assumed that a jump occupies a whole time step while in principle it should be instantaneous. Instead, a more accurate approach would be to integrate the state to find the time $t_{\text{jump}}$ when its norm reaches the random number $r$, apply a quantum jump at that time, and then keep propagating/integrating the resulting state starting again from $t_{\text{jump}}$. In this case, the corresponding analysis to find the analytical gradient would

be even more complex since it clearly deviates from the standard picture of having one propagator at every time step.

Therefore, one way to better handle the cumbersome matter of gradient calculation for quantum trajectories given any desired cost function, is to use automatic differentiation instead of relying on analytical gradient forms which, if existent, do not generally support efficient implementation without caching.

# CHAPTER 9

# OPEN-SYSTEM OPTIMAL CONTROL IMPLEMENTATION

## 9.1 Conditional graphing using TensorFlow

In order to use automatic differentiation for quantum trajectories, we must define the computational graph of every trajectory in terms of basic matrix operations hard-coded in the differentiator. One challenge in this process is the randomness inherent in each quantum trajectory which prevents the computational graph from having a fixed structure. Instead, the relationship between the inputs and the cost function is only determined at runtime by the choice of the random numbers entering the individual trajectory. Hence, our differentiator is equipped with the flexibility of dynamically creating the computational graph of each trajectory – a scenario called conditional graphing. Fig. 9.1 illustrates the type of conditional graph needed for quantum trajectories.

We implement our open quantum optimizer using the TensorFlow library for automatic differentiation and machine learning [1] similar to the closed-system implementation discusses in Chapter 5. Developed by Google's machine intelligence research group, TensorFlow allows for conditional graphing and dynamically setting the size of the computational graph at runtime. TensorFlow is easily integrated with Python and has a comprehensive basic set of operations for matrix algebra with predefined gradients. In addition, TensorFlow has support for distributed learning, by running different parts of the computational graph or multiple copies of the same graph on different machines in parallel. This is crucial for the implementation of quantum trajectories, as we will discuss below.

## 9.2 Techniques for handling trajectories

Another very important consideration for the optimizer is the ability to implement the number of trajectories needed for the simulation efficiently, which differs according to the

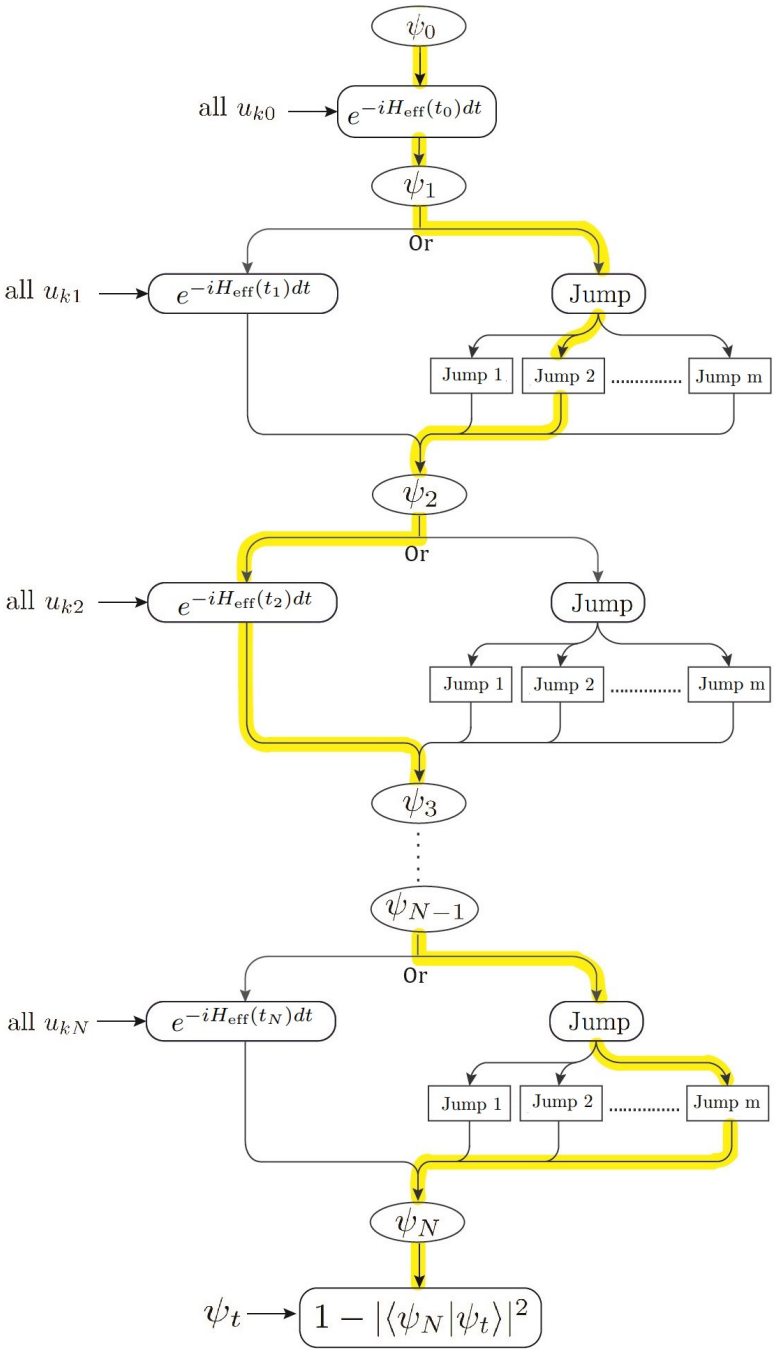


Figure 9.1: The computational graph of a quantum trajectory evolution must be conditional and allow for all possibilities of the forward path. Each forward path is only determined at runtime through the choice of random numbers entering the trajectory generation. At every timestep, the evolution either proceeds via the non-unitary Hamiltonian $H_{\text{eff}}$ or through a jump $\in \{c_l\}$ from one of the possible $m$ jump channels. The yellow path is an example of the many possible trajectories.

size and nature of each optimization problem. Fortunately, it is not always necessary to exhaustively sample the full statistics in each single iteration of the optimization process. Instead, it may be sufficient to generate a smaller number of trajectories which only partially represent the statistics. In the case where only a single trajectory is used per iteration, the procedure is known as Stochastic Gradient Descent (SGD) [65, 14, 129, 138]. This generally leads to non-monotonic convergence, often requires more iterations and can produce noisy pulses. A better way for our model is to include a moderate number of trajectories in each iteration so that every iteration is based on a batch of data. This procedure, where gradients applied in each iteration are generated by a batch of trajectories, is known as mini-batch SGD [110].

Using quantum trajectories allows for flexibility in the ways simulations are implemented, since trajectories can be constructed independently of each other. Whether to generate trajectories in series and/or in paralel, and how many trajectories should be grouped together, are examples of questions that the optimizer should address on a case-by-case basis, depending on the specific optimization problem. Hence, we implement different ways of grouping and propagating trajectories in our optimizer to better match the nature of each optimization problem. In particular, the following three techniques are used to handle trajectories generation.

### *9.2.1 Improved-sampling algorithm*

We present an algorithm for generating a balanced sample of trajectories per iteration using only a fraction of the needed number of trajectories. This leads to reducing the memory usage of the calculation while keeping convergence smooth and time efficient.

Suppose $m_{\text{tot}}$ is the batch size, i.e., the number of trajectories to be used in every iteration within mini-batch SGD. SGD leads to convergence even with relatively small batch sizes without significantly reducing the convergence speed as will be shown in the applications.

Therefore, SGD allows for limiting $m_{\text{tot}}$ to a small number which helps further reducing the complexity.

There are two potential limitations to this approach. First, control pulses will typically be noisy if a very small number of trajectories is used for every gradient update. To prevent this from happening, we utilize pulse smoothing cost functions to ensure that stochastic noise in the pulses is canceled. Second, there is an increase in the number of iterations required to reach convergence. To circumvent this limitation, we introduce a technique which only implements a subset of size $m_{\text{sim}}$ of the intended $m_{\text{tot}}$ trajectories. From this subset, we generate a better balanced sample for every iteration that makes convergence smooth and fast and also hugely reduces runtime and memory usage of the optimizer, especially if jumps are rare.

In many practical cases, dissipation and dephasing time scales are much longer than the time scale governing the dynamics of the system. For example, many optimized gates on superconducting qubits may take no more than $0.01 - 0.1\mu$s, while decoherence times of the qubit are orders of magnitude larger. In that case, dissipative terms in the Liouvillian will have a smaller impact on the dynamics and quantum jumps are less likely to occur. As a result, inside a statistically representative batch of trajectories, many trajectories will be identical to the no-jump trajectory. To eliminate the redundancy of generating the no-jump trajectory many times, our implementation allows for performing a test run that first generates the no-jump trajectory. Then, using the fact that the state norm of a quantum trajectory monotonically decreases over time [17] in the absence of jumps, we extract the final norm of the no-jump trajectory and identify it with the no-jump probability for the total evolution time. From then on, we only generate trajectories which do include jumps by controlling the range of the random numbers $r$. Consequently, all remaining generated trajectories in a batch are representatives of jump trajectories and the total number of trajectories to be generated can be reduced: if jumps are sufficiently rare, $m_{\text{sim}}$ is only

a small percentage of $m_{\text{tot}}$, with the precise fraction given by the probability of jumps. Finally, we perform a weighted average of the gradients of both the no-jump trajectory and the generated jump trajectories according to the calculated jump/no-jump probability. This algorithm is summarized below:

1. Generate the no-jump trajectory (random number $r = 0$).

2. Save the no-jump gradients $g_{\text{nj}}$ from this trajectory.

3. Extract the norm of the final state of this no-jump trajectory, $p = \langle\psi_N|\psi_N\rangle$ and assign it as the no-jump probability.

4. If $m_{\text{tot}}$ trajectories are needed, generate only

$$m_{\text{j}} = \lceil (1-p) m_{\text{tot}} \rceil \tag{9.1}$$

   jump trajectories with $r \in [p, 1)$, where $\lceil x \rceil$ denotes the integer ceiling of $x$. The reduced range for $r$ guarantees at least one jump will happen since the norm will definitely drop below $r$. After the jump, $r$ is reset to the full range $[0, 1)$ to simulate potential additional jumps.

5. Calculate the averaged jump gradient $g_{\text{j}}$ from the $m_{\text{j}}$ trajectories.

6. Calculate the net gradient of all $m_{\text{tot}}$ trajectories,

$$g = (1-p) g_{\text{j}} + p\, g_{\text{nj}} \tag{9.2}$$

If jumps are rare, this algorithm saves significant resources by only generating $m_{\text{sim}} = m_{\text{j}} + 1$ trajectories instead of $m_{\text{tot}}$. In addition, instead of just implementing randomly chosen trajectories every iteration, the algorithm will always generate a balanced sample every iteration/batch that represents jumps and no jumps with correct weights. This leads

to a much smoother and faster convergence than the case where $m_{\text{tot}}$ is a collection of (generally unbalanced) random events.

As a concrete example, consider the case of a problem with jump probability $(1-p) = 10\%$, and with a small batch size $m_{\text{tot}} = 10$ to reduce computational costs. Then, there is a chance that all 10 trajectories used in a given iteration could be identical to the no-jump trajectory. Since the occurrence of jumps is not uniform from iteration to iteration, convergence would be non-monotonic, and properly sampling the dynamics would require a bigger number of iterations. However, using the algorithm described above, each iteration is enforced to contain the same balanced statistics between jumps and no jumps, even for small sample size. Therefore, the improved-sampling algorithm is a crucial element of our optimizer that renders the memory requirements of an open-system problem similar to that of a closed system, if jumps are rare, as will be demonstrated in the applications.

### *9.2.2 Matrix-Vector exponential and clustering trajectories*

Another important computational bottleneck is the evaluation of the matrix exponential required for state propagation via the effective Hamiltonian. Our implementation eliminates the need to calculate the matrix exponential through matrix-matrix multiplication. Instead, when propagating a state $V_j = |\psi_j\rangle$ to $V_{j+1} = |\psi_{j+1}\rangle$ through the matrix exponential $e^A$, where $A = -i(H_{\text{eff}})_{j+1}dt$, we use an iterative Taylor series to reexpress the propagation as

$$V_{j+1} = e^A V_j = (1 + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \cdots)V_j = V_j + AV_j + \frac{1}{2!}A(AV_j) + \frac{1}{3!}A(A(AV_j)) + \cdots \quad (9.3)$$

The new state can thus be calculated by an iterative series of matrix-vector multiplications, reducing the complexity of propagation at every time step from $\mathcal{O}(n_{\text{Taylor}}d^3)$ which is required by matrix-matrix multiplication to $\mathcal{O}(n_{\text{Taylor}}d^2)$, where $n_{\text{Taylor}}$ is the number of Taylor expansion terms kept in the simulation. We note that the convergence of this iterative

approach could be enhanced in future implementations using Newton polynomials instead of Taylor expansion [119].

Using the improved-sampling algorithm, a batch of $m_{\text{sim}}$ initial states needs to be propagated per iteration. Instead of sequential generation of each trajectory, storing all gradients and then averaging them in the end, the above matrix-vector implementation allows for simultaneous propagation of a number of vectors. The procedure described in equation (9.3) can be generalized from $V_j$ representing a $d \times 1$ vector, to denoting a matrix of size $d \times m_{\text{sim}}$. Using this idea, we can cluster trajectories together and process them in a faster way than running them in series [79]. Clustering trajectories is particularly useful if the needed number of trajectories is much smaller than the Hilbert space dimension for big-sized problems or if the Hilbert space dimension is relatively small and there are a lot of unused resources when propagating one trajectory at a time. In our implementation, we treat the desired number of trajectories to be combined in each iteration (the batch size) as an adjustable parameter. It may be specified at runtime, and enables dynamically selecting the size of the computational graph. It also allows for combining several batches by applying their averaged gradients to the control parameters. Those two features together give reasonable flexibility in dividing the needed number of trajectories into batches of combined trajectories and in matching available computational resources to the concrete size and nature of each optimization problem.

### *9.2.3 Parallelization of trajectories*

While generating the needed statistics from a smaller number of clustered trajectories saves memory and runtime, a prime advantage of quantum trajectories is their high degree of parallelizability. We can utilize parallelization to further improve the efficiency and flexibility of the optimizer. Our implementation uses TensorFlow's distributed learning features to run a number of different clustered trajectories on different nodes in parallel. We have built an

interface between the SLURM manager for operating clusters [134] and TensorFlow allowing for the use of both synchronous and asynchronous training. Here, synchronous training refers to the situation when all compute nodes must finish their mini-batches together and their gradients are then averaged. Asynchronous training, by contrast, gives every compute node the ability to update the control parameters once its batch gradients are ready. Therefore, asynchronous training is essentially equivalent to performing multiple independent optimization iterations in parallel. Both types of training are of practical importance, depending on the needed statistics per iteration for stable convergence. Our implementation uses "between graph replication" which means that every node builds its own identical version of the computational graph. Then, communication between graphs is coordinated through a chief worker machine.

# CHAPTER 10

# OPEN-SYSTEM OPTIMAL CONTROL APPLICATIONS

## 10.1 Transmon qubit state transfer

To illustrate the improved-sampling algorithm, we consider a transmon qubit with $n = 4$ levels, initialized in the ground state $|g\rangle$. We wish to transfer the system to the first excited state $|e\rangle$. For the transmon, we assume a frequency difference between ground and excited levels of $\omega_{ge}/2\pi = 3.9$ GHz and an anharmonicity of $\alpha/2\pi =$ - 225 MHz. The control Hamiltonians $\{H_x, H_z\}$ couple to the $x$ and $z$ degrees of freedom of the qubit, so that the net Hamiltonian is given by

$$H = \omega_{ge} b^\dagger b + \frac{\alpha}{2} b^\dagger b (b^\dagger b - 1) + \Omega_x(t)(b^\dagger + b) + \Omega_z(t) b^\dagger b. \tag{10.1}$$

Here, $b$ and $b^\dagger$ are ladder operators for the transmon excitation level, truncated at an appropriate level ($n = 4$ in our case). The qubit is coupled to a heat-bath environment, resulting in relaxational dynamics with characteristic time $T_1$. Working at zero temperature, the corresponding jump operator is $b$. We utilize the following cost functions: state-transfer infidelity to maximize the fidelity between the final evolved state and $|e\rangle$, pulse-smoothing cost functions to generate smooth realizable control pulses and forbidden-state cost functions to forbid the occupation of the $n$-th level so that the truncation of the transmon levels remains valid.

First, we study the effect of relaxation on the results from the state-transfer optimization. In the closed-system case where relaxation is absent ($T_1 \to \infty$), we readily achieve state-transfer fidelities of 99.99% within a total evolution time of $T = 10$ ns, see Fig. 10.1. However, if the qubit is fairly lossy (taking, for example, $T_1 = 100$ ns), the previously determined pulse train only achieves a state-transfer fidelity of 96.2% since occupation of the $|e\rangle$ level is inevitably subject to dissipation. Re-running the optimization in the presence of $T_1$

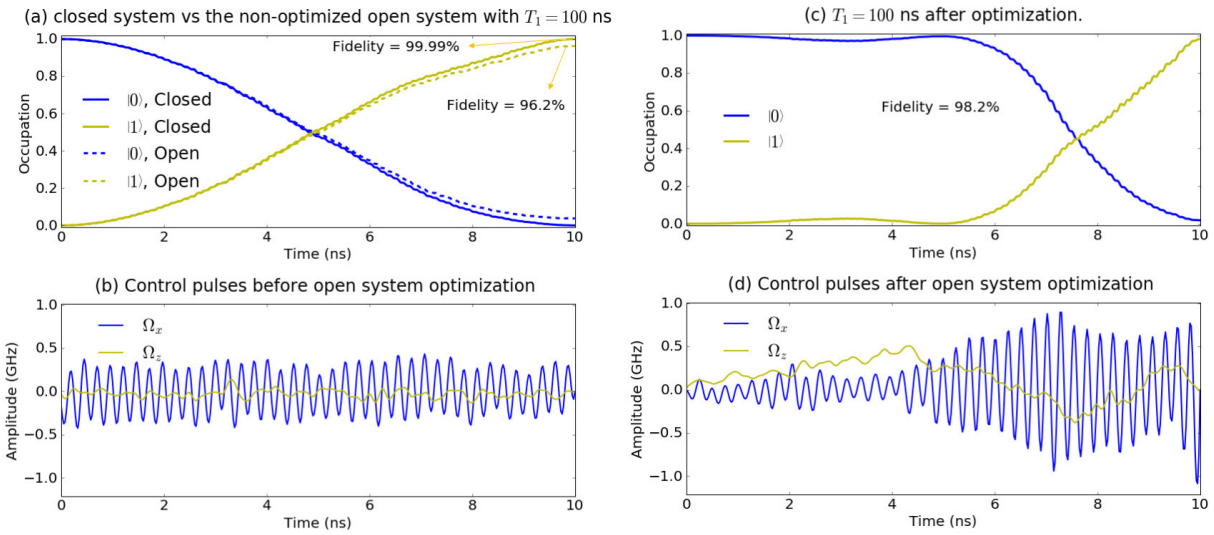


Figure 10.1: Optimizing state transfer from the ground to the first excited level in a lossy transmon qubit. Panels (a) and (c) show the occupation of both levels monitored over the pulse period while panels (b) and (d) show the pulse trains obtained from optimization. (a) In the absence of relaxation, the optimized solution reaches a closed-system fidelity of 99.99%. Then when this solution is applied in the presence of relaxation processes ($T_1 = 100\,\text{ns}$), the resulting state-transfer fidelity drops to 96.2%. (b) The corresponding pulse sequences. (c) Results from trajectory-based GRAPE. The new optimized solution raises the state-transfer fidelity to 98.2% (d) The optimized pulse minimizes relaxation effects by delaying the pulse as much as possible, then rapidly performing the transfer using increased power.

processes, we succeed in increasing the fidelity to 98.2%, so we gain around 2% even for this example of a rather lossy qubit, see Fig. 10.1.

Inspection of the results reveals that the optimizer aims to minimize the detrimental effects of relaxation by delaying the state-transfer operation as much as possible. This way, the total time period over which the state $|e\rangle$ is occupied, is reduced and there is, hence, a smaller time window where the system is sensitive to decay. While the closed-system case utilizes the whole time to perform the state transfer, the open dynamics state transfer is much more asymmetric in time, reflecting the asymmetry between the ground and the first excited state as far as decoherence is concerned. Note that the optimization cannot fully bring the fidelity back to 99.99% because the occupation of $|e\rangle$ is limited by the relaxation $e^{-t/T_1}$. Note that this result is also useful in optimizing the final time for the state transfer,

since the obtained pulse indicates the total time required for the state transfer, which is even relevant for the choice of the total time in the closed-system analysis.

Next, we discuss the choices of $m_{\text{tot}}$ and $m_{\text{sim}}$ needed for the optimization. Deploying our improved-sampling algorithm, we can perform the trajectory-based optimization at nearly the same computational cost as in the closed-system case. We ran simulations using different values of $T_1$ to obtain the probability of jumps for each case. Note that this probability changes with the iterations of the optimization since the Hamiltonian changes according to the updated control pulses. So, focusing on the maximum value we get for the probability of jumps, we get the results in Fig. 10.2 for different values of the ratio $T_f/T_1$ where $T_f$ is the final time of the state transfer.

We consider a case where the number of trajectories $m_{\text{tot}}$ required for good convergence is as big as 10,000, yet only a small number of trajectories actually needs to be simulated. As Fig. 10.2 shows for $T_f/T_1 < 10^{-2}$, the probability of jumps is less than 0.6%, allowing us to obtain the desired sampling by generating merely 60 trajectories. Even if the final time is increased to $0.1\,T_1$ like in Fig. 10.1, the probability of jumps is around 5% which is still a small fraction. In most of realistic applications, the sample size $m_{\text{tot}}$ does not have to be as large as 10,000, since using our algorithm allows for good convergence even for much smaller total numbers of trajectories. We compare the convergence for different choices of $m_{\text{tot}}$ using our algorithm in Fig. 10.3.

As shown in Fig. 10.3(d), we obtain good convergence even for a small $m_{\text{tot}} = 10$ and $m_{\text{sim}} = 2$ (i.e., only generating two trajectories per iteration) within around 100 iterations, reaching a target fidelity of 97.5%. By comparison, for a significantly larger number of trajectories $m_{\text{tot}} = 10,000$, we reach the same fidelity within 60 iterations. Therefore, we only need to double the number of iterations to simulate the system using a thousand times fewer trajectories per iteration. In addition, every iteration will be much faster since it only includes running two trajectories.

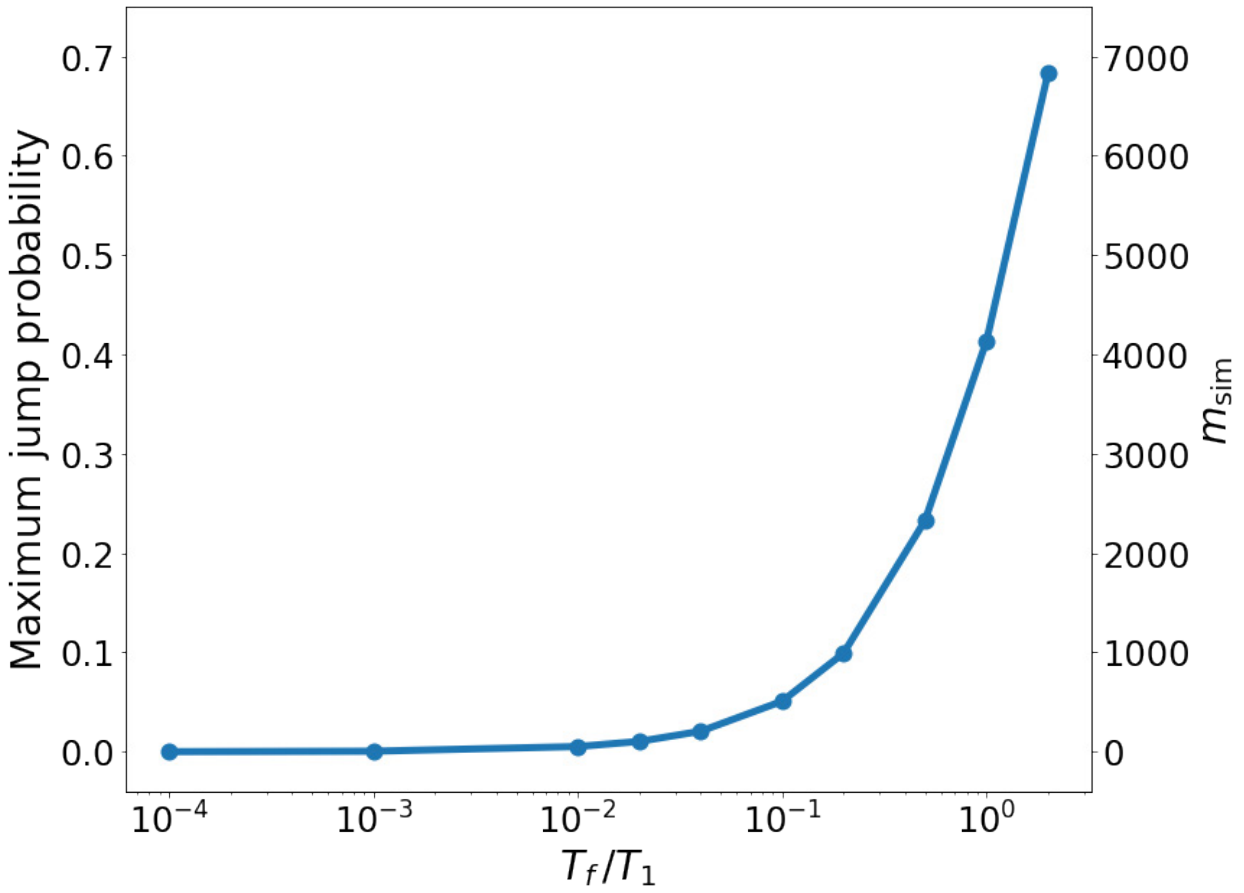


Figure 10.2: Maximum jump probability for the driven transmon qubit as a function of the simulation time $T_f$ measured in units of the relaxation time $T_1$. This represents the fraction of the number of trajectories that the improved-sampling algorithm will generate every iteration. In most realistic cases, the fraction does not exceed $1 - 5\%$, allowing for a significant reduction in computational costs. The right vertical axis represents the number of trajectories $m_{\mathrm{sim}}$ that needs to be simulated if $m_{\mathrm{tot}} = 10{,}000$.

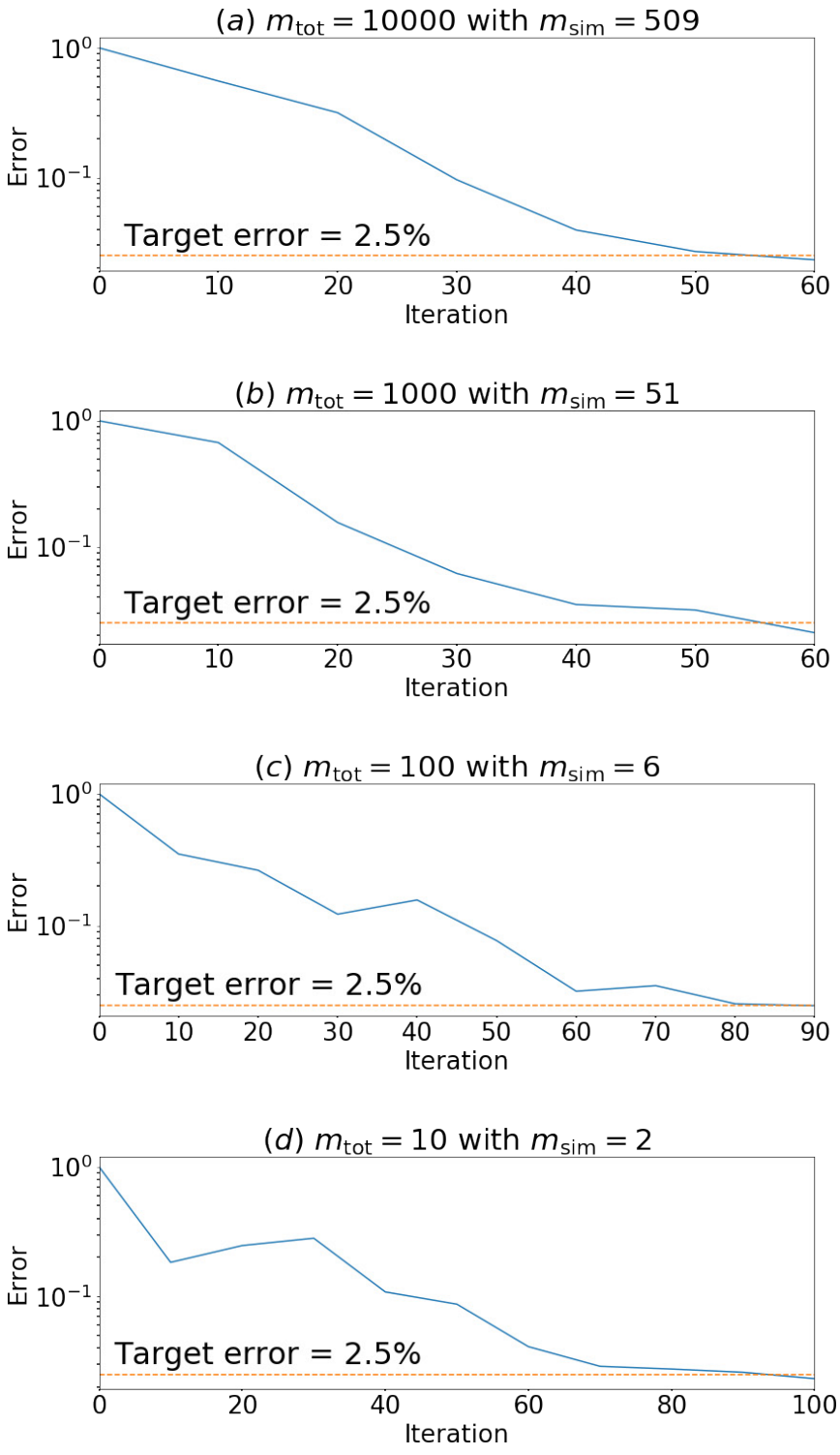


Figure 10.3: Convergence of the optimization algorithm to a target fidelity of 97.5% for different values of $m_{\text{tot}}$, using the improved-sampling algorithm. The reported fidelities are calculated using a sufficiently large number of trajectories.

Running the same optimization problem for $m_{\text{tot}} = 10$ but without the improved-sampling algorithm leads to convergence to the same target fidelity after 320 iterations. Therefore, using the improved-sampling algorithm needs only two trajectories per iteration for a total number of 200 trajectories to reach convergence, in comparison to 3,200 trajectories needed when not employing the algorithm.

## 10.2 Lambda system population transfer

### *10.2.1 Problem overview*

Having shown how the improved-sampling algorithm helps reduce the problem complexity for the toy example of a transmon state transfer, we next consider a more realistic case of driving nearly forbidden transitions in a three-level $\Lambda$ system as shown in Fig. 10.4.

Two levels of it, $|1\rangle$ and $|3\rangle$, are stable, i.e., the direct transition between $|1\rangle$ and $|3\rangle$ is forbidden. The third, intermediate level $|2\rangle$ can decay to either of the former two states and direct matrix elements allow one to drive the $|1\rangle \longleftrightarrow |2\rangle$ and $|3\rangle \longleftrightarrow |2\rangle$ transitions. The goal is to transfer the system from one stable state to the other while avoiding significant occupation of the intermediate state which is subject to errors from spontaneous dissipation.

### *10.2.2 Protocol 1: Raman Transitions*

We will compare two existing protocols to induce the desired transition. The first is the two-photon Raman transition in which the system is driven by two off-resonant pulses of amplitudes $\Omega_1$, $\Omega_2$, and detunings $\delta_1 = \delta_2 = \delta$. For convenience, it is assumed that each pulse couples solely to a single transition between the intermediate state and one of the stable states. Adiabatic elimination is an approximation that may be performed if the detuning $\delta$ is considered large [18]. This approximation assumes small occupation of the intermediate level, and hence, the system may be treated as an effective two-level system.

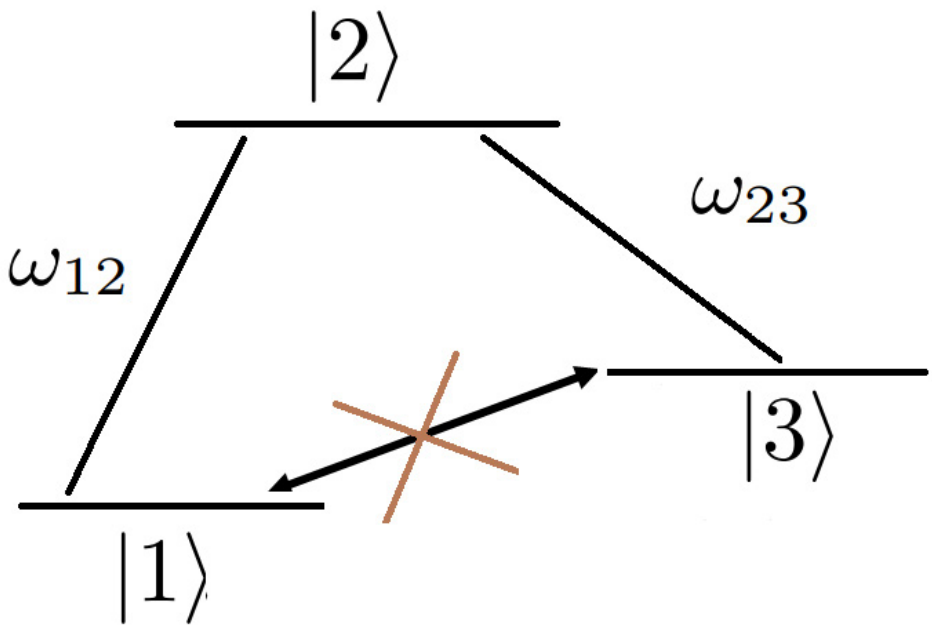


Figure 10.4: The $\Lambda$ system consists of ground state $|1\rangle$ and meta-stable excited state $|3\rangle$, as well as an intermediate lossy state $|2\rangle$. The frequencies $\omega_{12}$ and $\omega_{23}$ are distinct and no direct matrix element exists between $|1\rangle$ and $|3\rangle$. Hence, state transfer between them must invoke the intermediate state $|2\rangle$. The system is driven with a pulse $\zeta(t)$ that is optimized to maximize the state-transfer fidelity.

Under this assumption, this system is shown to be effectively driven by a Rabi oscillation of an effective frequency that is proportional to both $\Omega_1$ and $\Omega_2$. In this limit, the dissipation of the intermediate level is negligible since the level remains largely unoccupied during the transfer.

This approximation, however, is only valid if $\Delta = \delta_1 + \delta_2 \gg \Omega_1, \Omega_2$. If the transfer is desired to occur over shorter time scales, then the required effective Rabi frequency, and hence $\Omega_1$ and $\Omega_2$, must be increased. This may invalidate the adiabatic elimination condition, since needed larger detunings will cause the frequencies used by the two pulses to be too far from the transitions frequencies. In that case, it is not guaranteed that each pulse drives a single transition separately as assumed by the adiabatic elimination. Hence, this will result in the system deviating from the simplified picture of an effective two-photon process as we will show in the results subsection below.

### *10.2.3 Protocol 2: STIRAP*

The second commonly used protocol is the stimulated-Raman-adiabatic-passage (STIRAP) method. This involves the application of two partially overlapping pulses which adiabatically keep the $\Lambda$ system in a superposition of the two stable states without occupying the intermediate level [43]. First, a Stokes pulse [with amplitude $\Omega_S$(t)] is used to couple the two unoccupied states $|2\rangle$ and $|3\rangle$. Then, a pump pulse [with amplitude $\Omega_P$(t)] couples states $|1\rangle$ and $|2\rangle$ in such a way that makes direct transition from $|1\rangle$ to $|3\rangle$ possible. STIRAP assumes that the rotating wave approximation (RWA) is valid which causes the system to have three-time dependent eigenstates, one of them has no projection in the $|2\rangle$ state at all times [124]. This eigenstate is labeled the dark state $|\psi_d\rangle\,(t)$. If the pulses are changed adiabatically such that $\Omega_S$(t) is smoothly turned off while $\Omega_P$(t) peaks and then turns off, the system will remain in $|\psi_d(t)\rangle$ and reaches the desired state at the end without occupying the intermediate state at all as ensured by remaining in the dark state.

The required adiabaticity of the transfer, however, necessitates strong limits on the time needed for STIRAP transfer. In particular, the minimum time where pulses overlap is inversely proportional to the peak pulse amplitude used (in frequency units) [7]. The constant of proportionality is estimated from experimental data, and for around 95% STIRAP efficiency, it is estimated to be 10 [124]. The total time needed for a complete STIRAP transfer is usually around twice of the overlap time. This ensures that the delay between the two pulses gives maximum efficiency [7]. This poses a restriction on how fast the transfer can be accomplished. While the peak pulse amplitude could be increased to achieve shorter times, high pulse powers would eventually violate the RWA and result again in population of the intermediate level. Therefore, to achieve fast transfer between the two stable states in a $\Lambda$ system, existing techniques will inevitably involve partial population of the intermediate state which endangers the transfer fidelity because of its dissipative nature.

Recently, enhanced protocols which use shortcuts to adiabaticity [135] were presented to improve the speed of STIRAP while maintaining high transfer fidelities. Our optimal control results are comparable to those techniques in terms of speed and transfer efficiency.

### *10.2.4 Simulation details*

Different from the Raman and STIRAP protocols described above, we do not restrict the pulses to a single frequency component. Instead, we work with a single pulse of general form that can couple to both transitions. The Hamiltonian then takes the form

$$H = \sum_{i=1}^{3} \omega_i \left|i\right\rangle \left\langle i\right| + \zeta(t)(H_{12} + \alpha H_{23}), \tag{10.2}$$

with the definition

$$H_{ij} = \left|i\right\rangle \left\langle j\right| + \left|j\right\rangle \left\langle i\right|. \tag{10.3}$$

Here, $\alpha$ is a factor that relates the matrix elements of the two transitions. For our simulations, we used the sample values $\omega_1/2\pi = 0$ GHz, $\omega_2/2\pi = 5$ GHz, $\omega_3/2\pi = 1.8$ GHz and $\alpha = 1$. We associate state $|2\rangle$ with a relatively short relaxation time $T_1 = 20\,\text{ns}$ due to decay into either of the two stable states, and we set the target transfer time to 10 ns, while limiting the amplitude of $\zeta(t)$ to an experimentally reasonable maximum value of $3/2\pi\,\text{GHz}$.

Note that with these parameters, STIRAP requires a minimum of 42 ns for the full time of the protocol. Hence, the optimizer will search for a solution that is at least four times faster than STIRAP, but still maintains low occupation of $|2\rangle$ and high transfer fidelity.

### *10.2.5 Results*

With a maximum jump probability of around 10% (as calculated during simulation), the improved-sampling algorithm allows for a 90% reduction of the number of trajectories generated. Around 10 trajectories were used per iteration, thus accounting for an effective number of simulated trajectories of 100 per iteration. Convergence was reached with an optimized fidelity of 98.0%. To compare the solution against the two-photon Raman transition using the same total transfer time and pulse amplitude, trials with different values for the two pulse detunings and amplitudes were performed. The best solution yields only a fidelity of 84.1% and shows significant occupation of the intermediate level. The results are summarized in Fig. 10.5. In summary, our optimizer succeeds in high-fidelity population transfer in a $\Lambda$ system with a comparatively short transfer time for which standard protocols are significantly less efficient.

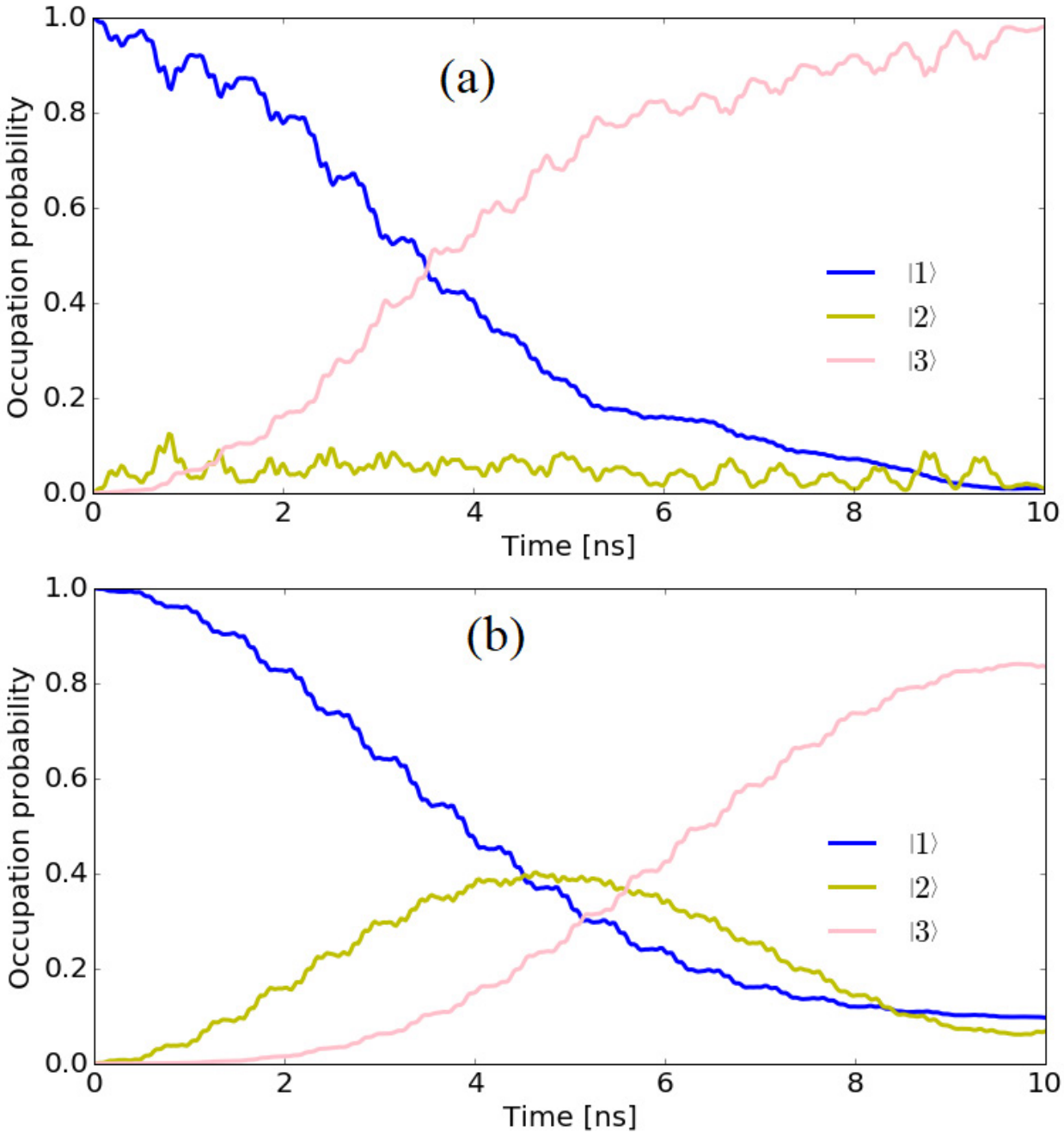


Figure 10.5: Results of driving the Lambda system with the optimized pulses vs. the usual two-photon Raman pulses. (a) The optimized pulse achieves a transfer fidelity of 98.0% by using several tones to properly limit the occupation of the lossy $|2\rangle$ state. (b) The two-photon Raman transition fails to limit the occupation of $|2\rangle$, and hence can only reach a fidelity of 84.1% within the parameter regime of our simulation.

## 10.3 Quantum Non Demolition (QND) readout of a transmon qubit allowing fast resonator reset

The previous two applications showed how our optimizer works in scenarios where jumps are rare. In the following application, we will deal with a situation where jumps are less rare, though rare enough to limit jump number per time step to one. We focus on the capabilities of the optimizer in a relatively big Hilbert space where a density-matrix–based approach would be difficult to implement.

### *10.3.1 Problem overview*

In cQED, a common technique to measure the state of a transmon qubit is to couple it to a readout resonator. This is described by the generalized Jaynes-Cummings model Hamiltonian

$$H = \omega_r a^\dagger a + \omega_q b^\dagger b + \frac{1}{2}\alpha b^\dagger b(b^\dagger b - 1) + g(a^\dagger b + a b^\dagger) + H_\zeta(t), \tag{10.4}$$

where $\omega_r$ and $\omega_q$ are the bare resonator and qubit frequencies, respectively, and $a$ is the lowering operator for photons inside the resonator. $b$ and $b^\dagger$ are the ladder operators for the transmon excitation number, truncated at an appropriate level. There are two jump operators for the total system, $a$ for photon loss with a rate of $\kappa$ and $b$ qubit decay with rate $\gamma = \frac{1}{T_1}$. $H_\zeta(t)$ is the resonator drive term responsible for generating the readout, and takes the form

$$H_\zeta(t) = \zeta(t)(a e^{i\omega_d t} + a^\dagger e^{-i\omega_d t}), \tag{10.5}$$

where $\omega_d$ is the drive frequency.

For the purpose of qubit readout, the circuit parameters are chosen such that the system is in the dispersive regime ($\Delta = |\omega_q - \omega_r| \gg g$) [128, 8]. In that case, the effective frequency of the resonator is AC-Stark shifted by a value that is dependent on the qubit state. Hence,

by driving the resonator at either of the two shifted frequencies, the amplitude-response of the readout tone can distinguish between the 0 and 1 states of the qubit. Alternatively, by driving at the bare resonator frequency, the qubit state can be extracted from information carried in the phase of the readout [8].

### *10.3.2 First optimization target: high readout fidelity*

An appropriate metric for successful readout is the single-shot readout fidelity $\mathcal{F}$ [84]

$$\mathcal{F} = 1 - \frac{p(0|1) + p(1|0)}{2}, \tag{10.6}$$

where $p(a|b)$ is the probability of measuring the qubit in state $a$ given it was prepared in state $b$. There are two main factors that limit the readout fidelity $\mathcal{F}$: noise in the measurement and qubit-decay processes that can make an excited-state trajectory look very similar to a ground-state one. While noisy readout could be mitigated by increasing the measurement time and hence enhancing the ability to average out the noise, this worsens the probability for spurious qubit decay during the measurement, leading to a decrease of the readout fidelity.

Existing readout protocols involve multiplying the readout signal by a filter function, and integrating it over the measurement time. The integration result is then compared against a set threshold in order to identify it with one of the underlying qubit states 0 or 1. Several filters exist to maximize $\mathcal{F}$ including the optimal linear filter [42] which we will utilize here.

One key area of improvement that we pursue here is the choice of the pulse $\zeta(t)$ that maximizes the fidelity. In most experiments, $\zeta(t)$ is taken to have a square-pulse envelope, rendering the time-dependent drive sinusoidal with a constant amplitude that is varied in order to maximize $\mathcal{F}$. Using our optimizer, we open up the possibility of a wider variety of pulse trains including multiple frequency components which may yield higher fidelities while maintaining readout speed.

### *10.3.3 Second optimization target: overcoming dressed dephasing and obtaining fast resonator reset*

Increasing the readout-pulse power can yield higher fidelities as it counteracts experimental noise. However, excessively high powers lead to increased photon occupations throughout the measurement which has several drawbacks.

One drawback pertains to the ability to perform the measurement in a manner that facilitates a fast cavity-reset process afterwards, allowing for new measurements to be started with minimum downtime. There are several protocols for both passive and active reset of the cavity [86, 20]. The time it takes to reset the cavity depends on how many photons are left in the cavity by the end of measurement. Ref. [15] shows that there is a power law relating the speed limit of resonator reset and the number of leftover resonator photons. For example, doubling the number of photons in the resonator requires an increase in the active resonator-reset time of at least 57%. Therefore, not limiting the cavity occupation number can significantly slow down subsequent resonator reset.

In addition, high photon numbers cause another significant problem, namely dressed dephasing [11] which results from higher-order corrections to the dispersive approximation, usually ignored in the regime of small photon numbers. As the number of photons increases, the quantum non-demolition (QND) nature of the readout is jeopardized by these extra dephasing channels.

In line with these insights, we utilize the flexibility of setting targets in our optimizer to search for a readout pulse that maintains high levels of fidelity whilst keeping the resonator occupation as low as possible.

### *10.3.4 Third optimization target: QND measurement*

One crucial element of the readout that needs to be maintained is its quantum non-demolition (QND) behavior. Typical readout measurements within the dispersive regime are expected

to keep the qubit state unchanged for sufficiently low photon occupation of the resonator [10], and hence qualify as QND measurements. The scale where the QND behavior of the readout breaks down is quantified by the critical number of photons $n_{\mathrm{crit}} = \Delta^2/4g^2$. (As $n_{\mathrm{crit}}$ is approached, higher-order terms in the perturbative dispersive approximation become important.)

Since the optimizer is based on the Hamiltonian (10.4), the simulation is not necessarily limited to the dispersive regime. In principle, this allows the optimization to employ high readout power without concerns about the validity of the dispersive approximation. However, large power endangers the QND nature as the dispersive regime breaks down [10]. The previous optimization target should help in that regard as it minimizes the number of photons in the cavity. To further ensure the QND nature of the readout, we add a QND-dedicated optimization target.

### *10.3.5 Cost functions*

We include three cost functions for achieving the three optimization targets mentioned above simultaneously. Starting with the readout-fidelity cost function, we first inspect the process of post-measurement decision making. For every trajectory, the output signal $s(t) = \left\langle (a + a^\dagger) \right\rangle (t)$ is convoluted according to

$$S = \int_0^{T_f} s(t)K(t)dt \tag{10.7}$$

with a filter kernel $K(t)$ that is used to enhance the distinguishability of the readout signals [42]. $T_f$ is the final measurement time. In order to determine the appropriate threshold value of $S$ distinguishing between readout of the qubit 0 and the 1 states, many trajectories are simulated and their corresponding values of $S$ are recorded. The histograms generated by the values of $S$ for both qubit states are fitted to Gaussians and the boundary is chosen

to minimize the overlap between the two Gaussians. Histograms take on Gaussian form because of both the existence of noise in the measurement and the different evolution of each trajectory resulting in a distribution of integrated trajectory signals. Fig. 10.6 shows an example of integrated readout signals.

As Fig. 10.6 suggests, one possible way to enhance readout fidelity is to increase the difference between the means of the two Gaussians to limit their overlap which is the main source of wrong decision making. Therefore, we implement the following fidelity cost function:

$$C_f = -\left(\frac{1}{T_f}\int_0^{T_f}[\bar{s_0}(t) - \bar{s_1}(t)]\ dt\right)^2. \tag{10.8}$$

Here, the bar denotes a trajectory average so that $\bar{s_i}(t)$ is the transmitted signal at time $t$ averaged over trajectories that all start in the initial state $|i\rangle$ with $i \in \{0, 1\}$.

For the second optimization target, the average resonator occupation number at the end of the measurement needs to be minimized to enable fast resonator reset. In addition, penalizing the average photon number during the entire measurement phase will make sure that dressed dephasing is suppressed. This also allows for potential termination of the measurement protocol at times smaller than the preset $T_f$ without accumulating large photon occupation. Hence, the second cost function was implemented in the form

$$C_r = \frac{1}{T_f}\sum_{i=0,1}\int_0^{T_f}\overline{\langle(a^\dagger a)\rangle_i}(t)dt. \tag{10.9}$$

where $i \in \{0, 1\}$ again refers to the initial state of the qubit.

Finally, for the QND optimization target, we add a cost function which rewards overlap between the final trajectory states and the corresponding initial states so that the qubit starting in the ground/excited state remains in the ground/excited state at the end of measurement. As we always start the measurement with the resonator in the ground state, this cost function further helps reduce the cost $C_r$ as it ensures that the cavity returns back to

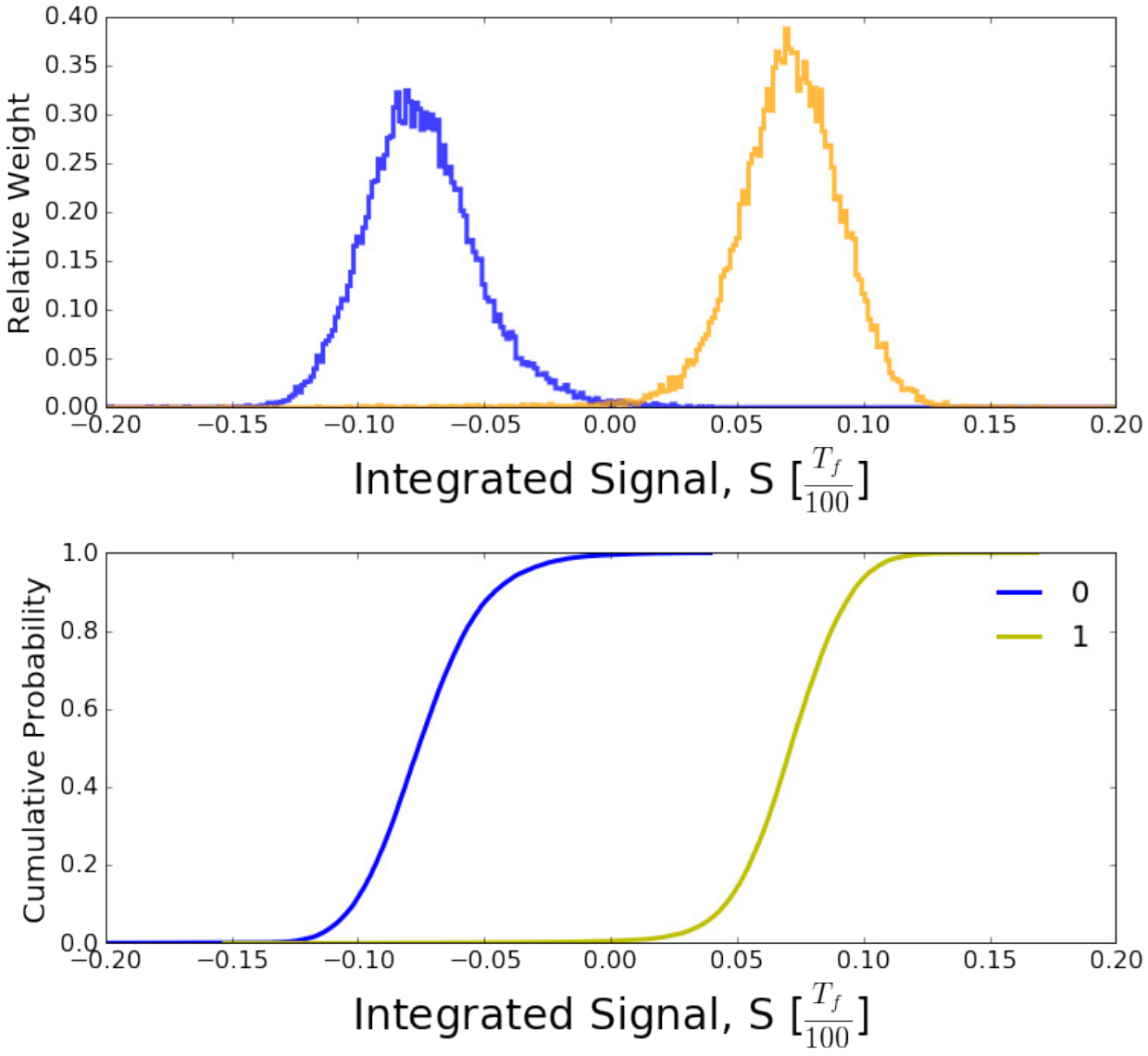


Figure 10.6: (a) Example of integrated readout signals for a qubit starting in the ground or excited state. The overlap between the two Gaussians contributes to the readout infidelity. One way to minimize the overlap between the distributions is by increasing the difference between the two Gaussian means. (b) The corresponding cumulative probabilities of both distributions. The decision threshold that maximizes the fidelity is at the integrated signal value that maximizes the difference between the two cumulative probabilities.

the ground state. This QND cost function is taken to have the form

$$C_q = 1 - \overline{|\langle\psi_f|\psi_i\rangle|^2}, \tag{10.10}$$

where $|\psi_f\rangle$ and $|\psi_i\rangle$ are the final and initial states of each trajectory, respectively. These three cost function are combined with other pulse-shaping constraints, and are assigned different weight factors which are set empirically by trial and error to improve convergence.

### *10.3.6 Implementation*

The problem is divided into two parts: optimization and classification. First, optimization is performed in such a way as to minimize the above-mentioned cost functions. To calculate the resulting readout fidelities, the diffusive trajectories produced by the resulting optimized pulse are mixed with Additive White Gaussian Noise of different powers, and then fed into an optimal-linear-filter classifier. The parameters used for the simulation are $\omega_q/2\pi = 4.6\,\mathrm{GHz}$, $\omega_r/2\pi = \omega_d/2\pi = 5\,\mathrm{GHz}$, $g/2\pi = 50\,\mathrm{MHz}$, $\kappa = 50\,\mathrm{M}s^{-1}$ and $\gamma = 1\,\mathrm{M}s^{-1}$. These parameters correspond to $n_{\mathrm{crit}} = 16$. We included 30 resonator levels and 3 transmon levels in the simulation. The simulated total time of measurement was taken to be $T_f = 100\,\mathrm{ns} = 0.1\,T_1$.

Following ref. [86], we denote pulse amplitudes in dimensionless form $A_n = A/A_{\mathrm{ph}}$. Here, $A$ is the absolute pulse amplitude, and $A_{\mathrm{ph}}$ is the reference amplitude which results in a steady-state resonator occupation of one photon. $n$ refers to the effective number of photons resulting from the used amplitude. We limit the pulse maximum amplitude to be $A_{16} = A_{n_{\mathrm{crit}}}$. The fidelity of the optimized pulse is compared against the fidelity of a constant square pulse of amplitude $A_{n_{\mathrm{crit}}}$.

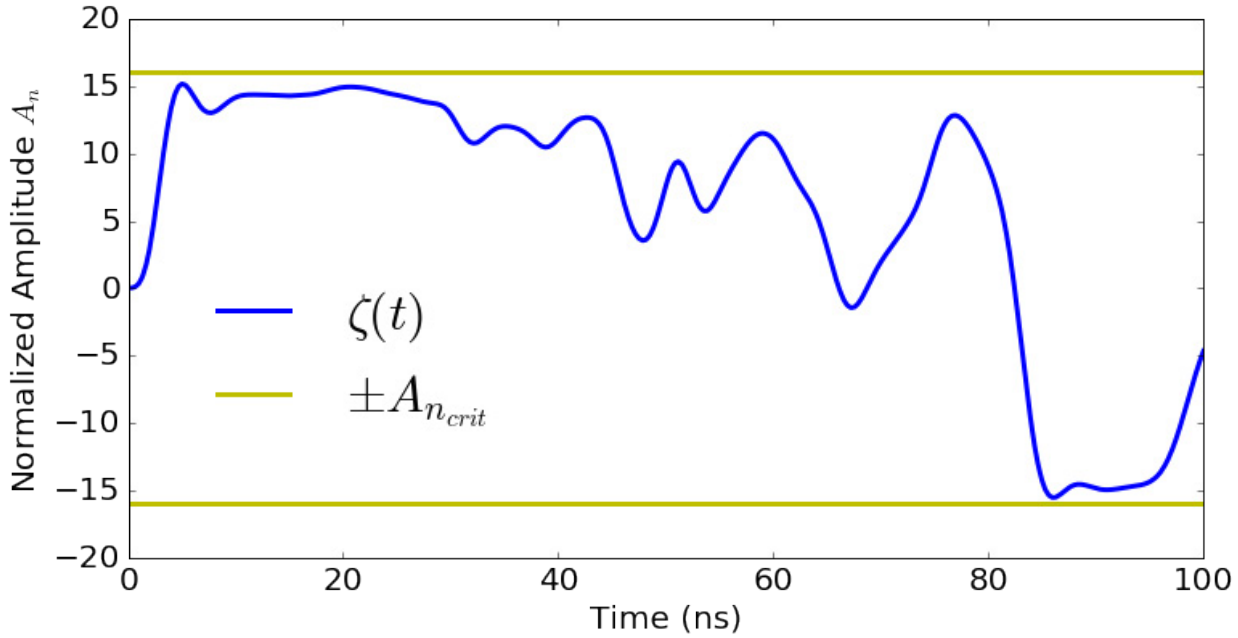


Figure 10.7: Optimized readout pulse within the amplitude limit. The pulse minimizes the weighted mixture of cost functions and uses only 58.3% of the power needed by the constant pulse.

### *10.3.7 Results*

We performed optimization with 8 parallel mini-batches, each of size $m_{\text{tot}} = 30$ using synchronous distributed training. With proper adjustments to the relative weights of different cost functions, the optimizer obtains the solution presented in Fig. 10.7. As ensured by the pulse-shaping cost functions, the pulse is smooth and starts and ends at near zero amplitudes. The resulting resonator occupation is shown in Fig. 10.8.

As evident from Fig. 10.8, the optimized pulse maintains a relatively low photon number during the measurement process, and significantly decreases the photon numbers towards the end of the pulse. The optimized pulse achieves final photon numbers of 0.09 and 0.04 for the qubit starting in the $|0\rangle$ and $|1\rangle$ states; respectively. The constant pulse, by contrast, leads to occupations of around 13 for both states, which is two orders of magnitude higher than the optimized results. Therefore, optimization significantly improves the time needed for resetting the resonator after the measurement with the resonator almost empty already.

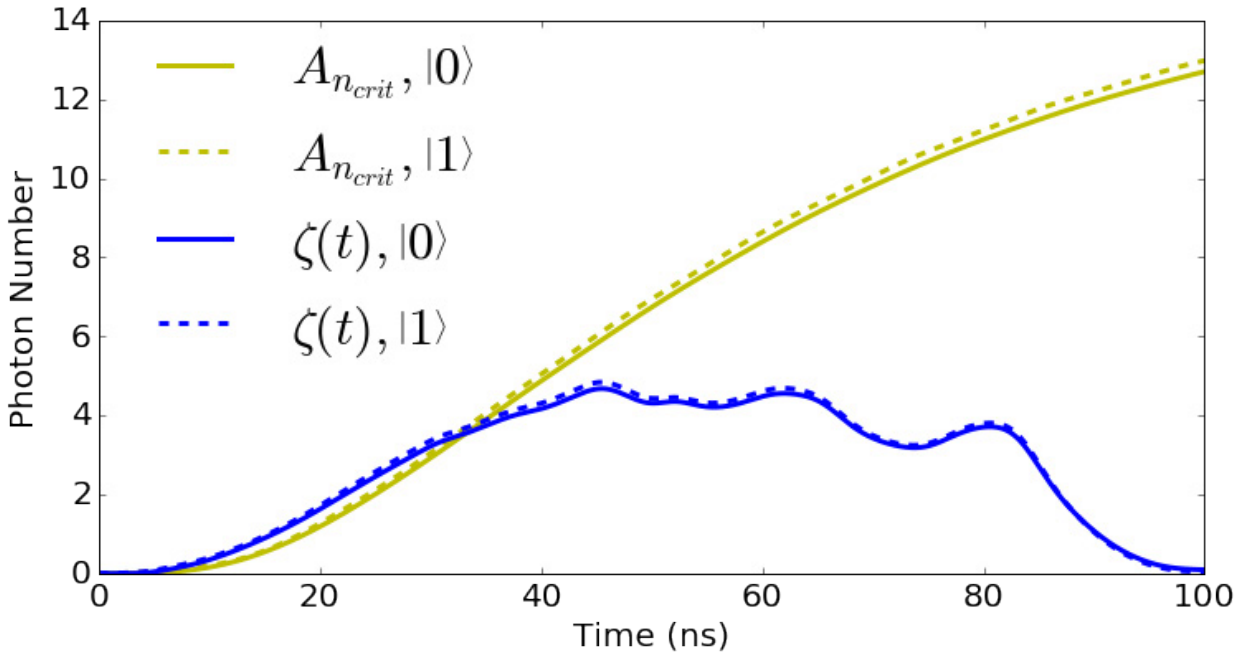


Figure 10.8: Resonator occupation as a function of measurement time for the optimized pulse $\zeta$(t) and the constant pulse of amplitude $A_{n_{\text{crit}}}$, with the qubit starting in the $|0\rangle$ and $|1\rangle$ states.

If the active reset protocol proposed in ref. [15] is used, then the reset is going to be 25 times faster than the non-optimized pulse case if the qubit is measured in the ground state, and 43 times faster if measured in the excited state.

Moreover, to illustrate how optimization affects the QND nature of the readout, the qubit occupation numbers are plotted in Fig. 10.9 for both pulses. The results show that if one were to use a constant-power readout pulse, then the QND behavior would be compromised since the ground state gets excited to an average occupation of 0.14. The optimized pulse, however, manages to bring this occupation down to 0.002, ensuring the ground state measurement process to be QND within 99.8%. As for the excited state, the QND nature is limited by relaxation processes. Due to relaxation with the specified rate $\gamma$, the qubit occupation number would decay to around 0.9 during the readout. However, due to potential qubit-cavity dressing, there are additional decay channels that may lower the final ideal occupation further. For instance, ref. [10] suggests that within the dispersive regime, the

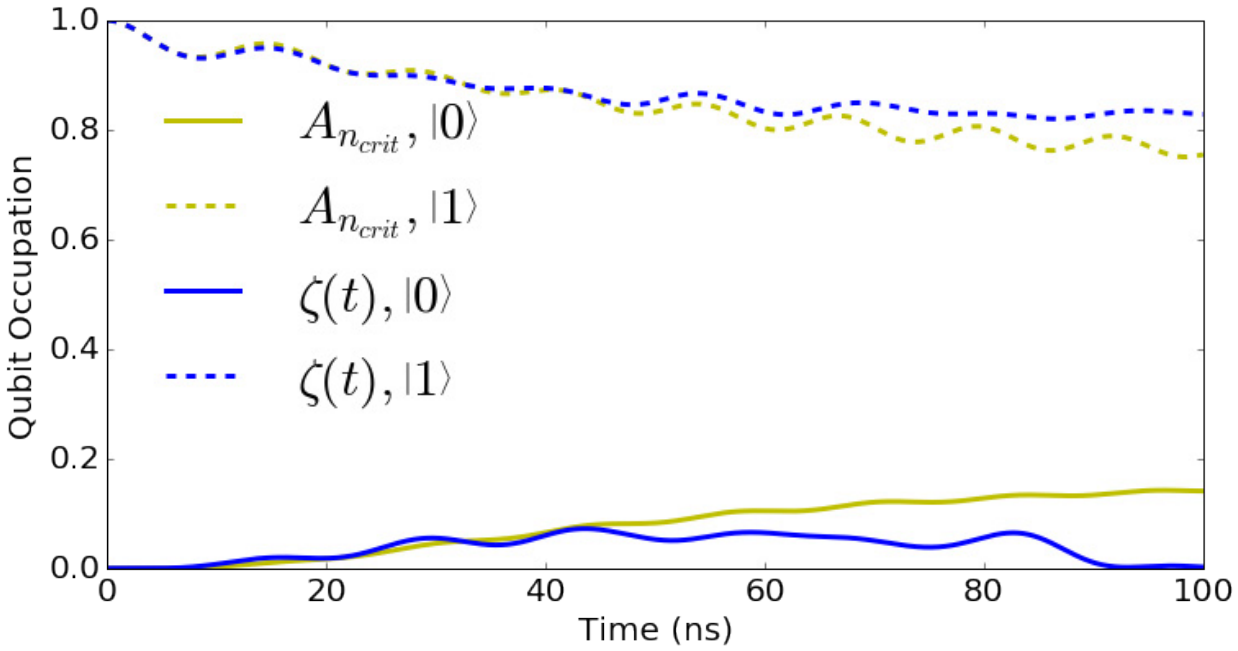


Figure 10.9: The qubit average occupation as a function of measurement time for both the optimized pulse $\zeta$(t) and the constant pulse of amplitude $A_{n_{\text{crit}}}$ with the qubit starting in the $|0\rangle$ and $|1\rangle$ states.

photon occupation of the resonator acts like an extra heat bath for the qubit. Fig. 10.9 shows that the constant pulse yields final occupation of 75.5% while the optimized pulse increases it to 82.8%, thus enhancing the QND character of the measurement protocol.

The QND and low-photon-number constraints ensure that the optimized pulse power is reduced from the maximum allowed power. Lowering the readout power could potentially cause the readout fidelity to decrease because of the presence of noise. To inspect whether readout power reduction negatively impacted the readout fidelities, we calculated the readout fidelities of the two pulses given different values for the noise power. The simulated noise power is again normalized in the same manner as the readout power, and we included normalized noise powers of up to $P_{20}$. The corresponding readout fidelities are presented in Fig. 10.10. The fidelities we obtain from both pulses are actually very close despite using almost half the total power. For most noise powers, the optimized pulse gives better fidelities. Therefore, we see that maintaining the QND nature of the readout and low photon-

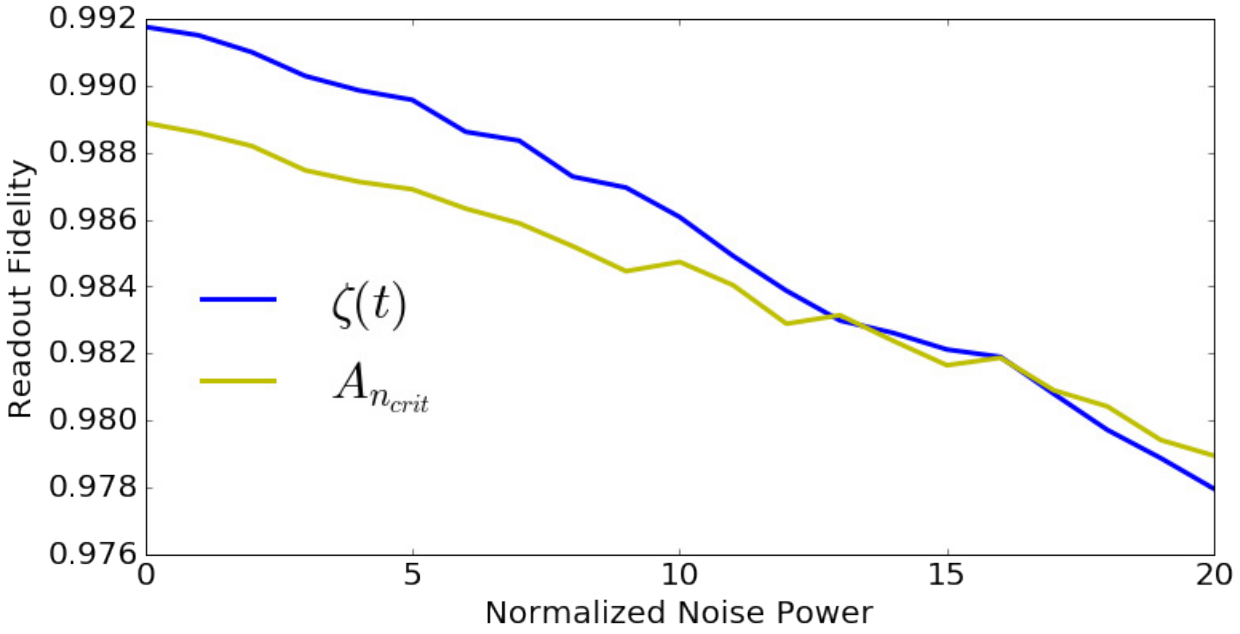


Figure 10.10: Readout fidelity of the optimized pulse compared to the constant, maximum-amplitude pulse.

occupation numbers was achieved in a way that does not harm the readout fidelity.

In summary, our implementation allows for optimizing complex open systems under several constraints without analytically calculating their gradients. In addition, the simulation complexity is significantly reduced. For example, in this application, the Hilbert space dimension is 90. A naive open-system GRAPE implementation would require propagating superoperators of dimension $8,100 \times 8,100$. Using direct integration of the equations of motion, this problem can be bypassed in favor of the time-evolution of density matrices of size $90 \times 90$. Instead, we process batches of 90-component vectors either in series or in parallel and achieve convergence with careful adjustment of convergence parameters. The results suggest that readout could be performed with optimized pulses to control several aspects of the measurement. The example we presented suggests that the readout fidelities are not necessarily hurt by using smaller integrated powers, while significant benefits could be gained by allowing for constantly and smoothly changing amplitude pulses that restore the QND nature of the measurement and also allow for much faster resonator reset.

# CHAPTER 11

# CONCLUSION & OUTLOOK

## 11.1 Conclusion

In conclusion, we have first presented quantum optimal control algorithm for closed systems harnessing two key technologies that enable fast and low-overhead numerical exploration of control signal optimization. The first key technology we used is automatic differentiation. We have demonstrated that automatic differentiation can be leveraged to facilitate effortless inclusion of diverse optimization constraints, needed to obtain realistic control signals tailored for the specific experimental capabilities at hand. Automatic differentiation dramatically lowers the overhead for adding new cost functions, as it renders analytical derivations of gradients unnecessary. For illustration, we have presented concrete examples of optimized unitary gates and state transfer, using cost functions relevant for applications in superconducting circuits. We emphasize that this is but one instance within a much larger class of quantum systems for which optimal control is instrumental, and the methods described here are not limited to the specific examples shown in this thesis.

The second key technology we have incorporated is the implementation of GPU-based numerical computations, which offers a significant speedup relative to conventional CPU-based code. The use of the TensorFlow library [1] hides the low-level details of GPU acceleration, allowing implementation of new cost functions at a high level. The reduction in computational time will generally depend on a number of factors including system type, Hilbert space size, and the specific hardware employed by the user. We observe that runtime speedup by an order of magnitude is not unusual when using a standard desktop PC, enabling the development of sophisticated quantum control without enormous investments into powerful computing equipment. The underlying libraries also have support for high-performance distributed computing systems for larger optimizations. Our software implementation is open

source and can be downloaded at: github.com/SchusterLab/quantum-optimal-control.

Afterwards, we have harnessed the concept of automatic differentiation to build a flexible optimizer for open quantum systems. The optimizer is based on quantum trajectories, leading to significant reduction in the computational overhead compared to approaches based on density matrices. Combinations of improved-sampling techniques, generating mini-samples of trajectories for stochastic gradient descent and parallelization of trajectories are used according to the application to take advantage of the quantum trajectories nature of the optimizer. The optimizer was then utilized for both small and moderately sized quantum systems with quantum jumps being rare or common, and showed quick convergence to results that enhance over existing protocols.

## 11.2 Outlook

### *11.2.1 Closed-system optimal control*

Our implementation for closed-system optimal control can be further enhanced in the future. This includes adding more features and improving the existing ones. Next, we discuss some of the additional features to be worked on in the future.

1. Extra realistic optimization targets could be added to the employed cost functions. For example, control over the final gate time and the bandwidth of the used pulses are potential targets to include in the future.

2. The ability to optimize over user-specified parameters can be included in the future. Currently, the optimizer can only vary the pulse amplitudes at every time-step. There are many practical applications where additional parameters need to be optimized. For example, the frequency of the pulse, qubit parameters like $E_J$ and $E_C$ and coupling between qubits and a resonator. This facilitates solving a wide range of optimization problems using optimal control.

3. Automatizing the tuning of convergence parameters based on the behavior of different costs is a desirable target in the future. Currently, the user sets fixed values for convergence parameters (like learning rate, relative weights for costs and method of gradient ascent) at the beginning of optimization. If convergence is not obtained, the user has to stop the optimization and re-build the computational graph with new convergence values. Automating this process will save time of trial and error performed by the user.

Enhancements to existing features of the optimizer can help users optimize for gates in a faster and more flexible manner. Features that can be enhanced include:

1. Better visualization of what the optimizer is trying to achieve every number of iterations will help users make sense of the resulting pulses.

2. Expanding the matrix exponential using Newton's or Chebyshev polynomials can speed up the time evolution.

3. The deployment of adaptive step size and Runge-Kutta methods for time evolution will enhance the speed and accuracy of the simulation.

In terms of applications, the optimizer can be used in the future to find universa gates of superconducting qubits in the shortest time possible. Speeding up gate times for superconducting qubits to $\sim 1$ ns without having an exceedingly large drive power and maintaining a high fidelity is a challenge we plan to address in future work. In addition, future work on the 0-$\pi$ qubit is planned. Optimizing gates for more realistic parameter regimes and including the $\zeta$ mode in the optimization are challenges that should be addressed in the future.

### *11.2.2 Open-system Optimal Control*

Our open-system optimizer will also benefit from the additions and enhancements mentioned in the previous section. In addition, there are some improvements that are particular to the

open-system optimizer.

1. Allowing multiple jumps per simulation time-step enhances the accuracy of the optimizer and can be implemented thanks to the flexibility of TensorFlow.

2. Including second-order optimization schemes like BFGS helps speeding up convergence. The only supported optimization method currently in the open-system optimizer is ADAM.

3. Implementing different kinds of quantum trajectories in the optimizer could be more suitable to specific optimization problems. For example, diffusive trajectories match the qubit-readout output observed in experiments and could be utilized to further enhance the readout fidelity.

Moreover, the optimal control pulses provide a promising tool for experimenters to use in their own studies, and we hope they can be useful in the future. Therefore, future work includes focused collaboration with experimental groups to implement optimized pulses in the lab.

# REFERENCES


[1] Martín Abadi et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, March 2016.

[2] Mohamed Abdelhafez, David I. Schuster, and Jens Koch. Gradient-based optimal control of open quantum systems using quantum trajectories and automatic differentiation. *Phys. Rev. A*, 99:052327, May 2019.

[3] M. Arioli, B. Codenotti, and C. Fassino. The Padé method for computing the matrix exponential. *Linear Algebra and its Applications*, 240:111–130, June 1996.

[4] Michael Bartholomew-Biggs, Steven Brown, Bruce Christianson, and Laurence Dixon. Automatic differentiation of algorithms. *Journal of Computational and Applied Mathematics*, 124(1-2):171–190, December 2000.

[5] Atilim G. Baydin, Barak A. Pearlmutter, Alexey A. Radul, and Jeffrey M. Siskind. Automatic differentiation in machine learning: a survey, April 2015.

[6] Nathan Bell and Michael Garland. Efficient sparse matrix-vector multiplication on CUDA. NVIDIA Technical Report NVR-2008-004, NVIDIA Corporation, December 2008.

[7] K Bergmann, H Theuer, and BW Shore. Coherent population transfer among quantum states of atoms and molecules. *Reviews of Modern Physics*, 70(3):1003, 1998.

[8] Alexandre Blais, Ren-Shou Huang, Andreas Wallraff, S. M. Girvin, and R. J. Schoelkopf. Cavity quantum electrodynamics for superconducting electrical circuits: An architecture for quantum computation. *Phys. Rev. A*, 69:062320, Jun 2004.

[9] Benjamin Block, Peter Virnau, and Tobias Preis. Multi-GPU accelerated multi-spin Monte Carlo simulations of the 2D Ising model. *Computer Physics Communications*, 181(9):1549–1556, September 2010.

[10] Maxime Boissonneault, Jay M Gambetta, and Alexandre Blais. Dispersive regime of circuit qed: Photon-dependent qubit dephasing and relaxation rates. *Physical Review A*, 79(1):013819, 2009.

[11] Maxime Boissonneault, JM Gambetta, and Alexandre Blais. Nonlinear dispersive regime of cavity qed: The dressed dephasing model. *Physical Review A*, 77(6):060305, 2008.

[12] Troy W. Borneman, Martin D. Hürlimann, and David G. Cory. Application of optimal control to CPMG refocusing pulse design. *Journal of Magnetic Resonance*, 207(2):220–233, December 2010.

[13] A. Borzı‘, J. Salomon, and S. Volkwein. Formulation and numerical solution of finite-level quantum optimal control problems. *Journal of Computational and Applied Mathematics*, 216(1):170–197, June 2008.

[14] Lon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT2010*, page 177186. Physica-Verlag HD, 2010.

[15] Samuel Boutin, Christian Kraglund Andersen, Jayameenakshi Venkatraman, Andrew J Ferris, and Alexandre Blais. Resonator reset in circuit qed by optimal control for large open quantum systems. *Physical Review A*, 96(4):042315, 2017.

[16] Mark D Bowdrey, Daniel KL Oi, Anthony J Short, Konrad Banaszek, and Jonathan A Jones. Fidelity of single qubit maps. *Physics Letters A*, 294(5-6):258–260, 2002.

[17] H. P. Breuer and F. Petruccione. *The theory of open quantum systems*. Oxford University Press, Great Clarendon Street, 2002.

[18] Etienne Brion, Line Hjortshøj Pedersen, and Klaus Mølmer. Adiabatic elimination in a lambda system. *Journal of Physics A: Mathematical and Theoretical*, 40(5):1033, 2007.

[19] Peter Brooks, Alexei Kitaev, and John Preskill. Protected gates for superconducting qubits. *Phys. Rev. A*, 87:052306, May 2013.

[20] Cornelis Christiaan Bultink, MA Rol, TE OBrien, Xiang Fu, BCS Dikken, Christian Dickel, RFL Vermeulen, JC de Sterke, Alessandro Bruno, RN Schouten, et al. Active resonator reset in the nonlinear dispersive regime of circuit qed. *Physical Review Applied*, 6(3):034008, 2016.

[21] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A Limited Memory Algorithm for Bound Constrained Optimization. *SIAM J. Sci. Comput.*, 16(5):1190–1208, September 1995.

[22] Bryan Catanzaro, Narayanan Sundaram, and Kurt Keutzer. Fast Support Vector Machine Training and Classification on Graphics Processors. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 104–111, New York, NY, USA, 2008. ACM.

[23] Qi M. Chen, Re B. Wu, Tian M. Zhang, and Herschel Rabitz. Near-time-optimal control for quantum systems. *Phys. Rev. A*, 92:063415, December 2015.

[24] Yi Chou, Shang-Yu Huang, and Hsi-Sheng Goan. Optimal control of fast and high-fidelity quantum gates with electron and nuclear spins of a nitrogen-vacancy center in diamond. *Physical Review A*, 91(5):052315, May 2015.

[25] M. A. Clark, R. Babich, K. Barros, R. C. Brower, and C. Rebbi. Solving lattice QCD systems of equations using mixed precision solvers on GPUs. *Computer Physics Communications*, 181(9):1517–1528, September 2010.

[26] W. J. Cody, G. Meinardus, and R. S. Varga. Chebyshev rational approximations to ex in [0, +) and applications to heat-conduction problems. *Journal of Approximation Theory*, 2(1):50–65, March 1969.

[27] A. D. Córcoles, Easwar Magesan, Srikanth J. Srinivasan, Andrew W. Cross, M. Steffen, Jay M. Gambetta, and Jerry M. Chow. Demonstration of a quantum error detection code using a square lattice of four superconducting qubits. *Nature Communications*, 6:6979, April 2015.

[28] Xiang Cui, Yifeng Chen, and Hong Mei. Improving Performance of Matrix Multiplication and FFT on GPU. In *Parallel and Distributed Systems (ICPADS), 2009 15th International Conference on*, pages 42–48, Washington, DC, USA, December 2009. IEEE.

[29] Andrew J Daley. Quantum trajectories and open many-body quantum systems. *Advances in Physics*, 63(2):77–149, 2014.

[30] P. de Fouquieres, S. G. Schirmer, S. J. Glaser, and Ilya Kuprov. Second order gradient ascent pulse engineering. *Journal of Magnetic Resonance*, 212(2):412–417, October 2011.

[31] S. Debnath, N. M. Linke, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe. Demonstration of a small programmable quantum computer with atomic qubits. *Nature*, 536(7614):63–66, August 2016.

[32] Joshua M. Dempster, Bo Fu, David G. Ferguson, D. I. Schuster, and Jens Koch. Understanding degenerate ground states of a protected quantum circuit in the presence of disorder. *Phys. Rev. B*, 90:094518, Sep 2014.

[33] P. Ditz and A. Borzi‘. A cascadic monotonic time-discretized algorithm for finite-level quantum control computation. *Computer Physics Communications*, 178(5):393–399, March 2008.

[34] Florianín Dolde et al. High-fidelity spin entanglement using optimal control. *Nature Communications*, 5, February 2014.

[35] N. Earnest, S. Chakram, Y. Lu, N. Irons, R. K. Naik, N. Leung, L. Ocola, D. A. Czaplewski, B. Baker, Jay Lawrence, Jens Koch, and D. I. Schuster. Realization of a $\Lambda$ system with metastable states of a capacitively shunted fluxonium. *Phys. Rev. Lett.*, 120:150504, Apr 2018.

[36] Nathan Don Earnest. *Engineering Light-Matter Interactions with Metastable States in a Heavy Fluxonium*. PhD thesis, The University of Chicago, 2019.

[37] D. J. Egger and F. K. Wilhelm. Adaptive Hybrid Optimal Quantum Control for Imprecisely Characterized Systems. *Phys. Rev. Lett.*, 112:240503+, June 2014.

[38] D. J. Egger and F. K. Wilhelm. Optimized controlled-Z gates for two superconducting qubits coupled through a resonator. *Superconductor Science and Technology*, 27(1):014001+, January 2014.

[39] Reuven Eitan, Michael Mundt, and David J. Tannor. Optimal control with accelerated convergence: Combining the Krotov and quasi-Newton methods. *Phys. Rev. A*, 83:053426, May 2011.

[40] John D. Farnum and David A. Mazziotti. Trigonometric mapping for the electric field strength in molecular optimal control theory. *Chemical Physics Letters*, 416(1-3):142–146, November 2005.

[41] K. Fatahalian, J. Sugerman, and P. Hanrahan. Understanding the Efficiency of GPU Algorithms for Matrix-matrix Multiplication. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, HWWS '04, pages 133–137, New York, NY, USA, 2004. ACM.

[42] Jay Gambetta, WA Braff, A Wallraff, SM Girvin, and RJ Schoelkopf. Protocols for optimal readout of qubits using a continuous quantum nondemolition measurement. *Physical Review A*, 76(1):012325, 2007.

[43] U Gaubatz, P Rudecki, S Schiemann, and K Bergmann. Population transfer between molecular vibrational levels by stimulated raman scattering with partially overlapping laser fields. a new concept and experimental results. *The Journal of Chemical Physics*, 92(9):5363–5376, 1990.

[44] Steffen J. Glaser, Ugo Boscain, Tommaso Calarco, Christiane P. Koch, Walter Köckenberger, Ronnie Kosloff, Ilya Kuprov, Burkhard Luy, Sophie Schirmer, Thomas Schulte-Herbrüggen, Dominique Sugny, and Frank K. Wilhelm. Training Schrödinger's cat: quantum optimal control. *Eur. Phys. J. D*, 69(12):1–24, 2015.

[45] M. H. Goerz. *Optimizing Robust Quantum Gates in Open Quantum Systems*. PhD thesis, 2015.

[46] Michael H. Goerz, Giulia Gualdi, Daniel M. Reich, Christiane P. Koch, Felix Motzoi, K. Birgitta Whaley, J. Vala, Matthias M. Müller, Simone Montangero, and Tommaso Calarco. Optimizing for an arbitrary perfect entangler. II. Application. *Phys. Rev. A*, 91:062307+, June 2015.

[47] Michael H. Goerz and Kurt Jacobs. Efficient optimization of state preparation in quantum networks using quantum trajectories. *Quantum Science and Technology*, 3(4):045005, Jul 2018.

[48] Michael H. Goerz, Felix Motzoi, K. Birgitta Whaley, and Christiane P. Koch. Charting the circuit QED design landscape using optimal control theory, August 2016.

[49] Caroline Gollub, Markus Kowalewski, and Regina de Vivie-Riedle. Monotonic Convergent Optimal Control Theory with Strict Limitations on the Spectrum of Optimized Laser Fields. *Phys. Rev. Lett.*, 101:073002, August 2008.

[50] Goren Gordon, Gershon Kurizki, and Daniel A. Lidar. Optimal dynamical decoherence control of a qubit. *Physical Review Letters*, 101(1):010403, Jul 2008.

[51] Peter Groszkowski, A Di Paolo, A L Grimsmo, A Blais, D I Schuster, A A Houck, and Jens Koch. Coherence properties of the 0- qubit. *New J. of Phys.*, 20(4):043053, Apr 2018.

[52] Lov K Grover. A fast quantum mechanical algorithm for database search. *arXiv preprint quant-ph/9605043*, 1996.

[53] Martin H. Gutknecht. A brief introduction to krylov space methods for solving linear systems. In *Frontiers of Computational Science*, page 5362. Springer Berlin Heidelberg, 2007.

[54] Simon Haykin. *Neural Networks: A Comprehensive Foundation.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 1994.

[55] R. Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural Networks, 1989. IJCNN., International Joint Conference on*, pages 593–605 vol.1. IEEE, June 1989.

[56] Reinier W. Heeres, Philip Reinhold, Nissim Ofek, Luigi Frunzio, Liang Jiang, Michel H. Devoret, and Robert J. Schoelkopf. Implementing a Universal Gate Set on a Logical Qubit Encoded in an Oscillator, August 2016.

[57] H. J. Hogben, M. Krzystyniak, G. T. P. Charnock, P. J. Hore, and Ilya Kuprov. Spinach A software library for simulation of spin dynamics in large spin systems. *Journal of Magnetic Resonance*, 208(2):179–194, February 2011.

[58] Zhibo Hou, Han-Sen Zhong, Ye Tian, Daoyi Dong, Bo Qi, Li Li, Yuanlong Wang, Franco Nori, Guo-Yong Xiang, Chuan-Feng Li, and Guang-Can Guo. Full reconstruction of a 14-qubit state within four hours. *New Journal of Physics*, 18(8):083036, 2016.

[59] Antony Jameson, Wolfgang Schmidt, and Eli Turkel. Numerical solution of the euler equations by finite volume methods using runge kutta time stepping schemes. In *14th fluid and plasma dynamics conference*, page 1259, 1981.

[60] H. Jirari. Optimal control approach to dynamical suppression of decoherence of a qubit. *EPL*, 87(4):40003, Sep 2009.

[61] J. R. Johansson, P. D. Nation, and Franco Nori. QuTiP: An open-source Python framework for the dynamics of open quantum systems. *Computer Physics Communications*, 183(8):1760–1772, August 2012.

[62] J. R. Johansson, P. D. Nation, and Franco Nori. QuTiP 2: A Python framework for the dynamics of open quantum systems. *Computer Physics Communications*, 184(4):1234–1240, April 2013.

[63] J.ín Kelly et al. Optimal Quantum Control Using Randomized Benchmarking. *Phys. Rev. Lett.*, 112:240504, June 2014.

[64] J.ín Kelly et al. State preservation by repetitive error detection in a superconducting quantum circuit. *Nature*, 519(7541):66–69, March 2015.

[65] Nikhil Ketkar. *Stochastic Gradient Descent*, page 113132. Apress, 2017.

[66] Navin Khaneja, Timo Reiss, Cindie Kehlet, Thomas Schulte-Herbrüggen, and Steffen J. Glaser. Optimal control of coupled spin dynamics: design of NMR pulse sequences by gradient ascent algorithms. *Journal of Magnetic Resonance*, 172(2):296–305, February 2005.

[67] Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, July 2015.

[68] Kyryl Kobzar, Thomas E. Skinner, Navin Khaneja, Steffen J. Glaser, and Burkhard Luy. Exploring the limits of broadband excitation and inversion pulses. *Journal of Magnetic Resonance*, 170(2):236–243, October 2004.

[69] Kyryl Kobzar, Thomas E. Skinner, Navin Khaneja, Steffen J. Glaser, and Burkhard Luy. Exploring the limits of broadband excitation and inversion: II. Rf-power optimized pulses. *Journal of Magnetic Resonance*, 194(1):58–66, September 2008.

[70] Jens Koch, V. Manucharyan, M. H. Devoret, and L. I. Glazman. Charging effects in the inductively shunted josephson junction. *Phys. Rev. Lett.*, 103:217004, Nov 2009.

[71] Jens Koch, Terri M. Yu, Jay Gambetta, A. A. Houck, D. I. Schuster, J. Majer, Alexandre Blais, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf. Charge-insensitive qubit design derived from the Cooper pair box. *Phys. Rev. A*, 76(4):042319, October 2007.

[72] Karsten König and Andreas Ostendorf. *Optically induced nanostructures: biomedical and technical applications*. Walter de Gruyter GmbH & Co KG, 2015.

[73] Robert L. Kosut, Matthew D. Grace, and Constantin Brif. Robust control of quantum gates via sequential convex programming. *Phys. Rev. A*, 88:052326, November 2013.

[74] Philip Krantz, Morten Kjaergaard, Fei Yan, Terry P Orlando, Simon Gustavsson, and William D Oliver. A quantum engineer's guide to superconducting qubits. *Applied Physics Reviews*, 6(2):021318, 2019.

[75] Vadim Krotov. *Global methods in optimal control theory*, volume 195. CRC Press, 1995.

[76] Victor W. Lee, Changkyu Kim, Jatin Chhugani, Michael Deisher, Daehyun Kim, Anthony D. Nguyen, Nadathur Satish, Mikhail Smelyanskiy, Srinivas Chennupaty, Per Hammarlund, Ronak Singhal, and Pradeep Dubey. Debunking the 100X GPU vs. CPU Myth: An Evaluation of Throughput Computing on CPU and GPU. *SIGARCH Comput. Archit. News*, 38(3):451–460, June 2010.

[77] Nelson Leung, Mohamed Abdelhafez, Jens Koch, and David Schuster. Speedup for quantum optimal control from automatic differentiation based on graphics processing units. *Physical Review A*, 95(4):042318, Apr 2017.

[78] Per J. Liebermann and Frank K. Wilhelm. Optimal Qubit Control Using Single-Flux Quantum Pulses. *Phys. Rev. Applied*, 6:024022+, August 2016.

[79] Alexey Liniov, Valentin Volokitin, Iosif Meyerov, Mikhail Ivanchenko, and Sergey Denisov. Increasing performance of the quantum trajectory method by grouping trajectories. In *Russian Supercomputing Days*, pages 136–150. Springer, 2017.

[80] Weifeng Liu and Brian Vinter. An Efficient GPU General Sparse Matrix-Matrix Multiplication for Irregular Data. In *2014 IEEE 28th International Parallel and Distributed Processing Symposium*, pages 370–381. IEEE, May 2014.

[81] S. Machnes, U. Sander, S. J. Glaser, P. de Fouquières, A. Gruslys, S. Schirmer, and Thomas Schulte-Herbrüggen. Comparing, optimizing, and benchmarking quantum-control algorithms in a unifying programming framework. *Phys. Rev. A*, 84:022305, August 2011.

[82] Shai Machnes, David J. Tannor, Frank K. Wilhelm, and Elie Assémat. Gradient optimization of analytic controls: the route to high accuracy quantum optimal control, July 2015.

[83] Yvon Maday and Gabriel Turinici. New formulations of monotonically convergent quantum control algorithms. *The Journal of Chemical Physics*, 118(18):8191–8196, May 2003.

[84] Easwar Magesan, Jay M Gambetta, Antonio D Córcoles, and Jerry M Chow. Machine learning for discriminating quantum measurement trajectories and improving readout. *Physical review letters*, 114(20):200501, 2015.

[85] Vladimir E. Manucharyan, Jens Koch, Leonid I. Glazman, and Michel H. Devoret. Fluxonium: Single cooper-pair circuit free of charge offsets. *Science*, 326(5949):113–116, 2009.

[86] Doug T McClure, Hanhee Paik, Lev S Bishop, Matthias Steffen, Jerry M Chow, and Jay M Gambetta. Rapid driven reset of a qubit readout resonator. *Physical Review Applied*, 5(1):011001, 2016.

[87] David C. McKay, Ravi Naik, Philip Reinhold, Lev S. Bishop, and David I. Schuster. High-Contrast Qubit Interactions Using Multimode Cavity QED. *Phys. Rev. Lett.*, 114:080501+, February 2015.

[88] Mazyar Mirrahimi, Zaki Leghtas, Victor V. Albert, Steven Touzard, Robert J. Schoelkopf, Liang Jiang, and Michel H. Devoret. Dynamically protected cat-qubits: a new paradigm for universal quantum computation. *New Journal of Physics*, 16(4):045014+, December 2013.

[89] Cleve Moler and Charles Van Loan. Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later : SIAM Review: Vol. 45, No. 1 (Society for Industrial and Applied Mathematics). *SIAM Review*, 45(1):3–49, August 2006.

[90] Gordon E Moore et al. Cramming more components onto integrated circuits, 1965.

[91] F. Motzoi, J. M. Gambetta, S. T. Merkel, and F. K. Wilhelm. Optimal control methods for rapidly time-varying Hamiltonians. *Phys. Rev. A*, 84(2):022307, August 2011.

[92] V. Nebendahl, H. Häffner, and C. F. Roos. Optimal control of entangling operations for trapped-ion quantum computing. *Phys. Rev. A*, 79(1):012312, January 2009.

[93] M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[94] R. Nigmatullin and S. G. Schirmer. Implementation of fault-tolerant quantum logic gates via optimal control. *New Journal of Physics*, 11(10):105032, October 2009.

[95] Kyoung-Su Oh and Keechul Jung. GPU implementation of neural networks. *Pattern Recognition*, 37(6):1311–1314, June 2004.

[96] Yukiyoshi Ohtsuki, Yoshiaki Teranishi, Peter Saalfrank, Gabriel Turinici, and Herschel Rabitz. Monotonically convergent algorithms for solving quantum optimal control problems described by an integrodifferential equation of motion. *Phys. Rev. A*, 75:033407, March 2007.

[97] Yukiyoshi Ohtsuki, Gabriel Turinici, and Herschel Rabitz. Generalized monotonically convergent algorithms for solving quantum optimal control problems. *The Journal of Chemical Physics*, 120(12):5509–5517, March 2004.

[98] Roberto Olivares-Amaya, Mark A Watson, Richard G Edgar, Leslie Vogt, Yihan Shao, and Alán Aspuru-Guzik. Accelerating correlated quantum chemistry calculations using graphical processing units and a mixed precision matrix multiplication library. *Journal of chemical theory and computation*, 6(1):135–144, 2009.

[99] José P. Palao and Ronnie Kosloff. Optimal control theory for unitary transformations. *Phys. Rev. A*, 68(6):062308+, December 2003.

[100] José P. Palao, Ronnie Kosloff, and Christiane P. Koch. Protecting coherence in optimal control theory: State-dependent constraint approach. *Phys. Rev. A*, 77(6):063412, June 2008.

[101] Agustin Di Paolo, Arne L Grimsmo, Peter Groszkowski, Jens Koch, and Alexandre Blais. Control and coherence time enhancement of the 0– qubit. *New J. Phys.*, 21(4):043002, Apr 2019.

[102] Line Hjortshøj Pedersen, Niels Martin Møller, and Klaus Mølmer. Fidelity of quantum operations. *Physics Letters A*, 367(1-2):47–51, 2007.

[103] Felix Platzer, Florian Mintert, and Andreas Buchleitner. Optimal Dynamical Control of Many-Body Entanglement. *Phys. Rev. Lett.*, 105:020501, July 2010.

[104] Rajat Raina, Anand Madhavan, and Andrew Y. Ng. Large-scale Deep Unsupervised Learning Using Graphics Processors. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 873–880, New York, NY, USA, 2009. ACM.

[105] P. Rebentrost, I. Serban, Thomas Schulte-Herbrüggen, and F. K. Wilhelm. Optimal Control of a Qubit Coupled to a Non-Markovian Environment. *Phys. Rev. Lett.*, 102(9):090401, March 2009.

[106] P. Rebentrost and F. K. Wilhelm. Optimal control of a leaking qubit. *Phys. Rev. B*, 79(6):060507, February 2009.

[107] Daniel M. Reich, Mamadou Ndong, and Christiane P. Koch. Monotonically convergent optimization in quantum control using Krotov's method. *The Journal of Chemical Physics*, 136(10):104103, March 2012.

[108] IBM Research and the IBM QX team. IBM Q Experience Documentation. Accessed 06-30-2019.

[109] I. Serban, J. Werschnik, and E. K. U. Gross. Optimal control of time-dependent targets. *Phys. Rev. A*, 71:053810, May 2005.

[110] Shai Shalev-Shwartz and Ambuj Tewari. Stochastic methods for l1-regularized loss minimization. *Journal of Machine Learning Research*, 12(Jun):1865–1892, 2011.

[111] Toby Sharp. Implementing Decision Trees and Forests on a GPU. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Computer Vision ECCV 2008*, volume 5305 of *Lecture Notes in Computer Science*, chapter 44, pages 595–608. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[112] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[113] Roger B. Sidje. Expokit: A Software Package for Computing Matrix Exponentials. *ACM Trans. Math. Softw.*, 24(1):130–156, March 1998.

[114] Thomas E. Skinner, Timo O. Reiss, Burkhard Luy, Navin Khaneja, and Steffen J. Glaser. Reducing the duration of broadband excitation pulses using optimal control with limited RF amplitude. *Journal of magnetic resonance (San Diego, Calif. : 1997)*, 167(1):68–74, March 2004.

[115] Shlomo E. Sklarz and David J. Tannor. Loading a Bose-Einstein condensate onto an optical lattice: An application of optimal control theory to the nonlinear Schrödinger equation. *Phys. Rev. A*, 66(5):053619, November 2002.

[116] A. Spörl, Thomas Schulte-Herbrüggen, S. J. Glaser, V. Bergholm, M. J. Storcz, J. Ferber, and F. K. Wilhelm. Optimal control of coupled Josephson qubits. *Phys. Rev. A*, 75(1):012302, January 2007.

[117] D. Steinkraus, I. Buck, and P. Y. Simard. Using GPUs for machine learning algorithms. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, ICDAR '05, pages 1115–1120 Vol. 2, Washington, DC, USA, August 2005. IEEE.

[118] Jung-Shen Tai, Kuan-Ting Lin, and Hsi-Sheng Goan. Optimal control of quantum gates in an exactly solvable non-markovian open quantum bit system. *Physical Review A*, 89(6):062310, Jun 2014.

[119] Hillel Tal-Ezer. On restart and error estimation for krylov approximation of w=f(a)v. *SIAM J. Scientific Computing*, 29:2426–2441, 2007.

[120] DavidJ Tannor, Vladimir Kazakov, and Vladimir Orlov. Control of Photochemical Branching: Novel Procedures for Finding Optimal Pulses and Global Upper Bounds. In J. Broeckhove and L. Lathouwers, editors, *Time-Dependent Quantum Molecular Dynamics*, volume 299 of *Nato ASI Series*, pages 347–360. Springer US, 1992.

[121] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[122] Zdeněk Tošner, Thomas Vosegaard, Cindie Kehlet, Navin Khaneja, Steffen J. Glaser, and Niels C. Nielsen. Optimal control in NMR spectroscopy: Numerical implementation in SIMPSON. *Journal of Magnetic Resonance*, 197(2):120–134, April 2009.

[123] Ivan S Ufimtsev and Todd J Martinez. Graphical processing units for quantum chemistry. *Computing in Science & Engineering*, 10(6):26–34, 2008.

[124] Nikolay V Vitanov, Andon A Rangelov, Bruce W Shore, and Klaas Bergmann. Stimulated raman adiabatic passage in physics, chemistry, and beyond. *Reviews of Modern Physics*, 89(1):015006, 2017.

[125] Brian Vlastakis, Gerhard Kirchmair, Zaki Leghtas, Simon E. Nigg, Luigi Frunzio, S. M. Girvin, Mazyar Mirrahimi, M. H. Devoret, and R. J. Schoelkopf. Deterministically Encoding Quantum Information Using 100-Photon Schrödinger Cat States. *Science*, 342(6158):607–610, November 2013.

[126] Leslie Vogt, Roberto Olivares-Amaya, Sean Kermes, Yihan Shao, Carlos Amador-Bedolla, and Alán Aspuru-Guzik. Accelerating resolution-of-the-identity second-order møller- plesset quantum chemistry calculations with graphical processing units. *The Journal of Physical Chemistry A*, 112(10):2049–2057, 2008.

[127] U. Vool, A. Kou, W. C. Smith, N. E. Frattini, K. Serniak, P. Reinhold, I. M. Pop, S. Shankar, L. Frunzio, S. M. Girvin, and M. H. Devoret. Driving forbidden transitions in the fluxonium artificial atom. *Phys. Rev. Applied*, 9:054046, May 2018.

[128] A. Wallraff, D. I. Schuster, A. Blais, L. Frunzio, R.-S. Huang, J. Majer, S. Kumar, S. M. Girvin, and R. Schoelkopf. Strong coupling of a single photon to a superconducting qubit using circuit quantum electrodynamics. *431*, page 162, Sep 2004.

[129] Linnan Wang, Yi Yang, Renqiang Min, and Srimat Chakradhar. Accelerating deep neural network training with inconsistent stochastic gradient descent. *Neural networks: The Official Journal of The International Neural Network Society*, 93:219229, Sep 2017.

[130] Paul Watts, J. Vala, Matthias M. Müller, Tommaso Calarco, K. Birgitta Whaley, Daniel M. Reich, Michael H. Goerz, and Christiane P. Koch. Optimizing for an arbitrary perfect entangler. I. Functionals. *Phys. Rev. A*, 91:062306+, June 2015.

[131] R. E. Wengert. A Simple Automatic Derivative Evaluation Program. *Commun. ACM*, 7(8):463–464, August 1964.

[132] Karl A. Wilkinson, Paul Sherwood, Martyn F. Guest, and Kevin J. Naidoo. Acceleration of the GAMESS-UK electronic structure package on graphical processing units. *Journal of computational chemistry*, 32(10):2313–2318, July 2011.

[133] Howard M. Wiseman and Gerard J. Milburn. *Quantum Measurement and Control*. Cambridge University Press, Nov 2009.

[134] Andy B Yoo, Morris A Jette, and Mark Grondona. Slurm: Simple linux utility for resource management. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 44–60. Springer, 2003.

[135] Brian B Zhou, Alexandre Baksic, Hugo Ribeiro, Christopher G Yale, F Joseph Heremans, Paul C Jerger, Adrian Auer, Guido Burkard, Aashish A Clerk, and David D Awschalom. Accelerated quantum control using superadiabatic dynamics in a solid-state lambda system. *Nature Physics*, 13(4):330, 2017.

[136] Wusheng Zhu, Jair Botina, and Herschel Rabitz. Rapidly convergent iteration methods for quantum optimal control of population. *The Journal of Chemical Physics*, 108(5):1953–1963, February 1998.

[137] Wusheng Zhu and Herschel Rabitz. A rapid monotonically convergent iteration algorithm for quantum optimal control over the expectation value of a positive definite operator. *The Journal of Chemical Physics*, 109(2):385–391, July 1998.

[138] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J. Smola. *Parallelized Stochastic Gradient Descent*, page 25952603. Curran Associates, Inc., 2010.