JONAS VINTHER MASTER'S THESIS

ROBUST QUANTUM OPTIMAL CONTROL VIA THE ADJOINT METHOD

ADVISOR: KARSTEN FLENSBERG
SUBMITTED: MAY 22, 2023

Contents

1 Abstract

Fault-tolerant quantum computing requires fast high-fidelity gates. Additionally, these gates should be insensitive to the dominating fluctuations of the physical system, thereby exhibiting robustness. Finding controls that realize all of this is the topic of Quantum Optimal Control and finding solutions to these problem remain elusive. This thesis concerns itself with finding approximate solutions through the use of sophisticated numerical methods. The presented methods are agnostic to the qubit platform and the results are exemplified by considering a superconducting qubit with multiple control lines. To faithfully represent the quantum system numerically a large number of basis states (~ 300) is required. The dynamics is simulated using adaptive algorithms that contain the error to a specified tolerance. The gradient-based optimization algorithm BFGS is employed and the derivatives of the objective function are efficiently calculated using the continuous adjoint method. Using this prescription both fast and high-fidelity gates are realized in simulation for different gate schemes. Problem formulations that promote robust controls that are insensitive to slowly varying offsets in the model parameters are treated by two different methods, the most efficient of which relies on a derived second order adjoint method. We find a robust X-gate that is insensitive to the specific value of the external flux, which display an infidelity below $3 \cdot 10^{-5}$ over a range of $10^{-3}\Phi_0$ for the external flux value.

2 Acknowledgements

I would like to thank my supervisor Karsten Flensberg for his contribution to this fruitful collaboration at the final part of my studies and for the chance to do research in such an enlightening and motivating environment, both at the Condensed Matter group and at NQCP.

I would also like to thank Svend Krøjer Møller first and foremost for his participation in my introduction to the study of Physics as my TA. Next, for reaching out to me with a thesis topic and thereby laying the foundation for this thesis and last but not least for his integral contributions to this thesis. Without the key insights provided by him and the rewarding discussions I have had with him, this thesis would not have come near this far.

In the context of life as a whole, I would like to thank my family for all the support they provide. It is more than what one could wish for and it enables me to comfortably do what I love.

Finally, thanks to all my friends for the late nights, the good times and for the friendships that may last a life time and thanks to the physics study at the University of Copenhagen as a whole, for all the fond memories that I will carry with me in the future.

3 Introduction

3.1 Outline of the thesis

In section 3, we provide a reminder of what quantum mechanics is and how it can be used for computation. This includes a brief overview of a handful the qubit platforms along with their respective challenges.

In section 4, superconducting qubits are presented more in depth. We explain how superconducting circuits form the Hamiltonian of a quantum system, and what the essential ideas are.This is capped off with a look into a subset of the possible superconducting qubits, namely the flux-qubit and its variations, with a final presentation of the superconducting qubit that is used in the rest of the thesis.

In section 5, we look into how quantum systems can be simulated in relation to the thesis topic. The essential aspects are choosing a basis for the representation and how to integrate the Schrödinger equation, both of which are discussed.

In section 6 we will describe what optimal control is and how it can be applied in the context of quantum dynamics. An overview of different definitions for the loss functions and pulse parametrizations is given and discussed. This is followed by an introduction to robust optimal control.

In section 7 the adjoint method is introduced. The optimizers used in this project are gradient based, and the adjoint method is the method for calculating gradients that is employed in this thesis. This method is extended to calculating second order derivatives and this is used in order to find robust solutions.

In section 8 the results are summarized and discussed, followed by an outlook into how these results can be realized in physical qubits.

3.2 Quantum Mechanics

Classically an object can be described by its position $\vec{r}$ and its momentum $\vec{p}$ and other properties such as mass m, charge and so on. Both $\vec{r}$ and $\vec{p}$ are usually parametrized wrt. the ever forward-propagating variable, time t. The rate of change in the position and the momentum is given by:

$$\dot{\vec{r}} = \frac{1}{m}\vec{p} \quad \text{and} \quad \dot{\vec{p}} = \vec{F} \tag{3.1}$$

where F is the force on the object in accordance with Newton's second law of motion. When considering conservative forces $\vec{F}$ can be written as the gradient to a scalar function, denoted the potential energy function $\vec{F} = -\nabla V(\vec{r})$. These dynamical laws can be rewritten in different ways, one of which is the Hamilton formalism [1]. Withing this framework, the dynamics are governed by the Hamiltonian which is the sum of the kinetic and potential energy:

$$H(\vec{q}, \vec{p}) = \sum_i \frac{p_i^2}{2m} + V(q_i) \tag{3.2}$$

The q_i and p_i are generalized coordinates, but can be regarded as momentum and position in the Newton sense. Then one can define the phase-space as the space spanned by the generalized coordinates. The point is that the object can then be said to exist at a point in phase-space and the dynamics are modelled as a flow given by Hamiltons equations. An example is that of the pendulum, which is visualized in figure 3.1

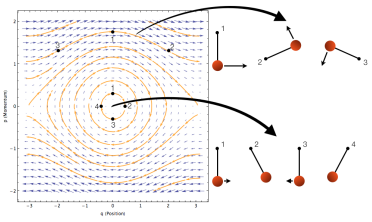


Figure 3.1: The dynamics of the planar pendulum, visualized in phase-space, with q and p being the x and y axis, respectively. The figure is from [2]. The vector field of the flow is given by Hamiltons equations:

$$\dot{q} \propto p \quad \text{and} \quad \dot{p} \propto \sin q$$

To summarize, classical mechanics can quite generally be modelled as a point in phase-space that flows about, where the flow is determined by the kinetic plus the potential energy of the system. Quantum mechanics is not much different and this will now be fleshed out.

In Quantum Mechanics, an object is described by its quantum state, which in Dirac notation looks like $|\psi\rangle$ and are called kets. This state exists in what is called Hilbert space, which is a vector space with an inner product. The dynamics of these states are governed by Schrödingers equation [3]:

$$i\hbar \frac{d}{dt}|\psi\rangle = \hat{H}|\psi\rangle \tag{3.3}$$

$\hat{H}$ is the hermitian operator associated with the energy of the system, ie. the Hamiltonian.[1] Mathematically, once you have a vector space, you also have a dual space which itself is a vector space. In Dirac notation these are referred to as $\langle\psi|$ and are called bras. The inner product is then simply a "bracket" $\langle\psi|\psi\rangle > 0$ (except for the zero vector) and can be regarded as the length of the state $|\psi\rangle$. We can relate these two spaces through the use of the "adjoint": $(|\psi\rangle)^\dagger = \langle\psi|$. Hermiticity is then defined as $\hat{H}^\dagger = \hat{H}$. The fact that $\hat{H}$ is hermitian means that the evolution is unitary. Unitarity means that $\hat{U}^\dagger = \hat{U}^{-1}$, which in turn means that the "length" of the state remains unchanged:

$$\frac{d}{dt}\langle\psi(t)|\psi(t)\rangle = 0 \tag{3.4}$$

Already, one can understand how a quantum mechanical state can be regarded as a vector, in the appropriate Hilbert space, and that the dynamics are reduced to rotations within this space. For a two-dimensional Hilbert space this can be visualized on the Bloch-sphere, which is described in section 3.3 and visualized for a generic example in figure 3.2 and for a specific example in fig. 8.5.

[1] The "ˆ" (hat) denotes that it is an operator.

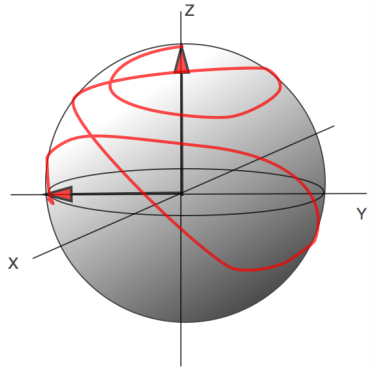


Figure 3.2: Dynamics in Quantum Mechanics are thought of as rotations in Hilbert which for a 2-dimensional Hilbert space can be visualized as a trajectory on the sphere. Figure from [4].

Continuing the comparison of classical and quantum mechanics, in order for $\hat{H}(\vec{q},\vec{p})$ to be an operator, the conjugate variables q_i and p_i themselves need to be promoted to operators. This is done by promoting the poisson bracket[2], where $\{q_i, p_j\} = \delta_{ij}$, to the commutator and multiplying by $i\hbar$:

$$[\hat{q}_i, \hat{p}_j] = \hat{q}_i\hat{p}_j - \hat{p}_j\hat{q}_i = i\hbar\delta_{ij} \tag{3.5}$$

The Schrödinger equation being a linear differential equation also means that any superposition of solutions is also a solution. This has far reaching consequences and in quantum computation it is used to parallelize computations in what is denoted Quantum Parallelism [5]. This concept will be explored further in section 3.4. Another important quantum phenomena with no classical counterpart is entanglement, which will be discussed in section 3.4.3.

[2] $\{f,g\} = \sum_i \frac{\partial f}{\partial q_i}\frac{\partial g}{\partial p_i} - \frac{\partial g}{\partial p_i}\frac{\partial f}{\partial q_i}$

3.3 Two level quantum systems and the Bloch Sphere

Before qubits and quantum computation is introduced, we first need to consider two level quantum systems, since a qubit is essentially any 2-level quantum system. The two levels, which is denoted the computational basis, can be a subspace of a larger Hilbert space with Hamiltonian $\hat{\mathcal{H}}$. Written in terms of the two logical states $|0\rangle$, $|1\rangle$ the effective Hamiltonian can be written in terms of Pauli-matrices, and has the general form:

$$\begin{pmatrix} \langle 0|\hat{\mathcal{H}}|0\rangle & \langle 0|\hat{\mathcal{H}}|1\rangle \\ \langle 1|\hat{\mathcal{H}}|0\rangle & \langle 1|\hat{\mathcal{H}}|1\rangle \end{pmatrix} = E_0(t)\mathbb{1} + \hbar\vec{w}(t)\cdot\vec{\sigma} \tag{3.6}$$

Since any hermitian 2×2 matrix can be written as a linear combination of the pauli-matrices, with real coefficients. The pauli-matrices written in the basis of the σ_z eigenstates, are:

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_x = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{3.7}$$

3.3.1 The Bloch sphere

The subscripts x, y and z are no coincidence since the operator[3]:

$$\hat{n}\cdot\vec{\sigma} \quad \text{with} \quad \hat{n} = \sin\theta\cos\phi\hat{x} + \sin\theta\sin\phi\hat{y} + \cos\theta\hat{z} \tag{3.8}$$

will have eigenstates (up to a choice of gauge or overall complex phase):

$$|+\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle \quad \text{and} \quad |-\rangle = \sin\frac{\theta}{2}|0\rangle - e^{i\phi}\cos\frac{\theta}{2}|1\rangle \tag{3.9}$$

with eigenvalues $+1$ and -1 respectively. It is thus advantageous to represent this state as a point on a unit sphere with the usual spherical coordinates as defined in eq. 3.8

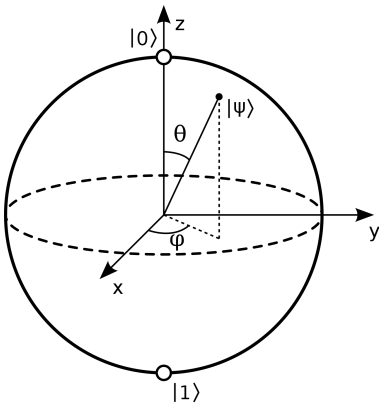


Figure 3.3: Visualization of the Bloch-sphere parametrized in spherical coordinates. Figure from [6].

[3] Here $\vec{\sigma}$ is a cartesian vector of operators and $\hat{n}$ is a cartesian unit vector.

Because now, as stated earlier, the time evolution of the state consist of rotations around on this Bloch sphere. A rotation about the axis $\hat{r}$ by an angle α can be written as:

$$R_{\hat{r}}(\alpha) = \exp(-i\alpha\hat{r}\cdot\vec{\sigma}/2) = \cos\frac{\alpha}{2}\hat{\mathbb{1}} + i\vec{r}\cdot\vec{\sigma}\sin\frac{\alpha}{2} \tag{3.10}$$

3.3.2 Physical examples and Rabi-oscillation

It is instructive to consider a physical system which is already a two level system. This could be a spin-½ particle in a magnetic field. The Hamiltonian is of the form of eq. 3.6 (as it must be). The spin operators are $S_i = \frac{\hbar}{2}\hat{\sigma}_i$ and it is easy to show that they would have the following equations of motion:

$$\dot{\vec{S}} = \vec{\omega}(t) \times \vec{S} \tag{3.11}$$

In accordance with the well known Larmor precession of spin in a magnetic field [7]. That is, the spin precess about the vector $\vec{w}(t)$ which can be directly translated to rotations of the state on the Bloch sphere.

A simple model for generating unitary evolution on the Bloch-sphere is to consider a two-level system with $|1\rangle$ having the highest energy and an operator coupling to an external oscillating pulse. Tt could be spin coupling to a B-field or a dipole-moment coupling to an electric field. Let us assume the latter, and that the moment and the field is parallel, such that:

$$\hat{\mathcal{H}} = \hat{\mathcal{H}}_0 - \vec{d}\cdot\vec{E}(t) = \hat{\mathcal{H}}_0 - \hat{d}E_0\cos(\omega_0 t + \phi) \tag{3.12}$$

If, in the computational basis, the operator $\hat{d}$ assumes the form:

$$\begin{pmatrix} \langle 0|\hat{d}|0\rangle & \langle 0|\hat{d}|1\rangle \\ \langle 1|\hat{d}|0\rangle & \langle 1|\hat{d}|1\rangle \end{pmatrix} = d_x\sigma_x + d_y\sigma_y \tag{3.13}$$

where both d_x and d_y are real and $\langle 1|\hat{d}|0\rangle = d_x + id_y$
The two-level Hamiltonian can be written as:

$$H = \frac{\hbar}{2}\begin{pmatrix} -\omega_q & \tilde{\Omega}^*\left(e^{i(\omega_0 t+\phi)} + e^{-i(\omega_0 t\phi)}\right) \\ C.C. & \omega_q \end{pmatrix} \tag{3.14}$$

Where $\tilde{\Omega} = -E_0(d_x + id_y) = -E_0|d|e^{i\varphi}$
It is then advantageous to go into the rotating frame of the σ_z-basis but with the frequency of the pulse, namely:

$$|\tilde{\psi}\rangle = U^\dagger|\psi\rangle = e^{-i\frac{\omega_0}{2}t\sigma_z}|\psi\rangle \tag{3.15}$$

Such that in this rotating frame, the Schrödinger equation is:

$$i\hbar\frac{d}{dt}|\tilde{\psi}\rangle = \frac{\hbar}{2}\begin{pmatrix} -(\omega_q - \omega_0) & \tilde{\Omega}^*\left(e^{i(2\omega_0 t+\phi)} + e^{-i\phi}\right) \\ C.C. & \omega_q - \omega_0 \end{pmatrix}|\tilde{\psi}\rangle \tag{3.16}$$

With the rotating-wave approximation, ie. discarding the fast-oscillating $e^{\pm i2\omega_0 t}$ terms, the above Hamiltonian becomes of the form:

$$\tilde{H} \approx \frac{\hbar}{2} \begin{pmatrix} -E_0|d|\cos(\phi+\varphi) \\ -E_0|d|\sin(\phi+\varphi) \\ -(\omega_q-\omega_0) \end{pmatrix} \cdot \vec{\sigma} \tag{3.17}$$

So the computational state rotates around this vector, such that any simple unitary evolution is possible, and is given by the applied electric-field. With no detuning ($\omega_0 = \omega_q$) the direction of the vector in the xy-plane is determined by the phase of the matrix element of the coupling operator and the phase of the pulse. In the case of zero detuning it is also easy to show that:

$$|\langle 0|0(t)\rangle|^2 = \cos^2\left(\frac{E_0|d|}{2}t\right) \tag{3.18}$$

Such that for a so called π-pulse, the state $|0(t)\rangle$ will transition to the state $|1\rangle$. The relation between the amplitude of the pulse E_0 and the time of the pulse T is $E_0 = \frac{\pi}{|d|T}$, or rather, that the area under the envelope function of the pulse integrates to $\frac{\pi}{|d|}$ during the time of the pulse. [5] [8]
It is also clear that this methodology breaks-down if there is no overlap of the two states, ie. $|d| \ll 1$, in that case one can make use of intermediate states. [9]

With this brief description of two level quantum systems and unitary evolution thereof, it is now possible to consider how these systems may be used for computation. This will be presented in the following section.

3.4 Quantum Computation

A classical computer stores information in bits which can be in one of two states, 0 or 1. These bits can be manipulated by logical gates. Gates act on one or more bits and produce a single output, an example could be the NOT-gate which simply flips the bit $0/1 \to 1/0$. All classical programmable computation consists of applying gates on bits.

Quantum digital computation[4] is not much different in that it also stores information in what is now called qubits, and apply gates on these qubits in an effort to achieve universal quantum computing power. Qubits are build from any two orthogonal quantum states. These states can be picked out from a zoo of states or they could simply be the two lowest lying states of some Hamiltonian. Different qubit platforms will be discussed in section 3.4.5. Applying a gate on one or more qubits consists of engineering the hamiltonians such that the resulting unitary evolution resembles the wanted logical gate. Consider eq. the pi-pulse with no detuning in section 3.3.2 if the phases are zero the unitary evolution will be:

$$\exp(i\pi\sigma_x/2) = -i\sigma_x \propto \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{3.19}$$

[4] As opposed to analog computing which will not be considered. For an example see [10]

which is equivalent to the NOT-gate in quantum logic, or more colloquially the X-gate. In general, the unitary evolution $U(t)$ will not match the target unitary gate U_{target} exactly. A measure for how well the two match is the gate fidelity, wich can be defined as [11]:

$$F_{\text{gate}} = \frac{1}{d^2}|(U_{\text{target}}^{\dagger}U(T_{\text{gate}}))|^2 \tag{3.20}$$

where d is the dimensionality of the matrix representation and T_{gate} is the gate time. This thesis concerns itself with realizing gates by finding the appropriate control pulses that maximizes the match between the unitary evolution and the desired gate.

In order to motivate the demand for a quantum computer the next section will present and disucss the strengths of a quantum computer compared to a classical computer.

3.4.1 Quantum Parallelism and readout

The differences between classical and quantum computation are in large part due to the superposition principle. The classical memory can only exist in one configuration of zeros and ones, whereas the quantum memory can essentially exist in all configurations at once:

$$\frac{1}{N}\left(|0,0,\ldots,0\rangle + |1,0,\ldots,0\rangle + \ldots + |1,1,\ldots,1\rangle\right) \tag{3.21}$$

or in any other linear combination thereof. Thus an algorithm can in essence be cast on every classical input at once. This is denoted quantum parallelism and it is one of the main reasons for the computational power of a quantum computer [5]. However, trouble arises when one wants to readout the answer for each of these queries to the algorithm, since measuring the quantum memory will collapse it to only a single configuration. It would then seem that next to nothing is gained, if it is necessary perform the computation many times in order to get statistics enough to describe the measurements. This is where the design of ingenious quantum algorithms become important. Because it is possible to encode the solution through constructive interference whilst suppressing the other results through destructive interference [12]. What this exactly means is best described by an example, eg. Grover's algorithm, which will also highlight the strength of quantum parallelism.

3.4.2 Example: Grover's algorithm

Grover's algorithm [13] is a database search algorithm. Consider the problem of finding an entry x_0 in an unsorted list of $N = 2^n$ entries. Classically, half the list need to be examined on average $\mathcal{O}(N/2)$. We will now show that Grover's algorithm only requires $\mathcal{O}(\sqrt{N})$. Prepare the memory in an equal superposition of every number which corresponds to indices of the database:

$$|\psi\rangle = \frac{1}{\sqrt{2^n}}\sum_{x}^{N}|x\rangle \tag{3.22}$$

This can be done using the Hadamard gate:

$$|\psi\rangle = H^{\otimes n}|0\rangle^{\otimes n} \quad \text{where} \quad H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \tag{3.23}$$

and the resulting state is visualized in fig. 3.4 in a manner that makes the following operations easy to understand. There are different ways to do the following, but the simplified way is to say, that the database is represented by the following unitary gate:

$$U_{x_0} = \mathbb{1} - 2|x_0\rangle\langle x_0| \tag{3.24}$$

The only effect U_{x_0} has when applied to the $|\psi\rangle$ state, is that the sign of the $|x_0\rangle$ state, contained in the superposition of $|\psi\rangle$, is flipped (as seen in fig. 3.5). After the U_{x_0} operator has been applied, the Grover operator is applied:[5]

$$U_\psi = 2|\psi\rangle\langle\psi| - \mathbb{1} \tag{3.25}$$

This operator inverts all the coeffecients of $|\psi\rangle$ around the mean value as seen in fig. 3.6

In this manner the spectral weight in $|\psi\rangle$ will accumulate around $|x_0\rangle$ by applying the operator $U_\psi U_{x_0}$ an appropriate amount of times.

$$(U_\psi U_{x_0})^k|\psi\rangle \simeq |x_0\rangle \tag{3.26}$$

The number of times these operators should be applied, denoted k, can be found by the following. Write the state as:

$$|\psi\rangle = \frac{\sqrt{N-1}}{\sqrt{N}}\left(\frac{1}{\sqrt{N-1}}\sum_{x\neq x_0}|x\rangle\right) + \frac{1}{\sqrt{N}}|x_0\rangle \tag{3.27}$$

$$= \cos(\phi)|x \neq x_0\rangle + \sin(\phi)|x_0\rangle \tag{3.28}$$

Where $\sin(\phi) = 1/\sqrt{N}$. This merits a geometrical representation where $|x \neq x_0\rangle$ is the x-axis unit vector and $|x_0\rangle$ is the y-axis unit vector in a 2-dimensional cartesian coordinate system. The state $|\psi\rangle$ starts out pointing almost parallel to the x-axis but at an angle ϕ and we want it to end up pointing along the y-axis. Applying U_{x_0} reflects the state in the x-axis, and it can then be shown that applying U_ψ reflects the state about the original state $|\psi\rangle$. So applying $U_\psi U_{x_0}$ rotates the state by angle 2ϕ and applying it k times yields:

$$(U_\psi U_{x_0})^k|\psi\rangle = \cos((2k+1)\phi)|x \neq x_0\rangle + \sin((2k+1)\phi)|x_0\rangle \tag{3.29}$$

For large N the starting angle is $\phi \simeq 1/\sqrt{N}$ and the appropriate amount of rotations is then approximately $k \simeq \frac{\pi}{4}\sqrt{N}$ such that only $\mathcal{O}(\sqrt{N})$ "look ups" in the database is necessary. [12] [5]

However, if the state does not perfectly align with $|x_0\rangle$ there is a finite chance that the algorithm returns the wrong answer. Therefore, the classical algorithm should also be allowed to trade-off accuracy with speed which is done in probabilistic algorithms. Often the quantum advantage disappears once the quantum and classical algorithms are put on equal footing [15], especially for the contemporary noisy-intermediate scale quantum computers [16]. This only means that

[5] Constructing this gate may seem implausible since it contains the state $|\psi\rangle$, but it can be build out of $\mathcal{O}(\log N)$ elementary gates [5].

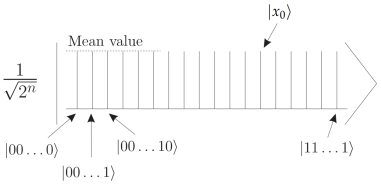


Figure 3.4: Initialization of the memory (this and subsequent figures are from [14])

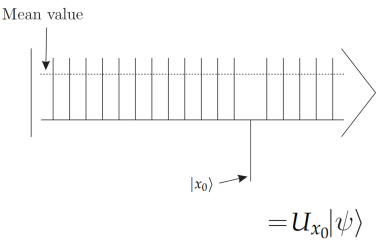


Figure 3.5: Flip the sign of the coefficient of $|x_0\rangle$ in $|\psi\rangle$

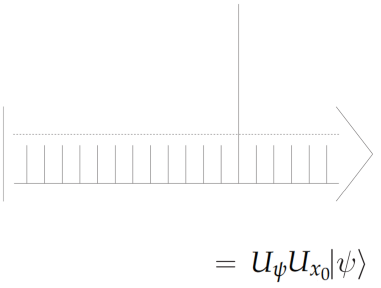


Figure 3.6: Inversion about the mean.

finding a problem that exhibit the possibility of a quantum advantage remains non-trivial [17], while incentivizing the need for fault-tolerant quantum computation and analog quantum simulators [18].

Nevertheless, other applications of quantum computers may be mentioned, such as the algorithm due to Shor [19], which factorizes primes in polynomial-time, rendering many contemporary encryption schemes broken.

Of greater importance to the physicist and the life-sciences dealing with the molecular level, is the possibility of simulating Quantum systems on a quantum computer, which, even for modestly sized systems is infeasible on classical computers.[6]

[6] Of course under certain conditions, approximations can be made and simulation of quantum systems on supercomputers remain an integral part of drug design in industry [20].

The problem of simulating quantum systems arises due to the sheer size of the Hilbert spaces. In many respects the Hilbert space is infinite but can be confined to a relevant finite subspace. However, still the Hilbert space grows exponentially in the number of degrees of freedom, since the resulting Hilbert space is a product of the individual Hilbert spaces. This means, that even just a quantum system containing 100 two-level DoF would require a classical memory able to contain a billion times the world's total current digital data. [7] Whereas, with a quantum computer, this would simply require 100 logical qubits[8].

[7] The collective amount of digital data is on the order of Zettabytes (10^{21}) [21] whereas 100 two-level systems amount to a Hilbert space with dimension $2^{100} = 10^{100/\log_2 10} \simeq 10^{30}$

[8] As stated in [22]: "Logical qubit: a redundantly encoded qubit in which quantum errors can be identified and corrected without corrupting the encoded qubit."

However, it remains difficult to manipulate quantum systems to the level necessary for realizing a universal quantum computer. This is due to the other novel phenomena in quantum mechanics, namely entanglement. Entanglement describes the fact that two seemingly individual quantum systems can only be precisely described, if they are entangled, by a combination of both systems. For an entangled state, the measurements in one system is correlated with measurements in the other. The information about the state exists, loosely spoken, between the two systems, such that only having access to one of the systems yields you next to no information about the state, depending on the degree of entanglement.

This leads to the paradoxical nature of creating qubits, in which one wants to isolate the quantum system from the environment in order to reduce it's entanglement to the environment and limit the resulting decoherence of the state. But at the same time the system needs to be coupled to the outside world in order to manipulate the qubit and make use of it. How coupling between the qubit and the environment leads to decoherence of the qubit, is described more precisely in the following section.

3.4.3 Density matrices

A quantum mechanical state that exists in a superposition of eigenstates wrt. some operator, will still yield different results for the corresponding observable, when prepared in the same initial state before each measurement. This means that measurement results for a single quantum system is already inherently probabilistic, and it is not immediately clear how to talk about the measurement results of

an ensemble of quantum systems.

The useful formalism for considering ensembles of quantum states is through the use of density matrices. For density matrices ρ, the trace is always equal to 1 and if the density matrix is diagonal, the entries along the diagonal can be interpreted as the percentage of the ensemble existing in the corresponding basis state.

So called pure states are when the whole ensemble consists of the same state and in these cases there is no big difference between using density matrices or state-vectors. However, for mixed states, density matrices provide a superior description. An example of a density matrix for a 2-state system could be an ensemble consisting of $0 < p < 1$ in the $|0\rangle$ state and therefore $1 - p$ in the $|1\rangle$ state, then:

$$\rho = p|0\rangle\langle 0| + (1-p)|1\rangle\langle 1| \doteq \begin{pmatrix} p & 0 \\ 0 & 1-p \end{pmatrix} \tag{3.30}$$

In general a 2×2 density matrix can be described by a 3 dimensional vector $\vec{p}$ with length $|\vec{p}| \leq 1$

$$\rho = \frac{1}{2}\left(\mathbb{1} + \vec{p} \cdot \vec{\sigma}\right) \tag{3.31}$$

That is to say, $\vec{p}$ can also be visualized together with the Bloch-sphere, where pure states still exist at the surface, but now mixed states exist in the bulk of the sphere, with the maximally mixed state existing at the origin with $|\vec{p}| = 0$

Density matrices also work well for describing entangled states and the decoherence it might entail. Consider eg. the entangled state:

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle_{\text{qubit}} \otimes |1\rangle_{\text{environment}} + |1\rangle_{\text{qubit}} \otimes |0\rangle_{\text{environment}}) \tag{3.32}$$

It's density matrix is found by:[9]

[9] In the $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$ basis.

$$\rho = |\Psi^+\rangle\langle\Psi^+| \doteq \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & ½ & ½ & 0 \\ 0 & ½ & ½ & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{3.33}$$

It is then possible to reduce this representation to only the subsystem that we can control. This is done in a sensible way, by tracing out the environment. More concretely, we can reverse the product Hilbert space into just the qubit Hilbert space, by doing a partial trace wrt. the environment. [10]

$$\rho_{\text{qubit}} = \text{Tr}_{\text{environment}}\rho \doteq \begin{pmatrix} ½ & 0 \\ 0 & ½ \end{pmatrix} \tag{3.34}$$

We see that by disregarding the correlation between the two systems, because we only have control of one, we end up with a complete mix of 50% $|0\rangle$ states and 50% $|1\rangle$ states. This is to showcase the fact that entanglement with the environment is what causes decoherence. The reason for tracing out the environment is because we do not control it or atleast the part that would be described as being stochastic, therefore one will also say noise is what decoheres the qubit.

[10]

$$\begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix} \to \begin{pmatrix} a+f & c+h \\ i+n & k+p \end{pmatrix}$$

Example of how to trace out the environment in eq. 3.33 using the partial trace.

3.4.4 Noise and decoherence

The dynamics of a pure state is simply derived from the Schrödinger equation, and is referred to as the Liouville-von Neumann equation:

$$\dot{\rho}_{\text{pure}} = -\frac{i}{\hbar}[H, \rho_{\text{pure}}] \tag{3.35}$$

However, when tracing out the environment new terms are introduced to the equations of motion and this is what accounts for decoherence of the state. The equation of motion for such an open quantum system, under suitable conditions such as markovianity[11], is called the Lindblad master equation:

$$\dot{\rho} = -\frac{i}{\hbar}[H, \rho] + \sum_i \left(L_i \rho L_i^\dagger - \frac{1}{2}\left(L_i^\dagger L_i \rho + \rho L_i^\dagger L_i \right) \right) \tag{3.36}$$

where L_i are so called jump operators [23]. In the Bloch-Redfield picture of system noise, the decoherence can be thought of as the result of two processes, relaxation and dephasing. Relaxation is decay from the excited to the ground state and is due to so called transverse noise that couples as σ_x and σ_y in the Hamiltonian of the qubit. Pure dephasing comes from fluctuations in the qubit frequency and is due to longitudinal noise that couples as σ_z in the Hamiltonian. In section 5.3 decoherence will be modeled, through the use of the following jump operators:

$$\text{Relaxation:} \qquad L_1 = \sqrt{\Gamma_1}|0\rangle\langle 1| \tag{3.37}$$

$$\text{Pure dephasing:} \qquad L_\varphi = \sqrt{2\Gamma_\varphi}|1\rangle\langle 1| \tag{3.38}$$

For a derivation of jump operators from a microscopic model, see [24]. An example of a density matrix that describes the decoherence of a qubit in the Bloch-Redfield model, is the following ρ_{BR}. If the system start out in a pure state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ the ρ_{BR} density matrix is written as:

$$\rho_{\text{BR}}(t) = \begin{pmatrix} 1 + \left(|\alpha|^2 - 1\right) e^{-\Gamma_1 t} & \alpha\beta^* e^{i(\omega_q - \omega_0)t} e^{-\left(\frac{\Gamma_1}{2} + \Gamma_\varphi\right)t} \\ \alpha^*\beta e^{-i(\omega_q - \omega_0)t} e^{-\left(\frac{\Gamma_1}{2} + \Gamma_\varphi\right)t} & |\beta|^2 e^{-\Gamma_1 t} \end{pmatrix} \tag{3.39}$$

Where $\hbar\omega_q = E_1 - E_0$ is the qubit frequency and ω_0 is due to the rotating frame as in 3.15. For sufficiently large times the system has decayed to the $|0\rangle$ state since excitations are ignored when working at sufficiently low temperatures due to the suppressing Boltzmann factor for the excitation rate:[12] $\Gamma_{\text{excitation}} = \exp(-\hbar\omega_q / k_B T)\Gamma_1$
The coherence time T_2 is defined from the transverse decay function:

$$T_2 = \frac{1}{\frac{\Gamma_1}{2} + \Gamma_\varphi} \tag{3.40}$$

and is used as the metric of the lifetime of the qubit. T_2 can be measured from Hahn echo experiments which refocuses slow noise whereas, if a Ramsey experiment is used, the coherence time is quoted as T_2^* which has no refocus such that $T_2^* \leq T_2$ [26]. It should be noted

[11] That is to say the probabilities of something happening is not correlated with the past.

[12] However, in experiment, the populace of the first excited state does not follow Boltzmann statistics at low enough temperature, where a measured residual population of around 1% is found [23][25]

that the transverse decay function may not only be of exponential form. It is dependent of the noise spectra defined later in sec. 5.3, such that for eg. $1/f$ noise, the transverse decay function assumes a gaussian form [27]:

$$e^{-\frac{\Gamma_1}{2}t}e^{-(\frac{t}{T_{\varphi,1/f}})^2} \tag{3.41}$$

Nevertheless T_2 and T_2^* are found from fitting exponential decay functions in the Hahn echo and Ramsey experiment respectively [27].

Relaxation is an irreversible error due to exchange of energy with the environment, but in principle, pure dephasing is a coherent error. Therefore, there exists procedures to combat the longitudinal noise by sequentially applying pi-pulses, which is known as a CPMG pulse sequence [28][29]. In general, combating pure dephasing is denoted dynamical error suppression.

There can be multiple noise sources which would have their own error rates Γ_1 and Γ_φ and with possibly different noise spectra. Diagnosing and classifying the different error contributions is key for developing good qubits since this can inform the material considerations and qubit design when developing qubits. [27] [30] [31]

Having established some of the essential theory around qubits, an overview of the different qubit platforms will be provided in the following.

3.4.5 Brief overview of qubit platforms

The following is a brief discussion[13] of de Leon et. al [30] in which they acknowledge 5 qubit platforms. A summary is given in figure 3.7.

[13] Only gate based quantum computers are considered. An example of a measurement based quantum computer are those based on photonics [32]

Platform	Where the quantum information is stored	Known sources of noise	
		Bulk materials	Surfaces and interfaces
Superconducting qubits	Energy eigenstates of Josephson junction–based electronic resonant circuits	• Substrate dielectric loss • Excess quasiparticles in superconducting metal cause dissipation and dephasing	• Uncontrolled oxides and contaminants host two-level systems, causing dissipation and dephasing • Surface spins and charges cause flux noise and charge noise, respectively
Gate-defined quantum dots	Spin states of electrons or holes confined in electrostatic potential	• High mobility required for individual dot formation • Nuclear spins limit T_2	• Charge traps and magnetic impurities at the dielectric interfaces • Interface inhomogeneity: Variation in valley splitting and spin-orbit coupling
Color centers	Electronic orbital and spin states	• Paramagnetic impurities and nuclear spins limit T_2 • Extended defects lead to strain and limit T_2*	Dangling bonds and electron traps at the surface affect T_2 and optical coherence for shallow NV centers
Ion traps	Electronic transitions within individual atomic ions	(Not a significant noise source)	Electric-field noise heats ion motion
Majorana zero modes	Non-Abelian topological phase of Majorana zero modes	Defect density in nanowires	Semiconductor-superconductor nanowire interface that creates a proximity hard gap

Figure 3.7: This table is taken from [30]. A collection of the gate fidelities and coherence times realized in experiment, that is quoted in the source, can be found in the appendix 10.1

Superconducting qubits are the focus of this thesis even though the methods, as will be apparent, extend to all qubit platforms. Mesoscopic[14] superconducting circuits give rise to quantum systems where the Hamiltonian is defined from the circuit elements, as shown in sec. 4.1. The wavefunctions of the energy states are superpositions of the flux in the system and thus the current in the system, or in the conjugate picture, a superposition of the number of cooper-pairs in the superconductors. The computational basis is usually the two lowest lying energy states of the system. In order to stay in the computational basis the energy difference to higher levels need to be different. This necessitates so called non-linear circuit elements that can provide an anharmonicity to the energy spectrum. This is usually done through Josephson junctions that add a cosine term to the hamiltonian as opposed to the quadratic term for conventional inductors. Gates are implemented by sending microwave pulses through a capacitive coupling. Superconducting qubits are one of the most mature platforms with up to 65 fully programmable qubits on one chip [33] and the implementation of a distance 5 surface code[15] [34].

[14] Mesoscopic refer to the intermediate scale between microscopic (nanometers) and macriscopic (millimeters).

[15] Surface codes refer to the paradigm of encoding a logical qubit in a 2d configuration of physical qubits, for which error correction can be made.

Quantum dots are confined quantum mechanical degrees of freedom. For gate-defined quantum dots the two-dimensional electron gas of a semiconductor can be confined to potential wells defined by the voltage in a series of electrodes. Electrons or holes will then occupy the discrete energy levels that exist in the potential well. The quantum mechanical degrees of freedom for this type of qubit is generally either spin-type or charge-type. For charge-type the computational states are the positional states of an electron in a double quantum dot. However, due to strong coupling with environmental noise these exhibit very short coherence times of $\sim 2ns$ [35]. For spin-types the information is stored in the spin states of electrons which can be separated by Zeeman-splitting. This in turn means the gate-times are generally longer since spin flip is usually slow due to the weak coupling [35]. However, effort have been made to make more sophisticated qubit schemes by mixing the different degrees of freedom.

Color centers refer to high purity crystals that host a single or few-atom impurities, whose states can be interacted with optically. The computational state typically consists of electron orbitals and spin eigenstates of the impurity. The most widely studied system is the nitrogen vacancy center in diamond, due to its high coherence time of several milliseconds [36] [37]. Gates can be applied through microwave pulses allowing gate times similar to superconducting qubits.

Trapped ions are atomic ions held in place by dynamical electric fields. The computational basis consists of the electronic states within each ion. Single qubit gates can thus be applied through lasers whereas two-qubit gates involve mediating the interaction through the motion of the ions. [38] [39] The conventional trapping geometry

consists of four rods placed in 3D, trapping the ions between them. However, 2D trapping geometries have been developed and would allow for better scaling capabilities. [40]

Topological qubits are build from quasi-particles that have yet to be experimentally verified unambiguously. However, the theoretical predictions for a quantum device comprised of such a system are quite promising. The quasi-particle are so called anyons that obey non-abelian statistic that is neither bosonic nor fermionic. If done correctly, the computational basis would be topologically protected from local perturbations which would mean extremely high coherence times. Likewise, the gate operations are implemented through the use of the non-abelian statistics in what is called "braiding", and is not victim to the same gate-infidelity sources as the other platforms, hopefully leading to perfect fidelities that would enable fault-tolerant quantum computation. A candidate for such an anyon is the Majorana zero mode in topological superconductors. [41] [42] However, even after extensive search no Majorana zero mode have conclusively been observed [43][44], and topological quantum computing remain a vision for now.

3.4.6 Measuring the qubit

To finally give an idea of how the qubit state is measured, an example of readout within superconducting qubits is given. In fact, this example is also related to how ion trap qubits can be measured. The method is called dispersive readout and it relies on weakly-coupling a resonator to the qubit. [27] That is, in the dispersive limit, where the qubit frequency ω_q and resonator frequency ω_r is far from each other compared to the coupling strength g, the qubit-resonator hamiltonian can be written to second order in $g/|\omega_q - \omega_r|$ and in the limit of only a few photons in the resonator, it becomes:

$$\mathcal{H}_{\text{disp}} = \left(\omega_r + \frac{g^2}{\Delta}\sigma_z\right)\left(a^\dagger a + \frac{1}{2}\right) + \frac{\omega_q + g^2/\Delta}{2}\sigma_z \tag{3.42}$$

where $\Delta = |\omega_q - \omega_r|$ is the qubit-resonator detuning. So the resonator frequency is shifted by a so-called dispersive shift that is dependent on the qubit state.[16] Measuring the qubit then simply constitutes measuring at which of the two frequencies $\omega_r \pm g^2/\Delta$ the resonator is resonant. This is the simplified version. There exists better ways to do this, eg. information is kept in the phase of the resonator probing signal, which can be mapped to a quantum trajectory of the continuous measurement of the qubit state. [27] [45]

[16] Ie. $\omega_r \rightarrow \omega_r + \frac{g^2}{\Delta}\sigma_z$ where σ_z is the z-pauli matrix acting on the computational basis of the qubit.

4 Superconducting Quantum Bits

The superconducting qubit platform is one of the more mature quantum information processing platforms [30]. It utilizes mesoscopic superconducting circuits as hosts for quantum states. The existence of a quantum system at this intermediate scale as opposed to the microscopic scale is due to the bosonic nature of the cooper pairs which make up almost all of the electronic states in the superconducting phase.[1] Similar macroscopic quantum behaviour is observed in superfluids and bosonic encoded qubits, which encode information in the electromagnetic modes of a 3D microwave cavity [22]. The hamiltonian that governs the dynamics of the resulting quantum system of the superconducting circuit is defined by the circuit elements. These can be engineered and tuned to create different energy spectra and matrix elements, thus these systems, as many other systems, are referred to as artificial atoms.

[1] For Bose-Einstein statistics a condensate phase appears below some critical temperature T_C with a fractional occupation of $1 - (\frac{T}{T_C})^{3/2}$ [46]

4.1 Circuit quantum electrodynamics

4.1.1 Quantization of electronic circuits

Following [47] and [48] the general procedure for quantizing electronic circuits will now be introduced.

An electric circuit is schematically defined by branches with different circuit elements which are joined together at nodes in a closed form as seen in figure 4.1 (b). A branch is characterized by the current running through it $I_b(t)$ and the voltage across it $V_b(t)$ (see fig. 4.1 (a).

Circuit elements largely consists of dispersive and dissipative elements. Dissipative elements such as resistance will not be accounted for here, which can be justified by only considering the mesoscopic devices consisting of superconducting materials at sufficiently low temperatures. However, dissipative elements can be included and affect the fluctutations of the quantum degrees of freedom, through the fluctuation-dissipation theorem [47].

Purely dispersive elements store energy in the electric and magnetic fields. With the following definition of branch fluxes and branch

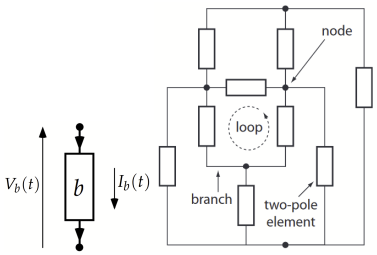


Figure 4.1: (a) Example of a branch with a circuit element b displaying the sign convention of the voltage and current associated with the branch. (b) Example of a generic circuit. Figures from [47]

charges:

$$\Phi_b(t) = \int_{-\infty}^{t} V_b(t')dt' \tag{4.1}$$

$$Q_b(t) = \int_{-\infty}^{t} I_b(t')dt' \tag{4.2}$$

one can talk about capacitive and inductive elements. A capacitive (inductive) element is a dispersive element for which the voltage $V_b(t)$ (current $I_b(t)$) can be written solely in terms of $Q_b(t)$ ($\Phi_b(t)$). The energy associated to the two types of linear dispersive elements are then:[2]

$$\varepsilon_C(t) = \int_{-\infty}^{t} V(t')I(t')dt' = \frac{1}{2C}(Q(t) - Q_{\text{offset}})^2 \tag{4.3}$$

$$\varepsilon_L(t) = \frac{1}{2L}(\Phi(t) - \Phi_{\text{offset}})^2 \tag{4.4}$$

The offsets are included for generality and arise from the fact that the circuit elements are integrated in a circuit. The offsets are continuous variables and may be due to applied external electromagnetic fields or simply from coupling to environmental degrees of freedom in which case they are considered noise terms.

If our goal is to find a Hamiltonian for the relevant degrees of freedom, which we can then quantize, we first have to find the Lagrangian and then Legendre transform it into the Hamiltonian. In order to choose phase-space variables for our Lagrangian, we recognize that if we assume we only consider electronic circuits with linear capacitive elements $V = Q/C$, we can rewrite eq. 4.3, through the use of eq. 4.1 as:

$$\varepsilon_C(t) = \frac{C}{2}\dot{\Phi}^2(t) \tag{4.5}$$

such that the flux can be regarded as the position variable with the linear capacitive elements providing the kinetic terms.

However, when writing up the Lagrangian one must remember the constraints set by Kirchoff's laws. There exists procedures for writing up the Lagrangian while satisfying Kirchoff's laws, which in the case of linear capacitive elements, is denoted the method of nodes [23].

However, for the circuits in this work, it's enough to simply figure out how to write the last variable in terms of the others, while satisfying Kirchoff's laws. The law relevant for the circuits considered in this thesis is the so called fluxoid quantization condition:

$$\sum_b \Phi_b + \Phi_{\text{ext}} = \Phi_0 k \tag{4.6}$$

where $\Phi_0 = h/2e$ is the flux quantum and k is an integer. That is, the sum of the branch fluxes (Φ_b) plus the external flux (Φ_{ext}) that is enclosed by the loop of the branches, must together equal an integer number of flux quantum.

After writing up the Lagrangian the Legendre transformation is done by first defining the conjugate momentum:

$$q_b = \frac{d\mathcal{L}}{d\dot{\Phi}_b} \tag{4.7}$$

[2] For examples of the diagrammatic notation used for the linear capacitive and inductive elements, see figure 4.2

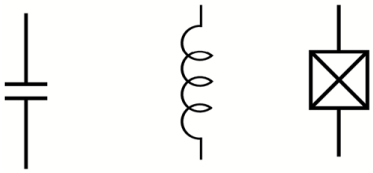


Figure 4.2: Examples of dispersive circuit elements encountered in this thesis. From left to right is the capacitor, inductor and the Josephson junction which is first introduced in section 4.2

with Poisson bracket value:

$$\{\Phi_n, q_m\} = \sum_i \frac{\partial \Phi_n}{\partial \Phi_i}\frac{\partial q_m}{\partial q_i} - \frac{\partial \Phi_n}{\partial q_i}\frac{\partial q_m}{\partial \Phi_i} = \delta_{nm} \tag{4.8}$$

which, once the conjugate variable pair is promoted to operators, is changed to the commutation relation.

$$[\hat{\Phi}_n, \hat{q}_m] = i\hbar\delta_{nm} \tag{4.9}$$

The Hamiltonian is found by:

$$\mathcal{H} = \sum_i q_i\dot{\Phi}_i - \mathcal{L} \tag{4.10}$$

which is then quantized via eq. 4.9
As usual it can be advantageous to work with dimensionless variables, such that:

$$\hat{\phi} \equiv \frac{2\pi}{h/2e}\hat{\Phi} = \frac{2\pi}{\Phi_0}\hat{\Phi} \tag{4.11}$$

$$\hat{n} \equiv \frac{\hat{q}}{2e} \tag{4.12}$$

$$[\hat{\phi}, \hat{n}] = i \tag{4.13}$$

where e is the electron charge, such that n is the number of cooper pairs and the flux is in terms of the reduced flux quantum $\Phi_0/2\pi$. Generally $\hat{\Phi}$ play the role of position and $\hat{q}$ the role of momentum.

The above introduction describe a general approach to the quantization of electronic circuits. A simple example will now be considered.

4.1.2 The LC-circuit: a Quantum Harmonic Oscillator

A simple dispersive circuit is that of the LC-circuit which contain a linear capacitance and a linear inductance (see fig. 4.3 (a)).
The Lagrangian for the LC-circuit is:

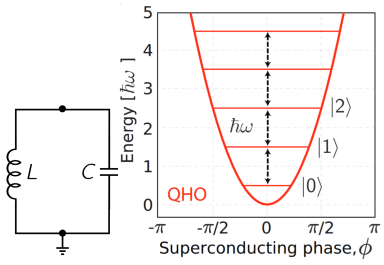
$$\mathcal{L} = \varepsilon_C - \varepsilon_L = \frac{C}{2}\dot{\Phi}^2 - \frac{\Phi^2}{2L} \tag{4.14}$$

which in a straightforward manner yields the quantized Hamiltonian:

$$\mathcal{H} = \frac{\hat{q}^2}{2C} + \frac{\hat{\Phi}^2}{2L} \tag{4.15}$$

This is reminiscent of the Quantum Harmonic Oscillator with frequency $\omega = 1/\sqrt{LC}$ and mass C. It is a recurrent theme that the capacitance is related to the inertia of the quantum mechanical system. It is a well-known result that the QHO exhibit an energy spectra with equal level spacing, as visualized in figure 4.3 (b). This has the consequence that is it impossible to have a well-defined computational basis, because any attempt at making transitions within this basis would also drive any other transition. This is best quantified by the anharmonicity $\alpha_\omega = \omega_{12} - \omega_{01} = (E_2 - E_1 - (E_1 - E_0))/\hbar$ which in the case of the LC-circuit is zero.

Therefore, it is of interest to consider a non-linear dispersive element which can induce different spacings between energy levels and thus a finite anharmonicity. To this end, the Josephson junction element will now be considered.



(a) (b)

Figure 4.3: (a) Schematic of an LC-circuit from [48]. (b) The corresponding quantum mechanical description, in this case a Quantum Harmonic Oscillator. Figure from [27]

4.2 The Josephson junction and the Transmon

4.2.1 The Josephson junction

A Josephson junction circuit element usually consists of a superconducting - insulator - superconducting interface. A difference in the phases of the superconducting states is expressed as a tunneling of cooper pairs across the insulating interface. A key property of a Josephson junction is it's Current-Phase-Relation (CPR) which can take many forms depending on the physics of the junction [49]. For this thesis, only junctions in the so-called ballistic limit are considered [50]. This is when the coherence length is much larger than the junction length. In this case the relation between energy and phase takes the form:

$$\varepsilon_J = -\Delta \sum_i \sqrt{1 - T_i \sin^2(\phi/2)} \tag{4.16}$$

where Δ is the superconducting gap and T_i is the transmission coefficient of the i-th conduction channel. For insulators the transmissions are small $T_i \ll 1$ and the energy can be expanded [51]:

$$\varepsilon_J \simeq -\Delta \sum_i \left(1 - \frac{T_i}{2} \sin^2(\phi/2)\right) \tag{4.17}$$

$$\simeq -\Delta \sum_i \frac{T_i}{4} \cos(\phi) + \Delta \sum_i \frac{T_i}{4} - \Delta \sum_i 1 \tag{4.18}$$

The last two terms only provide a shift in the zero-point energy so with the identification of the Josephson energy:

$$E_J = \frac{\Delta}{4} \sum_i T_i \tag{4.19}$$

The non-linear dispersive circuit element used in this thesis is:

$$\varepsilon_J = -E_J \cos\left(2\pi \frac{\Phi}{\Phi_0}\right) \tag{4.20}$$

We will now consider how this circuit element may help introduce a non-zero anharmonicity.

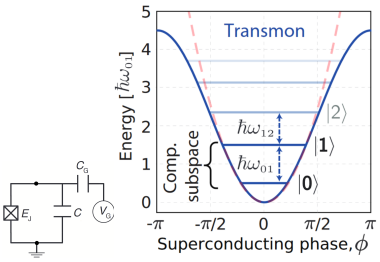
4.2.2 The Transmon

The simplest anharmonic circuit consists of a linear capacitor and a Josephson junction. This is visualized schematically in figure 4.4 (a) and the top node is referred to as a cooper pair island since it consists of an isolated superconducting material. The quantization of this circuit is rather straight forward and results in the Hamiltonian:

$$\mathcal{H} = \frac{(\hat{q} - q_{\text{offset}})^2}{2C} - E_J \cos\left(2\pi \frac{\hat{\Phi}}{\Phi_0}\right) \tag{4.21}$$

$$= 4E_C(\hat{n} - n_{\text{offset}})^2 - E_J \cos(\hat{\phi}) \tag{4.22}$$

with $E_C = \frac{e^2}{2C}$. The charge offset n_{offset} is a continuous variable and can be thought of as consisting of two contributions $n_{\text{offset}} =$



(a) (b)

Figure 4.4: (a) Schematic of a cooper pair box with a tuneable charge offset n_{offset} [50]. (b) Energy spectrum of the now anharmonic system [27]

$n_G + \delta n$. A gate voltage can be applied by a capactive coupling to the superconducting island, resulting in the tuneable $n_G = -\frac{C_G V_G}{2e}$ contribution [50]. The δn comes from capacitive coupling to the environment and should be regarded as a noise term.

The charge offset will affect the value of the energy levels as can be seen in figure 4.5. The sensitivity of the energies on n_{offset} is also clearly dependent on E_J/E_C. A larger ratio is interpretable as a larger mass of the system which coincides with less sensitivity to charge noise. This is important because as mentioned in section 3.4.4, uncontrolled changes in the qubit frequency ω_{01} causes dephasing. Therefore a large E_J/E_C is desirable. However, the anharmonicity also decreases with the lowering of E_C, namely $\alpha_\omega \simeq -E_C$ [50], which would minimize the chances for successful qubit operations. Therefore there exists a trade-off between charge sensitivity and anharmonicity, but as it turns out both can be accommodated if $20 \leq E_J/E_C \ll 5 \cdot 10^4$ These considerations provide the distinction between the first iterations of charge qubits $E_J/E_C \leq 1$ and the improvement, the Transmon qubit $E_J/E_C \sim 100$ [52] which has become the workhorse of the superconducting qubit platform scene.

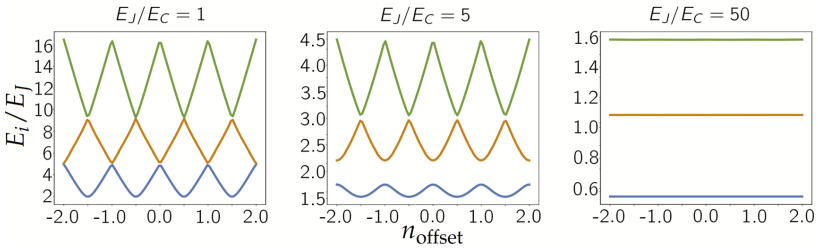


Figure 4.5: The energy of the three lowest lying states as a function of the charge n_{offset} for different ratios of E_J/E_C. A high E_J/E_C make the qubit insensitive to charge noise. Figure from [48]

However, considerations towards protection against relaxation can also be made. As will be stated more precisely in section 5.3, the T_1 time is estimated by Fermi's Golden rule and is thus loosely related to the overlap between the first and excited state, which for the Transmon is large, since the two states live in the same potential "valley". A way to decrease this overlap would be to let these states live in two distinct valleys, which is explored in the next section in the form of the flux qubit.

4.3 The flux-qubit and its variations.

There exists many variations of qubits based on superconducting circuits. They all rely on the Josephson circuit element to introduce anharmonicity to have a well defined computational subspace. There are thus two energy scales whos ratio help classify the different qubits, namely the charging energy E_C which may be due to a capacitor or the Josephson junctions capacitance, and the Josephson energy E_J. E_C is inversely proportional to the "mass" of the system, such that $E_J/E_C \gg 1$ usually means the system is insensitive to charge noise as seen in section 4.2.2. The ratio E_J/E_C distinguishes between

three types of superconducting qubits [53]. The charge qubit[3] with $E_J/E_C < 1$, the flux qubit with $E_J/E_C \sim 100$ and the phase qubit with $E_J/E_C \sim 10^6$. Furthermore, charge and phase qubits generally exhibit single-well potentials, while flux qubits exhibit double well potentials. In order to realize the double well potential in a flux qubit with a single Josephson junction a large inductance is needed. This could necessitate a large loop size and unwanted sensitivity to flux noise [53]. Instead, if the circuit consists of three Josephson junctions, two with Josephson energy E_J and one with αE_J, the loop size can be reduced while a double well potential is realized for $\alpha > 0.5$. [54] [55]

The persistent current qubit [56] (PCQ) is one such system, with $\alpha = 0.8$. The resulting potential is two-dimensional $V(\phi_1, \phi_2)$ with the double well being explicit along the $\phi = (\phi_1 - \phi_2)/2$ mode, as seen in fig. 4.6.

It can be hard to realize the large E_J/E_C, necessary for charge insensitivity, using Josephson junction capacitances alone since these also produce larger sensitivity to flux noise [57].

The capactively shunted flux qubit [57] (CSFQ) circumvents this by using small Josephson junctions and adding a large capacitor in parallel with the αE_J Josephson junction, which is results in a heavy ϕ mode and a light $\theta = (\phi_1 + \phi_2)/2$ mode, as seen by the delocalization in the θ direction in fig. 4.7. However, the CSFQ is operated at $\alpha = 0.4$ and is thus more reminiscent of the transmon but with larger anharmonicity due to the approximately quartic potential.

4.3.1 The double-shunted flux qubit

The double shunted flux qubit (DSFQ) [58] is based on ideas from both the PCQ and CSFQ. Both the ϕ and θ modes are made heavy by having a large capacitance in parallel with the two E_J junctions, as seen in fig. 4.8 (a). Furthermore, the α parameter is assumed tuneable, which can be realized as described in section 4.3.2. Such that the DSFQ can interpolate between the potentials of the PCQ and CSFQ.

The idea behind the DSFQ is that the states can live in a protected regime of $\alpha = 1$, where the $|0\rangle$ and the $|1\rangle$ are localized away from each other, realizing long T_1 times, and once gates are to be executed, they can be brought together by lowering the potential barrier separating them by decreasing α. Following the procedure of

[3] The transmon, however, is an example of a charge qubit with large E_J/E_C

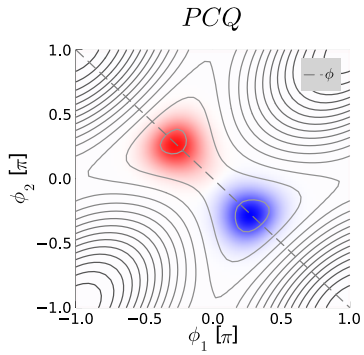


Figure 4.6: The contours show the potential landscape of the PCQ. The wavefunction of the ground state is colored in red while the first excited state is colored in blue. For this visualization $E_J/E_C = 100$

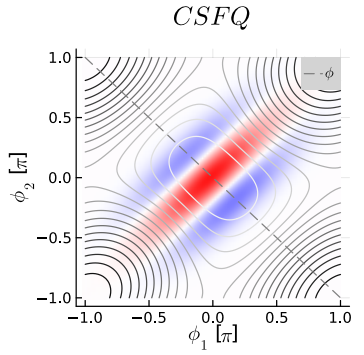
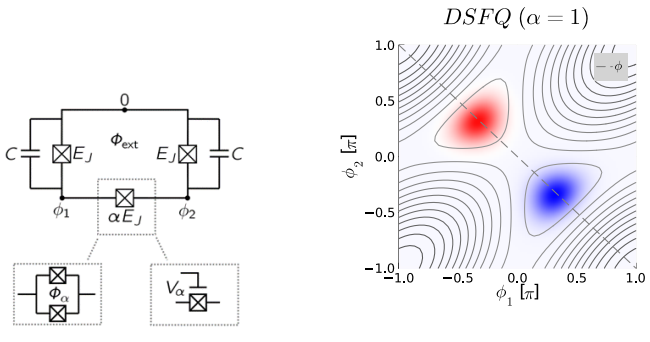


Figure 4.7: Compared to the PCQ, large Josephson junctions have been discarded in favour of a shunting capacitor on the αE_J junction, which results in a light θ mode. Additionally $\alpha = 0.4$



(a) (b)

Figure 4.8: (a) Schematic of the DSFQ [58]. The tuneability of α is discussed in section 4.3.2, but here it is drawn as either a SQUID or a SSmS junction. (b) The potential landscape of the DSFQ. It is reminiscent of the PCQ but now the heavy modes are due to shunting capacitors instead of large Josephson capacitances.

4.1 with the assumption of small Josephson capacitances, yields the Hamiltonian:[4,5]

$$\mathcal{H} = 4E_C(\hat{n}_1^2 + \hat{n}_2^2) - E_J\left(\cos\hat{\phi}_1 + \cos\hat{\phi}_2 + \alpha\cos(\hat{\phi}_1 - \hat{\phi}_2 + \phi_{\text{ext}})\right) \tag{4.23}$$

The model values used throughout the thesis is a Jospehson energy of $E_J = 2\pi\hbar \cdot 10\text{GHz}$ with $E_J/E_C = 100$. The external flux $\phi_{\text{ext}} = 2\pi\frac{\Phi_{\text{ext}}}{\Phi_0}$ affect the asymmetry between the two valleys[6], where an external flux of half a flux quantum is at the symmetry point. This means the computational basis is degenerate and will shield the qubit from dephasing, to first order. However, this is an artificial sweet spot since the insensitivity is only realized at this specific point. Therefore, the artificial sweet spot is avoided by choosing $\phi_{\text{ext}} = 0.995\pi$. For a discussion of the coherence times of the DSFQ see section 5.3.

The control line for gate pulses consists of a small capacitive coupling to the superconducting islands. Throughout this thesis, only a single drive Hamiltonian of

$$H_d = \frac{2eC_d}{C + C_d}V_d(t)\hat{n}_1 \equiv u(t)\hat{n}_1 \tag{4.24}$$

was considered, in addition to the $\alpha(t) \in [0.5, 1]$ control line. But nothing prevents the addition of an extra drive line coupling to $\hat{n}_2$, however, for simplicity, this was not investigated. The effect of the drive Hamiltonian on the system can be regarded as a change of the origin in momentum space, since:

$$4E_C\hat{n}_1^2 + u(t)\hat{n}_1 = 4E_C\left(\hat{n}_1 + \frac{u(t)}{8E_C}\right)^2 - \frac{u^2(t)}{16E_C^2} \tag{4.25}$$

which results in shifts of the origin on the order of ~ 0.2. This is to say that, even when undergoing dynamics, the state remain in a part of Hilbert space captured by the eigenstates of the $\hat{n}_i$ operators with eigenvalues ranging from ~ -10 to ~ 10.[7]

The wavefunctions of the lowest lying energy states of eq. 4.23 are visualized in figure 4.9. The value of α have a large impact on the energy eigenstates and the tuneability of this parameter is a key aspect of the DSFQ. Potential ways of tuning the Josephson energy will now be discussed.

4.3.2 Tuneable Josephson junction elements.

One way to achieve a tuneable E_J is through a circuit nicknamed the "DC SQUID" [47]. It consists of a loop of two Josephson junctions, tuned by an external flux Φ_{SQUID} that threads the loop. The energy of such a circuit element (while discarding the linear inductance of the loop) is:

$$\varepsilon_{\text{SQUID}} = -E_{J_1}\cos\phi_1 - E_{J_2}\cos\phi_2 \tag{4.26}$$

which can be rewritten as:

$$\varepsilon_{\text{SQUID}} = -E_{J_\Sigma}\sqrt{\cos^2\left(\frac{\Phi_{\text{SQUID}}}{2\Phi_0}\right) + d^2\sin^2\left(\frac{\Phi_{\text{SQUID}}}{2\Phi_0}\right)}\cos\phi \tag{4.27}$$

[4] The last term can be though of as $-\alpha E_J\cos\phi_\alpha$, where ϕ_α is found from the fluxoid quantization condition: $\sum_b \phi_b + \phi_{\text{ext}} = 2\pi k = \Phi_1 + \Phi_\alpha - \Phi_2 + \phi_{\text{ext}}$. It is important that the signs adhere to the convention set by the circuit diagram.

[5] For a derivation, see [48]

[6] Think of it as a seesaw, where the bottom of the potential valleys exist at each of their end of the seesaw. The qubit frequency is then largely determined by the height difference between the two ends of the seesaw.

[7] Where these numbers come from, will be apparent in the next chapter and when pulse values are encountered in fig. 8.1

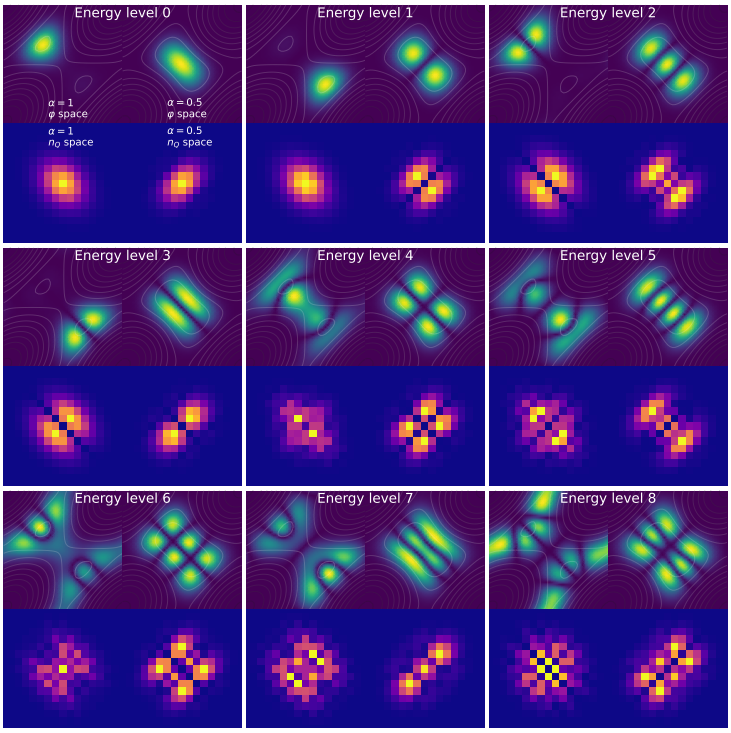


Figure 4.9: Visualization of the amplitude of the wavefunctions of the nine lowest lying energy states of the DSFQ eq. 4.23. They are shown both in the representation of the $\hat{\phi}$ eigenstates and the $\hat{n}$ eigenstates, denoted the ϕ an n_Q space respectively. Also shown is the effect of the α tuneable parameter on the potential and the resulting eigenstates. The domain of the ϕ space is $[-\pi, \pi]$ while for the n_Q space this is been restricted to $\{-8, -7, \ldots, 7, 8\}$ which still captures all the detail of the states. See section 5.1 for a discussion on the choice of basis.

Where $E_{J_\Sigma} = E_{J_1} + E_{J_2}$, $d = \frac{E_{J_1} - E_{J_2}}{E_{J_\Sigma}}$ and $\phi = (\phi_1 + \phi_2)/2 - \tan^{-1}\left(d \tan \frac{\Phi_{\text{SQUID}}}{2\Phi_0}\right)$ is the reparametrized mode of the junction. This device behaves as a single Josephson junction but with a tuneable Josephson energy [59].

Note that using this approach for a tuneable E_J in eq. 4.23 would result in $\Phi_{\text{ext}} \to \Phi_{\text{ext}} - \tan^{-1}\left(d \tan \frac{\Phi_{\text{SQUID}}}{2\Phi_0}\right)$. So when changing α by changing Φ_{SQUID} one should also take care to adjust Φ_{ext} if it is desired to stay at a fixed flux-bias.

Another approach is a Superconductor Semiconductor Superconductor (SSmS) Junction. For Josephson junctions with insulating layers between the superconductors, eq. 4.20 describes well the energy-phase relation of the junction, due to the occurence of only low-transmission channels. For SSmS we may assume this is also the case, except now, the transmission coefficients T_i of eq. 4.19 are tuneable through a voltage applied to the junction [60]. The derivation of eq. 4.20 led to an expression $\varepsilon_J \simeq -E_J \cos\phi + E_J$ such that the shift in the zero-point energy may vary. However, even any time-dependent shift in energy scale may be gauged away since the gauge $|\psi\rangle \to e^{i\varphi(t)}|\psi\rangle$ brings a $-\dot{\varphi}$ term to the Hamiltonian which may cancel this change.

An SSmS junction therefore allows for gate tuneable T_i which leads to a tuneable junction:

$$\varepsilon_J = -\frac{\Delta}{4} \sum_i T_i \cos\phi \tag{4.28}$$

This concludes the presentation of the theoretical considerations behind the quantum system considered throughout the rest of the thesis. In the following chapter the practical aspects of representing and simulating a quantum system numerically, will be discussed.

5 Simulating Quantum Systems

5.1 Choice of Basis

The math of quantum mechanics relies largely on Linear Algebra, this means that whenever we want to implement an operator or a state into the computer, we first have to pick a basis. A basis is just a set of orthonormal states that span your Hilbert space. They can be time-dependent which amount to viewing the Hilbert space in a rotating frame. The choice of basis have a large impact on the computation time and possibly also on the accuracy of the simulation.

The different choices and approximations available, while not being exhaustive, will now be discussed.

5.1.1 Canonical conjugate variables

The Hamiltonian can be written up as a function of two conjugate variables, such as position and momentum. In the context of superconducting qubits the operators $\hat{n}$ and $\hat{\phi}$ can be regarded as momentum and position, respectively. They each have corresponding eigenstates, which can be used as a basis. Using the $\hat{n}$ eigenstates is denoted the "charge basis" whereas utilizing the $\hat{\phi}$ eigenstates is called the "flux basis". In order to implement these bases, the spectrum of possible eigenvalues need to be discretized in some manner. When the Hamiltonian is periodic in the continuous variable ϕ, the charge counting operator will have a discretized spectrum, with an eigenvalue spacing given by 2π divided by the period. Therefore, in the context of eq. 4.23, the operator $\hat{n}$ will take on integer values which are interpreted as the number of cooper pairs on the corresponding island. Utilizing the natural discretization of $\hat{n}$ makes the charge basis a good choice of basis for periodic Hamiltonians, since the representation will be efficient in that a higher accuracy is reached with fewer number of basis states as compared to the flux basis.

In the **charge basis**, the matrix representation for $\hat{n}$ is simply a diagonal matrix with the eigenvalues along the diagonal. These are integers and we can define them in relation to the charge offset, such that the domain is around 0. Sorted from lowest to highest this

becomes:

$$\hat{n} \doteq \begin{pmatrix} -n_{\max} & & & \\ & -n_{\max}+1 & & \\ & & \ddots & \\ & & & n_{\max} \end{pmatrix} \equiv Q \tag{5.1}$$

It is then of interest to find the matrix representation of the operator

$$\cos\hat{\phi} = \frac{1}{2}(e^{i\hat{\phi}} + e^{-i\hat{\phi}}) \tag{5.2}$$

From almost any Quantum Mechanics textbook such as [3], we know that:

$$\langle\phi|n\rangle = \frac{1}{\sqrt{2\pi}} e^{i\phi n} \tag{5.3}$$

the matrix elements are then evaluated as:

$$\langle m|e^{-i\hat{\phi}}|n\rangle = \frac{1}{2\pi}\int_{-\pi}^{\pi}\langle m|e^{-i\hat{\phi}}|\phi\rangle\langle\phi|n\rangle d\phi = \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{-i\phi(m-n+1)}d\phi = \delta_{m,n-1} \tag{5.4}$$

such that we can define the matrix:

$$|m\rangle\langle m|e^{-i\hat{\phi}}|n\rangle\langle n| = \delta_{m,n-1}|m\rangle\langle n| \doteq \begin{pmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{pmatrix} \equiv A^\dagger \tag{5.5}$$

$A^\dagger$ is used to denote this matrix since the matrix is similar to a raising operator.

Now it is possible to write the operators of the Hamilton in eq. 4.23 in a matrix representation of the charge basis. Because there are two sets of canonical variables, the full Hilbert space is a product of two Hilbert spaces, such that:

$$\hat{n}_1^2 \doteq Q^2 \otimes \mathbb{1} \tag{5.6}$$

$$\hat{n}_2^2 \doteq \mathbb{1} \otimes Q^2 \tag{5.7}$$

$$\cos\hat{\phi}_1 \doteq \frac{1}{2}\left(A \otimes \mathbb{1} + A^\dagger \otimes \mathbb{1}\right) \tag{5.8}$$

$$\cos\hat{\phi}_2 \doteq \frac{1}{2}\left(\mathbb{1} \otimes A + \mathbb{1} \otimes A^\dagger\right) \tag{5.9}$$

$$\cos(\hat{\phi}_1 - \hat{\phi}_2 + \phi_{\text{ext}}) = \frac{1}{2}\left(e^{i\phi_1}e^{-i\phi_2}e^{i\phi_{\text{ext}}} + e^{-i\phi_1}e^{i\phi_2}e^{-i\phi_{\text{ext}}}\right) \tag{5.10}$$

$$\doteq \frac{1}{2}\left(e^{i\phi_{\text{ext}}}A \otimes A^\dagger + e^{-i\phi_{\text{ext}}}A^\dagger \otimes A\right) \tag{5.11}$$

In the **flux basis**, the matrix representation of $\hat{\phi}$ is diagonal and easily exponentiated. The $\hat{n}$ is then proportional to the derivative wrt. ϕ. This is approximated by finite differences such as:

$$f'(x_n) \approx \frac{f(x_{n+1}) - f(x_{n-1})}{2\Delta x} + \mathcal{O}(\Delta x^2) \tag{5.12}$$

$$f'(x_n) \approx \pm\frac{f(x_{n\pm1}) - f(x_n)}{\Delta x} + \mathcal{O}(\Delta x) \tag{5.13}$$

$$f''(x_n) \approx \frac{f(x_{n+1}) - 2f(x_n) + f(x_{n-1})}{\Delta x^2} + \mathcal{O}(\Delta x^2) \tag{5.14}$$

The reason for mentioning eq. 5.13 even though eq. 5.12 has better scaling with Δx is that the latter possess the caveat that it only compare pairs of points within two different subdomains, that is to say eg. the sequence $\{a, b, a, b, a, ...\}$ would have a constant first derivative of zero, even though that's clearly not the case. So in some contexts, eq. 5.13 might be advantageous. However, using eq. 5.12 and 5.14 yields:

$$\hat{n} \doteq \frac{-i}{2\Delta\phi} \begin{pmatrix} 0 & 1 & & & \\ -1 & 0 & 1 & & \\ & & \ddots & & \\ & & -1 & 0 & 1 \\ & & & -1 & 0 \end{pmatrix} \tag{5.15}$$

$$\hat{n}^2 \doteq \frac{-1}{\Delta\phi^2} \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & & \ddots & & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix} \tag{5.16}$$

in the flux basis.

5.1.2 Instantaneous Basis

The Hamiltonian is readily written up in either the charge or the flux basis, with the charge basis being the superior choice for periodic potentials. As indicated in section[1] 4.3.1 a charge cutoff on the order of $n_{\max} \sim 10$ is sufficient to model the system to many significant digits, which would equal a Hilbert space dimension of[2] $(2 \cdot 10 + 1)^2 = 441$. Therefore, it may be that these bases are not optimal in the sense that the dynamics may just as accurately be described by a smaller Hilbert space, which would greatly improve the simulation efforts. One informed guess of such a subset could be the eigenstates of the Hamiltonian, where it may be that perhaps only the $\sim$ 30 lowest lying energy states are necessary to model the system to just as many significant digits. However, with the α parameter changing, the eigenstates of the qubit Hamiltonian is also changing. Therefor, a time dependent basis is needed to stay in the description spanned by the energy eigenstates at each instant, denoted the instantaneous basis.

[1] Further evidence is provided in appendix figure 10.1

[2] A single mode is in the representation of $2n_{\max} + 1$ basis states and with two modes the Hilbert space dimensionality becomes $(2n_{\max} + 1)^2$

This can be formulated as the following: In the context of the DSFQ Hamiltonian eq. 4.23, the Hamiltonian consists of three terms:

$$\mathcal{H}(t) = H_0 + H_\alpha \alpha(t) + Vu(t) \tag{5.17}$$

Which for brevity and generalizability can be collected into a term defining the qubit $H(t) = H_0 + H_\alpha \alpha(t)$ and a (not necessarily small) perturbation to this hamiltonian $Vu(t)$. We can then define the collection of eigenstates that diagonalizes $H(t)$ at the instant t as:

$$D(t) = U^\dagger(t)(H_0 + H_\alpha \alpha(t))U(t) \tag{5.18}$$

Where D is diagonal with the energies as entrances. A basis change of:

$$\tilde{\psi} = U^\dagger(t)\psi \tag{5.19}$$

then yields the Hamiltonian:

$$\tilde{\mathcal{H}} = D(t) + i\dot{U}^\dagger(t)U(t) + U^\dagger(t)VU(t)u(t) \tag{5.20}$$

Where the centrifugal term $i\dot{U}^\dagger U$ is acquired from the time dependent basis not being an "inertial frame of reference".

It is worthwhile to clarify the construction of this representation. At any instant $\tilde{t}$ the $\alpha(t)$ parameter takes the value $\alpha(\tilde{t}) = \tilde{\alpha}$. There exists then a set of eigenstates for the Hamiltonian $H_0 + H_\alpha\tilde{\alpha}$. Each column of $U(\tilde{t})$ is equal to one of these eigenstates and the corresponding entry in $D(\tilde{t})$ is equal to the eigenvalue (ie. the energy). So with $(H_0 + H_\alpha\tilde{\alpha})|n_{\tilde{\alpha}}\rangle = E_{n,\tilde{\alpha}}|n_{\tilde{\alpha}}\rangle$ one may write:

$$U(\tilde{t}) = \begin{pmatrix} |0_{\tilde{\alpha}}\rangle & |1_{\tilde{\alpha}}\rangle & \dots & |N_{\tilde{\alpha}}^{\text{columns}}\rangle \end{pmatrix} \tag{5.21}$$

Where each ket is a column vector and N^{columns} would be the dimensionality of the Hilbert space representation. However, it remains unclear how to utilize this in practice, the $\dot{U}^\dagger U$ term especially seems a bit elusive, so in order to appreciate this description we have to consider:

$$\dot{D} = \frac{d}{dt}(U^\dagger H U) \tag{5.22}$$

which yields the equations:

$$\dot{D}_{ii} = (U^\dagger \dot{H} U)_{ii} \tag{5.23}$$

$$(\dot{U}^\dagger U)_{ij} = \frac{(U^\dagger \dot{H} U)_{ij}}{D_{ii} - D_{jj}} \quad \text{for} \quad i \neq j \tag{5.24}$$

This shifts the time dependence of the matrices onto the scalar function of $H(t)$ namely $\alpha(t)$, since $\dot{H} = H_\alpha\dot{\alpha}$. Evidently, equations for the diagonal of $\dot{U}^\dagger U$ are missing. This is due to the "Gauge Freedom" when diagonalizing H, since each eigenvector (present as columns in U) has a $\mathcal{U}(1)$ symmetry. It is possible to pick a gauge by specifying $\dot{U}^\dagger U$ along the diagonal, as will now be derived. The idea is to consider the consistency relation:

$$\dot{\mathbb{1}} = 0 = (\dot{U^\dagger U}) = \dot{U}^\dagger U + U^\dagger \dot{U} \tag{5.25}$$

Everywhere but the diagonal this is automatically given from eq. 5.24 since the expression is anti-hermitian. However, along the diagonal, it is clear that:

$$\mathcal{R}e((\dot{U}^\dagger U)_{ii}) = 0 \tag{5.26}$$

So the diagonal has to be imaginary, which also simply follows from the statement that the centrifugal term present in the Hamiltonian need be Hermitian. One choice is $(\dot{U}^\dagger U)_{nn} = i\omega_0$ which would effectively decrease the eigenvalues of $D(t)$, ie. this constitutes going to the rotating frame.[3]
Another fine choice of gauge is simply $(\dot{U}^\dagger U)_{ii} = 0$.

[3] It can also be understood as the extension of the basis change in eq. 5.19 to $U^\dagger(t) \to e^{i\omega_0 t}U^\dagger(t)$.

The idea is then that the $N_H \times N_H$ system[4] of $\mathcal{H}$ can be described in the instantenous basis by using only a subset N of eg. the lowest lying instantaneous energy states. That is to say, the square matrix U of eq. 5.21 becomes a rectangular matrix, by decreasing the number of columns from N_H to N such that the dimensionality of U is now $N_H \times N$ with $N < N_H$

In practice this means that for a fixed dynamics of $H(t)$, if we want to simulate many different instants of $u(t)$[5], it is only necessary to determine $U(t)$ once. Then it is possible, using the basis change discussed above, to simulate the dynamics using the small $N \times N$ matrices rather than the large $N_H \times N_H$ matrices. It is examined later whether this approach also keeps the accuracy of the simulation. It could be that using this subset of Hilbert space is not sufficient at describing the dynamics.

Finding $U(t)$ is done either by using eq. 5.24 to evolve it forwards in time. That is, using the notation of the Hadamard product[6] we may define $F_{ij} = 1/(D_{ii} - D_{jj})$ with the choice of gauge $F_{ii} = 0$, such that:

$$\dot{U}^\dagger = F \circ (U^\dagger \dot{H} U) U^\dagger \tag{5.27}$$

which is to be solved for $U(t)$.

In the case of discrete time, we may diagonalize H at each time step and store the result. Care then needs to be taken to fix the gauge and reorder the columns of U if a level crossing is traversed. However, even for just a two-qubit system $N_H = (2n_{\max} + 1)^4 \sim \mathcal{O}(10^5)$ it can already be cumbersome to find the rectangular U matrices with dimension $N_H \times N$, even though it only has to be done once.
Furthermore, it is quite restrictive to only have to consider a single instance of $H(t)$. That is to say, so far it is only possible to specify $\alpha(t)$ and then derive the rectangular matrices $U(t)$ of which it is then possible to efficiently simulate many different pulses. It would be more useful if the savings could also be done for different $\alpha(t)$ which includes only having to diagonalize big matrices once.
To this end, we can do the following. Define:

$$U_\alpha(t) = U^\dagger(t) H_\alpha U(t) \tag{5.28}$$
$$U_V(t) = U^\dagger(t) V U(t) \tag{5.29}$$

These are $N \times N$ matrices, with equations of motion:

$$\dot{U}_{\alpha/V} = \dot{U}^\dagger U U_{\alpha/V} - U_{\alpha/V} \dot{U}^\dagger U \tag{5.30}$$

The diagonal part of the hamiltonian simply obey the equation of motion:

$$\dot{D}_{ii} = (U_\alpha)_{ii} \dot{\alpha}(t) \tag{5.31}$$

whereas the centrifugal term is simply[7]:

$$\dot{U}^\dagger U = F \circ U_\alpha \dot{\alpha} \tag{5.32}$$

That is, the system can be described in the instantaneous basis by $N \times N$ matrices alone, with only having to revert to the N_H dimensions for

[4] N_H refer to the dimension of the Hilbert space representation such that $\mathcal{H}$ is an N_H by N_H matrix.

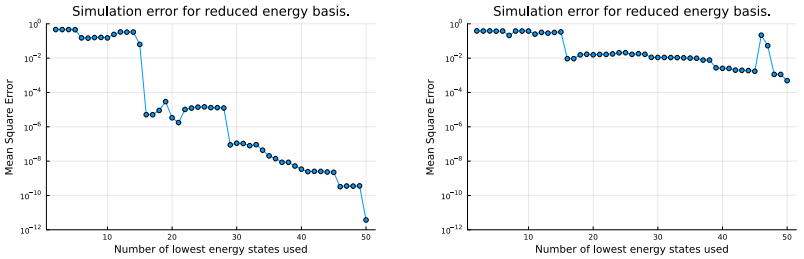
[5] Ie. trying out many different pulses on our qubit.

[6] This is simply element-wise matrix multiplication, ie. $(A \circ B)_{ij} = A_{ij} B_{ij}$

[7] For the particular choice of gauge: $(\dot{U}^\dagger U)_{ii} = 0$

the initial conditions. This is thus a promising avenue for simulating eg. two-qubit gates in the context of quantum optimal control, where you want to be able to simulate the system many times effectively.

However, the question remains, how large should N be in order to accurately describe the dynamics, if N is not much smaller than N_H nothing is gained. To this end, different instants of $\alpha(t)$ and $u(t)$ for a single-qubit gate was simulated using this approach and the dynamics were compared to a slow but accurate simulation using the charge basis. The comparisons were made by comparing the spectral weight as a function of time between the simulations made by the charge basis and the instantaneous basis, but for all intents and purposes, just think of it as the simulation error accrued when using the instantaneous basis. The simulation error as a function of the number of lowest lying energy states used (ie. the number of columns of U) are shown in figure 5.1. Increasing N decreases the simulation error, as it should. However, the two figures (a) and (b) are for two different sets of scalar functions $\alpha(t)$ and $u(t)$, which corresponds to different dynamics. As it turns out, it is not simple to say whether the lowest lying energy states properly describe the dynamics to the wanted accuracy since this is dependent on the dynamics. For one set of $\alpha(t)$ and $u(t)$, $N \sim 30$ yields a somewhat acceptable simulation error of around 10^{-7} while yielding a massive speedup in simulation time. However, for another set of dynamics, the simulation error for $N \sim 30$ would be on the order of 10^{-2}. Therefore, in generality this approach would not work since there is no clear bound on the simulation error. It would be interesting to investigate the interplay between the dynamics and the subset of the energy eigenstates that are needed to describe the dynamics. However, the instantaneous basis was no longer considered for this thesis as soon as it was apparent that it does not capture the dynamics in general.



(a) (b)

Figure 5.1: The simulation error accrued when using the instantaneous basis compared to the charge basis as a function of N. The dynamics for (a) was given by $\alpha_a(t)$ and $u_a(t)$ whereas for (b) another set of dynamics was simulated ($\alpha_b(t)$ and $u_b(t)$). It is clear that the simulation error decreases when increasing N but the simulation error is also dependent on the dynamics. The two points of increased simulation error in (b) are from simulations that returned error messages and should be disregarded.

5.1.3 Shift in eigenvalues

The eigenvalues of any Hamiltonian which ends up being used in the simulation can have a large impact on the simulation time. This is because, loosely speaking, a large eigenvalue $\hbar\omega_{\text{fast}}$ will bring a $e^{-i\omega_{\text{fast}}t}$ contribution to the dynamics. This constitute fast oscillations and therefore any "integration scheme" (as will be discussed in section 5.2) will have to sample the Schrödinger equation at many more points compared to if the dynamics did not contain these fast oscillations.

Therefore, it may be advantageous to shift the energy spectra of the Hamiltonian by the usual transformation $|\psi\rangle \to e^{i\omega_0 t}|\psi\rangle$. Good choices for ω_0 will either be the mean of the energies, ie. $\omega_0 = \bar{E}/\hbar = \frac{1}{N_H \hbar}\sum_i E_i$ or in the context of simulating the dynamics of a computational basis, simply use $\omega_0 = (E_0 + E_1)/(2\hbar)$

There is no reason not to use the rotating frame when simulating the dynamics since this eases the simulation efforts and practically, the implementation is simply $\mathcal{H} \to \mathcal{H} - \hbar\omega_0 \mathbb{1}$

5.1.4 Learned representation

A different approach was also investigated when trying to find the minimal basis that in general could accurately describe the dynamics. As in the case of section 5.1.2, the approach that will be described now, ended up not being able to describe the dynamics to the accuracy necessary for this thesis. However, the underlying idea seems interesting and it is still worth including here.

The idea is, that if we know which parts of Hilbert space the state will be contained in, we can sample these parts and then use optimization techniques to find a fewer set of states that can account for approximately all the spectral weight for each sample. The relevant parts of Hilbert space was presumed to be described by the lowest lying energy states of the Hamiltonian[8]:

$$H|n,\alpha\rangle = (H_0 + H_\alpha \alpha)|n_\alpha\rangle = E_{n,\alpha}|n_\alpha\rangle \tag{5.33}$$

where α is a scalar, just as before. The goal is to find a matrix $U(\theta)$, parametrized by a set of parameters $\{\theta\}$, which when used as a projection operator over the subspace of interest, get as close as possible to the identity. That is:[9]

$$\text{Find } \theta \text{ such that } U(\theta)U^\dagger(\theta)|n_\alpha\rangle \simeq |n_\alpha\rangle \text{ for } n \in [1,\dots n_{\max}] \text{ and } \alpha \in [0.5, 1] \tag{5.34}$$

The idea hinges on the dimensionality of $U(\theta)$ being $N_H \times N$ with N much smaller than N_H, such that when eq. 5.17 is to be simulated many times, the N-dimensional system[10]:

$$-i\frac{d}{dt}(U^\dagger(\theta^*)|\psi\rangle) = U^\dagger(\theta^*)\mathcal{H}(t)U(\theta^*)(U^\dagger(\theta^*)|\psi\rangle) \tag{5.35}$$

is much more effective to simulate and to approximately the same accuracy (hopefully). The θ^* is the solution of eq. 5.34.

The simplest approach for parametrizing $U(\theta)$ that was employed, was to simply let the θ's be the entrances of U.[11] Unitarity is then ensured by simply normalizing each column of U and then apply the Gram-Schmidt orthogonalization procedure [61]. The projection operator can also be defined as $(U^\dagger(\theta))^{-1}U^\dagger(\theta)$ where the pseudo-inverse is used. This works out to yield the same results but the second option is more readily implemented through eg. PyTorchs linear algebra library[12] while also being optimized for the ensuing backpropagation.

[8] Which in view of the conclusion of section 5.1.2 is not the case.

[9] This expressions assumes that $U(\theta)$ is unitary.

[10] This whole construction is much like the instantaneous basis of section 5.1.2, but instead of analytically deriving a set of basis states, we now employ machine learning to find the "best" basis states.

[11] $U = \begin{pmatrix} \theta_{00} & \theta_{01} & \dots & \theta_{0N} \\ \theta_{10} & \ddots & & \\ \vdots & & \ddots & \\ \theta_{N_H 0} & & & \theta_{N_H N} \end{pmatrix}$

[12] The "pseudo" in pseudo-inverse is necessary because U is not square. The python function employed was torch.linalg.pinverse

Stated more clearly, equation 5.34 can be solved as an optimization problem, as the following. Gather a "dataset" of the relevant wavefunctions that we want to represent, in this case $\{|n_\alpha\rangle\}$. The objective is, that these states, when described in terms of the basis defined by $U(\theta)$

$$|\tilde{\psi}_{n,\alpha}\rangle = (U^\dagger(\theta))^{-1}U^\dagger(\theta)|n_\alpha\rangle \tag{5.36}$$

remain the same.
Therefore, a suitable objective function, could be:[13]

$$\mathcal{L} = \sum_{n,\alpha}[\mathcal{R}e(|\tilde{\psi}_{n,\alpha}\rangle - |n_\alpha\rangle)]^2 + [\mathcal{I}m(|\tilde{\psi}_{n,\alpha}\rangle - |n_\alpha\rangle)]^2 \tag{5.37}$$

[13] Notice that the global phase survives the projection, so it is not necessary to make any considerations regarding the gauge.

The objective function eq. 5.37 simply describes how well the states $|n_\alpha\rangle$, sampled from the relevant parts of Hilbert space, survives the transformation of eq. 5.36. If $(U^\dagger(\theta))^{-1}U^\dagger(\theta)$ is the identity over all the different $|n_\alpha\rangle$, this approach will have been successful and the objective will have reached the global minimum $\mathcal{L} = 0$

Any optimization technique can then be employed to find the θ^* that minimizes $\mathcal{L}$. To this end, $U(\theta)$ was implemented using the python machine learning library PyTorch, such that the automatic differentiation could be used for the gradient based optimizer, ADAM (for a clarification of these terms, see section 6)

Such a minimization procedure was done with 30 basis states to be learned, ie. $N = 30$, with the dataset consisting of the[14] $n_{\max} = 19$ lowest lying energy states for 85 different α ranging from 0.5 to 1. So the double-sum of eq. 5.37 consists of $19 \cdot 85 = 1615$ summands.[15] After the optimization terminated, the resulting infidelity, as measured by the missing spectral weight when projecting onto $U^\dagger(\theta^*)$, can be seen in figure 5.2. The method does a perhaps surprisingly

[14] $n_{\max}$ of eq. 5.34 not eq. 5.1

[15] Excluding the factor 2 from the real part and the imaginary part.

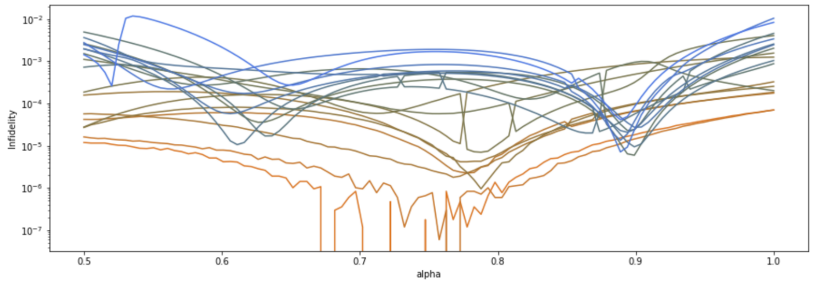


Figure 5.2: Infidelity when projecting onto $U^\dagger(\theta^*)$ as a function of α, each line represents an energy state, with orange being the lowest and blue the highest energy state. The discontinuities in the plot stem from level crossings. See figure 10.2 in the appendix for a visualization of the learned basis states.

good job, but the infidelities are too high for the ensuing simulations to be accurate enough to the number of significant digits wanted for this thesis. Therefore, no further work was done in this direction. However, it does seem like a viable option in other contexts.

The takeaway of this section is that despite the efforts of finding a non-trivial efficient and accurate representation, the charge basis remain the most faithful representation. The charge basis is therefore the necessary choice for the rest of the thesis since accurate simulations are desired. The next section will treat the question of whether

it is feasible to simulate the dynamics to the wanted accuracy in a sufficiently short time such that the trial-and-error nature of the optimization techniques remain unprohibited.

5.2 Integrating the Schrödinger equation

Integrating the Schrödinger equation means finding a solution to the initial value problem defined by the generally time dependent Hamiltonian. This can not be done in general and we revert to numerical methods in order to approximate these solutions. In the following two methods will be laid out and discussed. These may be regarded as a discrete (Trotter) and a continuous method (ODESolver).

5.2.1 Trotter decomposition

Formally, the solution to Schrödingers equation can be written in closed form, through the use of the time-ordering operator[16] $\mathcal{T}$. The time evolution operator is then given by:

$$\mathcal{U}(t) = \mathcal{T} e^{-\frac{i}{\hbar}\int_0^t H(t')dt'} \tag{5.38}$$

[16] $\mathcal{T}$ orders the operators according to time while disregarding their commutation relations: $\mathcal{T}A(t_1)B(t_2) = B(t_2)A(t_1)$ for $t_1 < t_2$ and any A and B (However, care should be made of fermionic operators in second quantization.)

Using the Lie-Trotter product formula [62], this can be written as:

$$\mathcal{U}(t) = \lim_{N\to\infty} \prod_{n=0}^{N} e^{-\frac{i}{\hbar}H(\frac{t}{N}n)\frac{t}{N}} \tag{5.39}$$

The idea is that if we go to high enough finite N the product will still aptly describe the time evolution operator:

$$\mathcal{U}(t) \simeq \prod_{n=0}^{N} e^{-\frac{i}{\hbar}H(\frac{t}{N}n)\frac{t}{N}} \tag{5.40}$$

One way to view the approximation is that, it is a truncation of the Baker-Campbell-Hausdorff formula (or more precisely the Zassenhaus formula) [63]:

$$e^{t(X+Y)} = e^{tX}e^{tY}e^{-\frac{t^2}{2}[X,Y]}e^{\frac{t^3}{6}(2[Y,[X,Y]]+[X,[X,Y]])}e^{\mathcal{O}(t^4)} \tag{5.41}$$

This yields one avenue to estimate the size of N as is explored in appendix 10.2.3 .

In practice the implementation of eq. 5.40 goes as the following. The time axis is divvied up into N evenly spaced pieces $t \to t_n = \Delta t n$. The time-dependent Hamiltonian is evaluated at each of the N times and represented as some matrix $H(\Delta t n) \to H_n$. Each of these matrices are multiplied by $-i\Delta t$ (with $\hbar = 1$) and then exponentiated either analytically or using numerical methods for exponentiation of matrices. A cumulative product of the resulting matrices is then done in order to approximate the time-evolution operator at each time step, ie:

$$\mathcal{U}(\Delta t m) \to \mathcal{U}_m = \prod_{n=0}^{m} \exp(-i\Delta t H_n) \tag{5.42}$$

Most of the computational effort is spent exponentiating matrices and doing matrix multiplication, the last of which can be done efficiently on a Graphics Processing Unit (GPU).

The strength of this method is that it is readily implemented and easily used for quantum optimal control[17]. That is because, as an example, the python Machine Learning module Pytorch can handle complex numbers and exponentiate matrices whilst being able to calculate gradients of objective functions via backpropagation.

[17] Quantum Optimal Control is described later in section 6

The trouble, however, lies in two complementary aspects related to the hyperparameter N. The fact that N needs to be sufficiently large for the simulation to converge necessitates a larger memory to store the all function evaluations that are needed for the gradient calculation, when done with backpropagation, as will be explained in section 6.3. Furthermore, if each matrix exponentiation is approximated numerically, a large N means that errors will accumulate to a too large degree when they are combined in eq. 5.42.[18]

[18] This was at least the experience when using the matrix exponentiation methods implemented in Python. In the progamming language Julia, the functions included a parameter that could tune the tolerance to zero. When this was done the effect of a worse simulation when increasing N was no longer present.

So, a small N means a mathematically bad convergence whilst a large N introduces computational errors and large computation times. Ie. this is seemingly not the way to go for precise and efficient simulations. This was at least the experience in the context of this thesis, where the normalization of the state decreased on the percent level when using the Pytorch implementation of matrix exponentials.

Therefore, the use of this method was abandoned and the following method was adopted instead.

5.2.2 Ordinary Differential Equation Solver

Solving differential equations numerically is done throughout a wide range of fields. It is thus a well-studied field and an extensive overview will not be provided. Instead, some of the schemes employed in this thesis will be mentioned and some of the terminology explained, with the general reference being to a standard textbook on the matter [64].

The objective simply stated is to approximate the solution $y(t)$ of:

$$\dot{y} = f(t, y) \tag{5.43}$$

Given an initial value $y(0) = y_0$. Most methods propagate y_0 forward, using f in order to obtain $y(t)$ at specific time-steps. The distribution of these time-steps may be dense in order to approximate the continuous solution.

The first distinction to be made is between Linear multistep methods and Runge-Kutta methods. Linear multistep methods takes into consideration the computed solutions at previous time steps when approximating the solution at the next time step. Runge-Kutta (RK) methods use only the current time-step and f [65]. RK methods denote a large family of numerical methods and is the focus of the following overview. In general, an RK-method generates a solution at

the next time step by the following:

$$y(t_{n+1}) = y(t_n) + \Delta t \sum_{i=1}^{s} b_i k_i(t_n) \tag{5.44}$$

where Δt is the step-size, s is the "stage" of the scheme and the k_i are given by:

$$k_i(t_n) = f(t_n + c_i \Delta t, y(t_n) + \Delta t \sum_{j=1}^{s} a_{ij} k_j(t_n)) \tag{5.45}$$

The scheme-dependent parameters a, b and c can be summarized using a Butcher tableau, see fig. 5.3.

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s,s} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$

Figure 5.3: Generic example of a Butcher Tableau which organizes the coefficients of the Runge-Kutta scheme. The coefficients c_i, a_{ij} and b_i fully characterize a Runge-Kutta method [66]. (The figure is from the same source)

If $a_{ij} = 0$ for $i \leq j$ then the scheme is explicit (ERK) since the k_j in that case only depend on the previous k_i's. If this is not the case the scheme is said to be implicit (IRK), and one needs to somehow solve for the k_j's.

However, if $a_{ij} = 0$ for $i < j$ it is said to be diagonally implicit (DIRK)[19] and if a_{ii} are the same for all i it is said to be singly diagonally implicit (SDIRK).

DIRK has the advantage of separating the problem of solving for the k_i's into sub-problems and with the extra condition that all eigenvalues of the a_{ij} matrix is the same (SDIRK), a set of these sub-problems can be solved at once with a single eigenvalue-decomposition [67].

Lastly, if an otherwise SDIRK method has $a_{11} = 0$ it is an explicit first stage, singly diagonally implicit method (ESDIRK), which seemingly allows for higher convergence order [68].

For differential equations there exists a notion of "stiffness",[20] which has no unambiguous definition but was allegedly[21] described by those who coined the term [69] as:

> "stiff equations are equations where certain implicit methods perform better, usually tremendously better, than explicit ones."

The order of an RK scheme refer to the exponent p of the term $\mathcal{O}(\Delta t^{p+1})$ that is "ignored" in the expression of eq. 5.44, that is, a p-th order RK method is accurate to $\mathcal{O}(\Delta t^p)$ in Δt.[22]

When the ODE has a natural decomposition $f = f_0 + f_1 + \ldots$, where each f_j is more simple than f and some of the terms are only mildly stiff, splitting methods can be employed. The idea is that this allows one to utilize the efficiency of an explicit scheme while only having to use an implicit method for calculating corrections [71].

Embedded methods refer to the case where two RK-methods are employed,[23] which almost share a Butcher tableau. This means, that through not much additional computational effort, it is possible to estimate the error and vary the "stepsize" (Δt) accordingly. These methods are therefore also known as adaptive methods. When the lower order RK-method only differ in the last row of the Butcher-tableau, with entries denoted by b_i^*, the estimated error is given by:

$$e(t_{n+1}) = y(t_{n+1}) - y^*(t_{n+1}) = \Delta t \sum_{i=1}^{s} (b_i - b_i^*) k_i(t_n) \tag{5.46}$$

[19] or semi-implicit

[20] Note, stiffness is an attribute of the relation between the ODE and the numerical schemes that seek to approximate its solutions. It has no relation to the solution itself.

[21] [65] state it in "Stiff differential equations" but the quote is not found in [69]. Nonetheless, the sentiment is sound [70].

[22] Furthermore, it is a theorem that $s \geq p$ and for $p \geq 5$ this changes to $s > p$. For a proof see p. 187-188 in [64]

[23] usually one of order p and the other of order $p + 1$

Lastly, stability can be defined through different tests usually labelled by a capital latter but these will not be described here. [72] [64]

Hopefully, it should now make sense to say, that the algorithm used in this thesis, is a composite algorithm consisting of the two Runge-Kutta methods:

- Vern9 [73]: Verner's "Most Efficient" 9/8 Runge-Kutta method [74].
- KenCarp47 [75]: An A-L stable stiffly-accurate 4th order seven-stage ESDIRK method with splitting.

For practical considerations made with regards to the implementations used in this thesis, see appendix 10.2.4.

So far, only dynamics described by the Schrödinger equation have been considered. This is also used throughout the rest of the thesis, but since this description is insufficient at describing decoherence of the qubit, the following chapter will estimate the decoherence of the qubit before reverting back to the closed quantum system description given by the Schrödinger equation.

5.3 Decoherence estimation

5.3.1 First order analytic expression

As mentioned in section 3.4.4 our control of the system is not exact and it will always exhibit some decoherence. To estimate this decoherence the density matrix formalism is used. It is assumed that the dynamics are of the form of eq. 3.36, with the jump operators given by eq. 3.37 and 3.38.

There exists numerical procedures to simulate the dynamics of such an open system (examples include [76] [77] [78] [79]) but these will not be considered here. Instead an analytical perturbative approach will be mentioned, which have been used in [80]. The idea is to expand $\rho(t)$ in terms of $\Gamma t = t/T_{\text{decoherence}}$ where Γ is related to the error rates of the open system:

$$\rho = \rho^{(0)} + \rho^{(1)} + \dots \tag{5.47}$$

Such that:

$$\rho^{(0)} = |\psi(t)\rangle\langle\psi(t)| \tag{5.48}$$

is the ideal system whose dynamics are purely given by the Schrödinger equation. The first order correction on the other hand follows:

$$\frac{d\rho^{(1)}}{dt} = -i[H, \rho^{(1)}(t)] + \mathcal{L}\rho^{(0)}(t) \tag{5.49}$$

which, for a time-independent system, has the solution:

$$\rho^{(1)}(t) = e^{-iHt} \int_0^t e^{iHt'} \mathcal{L}\rho^{(0)}(t') e^{-iHt'} dt' e^{iHt} \tag{5.50}$$

The decrease in fidelity due to decoherence can then be estimated by:

$$1 - F_\psi = 1 - \mathrm{Tr}[\rho^{(0)}\rho(t)] \simeq -\mathrm{Tr}[\rho^{(0)}(t)\rho^{(1)}(t)] \tag{5.51}$$

For simplicity a surrogate model was used, namely a two-dimensional system undergoing a sine pulse in the rotating-wave approximation akin to eq. 3.17. The idea was then to study the effect of the model parameters on the decoherence, but as it turns out, the resulting decoherence is agnostic to the model parameters. That is, with $\rho^{(0)}$ given as an arbitrary pure state on the bloch sphere, $\rho^{(1)}(t)$ could be found from eq. 5.50. Both terms were put into eq. 5.51 which was then averaged over the whole Bloch sphere, resulting in:

$$1 - \langle F_\psi \rangle_{\text{Bloch-Sphere}} \simeq \frac{\Gamma_1 + \Gamma_\varphi}{3} t \tag{5.52}$$

which is obviously only dependent on the error rates and not the model parameters. It should be noted that the cancellation happens when averaging over the Bloch-sphere. As it turns out, this is the same result that is achieved from the Bloch-Redfield density matrix (eq. 3.39):

$$1 - F_{\text{BR}} = 1 - |\alpha|^2(1 - e^{-\Gamma_1 t}) - 2|\alpha|^2|\beta|^2 e^{-\Gamma_2 t} - (|\alpha|^4 + |\beta|^4)e^{-\Gamma_1 t} \tag{5.53}$$

Since by averaging over the Bloch-sphere and only considering to first order, we again retrieve:

$$1 - \langle F_{\text{BR}} \rangle_{\text{Bloch-Sphere}} \simeq \frac{\Gamma_1 + \Gamma_\varphi}{3} t \tag{5.54}$$

In order to estimate the decrease in fidelity due to decoherence, it is therefore necessary to estimate the error rates Γ_1 and Γ_φ. This is done in the following.

5.3.2 Calculation of error rates

Following the derivation found in [48] [58] and the sources found within. The relaxation rate for a noisy channel λ is given by:[24]

[24] $\hbar = 1$ in the following.

$$\Gamma_1^\lambda = |\langle 0|\partial_\lambda \mathcal{H}|1\rangle|^2 S_\lambda(|\omega_q|) \tag{5.55}$$

where

$$S_\lambda(\omega) = \int_{-\infty}^{\infty} e^{-i\omega t}\langle \delta\lambda(t)\delta\lambda(0)\rangle dt \tag{5.56}$$

is the power spectral function of the associated noise channel, defined as the fourier transform of the auto-correlation function of the stochastic variable $\delta\lambda$.

For the DSFQ (section 4.3.1), the channels that limit the coherence are dephasing due to the external flux Φ_{ext} and relaxation due to dielectric loss, the last of which has the power spectral function:[25]

[25] The values used are, for the loss tangent $\tan\delta_{\text{diel}} = 2 \cdot 10^{-7}$ and a temperature of $T = 20\text{mK}$

$$S^{\text{diel}}(\omega) = \frac{\omega^2 \tan\delta_{\text{diel}}}{4E_C}\left[\coth\frac{\omega}{2k_B T} + 1\right] \tag{5.57}$$

Since dielectric loss is due to the large capacitances, the relevant matrix element is related to the two capacitively shunted modes, such that:

$$\Gamma_1^{\text{diel}} = (|\langle 0|\phi_1|1\rangle|^2 + |\langle 0|\phi_2|1\rangle|^2) S_\lambda(|\omega_q|) \tag{5.58}$$

The dephasing rate is given by: [26]

$$\Gamma_\varphi^\lambda = \sqrt{2} A_\lambda |\partial_\lambda \omega_q| |\ln(\omega_{\text{ir}} t)| \tag{5.59}$$

where the derivative can be found by:

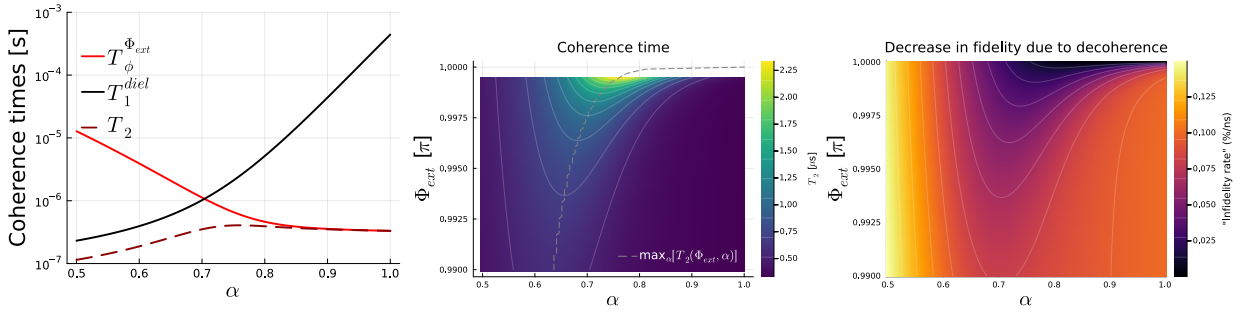
$$\partial_\lambda \omega_q = \langle \partial_\lambda \mathcal{H} \rangle_1 - \langle \partial_\lambda \mathcal{H} \rangle_0 \tag{5.60}$$

Such that in the case of dephasing due to external flux noise in the DSFQ:

$$\partial_{\Phi_{\text{ext}}} \omega_q = E_J \alpha \left(\langle \sin(\hat{\phi}_1 - \hat{\phi}_1 + \phi_{\text{ext}}) \rangle_1 - \langle \sin(\hat{\phi}_1 - \hat{\phi}_1 + \phi_{\text{ext}}) \rangle_0 \right) \tag{5.61}$$

The resulting coherence times are shown in figure 5.4 (a).

[26] The product of the infrared cutoff and characteristic time used is $\omega_{\text{ir}} t = 2\pi \cdot 10^{-6}$ and for the external flux $A_{\Phi_{\text{ext}}} = 10^{-6}\Phi_0$



(a) (b) (c)

Figure 5.4: (a) show the limiting relaxation time T_1^{diel} and dephasing time $T_\varphi^{\Phi_{\text{ext}}}$ and the resulting coherence time as function of α for $\phi_{\text{ext}} = 0.995\pi$. The coherence time for a range of ϕ_{ext} can be found in (b) and (c) show the "infidelity rate" defined by eq. 5.52 as: $\frac{\Gamma_1 + \Gamma_\varphi}{3}$, ie. it is approximately the fidelity decrease due to decoherence per nanosecond.

Firstly, it is noted that the coherence time is quite small ($< 1\mu$s). Secondly, regarding T_1, it is clear that the qubit will be in a protected regime when the barrier is up, with T_1 increasing 3 orders of magnitude compared to the single well potential regime. Unfortunately however, this is outweighed by the increased dephasing due to increased sensitivity to external flux variations. Efforts can be done to try and decrease this susceptibility such as the "Gradiometric DSFQ" in [58] or since dephasing is essentially a coherent error, dynamical error suppression techniques could be employed. However, using eq. 5.58 and the same model parameters, but for the Transmon qubit, yielded a relaxation time of $T_1 \simeq 4.3 \cdot 10^{-8}$s which is very much lower than what is realized in experiments. So perhaps a too low value is used for the loss tangent. At any rate, this indicates that the DSFQ does in fact exhibit protected behaviour, and the quoted coherence times are a lower bound at best. Furthermore, the coherence times are also dependent on the value of the external flux as seen in fig. 5.4 (b) with $\Gamma_\varphi^{\Phi_{\text{ext}}}$ completely disappearing at exactly $\Phi_{\text{ext}} = \Phi_0/2$, but this is due to eq. 5.59 being a first order result, so one would have to go to second order to retrieve a finite dephasing time. Figure 5.4 (c) display the "infidelity rate" which is defined by the expressions of the earlier section, namely $\frac{\Gamma_1 + \Gamma_\varphi}{3}$. For short times this can be regarded as the decrease in fidelity per nanosecond experienced by the qubit due

to the estimated error rates. This is to say, that in later sections when fidelities are quoted to many significant digits this is contradictory to the result shown here since anything beyond the third digit can be discarded due to decoherence effects. However, as already mentioned, these error rates may be larger than what is to be expected in experiment, and it assumed that it is safe to only consider a closed quantum system for the rest of the thesis.

This concludes the simulation concerns of the thesis and the focus will now shift on to how these considerations can be applied in an Optimal Control setting in what is called Quantum Optimal Control.

6 Quantum Optimal Control

Optimal Control encompasses the problem of optimization over dynamical variables. [81] [82] It pertains to dynamical systems for which it is possible to affect the evolution of the variable through control functions. Optimal control concerns itself with finding the control function that yields an evolution such that the dynamical variable optimizes some objective function.

It is a branch of mathematics and is thus accompanied by a rich theory pertaining to the existence and reach-ability of optimal solutions. However, this thesis is more focused on the numerics and application of the ideas behind optimal control in a specific example rather than the general theorems of the theory. Therefore, only an overview of the practical aspects that one should consider is provided.

The first step in optimization is to define the optimization problem. This is done by defining the objective function, also called the critical function or the loss function. Without loss of generality one can choose such that the goal is always to minimize this function. The next step consists of actually finding this minima, and this is where all the trouble lies. Firstly, the loss landscape will be described and then the methods of navigating this landscape in order to find a suitable minimum are presented.

6.1 Loss functions

Within the context of Machine Learning, one generally makes the distinction between regression and classification problems. The usual appropriate loss functions for each are L1 or L2 norm for regression[1], while cross entropy or the area of the receiver operator curve, proves useful for classification [84]. However, in the context of Quantum Optimal Control (QOC), these are not directly applicable. Instead measures of fidelity are used.

For example, if it is desired to transfer the ground state $|\psi(0)\rangle = |\psi_0\rangle$ to the first excited state, one could seek to minimize the infidelity:

$$L(|\psi(T)\rangle) = 1 - F = 1 - |\langle\psi_1|\psi(T)\rangle|^2 \tag{6.1}$$

At the same time, one could have two initial values and define the loss for a transfer of the two lowest lying states to the other, similar to an x-gate. Ie. with $|\psi^{(0)}(0)\rangle = |\psi_0\rangle$, $|\psi^{(1)}(0)\rangle = |\psi_1\rangle$, define:[2]

[1] This is also called the vector p-norm and referes to $|\vec{x}|_p = (\sum_i |x_i|^p)^{1/p}$ [83]

[2] It should be noted that the term swap-gate will be used throughout to refer to a gate that minimizes this loss and is not to be confused with the two qubit SWAP-gate found in litterature [85]

$$L^{\text{swap}}(\{|\psi^{(0)}(T)\rangle, |\psi^{(1)}(T)\rangle\}) = \frac{1}{2}\left(2 - |\langle\psi_1|\psi^{(0)}(T)\rangle|^2 - |\langle\psi_0|\psi^{(1)}(T)\rangle|^2\right) \tag{6.2}$$

In the picture of the Bloch-sphere, this will at best yield a π rotation about an axis in the xy-plane and is independent of the gauge and thereby the orientation of the axes of the Bloch-sphere. If one is interested in a specific rotation-axis for a specified gauge, the gate-infidelity can be defined as eq. 3.20, namely:

$$L^{\text{gate}}(U(T)|U_{\text{target}}) = 1 - \frac{1}{2^2}|\text{Tr}(U_{\text{Target}}^{\dagger}U(T))|^2 \tag{6.3}$$

Where $U(t)$ is the time-evolution operator for the computational basis. This loss seeks to provide a rotation axis that is aligned with the direction of U_{Target}, however it cannot discern between the $\hat{r}$ and $-\hat{r}$ of eq. 3.10, ie. the sign of the rotation axis.[3]

[3] If for some reason distinction between $\hat{r}$ and $-\hat{r}$ is desired, one could seek inspiration from eq. 3.10 for devising a suitable loss function.

It can also be advantageous to define loss functions that seek to maximize or minimize the occupation of some part of the Hilbert space during the evolution. Ie. the following can be used to minimize the gate time of a swap gate:

$$1 - \frac{1}{2T}\int_0^T (|\langle\psi_1|\psi^{(0)}(t)\rangle|^2 + |\langle\psi_0|\psi^{(1)}(t)\rangle|^2)dt \tag{6.4}$$

Since it seeks to maximize the time spent the in transitioned state. This is an example of an objective functional since it not only depends on the state at the final time T but also at all intermediate times.

One can also minimize the occupation in higher energy levels, such as to minimize the decoherence since higher energy levels usually have shorter coherence times.

$$L^{\text{contain}}(\psi(t)|\{\psi_f\}) = \frac{1}{T}\int_0^T \sum_{f\in F} |\langle\psi_f|\psi(t)\rangle|^2 dt \tag{6.5}$$

Where F are the forbidden states. In most cases, the subset of Hilbert space you want the state to be contained to is small, and the loss function is better written in terms of these allowed states:

$$L^{\text{contain}}(\psi(t)|\{\psi_a\}) = 1 - \frac{1}{T}\int_0^T \sum_{a\in A} |\langle\psi_a|\psi(t)\rangle|^2 dt \tag{6.6}$$

In addition to these loss functions, it is possible to include soft constraints on the controls by including the parameters of the controls in the loss function. This is analogous to regularization in deep learning literature. Examples found in litterature include minimizing the first and the second derivative of the pulse and the integral of the absolute square of the pulse [86] [9]. Hard constraints can be made manifest by choosing a proper parametrization of the pulse that enforces the constraint. Pulse parametrization is discussed in section 6.4.

Now, some of the gradient-based algorithms that exists for finding the minimum of such loss functions will be described.

6.2 Minimization procedures

Once a loss landscape is defined, the problem consists of finding the global minimum. Except for convex problems, this is usually infeasible due to the curse of dimensionality, with which the parameter space that needs to be searched grows exponentially with the number of parameters. Therefore, for high-dimensional problems, one often relies on good initialization of parameters and derivative-informed minimization procedures, in order to find the best local minimum.

6.2.1 Gradient Descent

Gradient Descent (GD) is the simplest gradient-based minimization algorithm. It simply updates the parameters by:

$$\vec{p}_{t+1} = \vec{p}_t - \eta \nabla \mathcal{L}(\vec{p}_t) \tag{6.7}$$

where η is the learning-rate or stepsize, which doesn't have to be constant (see section 6.2.3). With its simplicity comes a lot of drawbacks, such as slow convergence, stagnation in the presence of the numerous saddle-points [87] and ultimately no ability to escape even very sub-optimal local minima. The last point is mildly mitigated for problems where it is possible to partition the loss $\mathcal{L}$ into a sum of terms, where an update is made for each term, separately. This is denoted Stochastic Gradient Descent. Another augmentation is the addition of momentum to the GD-algorithm. The name aptly describes the idea, namely that the evaluation point can be prescribed some inertia, with the gradient describing the instantenous acceleration instead of speed. This results in a trajectory of the minimizer in parameter space, that resembles that of an inertial mass that rolls downhill. This can help the speed of convergence and the escape of flat regions or even small minima, but with the introduction of an extra hyperparameter that calls for new considerations [88]. Further improvements can be made by considering adaptive algorithms and in a deep learning context this culminates in the widely used ADAM algorithm [89] that uses exponential moving averages of the gradient and its absolute value, over the trajectory history, for each parameter. This results in an adaptive learning rate for each individual parameter and promotes steps for parameters with low-variance gradients, whilst also including the momentum extension. For this thesis ADAM was only used when working with Python and the Trotter decomposition of sec. 5.2.1, and despite the integration technique's shortcomings, ADAM worked well for all intents and purposes. However, when shifting to the Julia programming language, the minimization algorithm BFGS was more readily available and was thus used for all the results of the thesis.

6.2.2 BFGS

Much of the following is based on a standard textbook on the matter [90].
The above mentioned algorithms are first order derivative methods.

A well known second order method is Newton's method for optimization. Simply said, Newton's method is a root finding algorithm, ie. it finds the solution of $f(x) = 0$ using $f'(x)$. For optimization the root of $f'(x) = 0$ is wanted, thus it becomes a second order method. Newton's method is comparable to Gradient Descent (eq. 6.7) except it utilizes the curvature of the loss landscape when taking a step:

$$\vec{p}_{t+1} = \vec{p}_t - \eta H_{\mathcal{L}}^{-1}(\vec{p}_t)\nabla\mathcal{L}(\vec{p}_t) \tag{6.8}$$

Here $H_{\mathcal{L}}$ is the Hessian of the loss with respect to the parameters. However, if the Hessian is difficult to obtain, it is still possible to approximate it using so called quasi-Newton methods. The Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm is one such algorithm, where the gradient is now preconditioned by the matrix B [90]:

$$\vec{p}_{t+1} = \vec{p}_t - \eta B_t^{-1}\nabla\mathcal{L}(\vec{p}_t) \tag{6.9}$$

Where the update for the inverse approximate Hessian matrix B_t^{-1} is:

$$B_{t+1}^{-1} = \left(\mathbb{1} - \frac{s_t y_t^T}{y_t^T s_t}\right) B_t^{-1} \left(\mathbb{1} - \frac{y_t s_t^T}{y_t^T s_t}\right) + \frac{s_t s_t^T}{y_t^T s_t} \tag{6.10}$$

with $s_t = \vec{p}_{t+1} - \vec{p}_t = -\eta B_t^{-1}\nabla\mathcal{L}(\vec{p}_t)$ and $y_t = \nabla\mathcal{L}(\vec{p}_{t+1}) - \nabla\mathcal{L}(\vec{p}_t)$ and the arrows denoting vectors have been omitted for clarity. The best initialization of B_0^{-1} depends on the problem at hand, but it is possible to simply set it to be the identity matrix [90].

Utilizing more information about the loss landscape usually leads to faster convergence and in some cases to lower minima, at least this was the case for this thesis, where BFGS outperformed GD as seen fig. 6.1

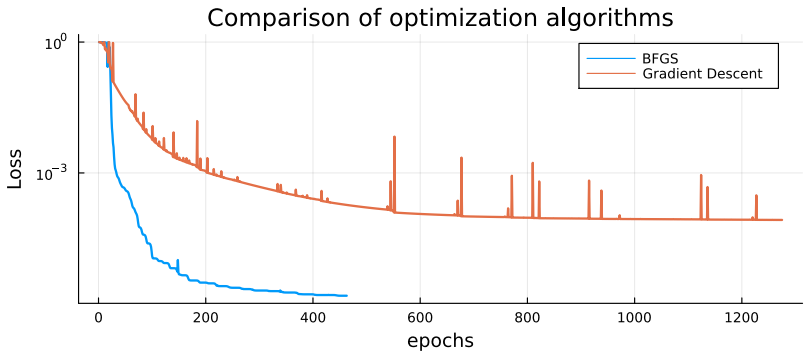


Figure 6.1: Loss history for some of the experiments described in sec. 8.5. Straight-out-of-the-box More-Thuente linesearch was used in both cases, and might bear some of the blame of the bad performance by GD since GD seemingly prefer different hyper-parameter settings.

The selection of the step length η remain a crucial part of any minimization algorithm. Thus, sophisticated schemes for estimating the optimal step length have been developed and the following will provide a brief look into this research topic.

6.2.3 Linesearch

Updating the parameters eg. as in equation 6.9 relies on two parts. An "update direction" which is based on derivative information and

a step length denoted by η. In general η is not constant, and the question arises of how to find the optimal step length given the search direction. This is the subject of linesearch which estimates the minimizer of:

$$\phi(\eta) = \mathcal{L}(\vec{p}_t + \eta S_t) \tag{6.11}$$

Where S_t is the search direction, which in the case of BFGS is $S_t = -B_t^{-1}\nabla\mathcal{L}(\vec{p}_t)$. The global minimizer of $\phi(\eta)$ is usually too costly to find and instead an inexact solution, which satisfy certain conditions, is estimated. For quasi-newton methods and BFGS in particular, certain conditions such as the curvature condition:[4]

$$(\vec{p}_{t+1} - \vec{p}_t)^T(\nabla\mathcal{L}(\vec{p}_{t+1}) - \nabla\mathcal{L}(\vec{p}_t)) > 0 \tag{6.12}$$

and the property that the gradients are sampled at points that allow the model to capture appropriate curvature information, is ensured by the Wolfe or strong Wolfe conditions [90]. The first condition is that the step length should ensure "sufficient decrease" in the objective function, stated as:

$$\phi(\eta) \leq \phi(0) + c_1\eta\phi'(0) \tag{6.13}$$

Where $c_1 \in [0, 1]$. Just imposing a decrease, ie. $c_1 = 0$ can lead to convergence only in the limit of $t \to \infty$ even for convex functions. Conversely $c_1 = 1$ would mean the function should be concave or at least have an even greater slope at a later point. Practical values for c_1 can be quite small, eg. $c_1 = 10^{-4}$. Even very small η will usually satisfy the sufficient decrease condition, so in order to ensure sufficient progress is made a second condition is imposed, namely the curvature condition:

$$\phi'(\eta) \geq c_2\phi'(0) \tag{6.14}$$

With $c_2 \in [c_1, 1]$, where for Newton or quasi-Newton methods a typical value is $c_2 = 0.9$.[5] The intuition behind this condition is that, since the slope $\phi'(0)$ is negative, the slope at η should be more flat. Because if it was not, it would imply further progress could be made by increasing η. A large positive slope at η would also satisfy this condition, so in order to ensure that η lies in the vicinity of a minimum an even stronger condition can be formulated:

$$|\phi'(\eta)| \geq c_2|\phi'(0)| \tag{6.15}$$

Thereby narrowing the region of acceptable η to relatively flat regions that simultaneously realize a sufficient decrease in the objective function. Equations 6.13 and 6.14 are the Wolfe conditions whereas 6.13 with 6.15 are the strong Wolfe conditions.

Sophisticated linesearch algorithms can be quite complicated and will not be explained in depth here. That being said, the general approach is to iteratively make new guesses for an appropriate η based on the information accumulated by evaluations of ϕ and its derivative. These pieces of information are used to create an interpolation which is analytically minimized. The procedure terminates once the imposed conditions are satisfied.

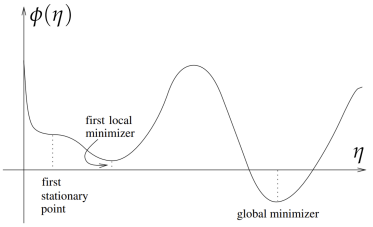


Figure 6.2: Example of generic linesearch problem defined by $\phi(\eta)$. Figure from [90]

[4] This is related to the positive definiteness of the approximate Hessian matrix and is different from the curvature condition of the Wolfe conditions below.

[5] This is the value that perhaps should be tuned differently for the GD algorithm in figure 6.1

The linesearch algorithm found to work the best for this thesis, and the one subsequently used, is the one by More and Thuente [91]. It satisfies the Strong Wolfe conditions and is accompanied by quite an extensive theoretical analysis.

With the loss landscape defined and the optimization techniques presented, it remain to be answered how to calculate the gradients of the loss function, necessary for the gradient based minimizers. In the following, the Machine Learning approach is presented in order to motivate the need for the alternative method actually employed for the results of the thesis.

6.3 Automatic differentiation

Automatic differentiation (AD) is the workhorse behind many of the impressive results coming out of Machine Learning. Below, it will quickly be fleshed out what it entails, what the shortcomings are and an example of an alternative that combats these shortcomings.

6.3.1 Forward and backward

The general idea behind AD is to leverage the chain-rule of differentiation, such that it is only necessary to define the derivative for simple mathematical operations and some constitutive functions from which most other expressions consists of. Then it is possible for the program to stitch together the wanted derivatives from the individual function evaluations and their accompanying derivatives.

Much of the following is based on a standard textbook on numerical optimization [90]. There exists two basic modes for stitching together the final derivative from the constituent functions, namely forward- and reverse mode. The names arise from the fact that for each computation there exists a computational graph (an example is visualized in figure 6.3) where reverse and forward refers to the way the algorithm traverse the computational graph when building the function derivative.

The **forward** mode propagates the derivative forward in the computational graph along with the calculation of the function. The idea is rather simple in that, since the value of an intermediate node in the computational graph is given as a function of all the parent nodes to that node:

$$w_i = w_i(\{w_j\}) \tag{6.16}$$

where the index j refer to the parents of the node i. The derivative of this node value with regards to say the first input x_1 is simply:

$$\frac{\partial w_i}{\partial x_1} = \sum_{j \text{ parents of } i} \frac{\partial w_i}{\partial w_j} \frac{\partial w_j}{\partial x_1} \tag{6.17}$$

This can be applied recurrently starting from $w_1 = x_1$ until $w_N = f(x_1)$ is reached and one has successfully calculated $\frac{\partial f}{\partial x_1}$. If f is multivariate this has to be done multiple times with a different version

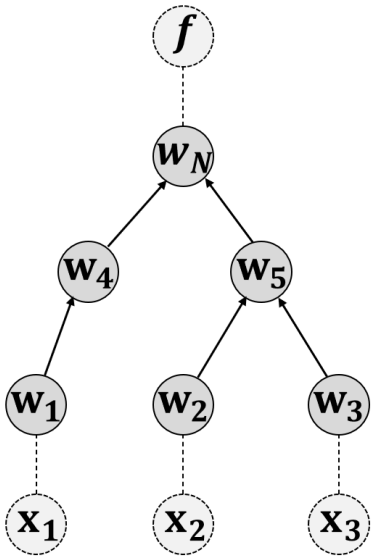


Figure 6.3: Example of a computational graph. The x_i are the input and f is the output. The w_i refer to intermediate stages of the calculation. Here w_4 is a child of w_1 and w_1 is the parent of w_4 and so on. In this context, the derivative of f wrt. x_i is desired.

of 6.17 for each x_i. This means that for $f : \mathbb{R}^n \to \mathbb{R}$ the computational overhead can be quite extensive compared to the reverse mode.

Reverse mode only forward propagates in order to store the node values. The derivative is then found, starting from the ultimate child node $\frac{\partial f}{\partial w_N} = 1$, by:

$$\frac{\partial f}{\partial w_j} = \sum_{i \text{ childs of } j} \frac{\partial f}{\partial w_i} \frac{\partial w_i}{\partial w_j} \tag{6.18}$$

until all the $\frac{\partial f}{\partial x_i}$ can be calculated at once. The downside is the storage of the node values, but this is manageable as is evident from the effectiveness of the back-propagation (reverse mode) algorithm in deep learning.

However, limitations still exists. In many cases such as using piecewise rational functions in place for trigonometric functions or when numerically approximating solutions to PDEs, a truncation error occurs: $\hat{f}(x) = f(x) + \tau(x)$. The truncation error $|\tau(x)|$ is usually small however its derivative need not be $\tau'(x)$ leading to a mismatch between the AD-derivative and the true derivative $\hat{f}'(x) \neq f'(x)$ Furthermore, the computational graph need to be properly defined meaning that slight care need to be made when implementing the function that needs to be differentiated.[6] So, quoting the conclusion from [90]

> In conclusion, automatic differentiation should be regarded as a set of increasingly sophisticated techniques that enhances optimization algorithms, allowing them to be applied successfully to many practical problems involving complicated functions. [...] Automatic differentiation should not be regarded as a panacea that absolves the user altogether from the responsibility of thinking about derivative calculations.

[6] Examples include omitting if statements and random-sampled values. In the latter case the reparameterization trick can be used as a solution [92].

6.3.2 The alternative

With respect to the concerns of this thesis, calculating derivatives for the result of a numerical solution to a differential equation, is the most important criteria. As it turns out, one can find differential equations for the derivatives, in at least two different forms that are analoguous to forward and reverse-mode. This results in no need for storage of function evaluations and computational graphs and it is possible to calculate the gradients to arbitrary accuracy (within bounds given by the low computational errors). These methods will be explored in section 7.

6.4 Pulse Parametrization

Pulse parametrization refers to how the pulse is expressed in terms of the parameters and how it is realized in the code. It is thus different from the physical realization of the pulse which is usually created by Arbitrary Waveform Generators [93]. The first distinction to be made is whether the pulse is in the discrete or continuous time

scheme. In the discrete case of section 5.2.1 the value of the pulse $u(\Delta t n) \to u_n$ is fixed during each timestep. Each u_n will then be the tuneable parameter. However, even this parametrization can be made continuous and it should be trivial to always revert back to discrete parametrization, so in the following only continuous parametrizations are considered.

The discrete parametrization is a piece-wise decomposition, formally written as:

$$u(t, \vec{p}) = \sum_n p_n \Theta(t - t_n)\Theta(t_{n+1} - t) \tag{6.19}$$

It is understood as a bar chart with the height defining the pulse-value during the corresponding time interval, as visualized in figure 6.4. It is therefore physically unrealizable but it has a trivial gradient wrt. $\vec{p}$.

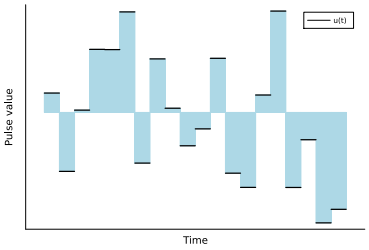


Figure 6.4: Example of using a piece-wise decomposition for pulse parametrization.

One way to accommodate the physical implementation of the signal in the simulation, is to transform the signal via the convolution [94]:

$$\tilde{u}(t) = \int_0^t h(t - \tau)u(\tau)d\tau \quad \text{with} \quad h(t) = \frac{e^{-t/t_r}}{t_r} \tag{6.20}$$

This results in an exponential rising edge from each pulse value to the next, where t_r determines the steepness of the rising edge. It would then be advantageous to fit this parameter to the given AWG used in the experiment, where one wants to implement the solutions found by simulations. In general it would be best to parametrize the signals such that they adhere to the experimental realizations, where it would even be possible to fit to the equpiment at hand. However, since this thesis was not coupled to an experiment, not much work was done in this regard. Nevertheless, the physical realizability was accommodated by only considering pulses that change at the time-scale set by the sampling rate of the AWG, which is on the order of 2.4 to 8.5GHz and seemingly beyond [95].

Before mentioning the two pulse parametrizations actually employed in the thesis. The parametrization of the other time-dependent scalar function, $\alpha(t, \vec{p})$, will be defined.

6.4.1 Parametrization of α

The general idea as explained in section 4.3.1, is to lower the "barrier" between the computational states and then apply the pulse. With $\alpha = 1$ the barrier is up and with $\alpha = 0.5$ the barrier is down, thus, unless else is stated, $\alpha(t, \vec{p})$ is defined as[7]:

$$\alpha(t, \vec{p}) = \begin{cases} 1 - (1 - \alpha_{\min})\frac{t}{T_a} & t \leq T_a \\ \alpha_{\min} & T_a \leq t \leq T - T_a \\ \alpha_{\min} + (1 - \alpha_{\min})\frac{t-(T-T_a)}{T_a} & T - T_a \leq t \end{cases} \tag{6.21}$$

That is, α is lowered linearly from $\alpha = 1$ to $\alpha = \alpha_{\min}$ in a time T_a, where it is left for a period of time in which the pulse is applied. The barrier is then raised linearly again, completely symmetric to the lowering.

[7] This is a rather simple definition and it was exactly chosen due to its simplicity. Other more complicated parametrizations are just as valid. It would be interesting to synthesize gates using only $\alpha(t)$, but that ended up outside the focus of this thesis.

T is the gate time and the pulse is applied in a time $T - 2T_a$. The parametrizations of the pulse, will now be defined.

6.4.2 DRAG-pulse

The simplest realizable pulse for quantum-gates is a cosine pulse as the one presented in section 3.3.2.

$$u(t, \vec{p}) = E(t) A \cos(\omega_d (t - T_a) + \varphi) \tag{6.22}$$

It is parametrized by an amplitude A, driving frequency ω_d and a phase φ. The envelope function $E(t)$ provides a physical ramp up and ramp down period towards the start and end of the pulse. The time scale of the ramp is given by T_r and the shape used throughout this thesis is $\propto \sin^2$, formally written as:

$$E(t) = \begin{cases} \sin^2\left(\frac{\pi}{2}\frac{t-T_a}{T_r}\right) & \text{if } T_a \leq t \leq T_a + T_r \\ 1 & \text{if } T_a + T_r \leq t \leq T_a + T_p + T_r \\ \sin^2\left(\frac{\pi}{2}\frac{T_a+T_p-t}{T_r}\right) & \text{if } T_a + T_p + T_r \leq t \leq T_a + T_p + 2T_r \\ 0 & \text{else} \end{cases} \tag{6.23}$$

Where the time the pulse is applied is broken into two terms $T - T_a = T_p + 2T_r$. This is only relevant for section 8.2

As noted in section 4.1.2 and 4.2 leakage out of the computational basis can occur. For the simple cosine pulse, this can be combated by adding a sine pulse with envelope $\dot{E}(t)$ and a scaling parameter λ as described in [96]. So for this thesis, a DRAG pulse refer to:

$$u(t, \vec{p}) = E(t) A \cos(\omega_d (t - T_a) + \varphi) + \lambda \dot{E}(t) A \sin(\omega_d (t - T_a) + \varphi) \tag{6.24}$$

With the parameters A, ω_d, φ, λ, T_a and T_r. The inclusion of T_r is debatable, however it is included in order to investigate how it will affect the minimization procedure. The total gate time T can also be included as a parameter as will be described in section 7.1.5, which is used to find a gate with minimized gate time as presented in sec. 8.2.

The DRAG pulse (eq. 6.24) and $\alpha(t)$ (eq. 6.21) are visualized in figure 6.5 along with some of the tuneable parameters.

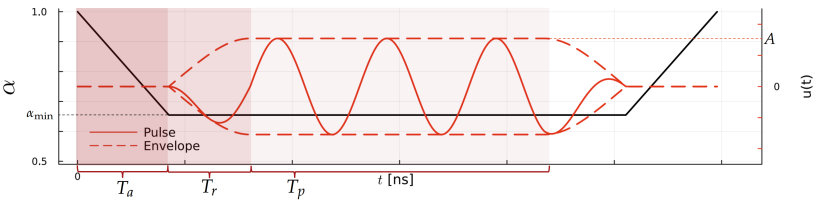


Figure 6.5: Visualization of the DRAG-pulse and $\alpha(t, \vec{p})$ parametrizations. The definition of some of the parameters is also shown. The envelope is sketched as $\pm A\sqrt{E^2 + (\lambda \dot{E})^2}$

This concludes the presentation of the simple parametrizations used in this thesis, which are parametrized by only a handful of parameters. In order to fully leverage the efficiency of the method to be developed in section 7, more complicated parametrizations which constitute the use of more parameters may be considered. This is done in the following.

6.4.3 Interpolation method

The advantage of the piece-wise decomposition is the flexibility to approximate a general function. The downside is the rather non-physical discrete nature of the parametrization. To accommodate a realistic implementation of the pulse, while keeping the flexibility of the parametrization, an extension of the piece-wise decomposition is considered, for which a continuous interpolation is made between each time-step.

Different methods exists for doing this. One way is the convolution method mentioned in the beginning of 6.4. However, the approach employed in this thesis is the following parametrization:

$$u(t, \vec{p}) = \frac{1}{\sum_k f(t - t_k)} \sum_i f(t - t_i) p_i \tag{6.25}$$

The function f is a 'weight' function that is peaked around zero. The idea is that, at an instant t, the pulse value is a weighted mean of the surrounding 'pillars' of the piece-wise decomposition, with decreasing weights the further the time step is from t. The weight function used is simply a gaussian:

$$f = e^{-x^2 \frac{\ln(1/r)}{\Delta t^2}} \tag{6.26}$$

where $r = f(\Delta t)$ is the hyper-parameter that describes how wide in time the weights reach, ie. it affects the steepness of the rising edge. The $\Delta t = t_{n+1} - t_n$ assumes a constant separation of time steps for simplicity. The sampling rate of the AWG determines how small Δt can be. An example for the value of r used in this thesis is shown in figure 6.6 and when r is small enough the piece-wise decomposition is retrieved as seen in figure 6.7.

The reason for denoting this method the interpolation method is that when r is small enough, eq. 6.25 with eq. 6.26 can be viewed as:

$$u(t_n + \delta t) \simeq \left(p_n e^{-\delta t^2 \frac{\ln(1/r)}{\Delta t^2}} + p_{n+1} e^{-(\delta t - \Delta t)^2 \frac{\ln(1/r)}{\Delta t^2}} \right) \tag{6.27}$$

with $0 < \delta t < \Delta t$ and where $N \simeq e^{-\delta t^2 \frac{\ln(1/r)}{\Delta t^2}} + e^{-(\delta t - \Delta t)^2 \frac{\ln(1/r)}{\Delta t^2}}$ is the normalization constant. So, in between time steps, the pulse value is an interpolation between the previous and the following parameter value, with a Gaussian rising edge in both ends.

The interpolation method was used as the most general parametrization in a continuous-time context, when it was desired to restrict the function space the least.

No constraints are put on the parameters wrt. the interpolation method, but for the DRAG-pulse and the α parametrization, certain constraints such as $0 \leq T_a \leq T/2$ and $\alpha_{\min} \in [0.5, 1]$ needed[8] to be enforced. How this was done is described in the following section.

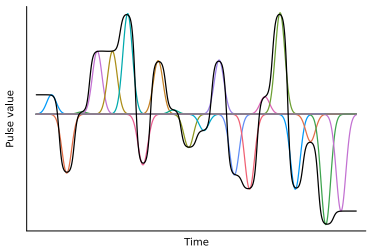


Figure 6.6: Example of using a weighted sum of Gaussian distributions for pulse parametrization, with $r = e^{-5} \simeq 0.67\%$. This represents the value of r used in the results section (sec. 8)

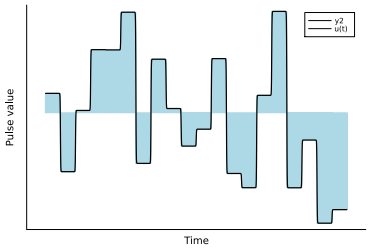


Figure 6.7: Example of using a weighted sum of Gaussian distributions for pulse parametrization, with $r = e^{-50} \simeq 2 \cdot 10^{-22}$

6.4.4 Limiting functions

Often a parameter is bounded and needs to somehow be restricted to lie within some interval. There are different ways to go about this,

[8] The $\alpha_{\min} \in [0.5, 1]$ is actually an artificial constraint, which was not posed by physical considerations. It is enforced in order to keep the interpretability as a lowering and raising of the potential barrier simple. Nonetheless, it was encoded as a hard constraint.

the easiest and often best of which, is to let the minimizer regard the parameter as unbounded, but then always pre-apply a limiting function to the parameter within the function calls. An example that is used primarily in this thesis is:

$$\sigma(\tilde{p}|p_{\min}, p_{\max}) = \frac{(p_{\max} - p_{\min})}{1 + e^{-\tilde{p}}} + p_{\min} \tag{6.28}$$

with the inverse[9]:

$$\sigma^{-1}(p|p_{\min}, p_{\max}) = -\ln\left(\frac{p_{\max} - p_{\min}}{p - p_{\min}} + 1\right) \tag{6.29}$$

An example is the limiting of $\alpha_{\min} \in [0.5, 1]$ in eq. 6.21 used throughout this thesis, ie. $\alpha_{\min} = \sigma(\tilde{\alpha}_{\min}|0.5, 1)$ where $\tilde{\alpha}_{\min} \in (-\infty, \infty)$ is the parameter contained in $\vec{p}$ that a minimizer can freely adjust. For binding a parameter to the positive real axis, the simple choice

$$p = \tilde{p}^2 \tag{6.30}$$

is used for this thesis.

This concludes the necessary considerations that need to be made when implementing a parametrization of a continuous scalar function wrt. this thesis.

The above sections constitute the necessary considerations for treating an optimization problem and with all of the above it is now possible to tackle a plethora of Quantum Optimal Control (QOC) problems. However, a subset of QOC problems remain unmentioned, namely those QOC problems that factor in model uncertainty, which may promote solutions that are insensitive to uncertainty in the model parameters. In this thesis, these problems are encompassed by the term Robust Optimal Control and the formulation of Robust Optimal Control problems will be formalized in the following section.

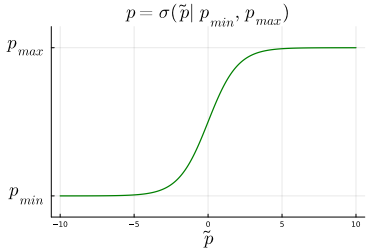


Figure 6.8: Visualization of the limiting function defined by eq. 6.28

[9] The inverse is used for initialization of the constrained parameter values.

6.5 Robust Optimal Control

Practical implementations of exact solutions will always be impossible to obtain. There will be noise in the system that can possibly create a large discrepancy between the actual evolution and the desired evolution.

In this section we will consider how to deal with the fact that system and control parameters can not be known to infinite precision.

6.5.1 Problem formulation

In optimal control the problem consists of minimizing a cost function while given the constraints set by the system dynamics. We will now consider the presence of an uncertainty in some of the parameters of the dynamics. Let p be the control parameters, then a denote the parameters with uncertainty. The sets of a and p may overlap entirely, only partially or not at all. Robust optimal control is then concerned

with optimizing the cost function wrt. p over some region in a, such as to promote solutions that are insensitive to the fluctuations in a.[10]

However, this region in a of the loss landscape still needs to be boiled down to a single scalar. That is, the distribution of the value of the loss in this region, needs to be summarized by summary statistics and combined into a single number.

Formally, let a follow some probability density function, $a \sim P(a|p)$. In general, the PDF can depend on the parameters p, eg. if some external magnetic field which is used for control increases in value, the variations in some otherwise ideally fixed external flux parameter could increase.

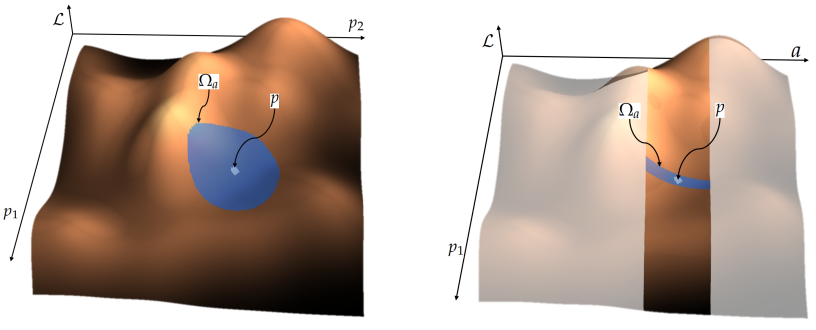
For simplicity however, consider $P(a|p)$ being a uniform distribution $P(a|p) = U(\Omega_a)$ where Ω_a describes the region for the possible values of the uncertain parameters, which may still depend on p if a and p share variables, but this complication is also disregarded in the following. Then the loss would depend on a stochastic variable $a \sim U(\Omega_a)$ and in order to have a well defined objective function it would need to be reduced to summary statistics. For this, one can invoke the law of the unconscious statistician[11] which states that the expectation value of some function of the stochastic variable a is:

$$\tilde{g}(p) = \mathbb{E}\left[g(a)\right] = \int g(a)P(a|p)da \tag{6.31}$$

with the understanding that a is a collection of variables to be integrated over, ie. da is a volume element in the space that the vector a lives in. With the above simplifications it is written as:

$$\mathbb{E}\left[g(a)\right] = \frac{1}{|\Omega_a|}\int_{\Omega_a} g(a)da \tag{6.32}$$

where $1/|\Omega_a|$ is the normalization constant of the uniform distribution. It is worthwhile to decipher what is meant by this expression with two examples. In the first example of figure 6.9 (a), the loss



(a) (b)

Figure 6.9: Visualization of a loss landscape and the integration region Ω_a of eq. 6.32 for the two different cases of $a = p$ and $a \cap p = \varnothing$, shown in (a) and (b) respectively. A simple Robust QOC problem can be posed by seeking to find the p that achieve the flattest and lowest lying blue region, which would constitute a low value for the loss function, that does not change much even when the uncertain parameters of the model fluctuate.

landscape of an objective function with two tuneable control parameters is visualized. In this specific example $a = p$ and the blue region denoted Ω_a is the integration region for the expectation value in eq. 6.32. The p of the figure refer to the parameter values assumed at the given step of the minimization procedure, so in the case of $a = p$ the integration region Ω_a moves about when the control parameters assume different values. In the opposite case of fig. 6.9 (b) where

[10] Ie. p could consist of the amplitude and driving frequency of eq. 6.22, while a may just contain ϕ_{ext} of eq. 4.23, if there is noise in the external flux. In this case there is said to be no overlap between p and a. If one also consider noise in the drive line and thus uncertainty in the control parameter related to the amplitude of the pulse, p and a share a parameter and would be said to have an overlap.

[11] It gets its name from the fact that most would not stop to consider whether the relation is true but would naively just apply it, since it is often described as the definition of the expectation value.

there is only one tuneable parameter and one uncertain parameter, which is not the same (ie. $a \cap p = \varnothing$), the integration region remain the same and is independent of the value of p. In this example the point p can only move along the 1-dimensional manifold defined by p_1 and insensitivity of the loss function is only encouraged in the orthogonal direction given by a.

An example of a meaningful objective function that promotes robustness could consist of a weighted sum of two terms. One term that seeks to minimize the expectation value of the loss:

$$\tilde{\mathcal{L}}(p) = \mathbb{E}\left[\mathcal{L}(p,a)\right] \tag{6.33}$$

and a second term that seeks to minimize deviations from this expectation value, which would mean a flat, and thus robust, loss landscape:

$$\mathrm{Var}(\mathcal{L}(p,a)) = \mathbb{E}\left[\left(\mathcal{L}(p,a) - \tilde{\mathcal{L}}(p)\right)^2\right] \tag{6.34}$$

It's usually not feasible to find the exact mean and variance, so these are often approximated by sampling values of the loss within this region:

$$\tilde{\mathcal{L}}(p) \approx \frac{1}{N}\sum_i^N \mathcal{L}(p,a_i) \tag{6.35}$$

$$\mathrm{Var}(\mathcal{L}(p,a)) \approx \frac{1}{N-1}\sum_i^N (\mathcal{L}(p,a_i) - \tilde{\mathcal{L}}(p))^2 \tag{6.36}$$

However, it scales poorly with the dimensionality of a since the density of sampled points decreases exponentially with the dimensionality. This is encompassed by the common term, the curse of dimensionality. This leads to a trade-off between run-time and accuracy of the method, depending on the size of N.

Another choice for a summary characteristic could be the maximum of the distribution:

$$\tilde{\mathcal{L}}(p) = \max_{a\in\Omega_a} \mathcal{L}(p,a) \tag{6.37}$$

Finding the maximum is in principle just as difficult as finding the minimum (even though the former is bounded to a smaller region). However, one can also approximate this approach which is referred to as linearization of the worst-case scenario [97][98]. This scheme assumes that the region in question is appropriately small, such that the loss function has little to no curvature within that region. Then an expansion in the deviation from the center of the region a_0 yields:

$$\max_{a\in\Omega_a} \mathcal{L}(p, a_0 + (a-a_0)) \simeq \mathcal{L}(p,a_0) + \max_{a\in\Omega_a}\left[(\nabla_a \mathcal{L}(p,a_0))^T (a-a_0)\right] \tag{6.38}$$

However, maximizing this is not quite straight forward, since it depends on how the region Ω_a is defined.

There are two trivial cases. If the limits of the parameters in a are independent of each other, the boundary is a box, ie. $(a_i - (a_0)_i) = \delta a_i \in [-m_i, m_i]$. Then the maximum is reached when:

$$\delta a_i = \mathrm{sign}(\partial_{a_i}\mathcal{L}(p,a_0))m_i \tag{6.39}$$

That is the vector δa goes to the corner of the box, that is nearest to the direction of the gradient (see figure 6.10). The other easy case is when the boundary is a sphere of radius m. Then the dot product is maximized when δa lies parallel to the gradient and assumes it's maximal length m:

$$\delta a_i = m \frac{\partial_{a_i} \mathcal{L}(p, a_0)}{|\partial_a \mathcal{L}(p, a_0)|} \tag{6.40}$$

These results can be generalized to any interpolation between the box and the sphere, that is, when the boundary is defined as:[12]

$$\sum_i \left(\frac{\delta a_i}{m_i} \right)^{2n} - 1 = 0 \tag{6.41}$$

where m_i decides the elongation along the corresponding axis, and n decides "how sharp the corners are". Then the optimum is found at :

$$\delta a_i = \text{sign}(\partial_{a_i} \mathcal{L}) \frac{m_i^{\frac{2n}{2n-1}} |\partial_{a_i} \mathcal{L}|^{\frac{1}{2n-1}}}{\left(\sum_j \left(m_j |\partial_{a_j} \mathcal{L}| \right)^{\frac{2n}{2n-1}} \right)^{\frac{1}{2n}}} \tag{6.42}$$

An example is visualized in figure 6.10.

It has now been described how it is possible to construct optimization problems that produces robust solutions by using summary statistics of the loss function such as mean, variance and maximum value. Focusing on the linearization of the worst-case scenario approach, it is instructive to discuss the effects of whether the sets of a and p overlap. This is done in the following section.

[12] The box is realized in the limit of $n \to \infty$ and the sphere is realized when $n = 1$ and $m_i = m$

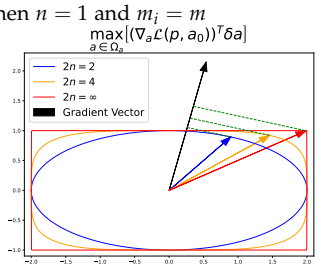


Figure 6.10: Examples of maximization of the dot product in eq. 6.38 for integration regions Ω_a defined by eq. 6.41. A two dimensional a is considered with the "elongations" given as $m_x = 2$ and $m_y = 1$. The vectors are found by eq. 6.42 for different values of n in eq. 6.41

6.5.2 Environment variables and control parameters

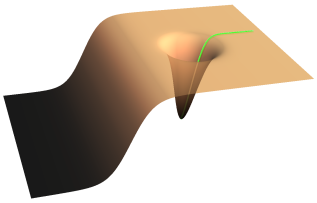
As previously stated, a denote the noisy parameters, whereas p denote the parameters that are to be tuned until an optimum is found. We will now examine what happens to the approach defined by eq. 6.38 in the two cases when $a = p$ and when $a \cap p = \varnothing$ (ie. they have no overlap whatsoever), in two easy to visualize examples.

First we examine $a = p$. In order to get a sense of how this augments the loss landscape, consider the problem where there are only two parameters, and that Ω_a is a circle with radius m (much like figure 6.9 (a)). Then with the above linearization of the worst-case scenario, the relevant loss function is:

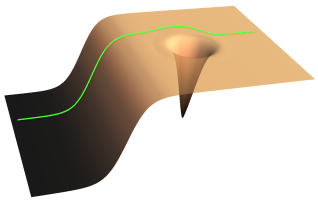
$$\tilde{\mathcal{L}}(p) = \mathcal{L}(p) + m |\nabla_p \mathcal{L}(p)| \tag{6.43}$$

A benign example is shown in figure 6.11. It is constructed by considering a loss landscape that consists of a slightly tilting plane with a local minimum that has a ramp down to a large global minimum.

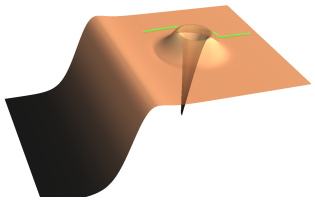
An optimization procedure is run on this loss landscape in order to see if the robustness considerations can steer the minimizer around the local minimum and into the global minimum. The optimization procedure is initialized such that the gradient head towards the local minimum and the algorithm employed is Gradient Descent with decaying momentum.



(a) $m = 0$ ie. no robustness considerations. The trajectory ends up in a local minima.



(b) m adequate. Steers the trajectory around the local minima and into the robust region.



(c) m too large. For this simple gradient descent algorithm, the trajectory is confined to a bad minima.

Figure 6.11: Constructed example of how different values of m affect the loss landscape and the minimization procedure. The trajectory of the minimizer is shown in green. The optimization algorithm consisted of Gradient Descent with decaying momentum (see sec. 6.2.1)

In fig. 6.11 (a) the landscape and the trajectory of the minimizer is visualized. With no robustness consideration and due to the initialization point, the minimizer end up in the local minimum.

In (b) and (c) different values of m in eq. 6.43 are considered and it is apparent how the regions with a slope are heightened. When m is of adequate size, a barrier is raised around the local minima and the minimizer is guided around it and out into the global minimum region.

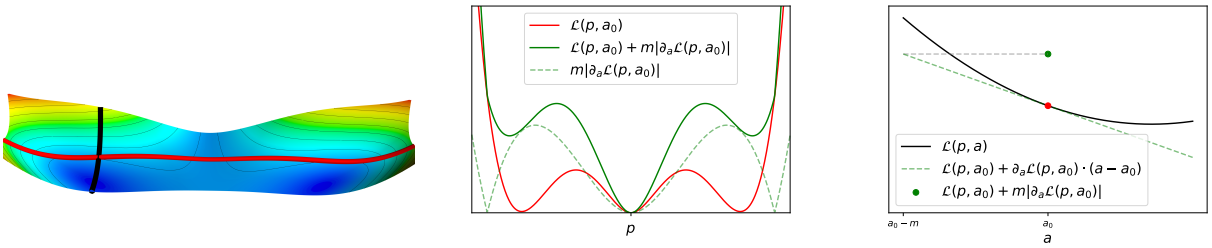
Conversely, when m is too large, the minimizer stagnates, since it is, by construction, trapped in a cauldron. This is because any region with slope is increased in loss value, rendering gradient-based optimization techniques useless.

This is a very constructed example, and it just goes to show that, so far, the linearization of the worst-case scenario approach does not seem like a promising avenue for producing robustness against fluctuations in the control parameters.

Consider now instead the case of a being a single parameter different from p which is also one-dimensional (much like the case of fig. 6.9 (b)). The loss landscape is still two-dimensional, but now it is only possible for the minimizer to move along the p-direction. We seek a solution that, in addition to being a minima, is also flat in the a-direction. Figure 6.12 (a) visualize such an example, with the red line running along p and the black line being parallel to the a-axis. The 1-dimensional manifold of these lines are plotted in (b) and (c) respectively. The setup is constructed such that there are 3 minima but two of them have some unwanted curvature, with regards to a, associated to them. Fig. 6.12 (c) shows how the gradient information is added to the loss, and fig. 6.12 (b) shows the original and the augmented loss.

In this example, it is apparent that the two minima remain minima but the flat minimum is promoted to a global minimum. So in the case of $a \cap p = \varnothing$ the linearization of the worst-case scenario approach seem useful in helping sophisticated gradient-based optimization techniques to find a robust solution. This will be investigated for a specific example in section 8.6, where it is also compared to the mean-value approach defined by equation 6.35. Therefore, it is helpful to also sketch this approach in the case of figure 6.12 which is done in the following.

The mean-value approach consists of calculating the loss at differ-



(a) (b) (c)

Figure 6.12: (a) shows the loss landscape of a Robust QOC problem with one tuneable parameter p that can take on values indicated by the red line and one noisy parameter a that fluctuate along the black line. Therefore it is desired to find a point on the red line that is simultaneously low-lying and flat in the a direction, which in this case would be the middle region. The one dimensional manifolds are shown in (b) and (c) with (b) visualizing the augmented loss function and (c) how this augmentation can be interpreted.

ent points along a and taking the mean value as the objective function. In the context of fig. 6.12 (c) let N points $a_i \in [a_0 - m, a_0 + m]$ be equidistantly distributed along the a-axis. The loss is then evaluated at each a_i and the mean is taken, such that the augmented loss becomes:

$$\tilde{\mathcal{L}}(p) = \frac{1}{N} \sum_i \mathcal{L}(p, a_i) \tag{6.44}$$

For a one-dimensional a this approach is feasible and employ more information than the previous method, which only used derivative information. A comparison of the results achieved by the two methods is done in section 8.6.

This concludes the robustness considerations relevant for this thesis, but for a discussion of additional methods, see the appendix section 10.3.3.

The time has finally come for a presentation of the adjoint method that has been mentioned and motivated for earlier in the thesis. It is the employed method for calculating derivatives of the objective functions encountered in this thesis and with objective functions akin to the one in eq. 6.43, a second order adjoint method will also be derived. All the derivations and discussion of the adjoint method are contained in the following chapter.

7 Adjoint Method

It has by now been made clear what Quantum Optimal Control is and what the challenges are when simulating quantum systems. In order to achieve results that will transfer from simulations to real world experiments, precise simulations are necessary[1]. This contain two aspects. For one, sufficiently large Hilbert space representations are necessary, in order to have a theoretical foundation for a precise description. Next, a precise numerical solver of the dynamics imply a method that can estimate and contain its errors. This usually means a lot of evaluations of the Schrödinger equation at different times.

On top of this, it is desired to find the sensitivity of the dynamics wrt. the parameters, such that gradient-based techniques can be used to find estimate solutions to the control problems. In view of the presentation of Automatic Differentiation in section 6.3, the otherwise widely used Backpropagation algorithm is not the optimal approach in this regard. Instead the adjoint method, which will be derived below, is much better suited for this. This is in part due to its efficiency and continuous time description as opposed to the discrete nature of the backpropagation algorithm.

The adjoint method is also tightly linked to the established theory of optimal control and is essentially the basis for standard Quantum Optimal Control algorithms such as GRAPE [99].[2] For further discussion see appendix 10.3.1.

What is meant by the adjoint method will now be laid out in a generic and continuous time context. But first, a clarification is provided, regarding the use of notation for matrix calculus.

[1] The term used here only includes solving the Schrödinger equation, and is therefore not sufficient for the "transferability". Considerations also need to be made towards model correctness and open system effects.

[2] However, the GRAPE-algorithm is inherently discrete in time and an approximation to first order in Δt is made during the derivation. Extensions have been made to accomodate this [100], but we will focus now on an exact continuous time description.

7.0.1 Preemptive note on matrix-calculus

This thesis assumes the "numerator layout" convention such that differentiation wrt. a vector a with dimensions $N \times 1$ is to be understood as:

$$\frac{d}{da}\begin{pmatrix} b_1 \\ \vdots \\ b_M \end{pmatrix} = \begin{pmatrix} \frac{db_1}{da_1} & \cdots & \frac{db_1}{da_N} \\ \vdots & \ddots & \\ \frac{db_M}{da_1} & & \frac{db_M}{da_N} \end{pmatrix} \tag{7.1}$$

ie. differentiation of a column vector by a column vector yields a matrix. In the same vein, differentiation of a scalar L by a column vector yields a row vector.

$$\frac{dL}{da} = \begin{pmatrix} \frac{dL}{da_1} & \cdots & \frac{dL}{da_N} \end{pmatrix} \tag{7.2}$$

This also means that the chain-rule is still given as: [3]

$$\frac{df(b(a))}{da} = \frac{\partial f}{\partial b}\frac{\partial b}{\partial a} \tag{7.3}$$

Derivatives and partial derivatives with respect to a vector (or scalar) a is denoted $d_a \equiv \frac{d}{da}$ and $\partial_a \equiv \frac{\partial}{\partial a}$ throughout the following discussion. Furthmore, the product rule applies as:

$$\frac{\partial u^T(x)v(x)}{\partial x} = u^T\frac{\partial v}{\partial x} + v^T\frac{\partial u}{\partial x} \tag{7.4}$$

However, in the following, complex valued entities are used, and since the complex conjugate is a different variable, ie. $\frac{\partial x^*}{\partial x} = 0$, the relevant relation is:

$$\frac{\partial u^\dagger(x)v(x)}{\partial x} = u^\dagger\frac{\partial v}{\partial x} \tag{7.5}$$

The seemingly missing term from the new product rule is instead introduced from $x \to \begin{pmatrix} x \\ x^* \end{pmatrix}$ but x^* will be trivially given from x so this only needs to be accounted for in the end of the derivation.

[3] eg. if f has dimensions $K \times 1$, the resulting dimensions are $L \times N = (K \times M)(M \times N)$

7.1 Derivation of the adjoint method

Following the derivation in the Stanford adjoint tutorial [101], the adjoint method will now be outlined.
The relevant objects are defined by the following:

- x is our state variable, ie. a vector of variables that each follow their own differential equation. For complex state variables it should be noted that x contain both x_i and x_i^*.[4]
- p are the parameters of the problem and therefor the variables with respect to which we want to minimize some objective function.
- $F(x, p)$ is the objective function or functional.
- $\tilde{h}(x, \dot{x}, p, t) = 0$ defines the dynamics, through the accompanying first order differential equation. As it stands it is written in generality, but for the purpose of clarity we will assume that: $\tilde{h}(x, \dot{x}, p, t) = \dot{x} - h(x, p, t) = 0$ throughout the following.[5]
- $\tilde{g}(x(0), p) = 0$ defines the initial conditions, which may also depend on the parameters, and again we assume that it takes the simple form: $\tilde{g}(x(0), p) = x(0) - g(p) = 0$

[4] In the context of Quantum Mechanics x should be regarded the vector representation of $|\psi\rangle$

[5] Again, h should be regarded as the matrix representation of $-i\hbar\mathcal{H}|\psi\rangle$ in the context of Quantum Mechanics, ie. the Schrödinger equation.

The discussion is structured as follows. First, the adjoint method is derived in the simplest case, which is when the objective function F is only dependent on the final state $x(T)$ and possibly the parameters p. The general approach of implementing the adjoint method is outlined and motivated for and then an example relevant for the thesis is considered. Next the derivation is extended to include objective functions that contain a functional term, ie. of the form $\int f(x(t), p)dt$. Lastly, a derivation with the final time T included in the parameters is considered and a closed form expression of the "adjoint state" is found.

7.1.1 The simplest case

The derivation relies on the idea of Lagrange multipliers and by choosing constraints for these introduced lagrange multipliers it's possible to retrieve a useful expression for the gradient of the loss. First define the full loss that includes the constraints defined by the dynamics and initial conditions h and g:

$$\mathcal{L} = L(x(T), p) + \int_0^T \lambda^\dagger(\dot{x} - h(x, p, t))dt + \mu^\dagger(x(0) - g(p)) \tag{7.6}$$

Then it is of interest to find the gradient:

$$d_p\mathcal{L} = \partial_x L d_p x(T) + \partial_p L + \int_0^T \lambda^\dagger(d_p\dot{x} - \partial_x h d_p x - \partial_p h)dt + \mu^\dagger(d_p x(0) - \partial_p g) \tag{7.7}$$

The problem with this equation is that it is hard to evaluate[6] $d_p x$ so it would be advantageous if these could somehow be eliminated, and that is exactly what is possible by chossing suitable relations for the adjoint state variables $\lambda^\dagger$ and $\mu^\dagger$. Firstly the term with $d_p\dot{x}$ is partially integrated:

$$\int_0^T \lambda^\dagger d_p\dot{x}dt = \lambda^\dagger d_p x|_0^T - \int_0^T \dot{\lambda}^\dagger d_p x dt \tag{7.8}$$

Inserting this into eq. 7.7 and collecting terms with respect to the $d_p x$:

$$\begin{aligned} d_p\mathcal{L} =& (\partial_x L + \lambda^\dagger)|_T d_p x(T) \\ &+ (\mu^\dagger - \lambda^\dagger)|_0 d_p x(0) \\ &+ \int_0^T (-\lambda^\dagger \partial_x h - \dot{\lambda}^\dagger) d_p x dt \\ &+ \partial_p L - \int_0^T \lambda^\dagger \partial_p h dt - \mu^\dagger \partial_p g \end{aligned}$$

It's now possible to choose the following useful conditions:

$$\lambda^\dagger(T) = -\partial_x L(x(T), p) \tag{7.9}$$
$$\mu^\dagger = \lambda^\dagger(0) \tag{7.10}$$
$$\dot{\lambda}^\dagger = -\lambda^\dagger \partial_x h \tag{7.11}$$

Because then the gradient becomes:

$$d_p\mathcal{L} = \partial_p L - \int_0^T \lambda^\dagger \partial_p h dt - \lambda^\dagger(0)\partial_p g \tag{7.12}$$

In order to see how this works in practice, let us consider a simple situation, wherein neither the loss or the initial conditions depend explicitly on the parameters p, such that:

$$\partial_p L = 0$$
$$\partial_p g = 0$$

Then the gradient is determined by

$$d_p\mathcal{L} = -\int_0^T \lambda^\dagger \partial_p h dt \equiv \nabla(0) \tag{7.13}$$

where the integral is written as a time-dependent entity[7]:

[6] This is not entirely true, as discussed in sec. 7.1.2

[7] The symbol for nabla is chosen to invoke the idea that it is a gradient that is being calculated.

$$\nabla(t) = \int_T^t \lambda^\dagger \partial_p h dt \tag{7.14}$$

such that the integral is a solution to the final value problem:

$$\dot{\nabla} = \lambda^\dagger \partial_p h \quad \text{with} \quad \nabla(T) = 0 \tag{7.15}$$

In order to solve for $\nabla(0)$ it is necessary to know $\lambda^\dagger(t)$ and $x(t)$ at each point in time. The adjoint state $\lambda^\dagger$ obey the final value problem:

$$\dot{\lambda}^\dagger = -\lambda^\dagger \partial_x h \quad \text{with} \quad \lambda^\dagger(T) = -\partial_x L(x(T)) \tag{7.16}$$

and the state x of course obey the initial value problem:

$$\dot{x} = h \quad \text{with} \quad x(0) = g \tag{7.17}$$

One way to solve 7.15 be to store a dense output of the $x(t)$ and $\lambda^\dagger(t)$ solutions and use these to evaluate 7.14. However, this would diminish the ability of the numerical solver to adaptively increase the precision by increasing the discretization, which would already be set in stone. Additionally it would not be memory efficient.

Instead, the x, $\lambda^\dagger$ and ∇ are propagated backwards in time simultaneously, giving rise to $\mathcal{O}(1)$ memory cost and allowing the sophisticated numerical solvers to keep the precision to a decided tolerance.

So in practice the calculation of the gradient looks like the following:

- Use an ODEsolver to find $x(T)$ from $x(0)$ using $\dot{x} = h$
- Initiate the adjoint state by the final value condition: $\lambda^\dagger(T) = -\frac{\partial L(x(T))}{\partial x(T)}$
- Initiate the gradient calculation by $\nabla(T) = 0$ (In a machine learning context this is analogous to resetting the gradient before backwards propagation)
- Define an "augmented" state which is a concatenation of $x(t)$, $\lambda^\dagger(t)$ and $\nabla(t)$. The dynamics of the augmented state are given by the above differential equations and can be written as:

$$d_t \begin{bmatrix} x(t) \\ \lambda^\dagger(t) \\ \nabla(t) \end{bmatrix} = \begin{bmatrix} h(x,p,t) \\ -\lambda^\dagger(t)\partial_x h(x,p,t) \\ \lambda^\dagger(t)\partial_p h(x,p,t) \end{bmatrix} \quad \text{with} \quad \begin{bmatrix} x(T) \\ \lambda^\dagger(T) \\ \nabla(T) \end{bmatrix} = \begin{bmatrix} x(T) \\ -\partial_x L(x(T)) \\ 0 \end{bmatrix} \tag{7.18}$$

- This augmented state is thus specified at $t = T$ and can be evolved backwards in time from $t = T$ to $t = 0$ using an ODEsolver.
- The gradient can then be read-off as: $d_p \mathcal{L} = \nabla(0)$

In view of section 6.3 on automatic differentiation, the adjoint method is thus analogous to the backpropagation algorithm, in the sense that "a forwards and a backwards call" is made. Ie. the system dynamics are evolved forwards in time, and the gradient is found by evolving the augmented state backwards in time. As it turns out, there is also a continuous time method analogous to the forward automatic differentiation method, which will now be discussed.

7.1.2 Motivation for the adjoint method:

The logic behind the above derivation was to eliminate the expressions that include $d_p x$ for the gradient of the loss. However, this is not strictly necessary, since $d_p x$ is given by:[8]

$$\dot{x} = h(x, p, t) \qquad \text{with} \quad x(0) = g(p) \tag{7.19}$$

$$d_p \dot{x} = \partial_x h d_p x + \partial_p h \qquad \text{with} \quad d_p x(0) = \partial_p g(p) \tag{7.20}$$

[8] For an example in the context of quantum optimal control, consider equation 10.15

These two equations can be simultaneously evolved forwards in time, such that the gradient of the objective function is given by the solution to:

$$d_p L(x(T), p) = \partial_x L d_p x(T) + \partial_p L \tag{7.21}$$

which is readily evaluated once $x(T)$ and $d_p x(T)$ is found.

However, this method does not scale as well in complexity wrt. the number of parameters as the adjoint method. This "forward sensitivity" method only requires one forward integration but of the following number of equations:

$$\text{Forward:} \quad N_x \times 1 + N_x \times N_p = N_x \times (N_p + 1) \tag{7.22}$$

Whereas the adjoint method requires one forward evolution and a backward evolution of:

$$\text{Forward:} \quad N_x \times 1 \tag{7.23}$$

$$\text{Backward:} \quad N_x \times 1 + 1 \times N_x + 1 \times N_p = 2N_x + N_p \tag{7.24}$$

ie. the adjoint method scales better with regards to the number of parameters, especially for large state vectors.

Furthermore, as [102] puts it, the setting of the adjoint state method offers mathematical flexiblity not realizable in conventional approaches, which will become clearer in the following sections. This does however, come at the cost of a mathematical and implementational overhead, but this thesis seeks to be an introductory guide to the adjoint method in a QOC setting, such that this overhead is diminished.

So in order to put the adjoint method in the context of Quantum Optimal Control, the following example will now be considered.

7.1.3 Example for the simplest case

Consider a Hamiltonian with two time-dependent scalar functions:

$$\mathcal{H}(p, t) = H_0 + H_\alpha \alpha(p, t) + V u(p, t) \tag{7.25}$$

where

$$\mathcal{H}(p, 0) = \mathcal{H}(p, T) = H_0 + H_\alpha \tag{7.26}$$

and

$$(H_0 + H_\alpha)\psi_i = E_i \psi_i \tag{7.27}$$

An objective function could then be to transfer the ground state to the first excited state:

$$L(x(T)) = 1 - x(T)^\dagger \psi_1 \psi_1^\dagger x(T) \tag{7.28}$$

with $x(t)$ obeying the Schrödinger equation:

$$\dot{x}(t) = -i\mathcal{H}(p,t)x(t) \qquad x(0) = \psi_0 \tag{7.29}$$

The final value problem for the adjoint state is then:

$$\dot{\lambda}^\dagger = i\lambda^\dagger\mathcal{H} \quad \text{with} \quad \lambda^\dagger(T) = x(T)^\dagger\psi_1\psi_1^\dagger \tag{7.30}$$

Since the domain is complex the state variable x consists of a concatenation of ψ and ψ^* which would also mean complimentary dynamics and adjoint state for the complex conjugate. However as mentioned, it is not necessary to explicitly keep track of this complication, since one is trivially given from the other.

Instead it will be enough to consider the gradient given as:

$$d_p\mathcal{L} = -\int_0^T \begin{pmatrix}\lambda^\dagger(t) & \text{CC.}\end{pmatrix} \partial_p \begin{pmatrix}-i\mathcal{H}(p,t)x(t) \\ \text{CC.}\end{pmatrix} dt \tag{7.31}$$

$$= -\int_0^T [2\mathcal{I}m(\lambda^\dagger H_\alpha x)\partial_p\alpha + 2\mathcal{I}m(\lambda^\dagger V x)\partial_p u]dt \tag{7.32}$$

Which also explains how the gradient wrt. real parameters of a real scalar end up being real as it should be.

By now it should be apparent how to evaluate this integral by simultaneously simulating the state and the adjoint state dynamics. So far, this example only considers an objective function which is dependent on the dynamics of one initial value, but this is not sufficient for qubit gate synthesis problem formulations, since both the dynamics of ψ_0 and ψ_1 need be considered. However, this can easily be accommodated by concatenating the initial value problems together, such that the state vector becomes an entity with dimensions $\mathbb{C}^{n\times 2}$ and initial value:

$$x(0) = \begin{bmatrix}\psi_0 & \psi_1\end{bmatrix} \tag{7.33}$$

and in the case of the objective function eq. 6.2 the adjoint state also has dimensions $\mathbb{C}^{n\times 2}$ with final value:

$$\lambda^\dagger(T) = \frac{1}{2}\begin{bmatrix}x_0^\dagger(T)\psi_1\psi_1^\dagger & x_1^\dagger(T)\psi_0\psi_0^\dagger\end{bmatrix} \tag{7.34}$$

The rest of the equations remain the same, except now, that the terms in 7.31 are simply modified by:

$$\lambda^\dagger\mathcal{O}x \to \text{Tr}\left(\lambda^\dagger\mathcal{O}x\right) \tag{7.35}$$

This consideration of including multiple initial values, together with the realization of a real-valued gradient, concludes the implementation techniques necessary for practical use of the adjoint method in a QOC context.

We will now build upon this base understanding by considering two extensions. Firstly, we consider an objective function which in addition to the term dependent on $x(T)$ also contain a functional term dependent on $x(t)$. Next, the inclusion of the gate-time T as a parameter is treated and lastly a closed form of the adjoint state is derived.

7.1.4 The general case

Consider now an objective function that is both dependent on the state vector at the final time but also at all the intermediate times. That is, consider the addition of a functional to the case in section 7.1.1, namely:

$$F(x,p) = L(x(T),p) + \int_0^T f(x(t),p)dt \tag{7.36}$$

The procedure is exactly the same, so writing up the objective function with the constraints yields the lagrangian:

$$\mathcal{L} = L(x(T),p) + \int_0^T (f(x(t),p) + \lambda^\dagger(\dot{x}-h))dt + \mu^\dagger(x(0)-g) \tag{7.37}$$

One is interested in the derivative, which is:

$$\begin{aligned} d_p\mathcal{L} = &\int_0^T (\partial_x f - \lambda^\dagger \partial_x h - \dot{\lambda}^\dagger) d_p x dt \\ &+ (\partial_x L + \lambda^\dagger)|_T d_p x(T) \\ &+ (-\lambda^\dagger + \mu^\dagger)|_0 d_p x(0) \\ &+ \partial_p L + \int_0^T (\partial_p f - \lambda^\dagger \partial_p h)dt - \mu^\dagger \partial_p g \end{aligned}$$

and choosing Lagrange multipliers such that the coefficient of the first three lines equal zero, yields:

$$\lambda^\dagger(T) = -\partial_x L(x(T),p) \tag{7.38}$$

$$\dot{\lambda}^\dagger = \partial_x f - \lambda^\dagger \partial_x h \tag{7.39}$$

$$\partial_p \mathcal{L} = \partial_p L + \int_0^T (\partial_p f - \lambda^\dagger \partial_p h)dt - \lambda^\dagger(0)\partial_p g \tag{7.40}$$

So the only complications of the addition of the functional term is the new $\partial_x f$ term in the dynamics of the adjoint state and the $\partial_p f$ term in the calculation of the gradient. [9]

[9] For completeness, it is noted that a forward sensitivity method still exists in this case, similar to eq. 7.21, namely that the gradient can be calculated in a forward propagation by:

$$d_p F = \partial_p L + \partial_x L d_p x(T) + \int_0^T (\partial_p f + \partial_x f d_p x)dt$$

7.1.5 Including the gate time as a parameter.

In the previous cases, it may be that the parameter vector p contained different time scales[10] T_i which may be individually tuneable, but with the ensuing total time remaining constant, eg. by:

$$\sum_i T_i = T \quad \text{(fixed in the case of sec. 7.1.1 and 7.1.4)} \tag{7.41}$$

It is now of interest to relax this condition and include the total time T as a parameter, which in the context of QOC would mean it is possible to optimize wrt. the gate-time.[11]

So, consider now a parameter vector p that contains the final time as a parameter:

$$p = \begin{pmatrix} \tilde{p} \\ T \end{pmatrix} \tag{7.42}$$

Where it may be that $\tilde{p}$ still contain the T_i, but the inclusion of the total time in the parameter vector p has become explicit. The procedure

[10] Such as T_a and T_r in the DRAG pulse parametrization eq. 6.24

[11] What this exactly means will become apparent in section 8.2, but simply stated it allows one to optimize for fast gates.

for deriving the adjoint method remains the same, so considering the general Lagrangian of eq. 7.37, the gradient can be written as:

$$d_p\mathcal{L} = \begin{pmatrix} d_{\tilde{p}}\mathcal{L} & d_T\mathcal{L} \end{pmatrix} \tag{7.43}$$

The $d_{\tilde{p}}\mathcal{L}$ part simply follows the same derivation as before (sec. 7.1.4), whereas the second part is:

$$\begin{aligned} d_T\mathcal{L} &= d_T L(x(T), p) + f(x(T), p) + \lambda^\dagger(T) h(x, \dot{x}, p, T) \\ &= \partial_x L d_T x(T) + \partial_p L d_T p + f(x(T), p) + \lambda^\dagger(T)(\dot{x}(T) - h(x, p, T)) \\ &= (\partial_x L + \lambda^\dagger(T)) d_T x(T) + \partial_p L d_T p + f(x(T), p) - \lambda^\dagger(T) h(x, p, T) \end{aligned}$$

ie. the only difference compared to not treating T as a parameter is to just treat T as any other parameter, but with the extra term

$$-\lambda^\dagger(T) h(x(T), p, T) + f(x(T), p)$$

added to the gradient wrt. T

It should be noted that this is not unique to this formulation since this extra term is simply:

$$\begin{aligned} d_t F(x(t), p)|_{t=T} &= \partial_x L(x(t), p) d_t x(t)|_{t=T} + f(x(T), p) \\ &= -\lambda^\dagger(T) h(x(T), p, T) + f(x(T), p) \end{aligned}$$

Since by definition $d_t x(t) = h$ and $\lambda^\dagger(T) = -\partial_x L(x(T), p)$.

We may also relax the assumption that the total time is contained explicitly in the parameters, and just consider a parameter vector p that contain the different time scales T_i which make up the total time T. Then it is allowed to let T not be fixed and the gradient of the objective function is simply given by eq. 7.40 with the addition of the extra term:

$$d_p\mathcal{L}(T \text{ not fixed}) = d_p\mathcal{L}(T \text{ fixed}) + (-\lambda^\dagger h + f)|_{t=T} \partial_p T \tag{7.44}$$

Hereby, it is straight forward to consider optimization problems which do not rely on fixed total time. The employed ODEsolver simply solves the dynamics till a given end time, which in this case is the continuous variable T. The gradients necessary for gradient-based optimization techniques are simply given by eq. 7.44.

The derivation of the adjoint method is concluded by deriving a closed form of $\lambda^\dagger(t)$, which is done in the following section.

7.1.6 Closed form of the adjoint state

It can be shown that, if $\lambda^\dagger(t)$ is given as

$$\lambda^\dagger(t) = \frac{\partial}{\partial x(t)} \left[-L(x(T), p) + \int_T^t f(x(t'), p) dt' \right] \tag{7.45}$$

It will obey

$$\dot{\lambda}^\dagger = -\lambda^\dagger \partial_x h + \partial_x f \quad \text{with} \quad \lambda^\dagger(T) = -\frac{\partial L}{\partial x(T)} \tag{7.46}$$

Extending on the derivation in [103], it goes as follows. Consider the definition of the time derivative:

$$\dot{\lambda}^\dagger = \lim_{\varepsilon\to 0^+} \frac{1}{\varepsilon}\left[\lambda^\dagger(t+\varepsilon) - \lambda^\dagger(t)\right] \tag{7.47}$$

$$= \lim_{\varepsilon\to 0^+} \frac{1}{\varepsilon}\left[\frac{\partial}{\partial x(t+\varepsilon)}\left(-L + \int_T^{t+\varepsilon} f dt\right) - \frac{\partial}{\partial x(t)}\left(-L + \int_T^{t} f dt\right)\right] \tag{7.48}$$

In order to reduce this expression, such that the limit can be done, it is noticed that the definition of $\dot{x} = h$ can be rewritten as:

$$x(t+\varepsilon) = \int_t^{t+\varepsilon} h dt + x(t) \tag{7.49}$$

Which may be expanded in ε:

$$x(t+\varepsilon) = x(t) + \varepsilon h + \mathcal{O}(\varepsilon^2) \tag{7.50}$$

Such that it is possible to write the derivative of any entity $A(t)$ wrt. $x(t)$ as:

$$\frac{\partial A(t)}{\partial x(t)} = \frac{\partial A(t)}{\partial x(t+\varepsilon)}\frac{\partial x(t+\varepsilon)}{\partial x(t)} = \frac{\partial A(t)}{\partial x(t+\varepsilon)}\left(1 + \varepsilon\partial_x h + \mathcal{O}(\varepsilon^2)\right) \tag{7.51}$$

Similarly, the integral in eq. 7.48 may be expanded as:

$$\int_T^{t+\varepsilon} f dt' = \int_T^t f dt' + \varepsilon f + \mathcal{O}(\varepsilon^2) \tag{7.52}$$

Using eq. 7.51 and 7.52, equation 7.48 may be rewritten as:

$$\dot{\lambda}^\dagger = \lim_{\varepsilon\to 0^+} \frac{1}{\varepsilon}\left[-\varepsilon\frac{\partial(-L + \int_T^t f dt')}{\partial x(t+\varepsilon)}\partial_x h + \varepsilon\frac{\partial f}{\partial x(t+\varepsilon)} + \mathcal{O}(\varepsilon^2)\right] \tag{7.53}$$

$$= -\lambda^\dagger(t)\partial_x h + \partial_x f \tag{7.54}$$

Thus validating the claim made in the beginning of this section.

The closed form of $\lambda^\dagger(t)$ was derived for completeness of the first order adjoint method theory chapter. However, it remains unclear how eq. 7.45 is to be interpreted due to the functional derivatives, so the expression will now be deciphered.

First off, in the context of Quantum Mechanics, we may define the propagator $\mathcal{U}(t,0)$ of the system $x(t) = \mathcal{U}(t,0)x(0)$ such that the first term in eq. 7.45 becomes:

$$\frac{\partial L(x(T),p)}{\partial x(t)} = \frac{\partial L(x(T),p)}{\partial x(T)}\frac{\partial x(T)}{\partial x(t)} = -\lambda^\dagger(T)\mathcal{U}(T,t) \tag{7.55}$$

Secondly, since the limits of the integral in the second term do not depend on $x(t)$, it is possible to use the Leibniz integral rule for differentiation under the integral sign as:

$$\frac{\partial}{\partial x(t)}\int_T^t f(x(t'),p)dt' = \int_T^t \frac{\partial f(x(t'),p)}{\partial x(t')}\frac{\partial x(t')}{\partial x(t)}dt' = \int_T^t \partial_x f(x(t'),p)\mathcal{U}(t',t)dt' \tag{7.56}$$

So the closed form in eq. 7.45 can be written as:

$$\lambda^\dagger(t) = \lambda^\dagger(T)\mathcal{U}(T,t) + \int_T^t \partial_x f(x(t'),p)\mathcal{U}(t',t)dt' \tag{7.57}$$

This concludes the theoretical and practical discussion of the first order adjoint method. It should be possible to tackle many different QOC problems with the adjoint method and this is done in section 8. In the following section a discussion of how the adjoint method can be employed to estimate robust control solutions is provided. To this end the second order adjoint method is derived.

7.2 Robust control within the adjoint method

We will now investigate how the adjoint method might help implement any of the ideas put forward in section 6.5. Already, the derivations in section 7.1 can be used to calculate the gradients of eq.[12] 6.35 and 6.36 as efficiently as possible. However, in these cases, even if the number of uncertain parameters is more than just a few, the curse of dimensionality indicates that you'd need an exponentially large sampling of the hyper volume in order to approximate the mean and variance truthfully.

One of the alternatives is eq.[13] 6.38 which relies on gradient information to find a robust solution. This will be the subject of this section where a second order adjoint method is fleshed out, in order to evaluate the gradient of a loss that contains a first order derivative. Furthermore, for an objective function that only contain zero order terms, the 2nd order derivative information could still be used as the hessian for a Newton-type optimization procedure.

As a reminder of the notation used so far, consider a set of parameters of the dynamics that undergo perturbations, these are denoted by a. The parameters p and a may be identical or just share a subset or be completely different, but p is the control parameters while a are some parameters of the model that experience noise which we would like the control solution to be insensitive to. [14]

The problem statement is then to minimize:

$$F(x(t),p,a) = L(x(T),p,a) + \int_0^T f(x(t),p,a)dt \tag{7.58}$$

over some region ω_a with regards to p, subject to the dynamical system:

$$\dot{x} = h(x,p,a,t) \tag{7.59}$$
$$x(0) = g(p,a) \tag{7.60}$$

And as also discussed in section 6.5 linearization of the worst-case scenario yields:

$$\tilde{F}(x,p) = F(x,p,a_0) + \partial_a F(x,p,a_0)\delta a \tag{7.61}$$

Where the vector δa is found from a maximum condition. If the uncertain parameters are independent and with different bounds

[12] These methods pose multiple QOC problems, one for each sampled point $a_i \in \Omega_a$ and try to solve them all at once using the same control for each.

[13] In this case, the objective function contain a first order derivative, eg. $\tilde{\mathcal{L}} = \mathcal{L} + |\nabla_a \mathcal{L}|^2$, such that in order to evaluate $d_p\tilde{\mathcal{L}}$ a second order derivative method is necessary.

[14] Eg. in the context of the QOC problems encountered in the thesis, the control parameters could be the amplitude, driving frequency and phase of some pulse, $p = \begin{pmatrix} A \\ \omega_d \\ \phi \end{pmatrix}$ whereas the noisy parameters could be an external flux but also the amplitude of the pulse due to noise in the drive line, $a = \begin{pmatrix} \Phi_{\text{ext}} \\ A \end{pmatrix}$, in which case they would share the parameter A.

$\delta a_i \in [-m_i, m_i]$, then:

$$\tilde{F}(x,p) = F(x,p,a_0) + \sum_i m_i |\partial_{a_i} F(x,p,a_0)| \tag{7.62}$$

However, in generality we may just write it as some function R:

$$\tilde{F}(x,p) = F(x,p,a_0) + R(\partial_a F(x,p,a_0)) \tag{7.63}$$

The problem consists now of finding the gradient of $\tilde{F}$ wrt. p:

$$d_p \tilde{F} = d_p F(x,p,a) + \frac{\partial R(\partial_a F(x,p,a_0))}{\partial(\partial_a F)} d_p(\partial_a F)^\dagger \tag{7.64}$$

The first term $d_p F$ is given by eq. 7.40 with the adjoint state subject to eq. 7.38 and 7.39. The gradient wrt. the noisy parameters is simply given by:[15]

$$\partial_a F = \partial_a L + \int_0^T (\partial_a f - \lambda^\dagger \partial_a h) dt - \lambda^\dagger(0) \partial_a g \tag{7.65}$$

Notice that it is the same $\lambda^\dagger(t)$ for both $d_p F$ and $\partial_a F$, which is a testament to the efficiency of the adjoint method.

The second term contain the novel object of interest:[16]

$$d_p(\partial_a F)^\dagger \tag{7.66}$$

The second order adjoint method will now be derived in order to evaluate $d_p(\partial_a F)^\dagger$.[17]

[15] Following the same procedure of defining the Lagrangian, partially integrating and choosing suitable constraints in order to eliminate the need to calculate $d_a x$ one simply recover the results of sec. 7.1.4 but with a instead of p

[16] It doesn't matter whether it's † or simply transposing since the gradient is real $(\partial_a F)^\dagger = (\partial_a F)^T = \nabla_a F$. Note that if $a = p$ we retrieve the Hessian:

$$d_p \nabla_p F = \begin{pmatrix} \frac{\partial^2 F}{\partial p_1^2} & \frac{\partial^2 F}{\partial p_2 \partial p_1} & \cdots \\ \frac{\partial^2 F}{\partial p_1 \partial p_2} & \frac{\partial^2 F}{\partial p_2^2} & \\ \vdots & & \ddots \end{pmatrix}$$

[17] It should be noted that akin to the discussion in section 7.1.2, the forward sensitivity method is even worse in this regard, since calculations of the form $d_p(d_a x_i)^\dagger$ yields $N_a \times N_p \times N_x$ equations and therefore a poor scaling towards the number of parameters of the model.

7.2.1 Calculation of second order derivative

Firstly, it is wise to ease the notation of the adjoint of $\partial_a F$ in eq. 7.65, namely:

$$(\partial_a F)^\dagger = (\partial_a L)^\dagger + \int_0^T ((\partial_a f)^\dagger - (\partial_a h)^\dagger \lambda) dt - (\partial_a g)^\dagger \lambda(0) \tag{7.67}$$

$$\equiv L_a + \int_0^T (f_a - h_a \lambda) dt - g_a \lambda(0) \tag{7.68}$$

Where the notation is simplified by $(\partial_a L)^\dagger \equiv L_a$ and so on. Now, with inspiration from [104] consider the Lagrangian:

$$\mathcal{G}_a = L_a + \int_0^T (f_a - h_a \lambda) dt - g_a \lambda(0) + \int_0^T [\phi^\dagger(\dot{x} - h) + \psi^\dagger(\dot{\lambda} - f_x + h_x \lambda)] dt \tag{7.69}$$

Ie. we have added the constraints of the state dynamics and adjoint state dynamics with accompanying lagrange multipliers to the gradient of F wrt. a (eq. 7.68). The second order adjoint method will be swiftly derived and then the equations and implementation considerations will be commented on.

The derivation simply follow the same approach used for the first order method, namely that we consider the variation of $\mathcal{G}_a$ wrt. p:

$$\begin{aligned} d_p \mathcal{G}_a =& \partial_x L_a d_p x(T) + \partial_p L_a + \int_0^T (\partial_x f_a d_p x + \partial_p f_a - \partial_x (h_a \lambda) d_p x - \partial_p (h_a \lambda) - h_a d_p \lambda) dt \\ & - \partial_p (g_a \lambda(0)) - g_a d_p \lambda(0) \\ & + \int_0^T \phi^\dagger (d_p \dot{x} - \partial_x h d_p x - \partial_p h) dt \\ & + \int_0^T \psi^\dagger (d_p \dot{\lambda} - \partial_x f_x d_p x - \partial_p f_x + \partial_x (h_x \lambda) d_p x + \partial_p (h_x \lambda) + h_x d_p \lambda) dt \end{aligned}$$

and[18] partially integrate the terms with $d_p\dot{x}$ and $d_p\dot{\lambda}$:

$$\int_0^T \phi^\dagger d_p\dot{x}dt = \phi^\dagger d_p x|_0^T - \int_0^T \dot{\phi}^\dagger d_p x dt$$
$$\int_0^T \psi^\dagger d_p\dot{\lambda}dt = \psi^\dagger d_p \lambda|_0^T - \int_0^T \dot{\psi}^\dagger d_p \lambda dt$$

[18] Note that terms such as $\partial_x(h_a\lambda)$ and $\partial_p(h_a\lambda)$ are written like this even though the only explicit x and p dependence exists in h_a. The reason for this is to keep the matrix calculus notation where eg. $\partial_p(h_a\lambda)$ is a matrix with dimensionality:

$$d_p\left[(N_a \times N_x)(N_x \times 1)\right] = N_a \times N_p$$

and $\partial_p(h_a)\lambda$ would not make sense wrt. the matrix notation.

Collecting terms wrt. $d_p x$ and $d_p\lambda$ yields:

$$\begin{aligned}
d_p\mathcal{G}_a =& (\partial_x L_a + \phi^\dagger)|_T d_p x(T) \\
&- \phi^\dagger(0) d_p x(0) \\
&+ \psi^\dagger(T) d_p\lambda(T) \\
&+ (-g_a - \psi^\dagger(0)) d_p\lambda(0) \\
&+ \int_0^T (\partial_x f_a - \partial_x(h_a\lambda) - \dot{\phi}^\dagger - \phi^\dagger\partial_x h - \psi^\dagger\partial_x f_x + \psi^\dagger\partial_x(h_x\lambda)) d_p x dt \\
&+ \int_0^T (-h_a - \dot{\psi}^\dagger + \psi^\dagger h_x) d_p\lambda dt \\
&+ \int_0^T (\partial_p f_a - \partial_p(h_a\lambda) - \phi^\dagger\partial_p h - \psi^\dagger\partial_p f_x + \psi^\dagger\partial_p(h_x\lambda)) dt + \partial_p L_a - \partial_p(g_a\lambda(0))
\end{aligned}$$

Where, by definition:

$$d_p x(0) = \partial_p g(p,a) \tag{7.70}$$
$$d_p\lambda(T) = -\partial_x L_x d_p x(T) \tag{7.71}$$

Consequently, one can choose $\phi^\dagger$ to follow the final value problem:

$$\dot{\phi}^\dagger = \partial_x f_a - \partial_x(h_a\lambda) - \phi^\dagger\partial_x h - \psi^\dagger\partial_x f_x + \psi^\dagger\partial_x(h_x\lambda) \tag{7.72}$$
$$\phi^\dagger(T) = -\partial_x L_a + \psi^\dagger(T)\partial_x L_x \tag{7.73}$$

and $\psi^\dagger$ to follow the initial value problem:

$$\dot{\psi}^\dagger = \psi^\dagger h_x - h_a \tag{7.74}$$
$$\psi^\dagger(0) = -g_a \tag{7.75}$$

Such that the gradient is given by:

$$d_p\mathcal{G}_a = \partial_p L_a - \partial_p(g_a\lambda(0)) - \phi^\dagger(0)\partial_p g + \int_0^T (\partial_p f_a - \partial_p(h_a\lambda) - \phi^\dagger\partial_p h - \psi^\dagger\partial_p f_x + \psi^\dagger\partial_p(h_x\lambda))dt \tag{7.76}$$

This looks more complicated than it is in most cases because it is written up in generality. The calculations will be simplified if eg. $f = 0$ or $\partial_p g = \partial_a g = 0$ and so on. An example that is not too complicated is considered in sec. 7.2.2.

The $\psi(t)$ of eq. 7.74 and 7.75 is simply the forward sensitivity method:

$$\psi(t) = -d_a x(t) \tag{7.77}$$

The minus sign is due to the sign of the initial value constraint $x(0) - g(p,a) = 0$ and it would be possible to absorb the minus sign into ψ for it to be exactly equal to the forward sensitivity method.

This also means that ψ is a matrix of dimensionality $N_x \times N_a$, which also applies to the ϕ of eq. 7.72 and 7.73.[19] However, $\phi(t)$

[19] The dimensionality is already apparent from when they were introduced to the Lagrangian. Ie. $N_a \times 1 = (?\times?)(N_x \times 1)$ must mean $\phi^\dagger$ and $\psi^\dagger$ are of dimensionality $N_a \times N_x$

is not as easily interpreted as $\psi(t)$, instead it is instructive to focus on how the equations are applied in practice. The thing to notice is that $\psi(t)$ is given by an initial value problem, where eq. 7.74 is only dependent on ψ and x so it is possible to solve these simultaneously by a forward evolution, in accordance with ψ simply being the forward sensitivity method. Next, both ϕ and λ are given by final value problems and can be solved simultaneously evolving backwards in time. This means that second order derivative information is achieved by a single forward and a single backwards evolution. The practical implementation will now be summarized:

- The forward evolution is done simultaneously for x and ψ, so by using an ODEsolver, $x(T)$ and $\psi(T)$ is found from:

$$d_t \begin{bmatrix} x(t) \\ \psi(t) \end{bmatrix} = \begin{bmatrix} h(x,p,a,t) \\ \partial_x h\psi - \partial_a h \end{bmatrix} \quad \text{with} \quad \begin{bmatrix} x(0) = g(p,a) \\ \psi(0) = -\partial_a g(p,a) \end{bmatrix} \tag{7.78}$$

- Next, the final values of λ and ϕ are prepared:

$$\begin{bmatrix} \lambda(T) \\ \phi^\dagger(T) \end{bmatrix} = \begin{bmatrix} -L_x \\ -\partial_x L_a + \psi^\dagger(T)\partial_x L_x \end{bmatrix} \tag{7.79}$$

- Then it is possible to calculate both the first order and second order derivatives by evolving everything backwards simultaneously. That is, consider the evolution of the augmented state:

$$d_t \begin{bmatrix} x(t) \\ \psi(t) \\ \lambda(t) \\ \phi^\dagger(t) \\ \nabla^a(t) \\ \nabla^p(t) \\ \mathbb{W}(t) \end{bmatrix} = \begin{bmatrix} h \\ \partial_x h\psi - \partial_a h \\ f_x - h_x\lambda \\ \partial_x f_a - \partial_x(h_a\lambda) - \phi^\dagger\partial_x h - \psi^\dagger\partial_x f_x + \psi^\dagger\partial_x(h_x\lambda) \\ \partial_a f - \lambda^\dagger\partial_a h \\ \partial_p f - \lambda^\dagger\partial_p h \\ \partial_p f_a - \partial_p(h_a\lambda) - \phi^\dagger\partial_p h - \psi^\dagger\partial_p f_x + \psi^\dagger\partial_p(h_x\lambda) \end{bmatrix} \tag{7.80}$$

  with $\nabla^a(T) = \nabla^p(T) = \mathbb{W}(T) = 0$ all the final values are given and the augmented state is evolved backwards in time.

- The respective gradients are then found by:

$$d_p F = \partial_p L - \nabla^p(0) - \lambda^\dagger(0)\partial_p g \tag{7.81}$$

$$\partial_a F = \partial_a L - \nabla^a(0) - \lambda^\dagger(0)\partial_a g \tag{7.82}$$

$$d_p(\partial_a F)^\dagger = \partial_p L_a - \partial_p(g_a\lambda(0)) - \phi^\dagger(0)\partial_p g - \mathbb{W}(0) \tag{7.83}$$

  such that eq. 7.64 can be evaluated.

It is instructive to consider an example, and in the context of the system considered in this thesis, namely the superconducting qubit denoted the DSFQ (eq. 4.23), the following robust QOC problem is posed.[20]

[20] In the notation of this section, the following example contain the simplifications:

$$g(p,a) \to g(a)$$
$$L(x(T),p,a) \to L(x(T),a)$$
$$f(x(t),p,a) \to 0$$

The $f \neq 0$ case will also be treated in the appendix 10.4.2

7.2.2 Robustness against external flux noise.

Take the Hamiltonian of eq. 4.23 and consider noise in the external flux parameter $\Phi_{\text{ext}} \rightarrow \Phi_{\text{ext}} + \delta\Phi_{\text{ext}}$. Let p be an arbitrary set of parameters excluding Φ_{ext} while the only noisy environment variable is the external flux $a = \Phi_{\text{ext}}$.[21] In fact a should refer to the dimensionless external flux $\phi_{\text{ext}} = 2\pi\Phi_{\text{ext}}/\Phi_0$ but to evoke the idea of magnetic flux Φ is used in the following, where the subscript "$_{\text{ext}}$" is omitted for brevity. Write the Hamiltonian of eq. 4.23 as:

[21] It would be interesting and rather simple to also add E_J and perhaps the control parameter $\alpha_{\min}$ to a, however, this did not make it into the thesis.

$$\mathcal{H}(p, \Phi, t) = H_0 + H_\alpha(\Phi)\alpha(p, t) + Vu(p, t) \tag{7.84}$$

with $H_\alpha(\Phi) \doteq \frac{-E_J}{2}\left(e^{i\Phi}A \otimes A^\dagger + e^{-i\Phi}A^\dagger \otimes A\right)$ if working in the charge basis as defined in section 5.1.1.

The transfer of $|0\rangle$ to $|1\rangle$ will be considered for simplification of the derivation, but it is straight forward to expand the following to consider a loss function dependent on both computational states, as described in section 7.1.3. Again, the initial conditions are defined from the eigenvalue equation:

$$(H_0 + H_\alpha(\Phi))\psi_i(\Phi) = E_i(\Phi)\psi_i(\Phi) \tag{7.85}$$

Such that the initial conditions are dependent on Φ

$$x(0) = g(a) = \psi_0(\Phi) \tag{7.86}$$

The dynamics are simply governed by the Schrödinger equation:

$$h(x, p, \Phi, t) = -i\mathcal{H}(p, \Phi, t)x(t) \tag{7.87}$$

The objective function is the infidelity of the state transfer:

$$F(x, p, \Phi) = L(x(T), \Phi) = 1 - x(T)^\dagger\psi_1(\Phi)\psi_1(\Phi)^\dagger x(T) \tag{7.88}$$

The robust objective function is:

$$\tilde{L} = L(x(T), \Phi) + m|\partial_\Phi L| \tag{7.89}$$

The gradient of the first term $d_p L$ (along with $\partial_\Phi L$) is calculated by:[22]

[22] The last two terms of eq. 7.93 are the $\partial_a L$ and $-\lambda^\dagger(0)\partial_a g$ terms of eq. 7.82.

$$\lambda(T) = \psi_1(\Phi)\psi_1(\Phi)^\dagger x(T) \tag{7.90}$$

$$\dot{\lambda} = -i\mathcal{H}\lambda \tag{7.91}$$

$$d_p L = -2\int_0^T \left[\mathcal{I}m\left(\lambda^\dagger H_\alpha x\right)\partial_p\alpha + \mathcal{I}m\left(\lambda^\dagger V x\right)\partial_p u\right]dt \tag{7.92}$$

$$d_\Phi L = -2\int_0^T \mathcal{I}m\left(\lambda^\dagger H'_\alpha x\right)\alpha dt + 2\mathcal{R}e\left(x^\dagger(T)\psi_1'\psi_1^\dagger x(T)\right) - 2\mathcal{R}e\left(\lambda^\dagger(0)\psi_0'\right) \tag{7.93}$$

Where[23] $H'_\alpha \doteq \frac{-iE_J}{2}\left(e^{i\Phi}A \otimes A^\dagger - e^{-i\Phi}A^\dagger \otimes A\right)$ in the charge basis and ψ_i' is found from differentiating the Schrödinger equation:

[23] In operator form:
$H'_\alpha = E_J \sin(\hat{\phi}_1 - \hat{\phi}_2 + \phi_{\text{ext}})$

$$\psi_i' = \sum_{n\neq i} \frac{\psi_n^\dagger H'_\Phi \psi_i}{E_i - E_n}\psi_n \quad \text{with} \quad \psi_i^\dagger\psi_i' = 0 \tag{7.94}$$

The gradient of the second term is found as:

$$d_p\left(m|\partial_\Phi L|\right) = \text{sign}(\partial_\Phi L) m d_p(\partial_\Phi L) \tag{7.95}$$

and calculated by:

$$\phi(T) = (\psi_1 \psi_1'^\dagger + \psi_1' \psi_1^\dagger) x(T) - \psi_1 \psi_1^\dagger \psi(T) \tag{7.96}$$

$$\dot{\phi} = -i\mathcal{H}\phi - i\alpha H_\alpha' \lambda \tag{7.97}$$

$$\psi(0) = -\psi_0'(\Phi) \tag{7.98}$$

$$\dot{\psi} = -i\mathcal{H}\psi + i\alpha H_\alpha' x \tag{7.99}$$

$$\begin{aligned} d_p(\partial_\Phi L) = -2 \int_0^T \Big[& -\mathcal{I}m\left(x^\dagger H_\alpha' \lambda\right) \partial_p \alpha \\ & + \mathcal{I}m\left(\phi^\dagger H_\alpha x\right) \partial_p \alpha + \mathcal{I}m\left(\phi^\dagger V x\right) \partial_p u \\ & + \mathcal{I}m\left(\psi^\dagger H_\alpha \lambda\right) \partial_p \alpha + \mathcal{I}m\left(\psi^\dagger V \lambda\right) \partial_p u \Big] dt \end{aligned} \tag{7.100}$$

This is the first example of an application of theory established in section 7.2.1, so it instructive to clarify where each term comes from. The two terms in eq. 7.96 are from $-(\partial_x L_a)^\dagger$ and $(\partial_x L_x)^\dagger \psi(T)$ respectively.
The two terms in eq. 7.97 come from $-h_x \phi$ and $-(\partial_x (h_a \lambda))^\dagger$, the last of which is calculated through the use of:

$$\partial_x (h_a \lambda) = \partial_x (i\alpha x^\dagger H_\alpha' \lambda) = (i\alpha H_\alpha' \lambda)^\dagger \tag{7.101}$$

The next equation is of course just $-\psi_0'(\Phi) = -\partial_a g$ with eq. 7.99 being the forward sensitivity method, up to a sign. The terms in eq. 7.100 are from $-\partial_p (h_a \lambda)$, $-\phi^\dagger \partial_p h$ and $\psi^\dagger \partial_p (h_x \lambda)$ of eq. 7.83

With the approach of defining augmented states and evolving back and forwards in time using ODEsolvers being described multiple times, it should be apparent by now how to implement gradient-based optimization techniques for solving QOC problems, where the derivatives are calculated efficiently to second order by the adjoint method.

In this thesis, several QOC problems are posed and treated with this approach, the result of which are presented in the following section.

8 Results

The foundation for actually carrying out the optimization of QOC problems have been laid out in the previous chapters. We are now at a point where we can consider different QOC problems each with their own non-trivial aspect.

As previously stated, the system that is considered is the superconducting qubit denoted DSFQ (eq. 4.23). The system dynamics are simulated to a high accuracy (with significant digits at the $\sim 10^{-8}$ level)[1] using ODEsolvers 5.2.2 and a sufficiently precise representation, which in this case works out to be the charge basis 5.1.1 with $n_{\max} = 8$ resulting in $(2n_{\max}+1)^2 = 289$ basis states. All the quoted fidelities are for simulations of the closed quantum system. For a discussion of how the openness would affect the fidelities see section 5.3.

[1] Arguments for this statement is provided in appendix 10.2.1

The different QOC problems will now be presented and discussed in a sequential manner.

8.1 Optimized DRAG swap-gate

As a first example of quantum optimal control, a simple optimization problem is considered. We wish to construct a swap-gate in a finite time $T = 30$ns for the system described in section 4.3.1. The objective function is therefore given by eq. 6.2.

A rather high fidelity can be achieved by a pulse that comes from analytical means and human tuning and this result will be considered the baseline solution. The time dependence of α is given by eq. 6.21, and in the case of the baseline solution, the pulse is parametrized as eq. 6.22, with parameters given by p_0 in table 8.1. The amplitude and driving frequency is included as the following scaled parameters, in order to be more interpretable:

$$\tilde{A} \equiv A \left(\frac{\pi}{|\langle 1_{\alpha_{\min}} | \hat{n} | 0_{\alpha_{\min}} \rangle| (T_p - T_r)} \right)^{-1} \tag{8.1}$$

$$f \equiv \frac{\hbar \omega_d}{E_{1,\alpha_{\min}} - E_{0,\alpha_{\min}}} = \frac{\omega_d}{\omega_q} \tag{8.2}$$

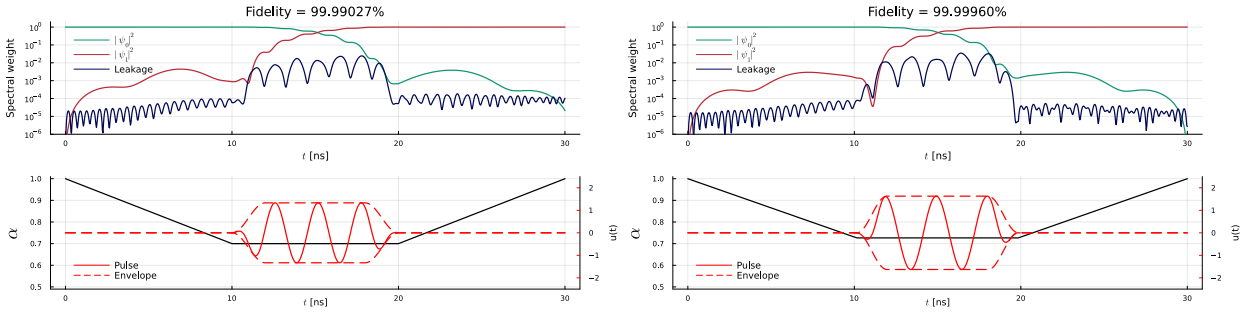
where $\tilde{A} = 1$ is an analytical pi-pulse and f measures the driving frequency ω_d relative to the qubit frequency ω_q. This is advantageous because ω_q is dependent on $\alpha_{\min}$, since $(H_0 + \alpha_{\min} H_\alpha)|n_{\alpha_{\min}}\rangle = E_{n,\alpha_{\min}} |n_{\alpha_{\min}}\rangle$.

Parameter	p_0	$p_{\text{semi-opt}}$	p_{opt}
T_r	2	2	1.8096
T_a	10	10	10.1514
$\alpha_{\min}$	0.7	0.7	0.726536
φ	0	0.5418	2.8197
f	0.9778	0.9763	0.9719
$\tilde{A}$	1	1.01556	1.0167
λ	0	0.05289	0.0357
Infidelity	$9.722 \cdot 10^{-5}$	$2.905 \cdot 10^{-5}$	$4.049 \cdot 10^{-6}$

Table 8.1: Table of parameter values. See sec. 6.4 and fig. 6.5 for a clarification of the parameters. The T_p found there is fixed by $2T_a + 2T_r + T_p = T = 30$ns. The green color refer to the fact that the parameter has been optimized. The infidelity refer to the swap infidelity defined by eq. 6.2.

For the baseline only the parameter f is scanned rather crudely and chosen to four digits. From this set of parameters an optimization procedure is initialized, which can adjust the highlighted parameters of table 8.1, yielding the $p_{\text{semi-opt}}$ column. The pulse is parametrized as the DRAG-pulse given in eq. 6.24.

Lastly, a second optimization procedure with three additional parameters is intialized from $p_{\text{semi-opt}}$ yielding p_{opt}. A comparison between the baseline pulse and the optimized drag pulse p_{opt} is made in figure 8.1. What is plotted in the top figures is the spectral weight



(a) Baseline swap gate (p_0 of table 8.1)

(b) Optimized DRAG pulse swap gate (p_{opt})

Figure 8.1: Visualization of the control functions $\alpha(t, \vec{p})$ and $u(t, \vec{p})$ (bottom row) and the resulting dynamics expressed as the spectral weight wrt. the instantaneous basis of sec. 5.1.2 (top row) for p_0 (a) and p_{opt} (b) respectively. The fidelity refer to the swap fidelity ie. $1 - L^{\text{swap}}$ with L^{swap} that of eq. 6.2.

of the state initialized as $\psi(0) = |0_{\alpha=1}\rangle$ as a function of time. The $|\psi_0|^2$ refer to the absolute square of the projection of $\psi(t)$ on to the zeroth eigenstate of the Hamiltonian $H_0 + \alpha(t)H_\alpha$, that is the ground-state of the Hamiltonian at that instant. Simply said, what is plotted is the absolute square of the projection on to the instantaneous basis (as defined in section 5.1.2). Leakage simply refers to the amount of spectral weight that is not contained in the computational space.

The notable feature is that the gate-fidelity is largely determined by the leakage at time T and clearly this is lower for the optimized pulse, in accordance with the lower gate-infidelity as quoted in table 8.1. The reason that p_{opt} is not too different from p_0 is because the initialization p_0 is already near a minimum.

A slightly less trivial QOC problem is that of finding a fast high-fidelity gate. This problem is treated in the next section.

8.2 Fast DRAG swap gate

Firstly, one should define the desired toleration to the trade-off between the speed of the gate and the fidelity of the gate, since these are mutually competing. To this end, one could define the QOC problem as "find the fastest gate that is below some threshold infidelity", where one would have to decide one a threshold.

Nevertheless, one way to go about solving this QOC problem, would be to do an optimization procedure, as in section 8.1, for different fixed T's and thereafter choosing the fastest solution that is also below the threshold in gate-infidelity. Another method, which is straight forward in this "continuous time" framework, is to let the optimizer minimize the gate time, which would then be a continuous parameter, in addition to the gate-infidelity. The objective function

will thereby consist of two terms. One term that minimizes the gate infidelity, and a term that minimizes the gate time. Whenever multiple terms are present in the objective function, it is important to give each term suitable weights. These weights may depend on the value of the terms themselves and this can be quite advantageous [105]. This is also done here, where the gate-time objective is weighted by a function of the infidelity. The weight function is chosen such that when the infidelity is large, the gate-time term is near zero and the minimizer will focus on getting the infidelity low. Then, when the infidelity is low enough this weight increases until reduction of the gate-time is the most important objective. The weighting function can be rather arbitrary, but the wanted attributes are accomplished with the definition as seen in figure 8.2, namely:

$$w(L) = \frac{\ln(L)^4}{75 \cdot 10^5} \tag{8.3}$$

The total objective function with the gate-infidelity defined by eq. 6.2 is then:

$$\mathcal{L} = L^{\text{swap}} + w(L^{\text{swap}})T \tag{8.4}$$

The calculation of the gradient of such an objective function is described in section 7.1.5 and succinctly given by eq. 7.44.[2] The optimization procedure was initialized from $p_{\text{semi-opt}}$ and yielded the result stated in table 8.2. The pulse and the corresponding simulation is visualized in figure 8.3.

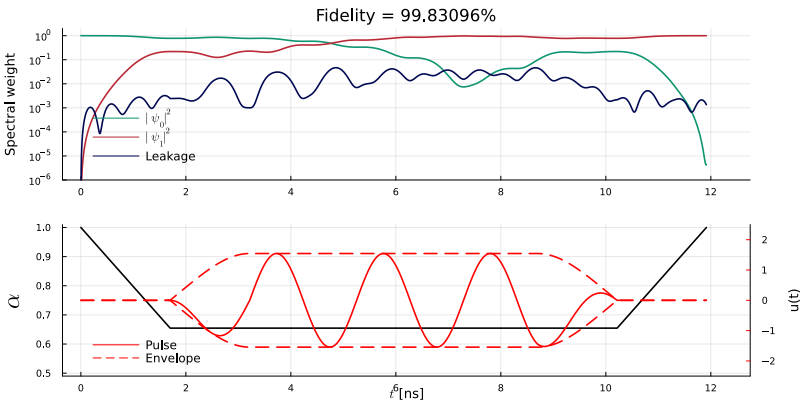


Figure 8.3: Visualization of the resulting control functions and associated simulation of p_{fast}, when employing eq. 8.4 as a loss function with an initialization from $p_{\text{semi-opt}}$. The parameter values of p_{fast} can be found in 8.2. Note the resulting gate time of only 11.9157ns

Note the gate-time of $T \sim 11.9$ns as opposed to the $T = 30$ns of the pulses in sec. 8.1 while still achieving a gate fidelity of $\sim 99.8\%$. This is of course a rather low fidelity and in practice the infidelity threshold would be set a little lower, resulting in a longer gate but with higher fidelity. However, it is not immmediately clear where the choice of threshold infidelity was made, but it is somehow related to the weighting function $w(L)$.

In order to understand this threshold and investigate the usefulness of optimizing for both gate-fidelity and gate-time simultaneously, a scan was made across the gate-time. That is, for different fixed gate times T, an optimization procedure akin to section 8.1 was made, just

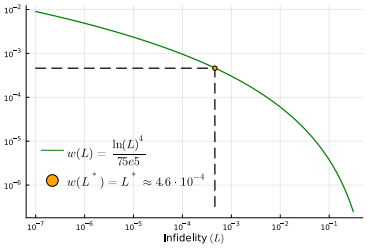


Figure 8.2: Example of a suitable weighting function for the QOC problem that consists of minimizing the gate-time in addition to the gate-infidelity.

[2] The only noteworthy effect is perhaps that the final value of the adjoint state is:

$$\lambda^\dagger(T) = -\partial_x \mathcal{L} = (1 + w'(L^{\text{swap}})T)\partial_x L^{\text{swap}}$$

but this is simply given by the definition, so it is nothing new.

Parameter	p_{fast}
T_r	1.517
T_a	1.7006
T_p	5.4799
$T = 2T_a + 2T_r + T_p$	11.9157
$\alpha_{\min}$	0.65459
φ	0.0326
f	0.928
$\tilde{A}$	1.285
λ	-0.4366
Infidelity	$1.69 \cdot 10^{-3}$

Table 8.2: The optimized parameter values of p_{fast} that is visualized in fig. 8.3

as the first method mentioned in the beginning of this section. This yields the green dots in figure 10.4.

Then, for the objective function in eq. 8.4 to be near a minimum, the following must hold:

$$L^{\text{swap}} \simeq w(L^{\text{swap}})T \tag{8.5}$$

because either $w(L^{\text{swap}}) \simeq 0$ which means the fidelity is too bad and needs first to converge or $w(L^{\text{swap}}) \gg L^{\text{swap}}$ in which case it would reduce the gate-fidelity by decreasing the gate-time. So, we can regard:

$$L^{\text{swap}} + w(L^{\text{swap}})T = \mathcal{O}(L^{\text{swap}}) \quad \text{where} \quad \mathcal{O}(L) \in [10^{-0.5}L, 10^{0.5}L] \tag{8.6}$$

as a relation that gives the minima of L^{swap} as a function of T. This is shown as the red-band in figure 10.4 for the $w(L)$ given by eq. 8.3 and as grey bands for $w \to w(L)/10^n$ for $n \in \{1, 2, 3, 4\}$

For a chosen weighting function, $\min(\mathcal{L})$ must then lie on the accompanying band. We see that the optimization procedure automatically finds the shortest pulse that obey this relation, since it lies almost directly at the cross-over between the red-band and the interpolation between the closest green-dots, which indicate the best achievable fidelity for that gate-time (and the given model-parameters/pulse-parametrization). It is then clear, that if a lower gate-infidelity is wanted, we could have chosen the weighting function $w \to w(L)/10^3$ which probably would have yielded a gate time $T \approx 17$ns with gate-infidelity of $\approx 10^{-5}$. Ie. it is argued that one simply defines the wanted gate-infidelity by choosing a suitable weighting function and this optimization procedure will automatically find the shortest pulse that achieves the wanted gate-infidelity.

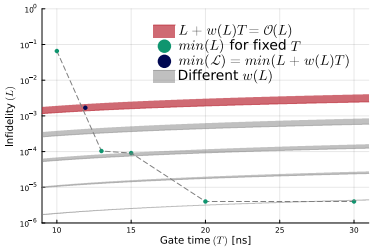


Figure 8.4: The green dots correspond to 5 different QOC problems akin to the one of sec. 8.1 but with T fixed to $\{10, 13, 15, 20, 30\}$ns respectively. The dark blue dot corresponds to a minimization of eq. 8.4 initialized from $T = 30$ns but with T now tuneable. It is argued that the minimization procedure automatically finds the crossing of the red band and the interpolation between the green dots, and therefore the fastest possible gate for the given infidelity threshold.

8.3 Non-commuting swap gates

So far, only swap-gate synthesis has been considered through the use of the objective function 6.2. To illustrate the capabilities of synthesizing any single-qubit gate, the QOC problem of finding two orthogonal swap-gates will now be treated.

The initial need to formalize the objective function as 6.2 instead of as 6.3 with, say, $\mathcal{U}_{\text{target}} = \sigma_x$ is because the latter is dependent on the choice of gauge, ie. the orientation of the coordinate system of the Bloch-sphere. One way to fix the gauge, is to first optimize for a swap-gate. The energy-eigenstates are then determined numerically once on the machine. This may yield an arbitrary gauge, such that the resulting unitary evolution, in general, has small complex numbers along the diagonal and two complex numbers, with different phases along the off-diagonal:

$$\begin{pmatrix} \langle 0|\psi_0(T)\rangle & \langle 0|\psi_1(T)\rangle \\ \langle 1|\psi_0(T)\rangle & \langle 1|\psi_1(T)\rangle \end{pmatrix} = \begin{pmatrix} \delta_1 & |z_1|e^{i\phi_1} \\ |z_2|e^{i\phi_2} & \delta_2 \end{pmatrix} \tag{8.7}$$

Parameter	p_{σ_y}
T_r	2
T_a	10
$\alpha_{\min}$	0.7
φ	1.841
f	0.9985
$\tilde{A}$	1.040
λ	-0.2772
Gate infidelity	$4.235 \cdot 10^{-5}$

Table 8.3: Parameter values for p_{σ_y}. The X-gate is given by $p_{\sigma_x} = p_{\text{semi-opt}}$ of table 8.1. Gate fidelity refers to eq. 6.3 for which $1 \leq L^{\text{gate}}/L^{\text{swap}}$. However, due to rounding of the swap fidelity, this is not examinable.

The x- and y-axis of the coordinate system of the Bloch-sphere can then be defined as:

$$\tilde{\sigma}_x \equiv \begin{pmatrix} 0 & e^{-i\frac{\phi_2-\phi_1}{2}} \\ e^{i\frac{\phi_2-\phi_1}{2}} & 0 \end{pmatrix}, \quad \tilde{\sigma}_y \equiv \begin{pmatrix} 0 & -ie^{-i\frac{\phi_2-\phi_1}{2}} \\ ie^{i\frac{\phi_2-\phi_1}{2}} & 0 \end{pmatrix} \tag{8.8}$$

The objective function 6.3 together with $\tilde{\sigma}_y$ can be used to find a swap-gate that is orthogonal the original swap-gate. The resulting evolution on the Bloch-sphere of the two pulses is visualized in figure 8.5

Evidently the two pulses constitute a rotation along the x- and y-axis respectively. Coincidentally the y-rotation is along the negative y-direction, this is due to the objective function still being agnostic towards $\pm\mathcal{U}_{\text{target}}$.[3] The pulse parametrization is still given by the DRAG parametrization of eq. 6.24. The optimization procedure was initialized from $p_{\text{semi-opt}}$ which also plays the role of the swap-gate that fixes the gauge and thereby become the designated X-gate. The resulting optimized parameters p_{σ_y} for the Y-gate are listed in table 8.3. The pulses and the corresponding simulations are visualized in figure 8.6

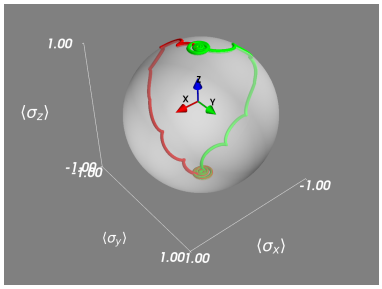
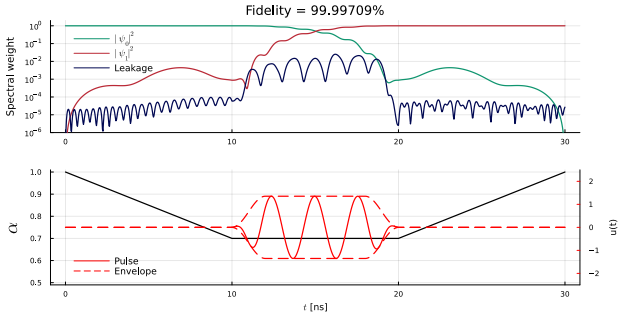
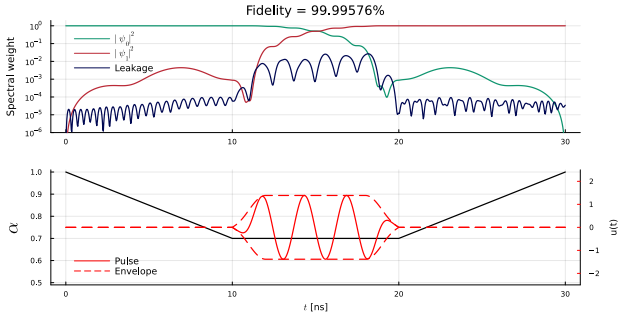


Figure 8.5: Evolution on the Bloch sphere in the rotating frame, with the x-axis and the red trajectory given by the evolution with $p_{\text{semi-opt}}$. The green trajectory is evolution of the optimized Y-gate given by p_{σ_y}

[3] This has no practical consequences, and in this case one could simply let $-y \to x$ and $x \to y$ in fig. 8.5 in order to quote the swap-gates as rotations along the positive x and y axis.



(a) X-gate ($p_{\text{semi-opt}}$)



(b) Y-gate (p_{σ_y})

Figure 8.6: Visualizations of the control functions and accompanying simulations for the X-gate defined by $p_{\text{semi-opt}}$ and the Y-gate (defined by p_{σ_y}) found by optimization of eq. 6.3 with $U_{\text{target}} = \tilde{\sigma}_y$ initialized from $p_{\text{semi-opt}}$. The fidelities refer to the swap-fidelity.

The gates implemented so far have relied on the lowering of α which increases the overlap between the computational states and allows for state transfer. In the next section a swap gate that relies on higher lying states is treated, because the barrier will be kept in the protected regime $\alpha(t) = 1$ during the gate.

8.4 Swap gate for qubit in protected regime

As seen from the coherence times 5.4 the T_1 time is greatly reduced when lowering the barrier by decreasing α. Therefore, it could be informative to investigate how well a pulse can send the computational basis above and across the barrier and thus implement a swap gate for a qubit in the protected regime. For this the interpolation method pulse parametrization of section 6.4.3 is used.[4] It was parametrized by 200 parameters spaced evenly across the 30ns and a simple mechanism[5] to ensure that the pulse started and ended with a value of zero ($u(0) = u(T) = 0$). An initialization of $u(t) = 0$ will be hard for the optimizer to work with since the objective function would have next to no dependence on the parameters. Other initializations are available,

[4] The DRAG-pulse is too simple to achieve this and also one of the key motiviations of deriving the adjoint method was so one would have the ability to optimize for a lot of control parameters.

[5] It could either be an envelope function as in the case of eq. 6.23 or let the tuneable parameters span the duration $[0 + \Delta T, T - \Delta T]$ with the outermost parameters forced to stay zero.

such as a $u(t) = 1$ or similar. Instead a random parametrization was used with $p_i \sim U(-0.5, 0.5)$, where U is the uniform distribution.

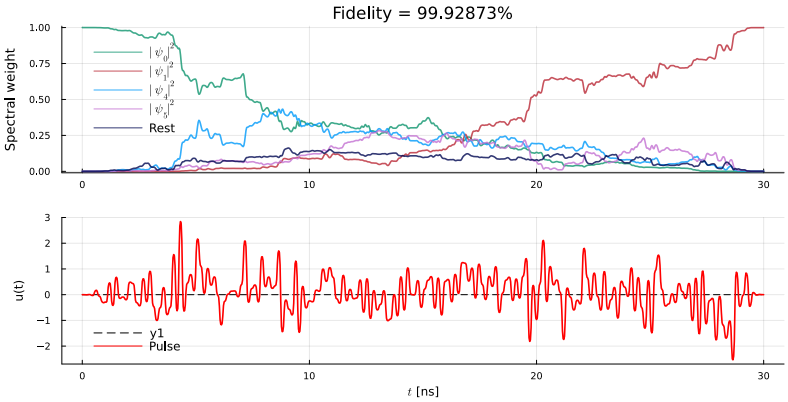
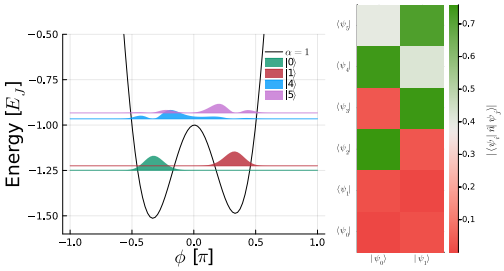


Figure 8.7: Optimized pulse for a swap gate, parametrized by the interpolation method of sec. 6.4.3 with a sampling rate of $200/30\text{ns} = 6.7\text{GHz}$. The evolution of the computational states can be seen as utilizing the 4th and 5th excited state in order to shoot over the barrier.

The objective function was simply L^{swap} of eq. 6.2 which yielded the result seen in figure 8.7 The spectral weight of ψ_4 and ψ_5 is plotted alongside the computational basis, because these are the lowest lying states with non-negligible overlap wrt. the pulse-coupling operator $\langle i|\hat{n}|j\rangle$ (see figure 8.8).



(a) Wavefunctions in ϕ basis. (b) Matrix elements

Figure 8.8: (a) The wavefunctions of the DSFQ (eq. 4.23) for $\alpha = 1$ along the $\phi = (\phi_1 - \phi_2)/2$ mode. (b) show why the 4th and 5th excited state is utilized.

The spectral weight as a function of time is easy to interpret. As seen in fig. 8.7 the $|0\rangle$ state is semi-transferred to a higher-lying state in the same well. Then it is transferred across to the other well, and then transferred to the $|1\rangle$ state, loosely speaking.[6] It also spends some time outside these necessary states, with an average spectral weight of 0.071 during the 30ns. This can be lowered to 0.015 as done in appendix 10.4.2 by the addition of 6.6 to the objective function, with the allowed states being the 4th and 5th states in addition to the computational basis.

[6] See the appendix figure **??** for an even clearer example.

A fourier analysis of the pulse yields the spectrum seen in figure 8.9 which is compared to the transition frequencies $\omega_{ij} = (E_i - E_j)/\hbar$. Evidently the ω_{41}, ω_{40} and ω_{54} frequencies are largely present in the pulse. Also shown in figure 8.9 is the time-averaged spectral weight

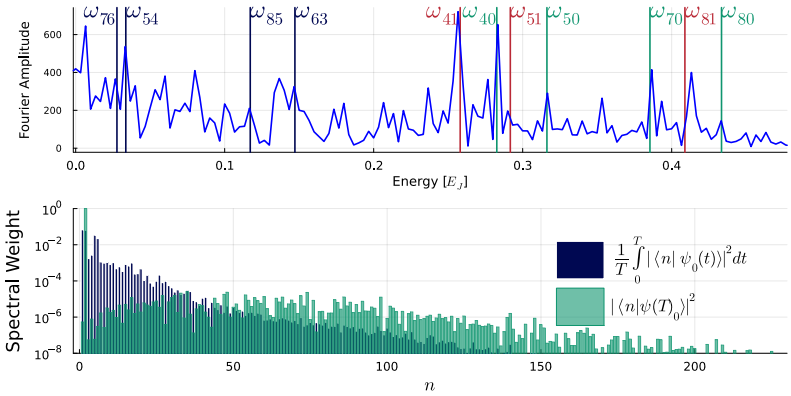


Figure 8.9: Top row: Fourier amplitudes of the pulse in figure 8.7 compared to the transition frequencies ω_{ij}. Bottom row: spectral weight distribution of both the final state after the evolution and the temporal mean of the state during the evolution. This showcase the need for a large Hilbert space in order to capture the dynamics faithfully.

for the different energy levels and also the final spectral weight

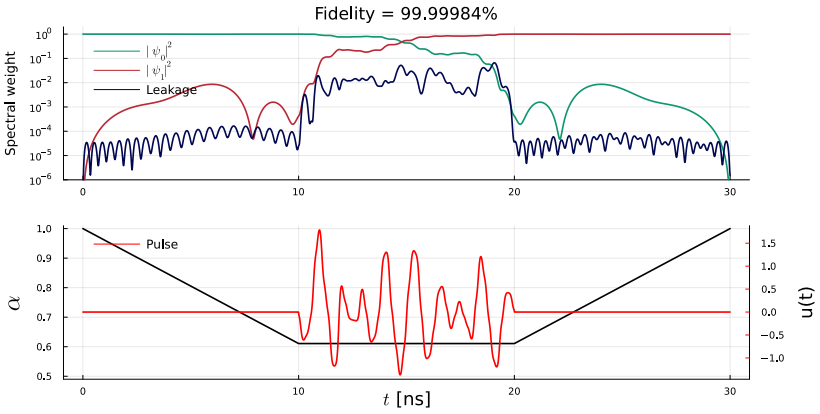
distribution of the state initialized as $|0_{\alpha=1}\rangle$. This showcase the need for rather large Hilbert spaces for faithful representations as argued in section 5. For all the results presented a Hilbert space dimension of at least 289 was chosen as a suitable compromise between speed and accuracy. The spectral weight distribution also indicates that this is adequate up to the number of significant digits quoted for the fidelities.

Keeping with the idea of leveraging the capabilities of the adjoint method to optimize more than a handful parameters, a less trivial pulse will now be considered on the same QOC problem of sec. 8.1

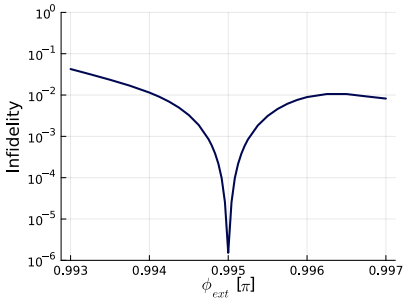
8.5 Arbitrary Waveform Generated-pulse swap gate

In order to investigate the limit of the single qubit-gate fidelity of the given qubit model, and simultaneously exploit the effectiveness of the adjoint method, a pulse with more parameters was optimized. The idea is the same as in sec. 8.1, in that the barrier is lowered linearly and a pulse is applied. However, now the pulse is parametrized just as in the case of section 8.4 described in section 6.4.3 but with a time-varying $\alpha(t)$ (eq. 6.21). It was initialized from $u(t) = 0$ and $\alpha_{\min} = 0.55$ which, in contrast to sec. 8.4, is possible due to the lowering of the barrier introducing a slight overlap and thus an adjoint state different from zero, allowing for non-zero initial gradients. The pulse is only applied during the time from 10ns to 20ns, ie. $T_a = 10$ns was fixed during the optimization.[7] So the only tuneable parameters was the 50 parameters of the interpolation method for the pulse and $\alpha_{\min}$ for $\alpha(t)$. The optimized pulse and accompanying simulation is visualized in figure 8.10 (a). The pulse is rather non-trivial and the fidelity of the swap-gate is again limited by the leakage at time T. In anticipation to the next section, concerning robustness, the sensitivity of the fidelity of the swap-gate to the external flux parameter is shown in figure 8.10 (b). The sensitivity is measured by intializing a new system for each ϕ_{ext} and just applying the same scalar functions $\alpha(t)$ and $u(t)$ for each system and measuring the resulting swap-fidelity. It is clearly optimized for the value $\phi_{\text{ext}} = 0.995\pi$ that was used during

[7] This was done for simplicity but one could just as well have let T_a be a parameter, which would have necessitated even more pulse parameters and thus a further showcase of the strength of the adjoint method. Alas this was not done.



(a) Optimized arbitrary pulse and accompanying evolution.



(b) Robustness to ϕ_{ext}

Figure 8.10: (a) Optimization of an arbitrary pulse parametrized as in sec. 6.4.3 with a sampling rate of 50/10ns = 5GHz initialized as $u(t) = 0$. The duration of the pulse was fixed to 10ns but the $\alpha_{\min}$ parameter remained tuneable and was initialized with a value of 0.55. (b) Shows the large sensitivity of the swap fidelity to the value of the external flux for this pulse.

optimization and the fidelity falls off quickly with deviations from this value.[8] A discrepancy in the external flux of only $5 \cdot 10^{-4}\Phi_0$ brings the fidelity down by 4 orders of magnitude. It should be noted that this is at least just as bad for other parametrizations (see appendix fig. 10.6). In experiment the flux noise amplitude [59] is on the order of $1.4 \pm 0.2\mu\Phi_0$ ie. $\sim 1/500$th of what is of concern here.

Nevertheless, it is useful to consider gates that will be insensitive to the specific model parameters since these will fluctuate at some level. This is done in the following section.

[8] It goes approximately as $\sim 10^4 \Delta\phi_{\text{ext}}^2$, also recall that $\phi_{\text{ext}} = 2\pi\Phi_{\text{ext}}/\Phi_0$

8.6 Robustness to external flux variation

There are several ways to go about producing robust pulses two of which have been laid out in section 6.5. These methods pose QOC problems that rely on information about the loss landscape wrt. the noisy parameters in order to promote robust solutions. Both methods will be applied in the following. The only noisy parameter for which insensitivity is pursued is the external flux, but at least the first method would be able to also consider insensitivity to eg. α_{min} and E_J variations in addition to noise in ϕ_{ext}. Noise in the external flux is related to dephasing of the qubit as the qubit frequency is practically linear in ϕ_{ext} within reasonable ranges [58].

8.6.1 Derivative information method

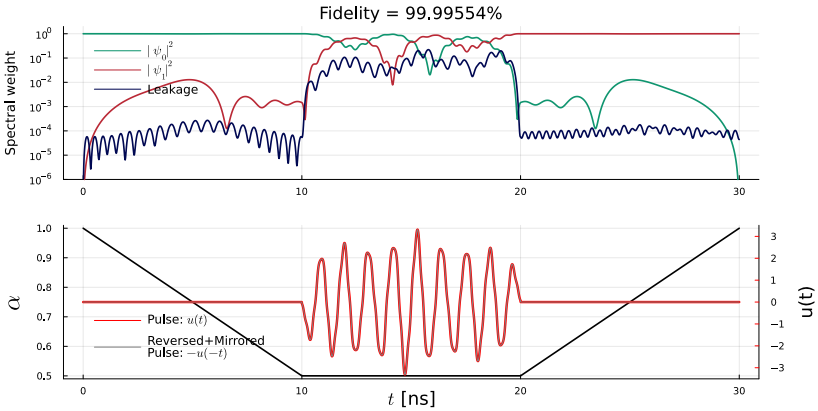
The first method rely on the gradient information of the loss landscape, to approximate the sensitivity of the objective function wrt. the fluctuating parameter, as described in section 6.5. To this end second order derivatives are desired and the second order adjoint method calculates these efficiently. The objective function employed consists of the extension of the objective function in eq. 7.89 to consider the evolution of both computational states:

$$\mathcal{L} = L^{\text{swap}} + m|\partial_{\phi_{\text{ext}}} L^{\text{swap}}| \tag{8.9}$$

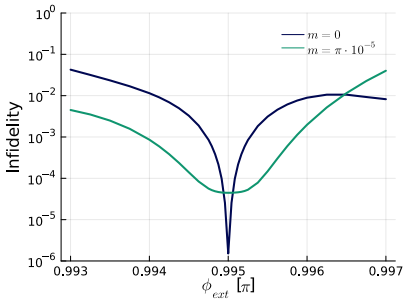
Similarly, with the simple extensions described in sec. 7.1.3, the gradient calculations can be done by eq. 7.90 - 7.100.

The parametrization is exactly the same as in sec 8.5 and with the same initialization, the only difference is the objective function. The optimization procedure yielded the result seen in figure 8.11 (a). An new curve is added to the plot for the pulse, which show that the pulse is completely anti-symmetric $u(t) = -u(-t)$. This is in accordance with the results obtained in [106] for which anti-symmetry of the pulse yields protection to second order in $\Delta\phi_{\text{ext}}$. The insensitivity of the gate[9] is verified in fig. 8.11 (b) and it can be shown that the pulse is in fact protected to second order since it scales as a quartic away from the minima at $\phi_{\text{ext}} = 0.995\pi$ (see appendix figure 10.8). However, the robustness comes at the cost of the minima being worse. So the derivative information method has helped the optimizer find a flat but slightly higher lying minima.

[9] It should be noted that the metric used only verifies that the gate still correspond to a rotation in the xy-plane of the Bloch-sphere. It does not say anything about whether two originally orthogonal gates remain orthogonal when subject to fluctuations. If the angle between them change this would diminish the applicability of the gates. Therefore, the quoted metric is not the end all be all when it comes to assessing the robustness of the gate.



(a) Optimized derivative-informed robust arbitrary pulse



(b) Robustness to ϕ_{ext}

Figure 8.11: (a) Shows the result of a QOC problem akin to sec. 8.5 but with the objective function given by eq. 8.9 which promotes robust solutions. Note the pulse is completely assymetric which protects the gate to second order in dephasing noise as stated in [106]. This is verified in (b) where the gate infidelity goes quartically in the dephasing noise caused by fluctuations in the external flux (see also fig. 10.8 in the appendix)

In order to investigate how well it is possible to retrieve both a flat and low-lying minima, the other less efficient but more accurate method will now be considered.

8.6.2 Sampling method

As established in section 6.5 eq. 6.35 can be used to develop robust controls. It is an inefficient procedure wrt. the number of uncertain parameters, but otherwise the procedure is accurate since it makes use of direct information of the loss sensitivity. Since only Φ_{ext} fluctuations will be considered, this approach is doable and will represent the best achievable gate insensitivity to external flux variations.

The implementation works as follows. A control pulse is initialized as previously with $T_a = 10\text{ns}$ (which is fixed), $\alpha_{\min} = 0.55$ (to be optimized) and $u(t) = 0$ (parametrized as the interpolation method with 50 tuneable parameters). These controls are applied on five different systems characterized by the their value of $\phi_{\text{ext}} \in \{0.994, 0.9945, 0.995, 0.9955, 0.996\}\pi$ which are propagated forward in order to evaluate the swap-infidelity for each system. The objective function is the mean of these 5 infidelities.

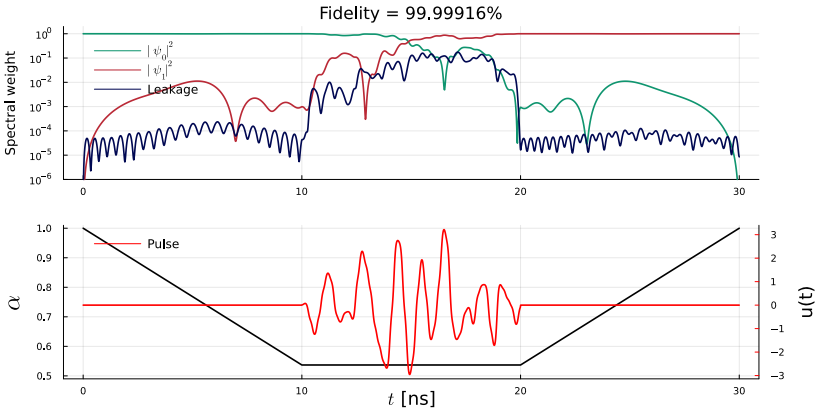
$$\mathcal{L} = \frac{1}{5}\sum_{i=0}^{5} L_i^{\text{swap}} \tag{8.10}$$

Note that it is a single control pulse that is applied to five different systems. The results are visualized in figure 8.12. The pulse is highly non-trivial, with none of the symmetry exhibited by the derivative informed robust pulse. It should be noted that during the optimization procedure, the pulse in fig. 8.12 started out being anti-symmetric with vanishing area under the cover and only began deforming to the pulse shown once the mean infidelity had hit $\sim 10^{-4}$.[10] Therefore, the pulse seemingly went beyond the analytical results of [106] and instead of raising the order of the error-suppression $\mathcal{O}(\Delta\phi_{\text{ext}}^n)$ simply focused on lowering the scaling constants.[11] It would thus be interesting to study this pulse in order to understand how the insensitivity arises and how it may be applied in other circumstances.

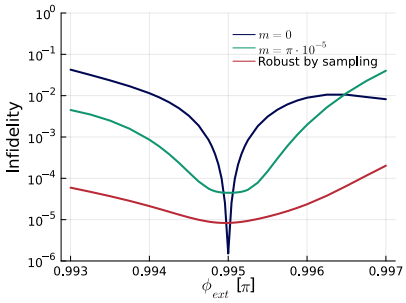
As seen in fig. 8.12 (b) it provides even better insensitivity to the

[10] These two regimes correspond to one third and two thirds of the training time, respectively. A video can be found in [107]

[11] The derivate informed pulse yielded a quartic scaling of the infidelity to $\Delta\phi_{\text{ext}}$ whereas this method yields a quadratic scaling but with a lower scaling constant of ~ 10 as oppossed to the $\sim 10^4$ of the non-robust pulse.



(a) Optimized sample-informed robust arbitrary pulse



(b) Robustness to ϕ_{ext}

Figure 8.12: (a) Shows the result of a QOC problem akin to sec. 8.5 but with the objective function given by eq. 8.10. This method is less efficient but utilizes more direct information than the method of sec. 8.6.1 and therefore achieves better insensitivity to fluctuations in the external flux, with a gate infidelity below 10^{-4} for a large range of external flux values.

value of the external flux, and represents the best attainable high-fidelity robust swap-gate. This means there is room for improvement for the derivative information method. It would be interesting to investigate whether a varying m eg. as a function of swap-infidelity as in sec. 8.2 would help the derivative informed method to obtain an even better optimum. Since this perhaps could help steer the anti-symmetric pulse of figure 8.11 into the form of fig. 8.12 by promoting the complicated shape of figure 8.10. A result in this regard can be found in appendix figure 10.7, where an initialization of $p \sim U(-0.5, 0.5)$ was used instead of $u(t) = 0$ for the procedure of sec. 8.6.1. This lead to an asymmetric pulse which still doesn't obtain the same profile as the sampling method. It would therefore be interesting to consider methods that somehow leverage the efficiency of the derivative information method, while utilizing the high-information content of the sampling method. One such method could be to use the derivative information method, but for a system, where at each step of the minimzer, the noisy parameter takes on a randomly selected value. This also introduces stochasticity into the optimization algorithm, which is something that has seen great applicability within the Machine Learning community. However, this is left as a research subject for future projects.

The end of this discussion concludes the results section.

9 Conclusion and further work

9.1 Conclusion

The work of this thesis relies on accurate simulation of the dynamics governed by the Schrödinger equation, in order to accommodate the gap between experiment and theory, such that the results obtained in simulation may also be exhibited in experiment. This necessitates the use of a large number of basis states in order to faithfully represent the quantum system numerically. The evolution of the quantum system is simulated using adaptive Runge-Kutta methods. These sophisticated integration schemes are able to assess the simulation error and consequently suppress the error to a prespecified threshold and together with the faithful representation, this realized a high number of significant digits for the results.

Estimation of the solutions to the diverse Qauntum Optimal Control problems are found through the use of Gradient-based optimization techniques. The gradients of the objective function are best calculated with the adjoint method, which is both memory efficient and scales well in the number of tuneable parameters. The calculation of both first and second order derivatives from the adjoint method are derived using Lagrange multipliers. The second order derivatives were used to efficiently treat robust QOC problems. The methodology of the thesis remain agnostic to the qubit platform, and was therefore only applied on a single example, namely that of the superconducting qubit denoted the DSFQ.

The overall methodology was successful at realizing the desired gates in simulation. The parameters for a DRAG-pulse corresponding to a high fidelity gate was found and from a single optimization procedure the pulse for the fastest gate with a fidelity above some prespecified threshold was found. Additionally, two pulses corresponding to two different non-commuting operators wrt. the computational basis was found.

Utilizing a different pulse parametrization, which was not of sinusoidal form, a gate which flips the qubit state was realized in the protected regime of the DSFQ by employing higher lying energy states during the evolution.

Lastly, pulses which correspond to high fidelity robust gates was found, where robustness refer to the fact that the gate fidelity was insensitive to the specific value of the external flux of the system, within a range of $10^{-3}\Phi_0$. The robust solutions were found by two

different methods, namely an efficient derivative informed method and a brute force sampling method, both of which were successful at reducing the sensitivity of the gate fidelity to the slow-varying fluctuations in the external flux.

9.2 Further Work

9.2.1 Adjoint method for stochastic differential equations.

The system was only simulated as a closed system and the ensuing Quantum Optimal Control solutions were only developed in this context. However, any physical qubit is an open system and should be modelled as thus. This can generally be done by describing the dynamics by a stochastic Schrödinger equation or using the density matrix formalism. Therefor it would be necessary to be able to calculate gradients for stochastic ODEs in order to employ gradient based optimization techniques. To this, an extension of the adjoint method can be made to include sensitivity analysis for stochastic ODEs [108]. Nonetheless, experimental application of software-defined pulses (ie. pulses found by simulation) has been made and resulted in a high fidelity of 99.4% showcasing the practical applicability of the work of this thesis [109].

9.2.2 Model free data driven gradient based quantum optimal control

A more speculative but nonetheless interesting approach, is to consider the derivation of section 7.1.3 where for simplicity we will now only consider the determination of the pulse parameters. The gradient of the fidelity of the state transfer from $|0\rangle$ to $|1\rangle$ is then:

$$d_p\mathcal{L} = -2\int_0^T \mathcal{I}m(\lambda^\dagger V x)\partial_p u dt \tag{9.1}$$

The point now, is that since the objective function is only dependent on $x(T)$, the adjoint state is given by the simple form of eq. 7.57, namely:

$$\lambda(t) = U(t,T)\lambda(T) = U(t,T)|1\rangle\langle 1|x(T) \tag{9.2}$$

This means, that the integrand is given by:

$$\lambda^\dagger(t)Vx(t) = x^\dagger(T)|1\rangle\langle 1|U(T,t)VU(t,0)x(0) \tag{9.3}$$

The point is that if it is possible to measure these entities on an actual quantum device, it would be possible to use gradient-based optimization techniques on a system of which you do not have to know the specific Hamiltonian. However, it is not immediately clear how one would measure the imaginary part of the integrand and one would most likely have to know how to apply the operator V.

An effort in this direction have already been made by the so called data-driven GRAPE method with an accompanying succesful application in experiment. [110] [94]

In view of eq. 9.3 the data-driven GRAPE method seemingly constitute measuring the $x^{\dagger}(T)|1\rangle$ term on the device while still numerically estimating $\langle 1|U(T,t)VU(t,0)x(0)$ by simulation.

Therefore, the data-driven GRAPE approach consitute the half-way point between the work of this thesis and a fully model free data driven approach. However, the adjoint method is what underlies it all, when it comes to calculation of the gradients, which is also referred to as sensitivity analysis.

9.2.3 Two qubit QOC with MPS representation

The naive approach for doing quantum optimal control for two-qubit gates would constitute working with a Hilbert space representation of dimension $(2n_{\max}+1)^4 \sim \mathcal{O}(10^5)$ which is impractical to say the least. Methods such as the instantaneous basis (sec. 5.1.2) seem promising on paper, but even for single qubit systems, the representation seized to capture the dynamics faithfully. A more sophisticated method for reducing the representation to its most important constituents is that of Matrix Product State Representations [111], which have already inspired algorithms in the Machine Learning community that to deal with high-dimensional problems [112] [113] [114].

This is a promising avenue for dealing with the problem of simulating the quantum system and would help make it possible to do QOC on two-qubit gate synthesis.

10 Appendix

10.1 Introduction

10.1.1 Brief overview of qubit platforms

Qubit Platform	$F_{\text{Single gate}}$	$F_{\text{Two qubit gate}}$	F_{readout}	T_1	T_2	$T_{2,\text{Dynamical Decoupling}}$
Superconducting qubits	99.97% (47) ($\sim 10-100ns$) (48,49,45)	99.5% (47)	99.8% (47)	$\sim 100\mu s$(45)	-	$\sim 20-100\mu s$ (8,45)
Gate-defined quantum dots	99.6% ($\sim 100ns$) (147,148)	98% ($\sim 200ns$)	-	-	-	$> 300\mu s$ (164)
Color centers	99.9952% ($\sim 50ns$) (204)	99.2% ($\sim 1\mu s$) (205)	-	-	1.8ms (231)	-
Ion traps	99.9999% ($10ns-5\mu s$) (269)	99.9% ($0.5-100\mu s$) (272,273)	$> 99.9\%$(269,274-276)	Practically unlimited	-	$\geq$ 1hour (270)

Table 10.1: Collection of the gate fidelities and coherence times quoted in [33]. The numbers in parenthesis refer to the number for the citation found in [33].

10.2 Simulating Quantum Systems

10.2.1 Estimation of number of signifcant digits.

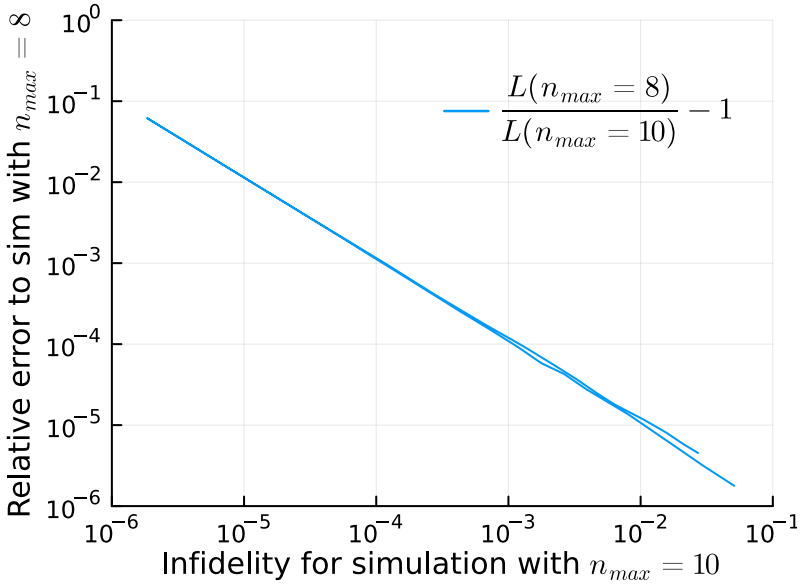


Figure 10.1: Estimation of the impact the choice of $n_{\max}$ in eq. 5.1 has on the simulation accuracy. A comparison is made that shows that even for the infidelities at the 6 digit level the relative simulation error is contained at the percent level. This is also the case for all quoted fidelities, for which there was no discrepancy between $n_{\max} = 8$ used during the optimization and the $n_{\max} \geq 10$ that was used for the presentation of the final result.

10.2.2 Learned representation

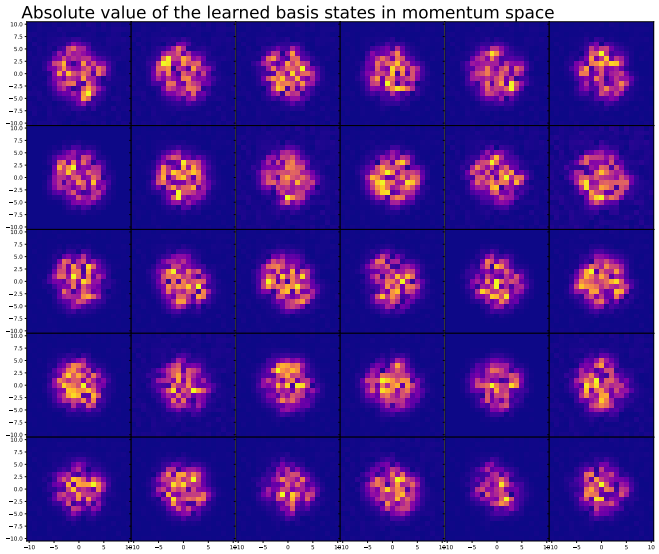


Figure 10.2: Visualization of the columns in $U(\theta)$ after normalization and Gram-Schmidt orthogonalization for which the columns then correspond to wavefunctions. The amplitude of the wavefunctions projected on to the charge basis is shown.

10.2.3 Trotter decomposition

A theory of Trotter error is no trivial task [115] especially for time-dependent Hamiltonians. The following constitute an estimation made by the author, that is not necessarily well-founded in theory. It is included here as a simple (perhaps dubious) example for inspiration to more well-founded estimations of Trotter error.

Therefore, if anyone are inspired by the following to do the same, they should do so at their own discretion. That being said, the method returned sensible estimations which will now be presented.

For the preliminary estimation of N in eq. 5.40 a one-dimensional version of the DSFQ Hamiltonian (eq. 4.23) will be considered.

$$H = 4E_C n_Q^2 - 2E_J \cos\phi + \alpha(t)E_J \cos(2\phi - \Phi_{ext}) \tag{10.1}$$

The commutator of the Hamilton evaluated at different times is wanted and with a rudimentary Poisson bracket method, one arrive at:

$$[H(t_1), H(t_2)] = -i\hbar 4^2 E_C E_J(\alpha(t_1) - \alpha(t_2)) \sin(2\phi - \Phi_{ext}) n_Q \tag{10.2}$$

For $T = 30ns$, $E_J = 2\pi 10GHz$ and $E_C = \frac{E_J}{400}$ and if one assume that "much less than one" ($\ll 1$) means "equal to 0.03" ($= 0.03$) and if we assume the largest change in α is $\frac{1}{2}$, then:

$$N = \frac{4TE_J}{2\sqrt{400} \cdot 0.03} \approx 6283 \tag{10.3}$$

With a control pulse the Hamiltonian becomes:

$$H = 4E_C n_Q^2 - 2E_J \cos\phi + \alpha(t)E_J \cos(2\phi - \Phi_{ext}) + u(t)n_Q \tag{10.4}$$

Which yields:

$$[H(t_1), H(t_2)] = i\hbar[4^2 E_C E_J(\alpha_2 - \alpha_1)\sin(2\phi - \Phi_{ext})n_Q + 2E_J(u_2 - u_1)\sin\phi + 2(\alpha_2 - \alpha_1)E_J(u_1 + u_2)\sin(2\phi - \Phi_{ext})] \tag{10.5}$$

There are different ways to assess the magnitude of the resulting operator and these are usually related to the eigenvalues of the operator. One way could (perhaps) be to add the largest eigenvalue in quadrature and if it is assumed that u is of the order $\mathcal{O}(E_J)$ then $|u_2 - u_1| \sim 2E_J \sim u_1 + u_2$, resulting in

$$N = \frac{T}{0.03}\left[\sqrt{(4^2 E_C E_J \frac{1}{2})^2 + (4E_J^2)^2 + (2E_J^2)^2}\right]^{1/2} = \frac{TE_J}{0.03}\left[\left(\frac{8}{400}\right)^2 + 20\right]^{1/4} \approx 132874 \tag{10.6}$$

Another way could perhaps be:

$$N = \frac{T}{0.03}\left[4^2 E_C E_J \frac{1}{2} + 4E_J^2 + 2E_J^2\right]^{1/2} = \frac{TE_J}{0.03}\left[\frac{8}{400} + 6\right]^{1/2} \approx 154162 \tag{10.7}$$

But these yield approximately the same order of magnitude.

10.2.4 Programming Language

As already noted the progamming language Python has been employed for this thesis. However, during the thesis it was realized that the progamming language Julia offered better methods in that in that some of the desired functions was already implemented and most importantly because it is much faster than Python, without being more complex.

This is succinctly described by a GitHub repository made by Chris Rackauckas [116] in which the figure 10.3 is contained. It should be noted this comparison is made for calculation of derivatives using an implemented adjoint method. These methods were not used for the calculation of derivatives in this method. The adjoint method was implemented by the author and can be found in [117]. This is both because the extension to complex ODEs (as is the case for the Schrödinger equation) remained elusive (at least for the torchdiffeq package) and also because it was certain a second order adjoint

Relative Performance of Forward Pass (Lower is Better)

Number of ODEs	3	28	768	3,072	12,288	49,152	196,608	786,432
DifferentialEquations.jl	1.0x	1.0x	1.0x	1.0x	1.0x	1.0x	1.0x	1.0x
DifferentialEquations.jl dopri5	1.0x	1.6x	2.8x	2.7x	3.0x	3.0x	3.9x	2.8x
torchdiffeq dopri5	4,900x	190x	840x	220x	82x	31x	24x	17x

Relative Performance of Gradient Pass (Lower is Better)

Number of Parameters	3	4	256	1,024
DifferentialEquations.jl	1.0x	1.0x	1.0x	1.0x
torchdiffeq dopri5	12,000x	1200x	----	----

Figure 10.3: Citing [116] "Shown are the timings relative to the fastest method (lower is better). For approximately 1 million ODEs and less, torchdiffeq was more than an order of magnitude slower than DifferentialEquations.jl on every tested problem, and many times substantially slower. Though note that the relative performance of torchdiffeq does increase as the number of ODEs increases. Additionally, torchdiffeq either exhibited slower gradient calculations or the gradient calculation diverged."

method was needed which is not found in these libraries. The take-away is thus, that Julia offered much faster ODEsolvers than python and the ODEsolvers of Julia [118] was subsequently used.

10.3 The Adjoint Method

10.3.1 In the context of Quantum Optimal Control

One of the celebrated results of Optimal Control Theory is Pontryagin's Maximum Principle (PMP) [82]. Stated crudely, it is based on formulating the optimization problem with constraints as a Lagrangian through the use of Lagrange Multipliers. However, instead the Hamiltonian formalism is used, which is related to the Lagrangian by the usual Legendre transformation. The PMP then provides a necessary (not sufficient) condition for the maximization of this Hamiltonian (called the pre-Hamiltonian). In essence, this is done by relating the optimal control to both the state, whose dynamics are governed by the system model at hand, and to the lagrange multiplier (which is called the adjoint state below) whose dynamics are given by the PMP as a final value problem. This is of course only possible under a set of suitable conditions, which will not be discussed here since this is only a brief overview of the established theory. However, these conditions include a constraint on the control function. The optimal control problem can then be solved by finding the initial value of the adjoint state, instead of having to search the infinite dimensional space of control functions. Following [119], consider as an example the quantum system:

$$\dot{\psi} = -i(H_0 + Vu(t))\psi \quad \text{with} \quad \psi(0) = \psi_0 \tag{10.8}$$

and the quantum optimal control problem of maximizing

$$F = \langle\psi(T)|\hat{O}|\psi(T)\rangle - \alpha \int_0^T u^2(t)dt \tag{10.9}$$

where the hermitian operator $\hat{O}$ could be the projection operator onto a target state, and the second term is the minimization of the control fluence, a necessary constraint for the straight-forward application of the PMP. By introducing the time-dependent lagrange multiplier $\lambda(t)$, the lagrangian is formed:

$$\mathcal{L} = \langle\psi(T)|\hat{O}|\psi(T)\rangle + \int_0^T \left(-\alpha u^2(t) + 2\mathcal{R}e\left[\lambda^\dagger(\dot{\psi} - i(H_0 + Vu(t))\psi)\right]\right) dt \tag{10.10}$$

Applying the PMP then yields:

$$\dot{\lambda} = -i(H_0 + Vu(t))\psi \tag{10.11}$$

$$\lambda(T) = \hat{O}\psi(T) \tag{10.12}$$

$$\alpha u(t) = \mathcal{I}m\left[\lambda^\dagger(t)V\psi(t)\right] \tag{10.13}$$

such that finding the optimal control $u(t)$ constitutes solving eq. 10.11 to 10.13 together with 10.8 for $\lambda(0)$. However, as proven in [119], the

inclusion of the constraint comes at the cost of complete controllability, in the sense that $\alpha \neq 0$ means that a 100% overlap cannot be achieved (in the case of $\mathcal{O} = |\psi_1\rangle\langle\psi_1|$)

Instead of using constrained controls, a weak-PMP can be formulated, for which the optimum condition of the PMP is switched out with an extremum condition instead. This leads to gradient-based optimization techniques such as the GRAPE algorithm [99]. However, GRAPE seems to be inherently discrete and only to first order, so instead focus of this thesis is on the adjoint method in the continuous context for which no Δt approximation is needed.

10.3.2 The adjoint state as a term in the objective function

The first encounter by the author of the "linearization of the worst-case scenario" approach (mentioned sec. 6.5) was in [120] and [121] in which case the norm of the adjoint state itself is added to the objective function. Since the adjoint state in the case of an objective function (not functional, see eq. 7.45) this augmentation promotes insensitivity of the loss to the initial state of the solution to Schrödingers equation and in this sense, it seems insensible to adopt this strategy to robust Quantum Optimal Control. Nonetheless, it is worthwhile mentioning, since it may be that there exists sensible ways of augmenting the objective function by the adjoint state, which would prove useful. At any rate, it would be worthwhile to consider.

10.3.3 Robustness methods

Barnes et. al [106] describe an analytical tool for analyzing and producing pulses that are robust with regards to dephasing. The robustness is interpreted from complex-valued curves but to second order they usually lead to anti-symmetric pulses or symmetric pulses with zero area. In section 8.6 a noise in the external flux is considered and this is related to dephasing of the system. Therefore, as already mentioned, it is interesting to note that the resulting pulse in figure 8.11 is also anti-symmetric.

The derivative method described in [86] focus on also minimizing:

$$\int_0^T \sum_i |\langle \partial_{a_i}\psi(t)|\partial_{a_i}\psi(t)\rangle|^2 dt \tag{10.14}$$

in order to produce robust pulses, where $|\partial_a\psi(t)\rangle$ is simply defined by the partial derivative of the Schrödinger equation:

$$\frac{-i}{\hbar}\frac{d|\partial_{a_i}\psi(t)\rangle}{dt} = \frac{\partial H}{\partial a_i}|\psi(t)\rangle + H|\partial_{a_i}\psi(t)\rangle \tag{10.15}$$

ie. the forward method. It would be interesting and worthwhile to consider how this may be implemented with the adjoint method.

(Since, loosely speaking, the above derivative method consists of adding the $\partial_a x$ to the objective function, this may well just be implemented by scaling the derivative informed method of eq. 6.38 by the adjoint state, since "$\partial_a x = \partial_a L/\partial_x L = \partial_a L/\lambda$")

10.4 Results

10.4.1 Fast SWAP pulse

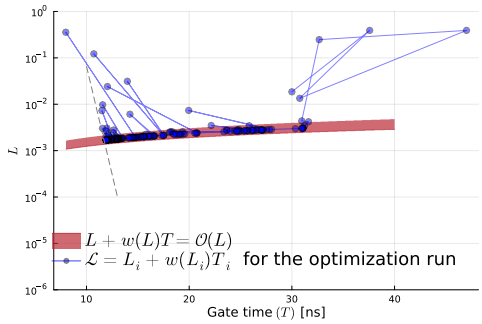


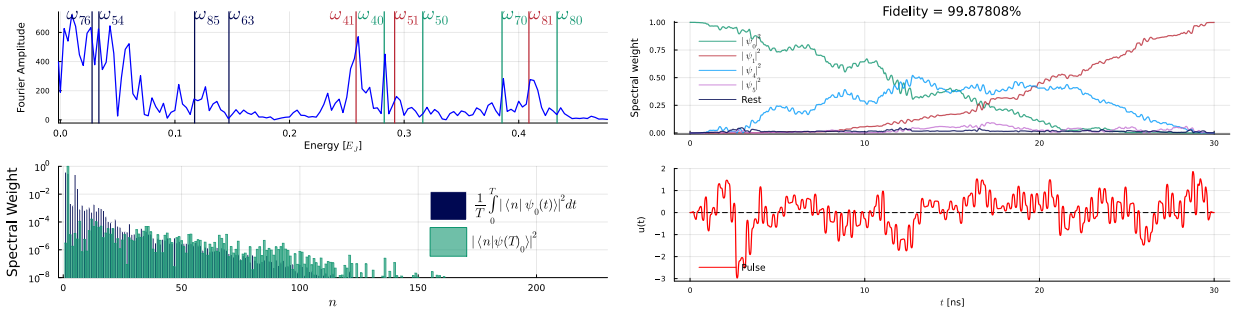
Figure 10.4: This figure shows the relation between the swap infidelity and the gate time during the optimization procedure which support the claim that the minima of the augmented objective function is restricted to the bands.

10.4.2 Swap gate for protected regime

The result shown in figure 10.5 are very much related to the QOC problem of section 8.4, except the objective function is extended to include a term that promote the constraint of the dynamics of the state to an "allowed" subset of Hilbert space. That is, the objective function is:

$$\mathcal{L} = L^{\text{swap}} + 1 - \frac{1}{T}\int_0^T \sum_{a\in\{0,1,4,5\}} |\langle\psi_a|\psi(t)\rangle|^2 dt \tag{10.16}$$

This results in the blue line of the top right figure of figure 10.5 (denoted "Rest") being suppressed, which is also clear from the distribution of spectral weight shown in 10.5 (b).



(a)

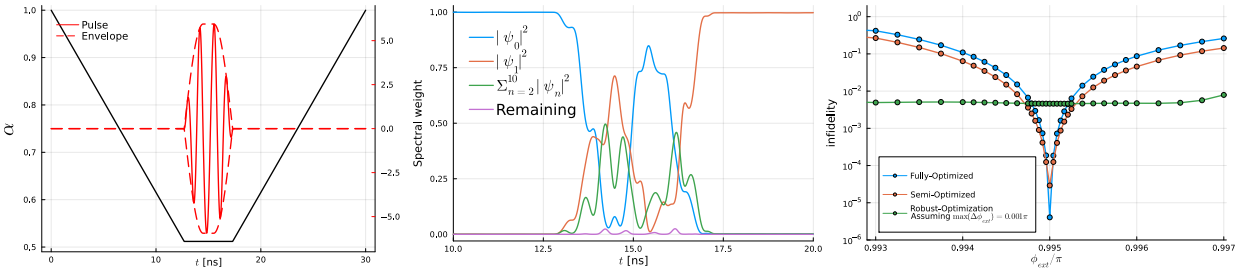
(b)

Figure 10.5: caption

10.4.3 Derivative informed robustness

The inclusion of $f \neq 0$ to the derivative informed robust objective function of eq. 6.38 was also tried out. However, this was only done for the simple DRAG pulses and therefore the minimizer was not quite able to both minimize the functional whilst finding a robust solution. But the gradient calculation was sound, which validates the derivation of sec. 7.2.1 with $f \neq 0$. It would have been interesting to have done the same for the more general interpolation method parametrization.

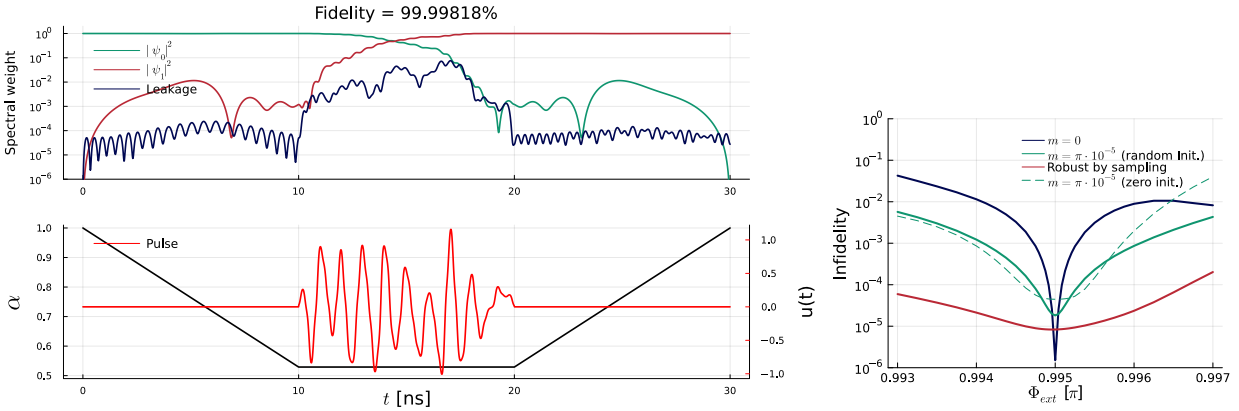
However, robustness by itself was achieveable by the simple DRAG-pulses as is evident from figure 10.6 (c)



(a) Control Pulse (b) Occupation (c) Robustness to Φ_{ext}

Figure 10.6: Fidelity at $\Phi_{\text{ext}} = 0.995\pi$ is 99.55%

Figure 10.7 is achieved by posing almost exactly the same robust QOC problem as in sec. 8.6.1, except in this case the pulse was initialized from a uniform distribution as opposed to being initialized as zero. That is, the parameters of the interpolation method used, was intialized as $p \sim U(-0.5, 0.5)$. The insensitivity is shown in fig. 10.7 (b) and whilst being a little different from the case of sec. 8.6.1 it still exhibits flatness and thereby robustness.



(a) Occupation

(b) Robustness to Φ_{ext}

Figure 10.7: Caption

This last figure (fig. 10.8) shows the scaling of the

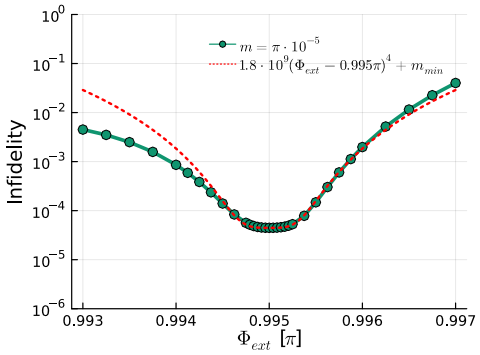


Figure 10.8: This figure shows that the sensitivity of the pulse of fig. 8.11 scales quartically. Likewise fits can done for all the other sensitivity profiles, but in those cases a quadratic scaling is needed, and therefore they are said to be sensitive to second order. (However, in the case of fig. 8.12 the scaling constant is simply lowered, rendering a quadratic but less sensitive sensitivity-profile.

Bibliography

[1] A.L. Fetter and J.D. Walecka. *Theoretical Mechanics of Particles and Continua*. Dover Books on Physics. Dover Publications, 2003.

[2] Building 37: Allan K. and Robert J. Hamilton-jacobi (part 1). https://bldg37.wordpress.com/2012/11/28/hamilton-jacobi-part-1/. Accessed: 2023-05-08.

[3] J. J. Sakurai and Jim Napolitano. *Modern Quantum Mechanics*. Cambridge University Press, 2 edition, 2017.

[4] Ben Criger, Daniel Park, and Jonathan Baugh. Few-qubit magnetic resonance quantum information processors: Simulating chemistry and physics. *Advances in Chemical Physics*, 154, 10 2012.

[5] J. Eisert and M. M. Wolf. Quantum computing. 2004.

[6] CC BY-SA 3.0 via Wikimedia Commons Smite-Meister. Wikipedia entry on bloch sphere. https://commons.wikimedia.org/wiki/File:Bloch_sphere.svg. Accessed: 2023-05-08.

[7] J. Daintith. *A Dictionary of Physics*. Oxford Paperback Reference. OUP Oxford, 2009.

[8] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[9] Mohamed Abdelhafez, Brian Baker, Andrá s Gyenis, Pranav Mundada, Andrew A. Houck, David Schuster, and Jens Koch. Universal gates for protected superconducting qubits using optimal control. *Physical Review A*, 101(2), feb 2020.

[10] Arnab Das and Bikas K. Chakrabarti. Colloquium: Quantum annealing and analog quantum computation. *Rev. Mod. Phys.*, 80:1061–1081, Sep 2008.

[11] Easwar Magesan, Robin Blume-Kohout, and Joseph Emerson. Gate fidelity fluctuations and quantum process invariants. *Physical Review A*, 84(1), jul 2011.

[12] Ray LaPierre. *Grover Algorithm*, pages 163–176. Springer International Publishing, Cham, 2021.

[13] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 212–219, New York, NY, USA, 1996. Association for Computing Machinery.

[14] P. Kaye, R. Laflamme, and M. Mosca. *An Introduction to Quantum Computing*. OUP Oxford, 2007.

[15] E. M. Stoudenmire and Xavier Waintal. Grover's algorithm offers no quantum advantage, 2023.

[16] Guillermo González-García, Rahul Trivedi, and J. Ignacio Cirac. Error propagation in nisq devices for solving classical optimization problems. *PRX Quantum*, 3:040326, Dec 2022.

[17] Raffaele Santagati, Alan Aspuru-Guzik, Ryan Babbush, Matthias Degroote, Leticia Gonzalez, Elica Kyoseva, Nikolaj Moll, Markus Oppel, Robert M. Parrish, Nicholas C. Rubin, Michael Streif, Christofer S. Tautermann, Horst Weiss, Nathan Wiebe, and Clemens Utschig-Utschig. Drug design on quantum computers, 2023.

[18] Rahul Trivedi, Adrian Franco Rubio, and J. Ignacio Cirac. Quantum advantage and stability to errors in analogue quantum simulators, 2022.

[19] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, oct 1997.

[20] A. Heifetz. *Quantum Mechanics in Drug Discovery*. Methods in Molecular Biology. Springer US, 2020.

[21] Irina Kolchurina, Yulia Klimashina, Vera Shipunova, Lyudmila Sabanova, Kristina Basite, and Maria Kolchurina. Practice-oriented technologies in the educational process as the basis for the economic development of society. 01 2020.

[22] Morten Kjaergaard, Mollie E. Schwartz, Jochen Braumüller, Philip Krantz, Joel I.-J. Wang, Simon Gustavsson, and William D. Oliver. Superconducting qubits: Current state of play. *Annual Review of Condensed Matter Physics*, 11(1):369–395, 2020.

[23] S.E. Rasmussen, K.S. Christensen, S.P. Pedersen, L.B. Kristensen, T. Bækkegaard, N.J.S. Loft, and N.T. Zinner. Superconducting circuit companion—an introduction with worked examples. *PRX Quantum*, 2(4), dec 2021.

[24] Frederik Nathan and Mark S. Rudner. Universal lindblad equation for open quantum systems. *Physical Review B*, 102(11), sep 2020.

[25] X. Y. Jin, A. Kamal, A. P. Sears, T. Gudmundsen, D. Hover, J. Miloshi, R. Slattery, F. Yan, J. Yoder, T. P. Orlando, S. Gustavsson, and W. D. Oliver. Thermal and residual excited-state population in a 3d transmon qubit. *Physical Review Letters*, 114(24), jun 2015.

[26] Bas Hensen, Wister Wei Huang, Chih-Hwan Yang, Kok Wai Chan, Jun Yoneda, Tuomo Tanttu, Fay E. Hudson, Arne Laucht, Kohei M. Itoh, Thaddeus D. Ladd, Andrea Morello, and Andrew S. Dzurak. A silicon quantum-dot-coupled nuclear spin qubit. *Nature Nanotechnology*, 15(1):13–17, dec 2019.

[27] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver. A quantum engineer's guide to superconducting qubits. *Applied Physics Reviews*, 6(2):021318, jun 2019.

[28] H. Y. Carr and E. M. Purcell. Effects of diffusion on free precession in nuclear magnetic resonance experiments. *Phys. Rev.*, 94:630–638, May 1954.

[29] S. Meiboom and D. Gill. Modified Spin-Echo Method for Measuring Nuclear Relaxation Times. *Review of Scientific Instruments*, 29(8):688–691, 12 2004.

[30] Nathalie P. de Leon, Kohei M. Itoh, Dohun Kim, Karan K. Mehta, Tracy E. Northup, Hanhee Paik, B. S. Palmer, N. Samarth, Sorawis Sangtawesin, and D. W. Steuerman. Materials challenges and opportunities for quantum computing hardware. *Science*, 372(6539):eabb2823, 2021.

[31] Morten Kjaergaard, Mollie E. Schwartz, Jochen Braumüller, Philip Krantz, Joel I.-J. Wang, Simon Gustavsson, and William D. Oliver. Superconducting qubits: Current state of play. *Annual Review of Condensed Matter Physics*, 11(1):369–395, 2020.

[32] H. J. Briegel, D. E. Browne, W. Dür, R. Raussendorf, and M. Van den Nest. Measurement-based quantum computation. *Nature Physics*, 5(1):19–26, jan 2009.

[33] IBM. IBM quantum experience backends. `https://quantum-computing.ibm.com/`. Accessed: 2023-05-08.

[34] Rajeev Acharya, Igor Aleiner, Richard Allen, Trond I. Andersen, Markus Ansmann, Frank Arute, Kunal Arya, Abraham Asfaw, Juan Atalaya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Joao Basso, Andreas Bengtsson, Sergio Boixo, Gina Bortoli, Alexandre Bourassa, Jenna Bovaird, Leon Brill, Michael Broughton, Bob B. Buckley, David A. Buell, Tim Burger, Brian Burkett, Nicholas Bushnell, Yu Chen, Zijun Chen, Ben Chiaro, Josh Cogan, Roberto Collins, Paul Conner, William Courtney, Alexander L. Crook, Ben Curtin, Dripto M. Debroy, Alexander Del Toro Barba, Sean Demura, Andrew Dunsworth, Daniel Eppens, Catherine Erickson, Lara Faoro, Edward Farhi,

Reza Fatemi, Leslie Flores Burgos, Ebrahim Forati, Austin G. Fowler, Brooks Foxen, William Giang, Craig Gidney, Dar Gilboa, Marissa Giustina, Alejandro Grajales Dau, Jonathan A. Gross, Steve Habegger, Michael C. Hamilton, Matthew P. Harrigan, Sean D. Harrington, Oscar Higgott, Jeremy Hilton, Markus Hoffmann, Sabrina Hong, Trent Huang, Ashley Huff, William J. Huggins, Lev B. Ioffe, Sergei V. Isakov, Justin Iveland, Evan Jeffrey, Zhang Jiang, Cody Jones, Pavol Juhas, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Tanuj Khattar, Mostafa Khezri, Mária Kieferová, Seon Kim, Alexei Kitaev, Paul V. Klimov, Andrey R. Klots, Alexander N. Korotkov, Fedor Kostritsa, John Mark Kreikebaum, David Landhuis, Pavel Laptev, Kim-Ming Lau, Lily Laws, Joonho Lee, Kenny Lee, Brian J. Lester, Alexander Lill, Wayne Liu, Aditya Locharla, Erik Lucero, Fionn D. Malone, Jeffrey Marshall, Orion Martin, Jarrod R. McClean, Trevor Mccourt, Matt McEwen, Anthony Megrant, Bernardo Meurer Costa, Xiao Mi, Kevin C. Miao, Masoud Mohseni, Shirin Montazeri, Alexis Morvan, Emily Mount, Wojciech Mruczkiewicz, Ofer Naaman, Matthew Neeley, Charles Neill, Ani Nersisyan, Hartmut Neven, Michael Newman, Jiun How Ng, Anthony Nguyen, Murray Nguyen, Murphy Yuezhen Niu, Thomas E. O'Brien, Alex Opremcak, John Platt, Andre Petukhov, Rebecca Potter, Leonid P. Pryadko, Chris Quintana, Pedram Roushan, Nicholas C. Rubin, Negar Saei, Daniel Sank, Kannan Sankaragomathi, Kevin J. Satzinger, Henry F. Schurkus, Christopher Schuster, Michael J. Shearn, Aaron Shorter, Vladimir Shvarts, Jindra Skruzny, Vadim Smelyanskiy, W. Clarke Smith, George Sterling, Doug Strain, Marco Szalay, Alfredo Torres, Guifre Vidal, Benjamin Villalonga, Catherine Vollgraff Heidweiller, Theodore White, Cheng Xing, Z. Jamie Yao, Ping Yeh, Juhwan Yoo, Grayson Young, Adam Zalcman, Yaxing Zhang, and Ningfeng Zhu. Suppressing quantum errors by scaling a surface code logical qubit, 2022.

[35] Guang-Wei Deng, Nan Xu, and Wei-Jie Li. *Gate-Defined Quantum Dots: Fundamentals and Applications*, pages 107–133. Springer International Publishing, Cham, 2020.

[36] Marcus W. Doherty, Neil B. Manson, Paul Delaney, Fedor Jelezko, Jörg Wrachtrup, and Lloyd C.L. Hollenberg. The nitrogen-vacancy colour centre in diamond. *Physics Reports*, 528(1):1–45, 2013. The nitrogen-vacancy colour centre in diamond.

[37] F. Jelezko. Single defect centres in diamond: A review. *physica status solidi (a)*, 203:3207 – 3225, 10 2006.

[38] M. Saffman, T. G. Walker, and K. Mølmer. Quantum information with rydberg atoms. *Rev. Mod. Phys.*, 82:2313–2363, Aug 2010.

[39] Harry Levine, Alexander Keesling, Ahmed Omran, Hannes Bernien, Sylvain Schwartz, Alexander S. Zibrov, Manuel Endres, Markus Greiner, Vladan Vuletić, and Mikhail D. Lukin. High-fidelity control and entanglement of rydberg-atom qubits. *Phys. Rev. Lett.*, 121:123603, Sep 2018.

[40] J. Chiaverini, R. B. Blakestad, J. Britton, J. D. Jost, C. Langer, D. Leibfried, R. Ozeri, and D. J. Wineland. Surface-electrode architecture for ion-trap quantum information processing, 2005.

[41] N. Read and Dmitry Green. Paired states of fermions in two dimensions with breaking of parity and time-reversal symmetries and the fractional quantum hall effect. *Phys. Rev. B*, 61:10267–10297, Apr 2000.

[42] Xiao-Liang Qi and Shou-Cheng Zhang. Topological insulators and superconductors. *Rev. Mod. Phys.*, 83:1057–1110, Oct 2011.

[43] J. Chen, B. D. Woods, P. Yu, M. Hocevar, D. Car, S. R. Plissard, E. P. A. M. Bakkers, T. D. Stanescu, and S. M. Frolov. Ubiquitous non-majorana zero-bias conductance peaks in nanowire devices. *Phys. Rev. Lett.*, 123:107703, Sep 2019.

[44]

[45] Kurt Jacobs and Daniel A. Steck. A straightforward introduction to continuous quantum measurement. *Contemporary Physics*, 47(5):279–303, sep 2006.

[46] J.F. Annett. *Superconductivity, Superfluids and Condensates*. Oxford Master Series in Physics. OUP Oxford, 2004.

[47] Uri Vool and Michel Devoret. Introduction to quantum electromagnetic circuits. *International Journal of Circuit Theory and Applications*, 45(7):897–934, jun 2017.

[48] Anders Enevold Dahl. Engineering superconducting qubits. `https://nbi.ku.dk/english/theses/masters-theses/anders-enevold-dahl/`. Accessed: 2023-05-08.

[49] A. A. Golubov, M. Yu. Kupriyanov, and E. Il'ichev. The current-phase relation in josephson junctions. *Rev. Mod. Phys.*, 76:411–469, Apr 2004.

[50] Albert Hertel, Laurits Andersen, Natalie Pearson, Malcolm Connolly, Valentina Zannier, Lucia Sorba, Liu Yu, Peter Krogstrup, Geoffrey Gardner, Michael Manfra, et al. Superconducting gatemon qubits based on selective-area-grown semiconductor materials. In *APS March Meeting Abstracts*, volume 2019, pages C29–012, 2019.

[51] Dennis Willsch, Dennis Rieger, Patrick Winkel, Madita Willsch, Christian Dickel, Jonas Krause, Yoichi Ando, Raphaël Lescanne, Zaki Leghtas, Nicholas T. Bronn, Pratiti Deb, Olivia Lanes,

Zlatko K. Minev, Benedikt Dennig, Simon Geisert, Simon Günzler, Sören Ihssen, Patrick Paluch, Thomas Reisinger, Roudy Hanna, Jin Hee Bae, Peter Schüffelgen, Detlev Grützmacher, Luiza Buimaga-Iarinca, Cristian Morari, Wolfgang Wernsdorfer, David P. DiVincenzo, Kristel Michielsen, Gianluigi Catelani, and Ioan M. Pop. Observation of josephson harmonics in tunnel junctions, 2023.

[52] 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. *Physical Review A*, 76(4), oct 2007.

[53] Xiu Gu, Anton Frisk Kockum, Adam Miranowicz, Yu xi Liu, and Franco Nori. Microwave photonics with superconducting quantum circuits. *Physics Reports*, 718-719:1–102, 2017. Microwave photonics with superconducting quantum circuits.

[54] Caspar H. van der Wal, A. C. J. ter Haar, F. K. Wilhelm, R. N. Schouten, C. J. P. M. Harmans, T. P. Orlando, Seth Lloyd, and J. E. Mooij. Quantum superposition of macroscopic persistent-current states. *Science*, 290(5492):773–777, 2000.

[55] I. Chiorescu, P. Bertet, K. Semba, Y. Nakamura, C. J. P. M. Harmans, and J. E. Mooij. Coherent dynamics of a flux qubit coupled to a harmonic oscillator. *Nature*, 431(7005):159–162, sep 2004.

[56] T. P. Orlando, J. E. Mooij, Lin Tian, Caspar H. van der Wal, L. S. Levitov, Seth Lloyd, and J. J. Mazo. Superconducting persistent-current qubit. *Physical Review B*, 60(22):15398–15413, dec 1999.

[57] Fei Yan, Simon Gustavsson, Archana Kamal, Jeffrey Birenbaum, Adam P Sears, David Hover, Ted J. Gudmundsen, Danna Rosenberg, Gabriel Samach, S Weber, Jonilyn L. Yoder, Terry P. Orlando, John Clarke, Andrew J. Kerman, and William D. Oliver. The flux qubit revisited to enhance coherence and reproducibility. *Nature Communications*, 7(1), nov 2016.

[58] Svend Krøjer, Anders Enevold Dahl, Kasper Sangild Christensen, Morten Kjaergaard, and Karsten Flensberg. Fast universal control of a flux qubit via exponentially tunable wavefunction overlap, 2023.

[59] M. D. Hutchings, J. B. Hertzberg, Y. Liu, N. T. Bronn, G. A. Keefe, Markus Brink, Jerry M. Chow, and B. L. T. Plourde. Tunable superconducting qubits with flux-independent coherence. *Physical Review Applied*, 8(4), oct 2017.

[60] T. W. Larsen, M. E. Gershenson, L. Casparis, A. Kringhøj, N. J. Pearson, R. P. G. McNeil, F. Kuemmeth, P. Krogstrup, K. D. Petersson, and C. M. Marcus. Parity-protected superconductor-semiconductor qubit. *Physical Review Letters*, 125(5), jul 2020.

[61] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in Atlantic History & Culture. Johns Hopkins University Press, 1983.

[62] H. F. Trotter. Approximation of semi-groups of operators. *Pacific Journal of Mathematics*, 8(4):887 – 919, 1958.

[63] Wilhelm Magnus. On the exponential solution of differential equations for a linear operator. *Communications on Pure and Applied Mathematics*, 7(4):649–673, 1954.

[64] *Runge–Kutta Methods*, chapter 3, pages 143–331. John Wiley Sons, Ltd, 2016.

[65] John D. Cook Consulting. Ode solver landscape. https://www.johndcook.com/blog/2020/06/12/ode-solver-landscape/. Accessed: 2023-05-11.

[66] Julien Alexandre Dit Sandretto and Alexandre Chapoutot. Validated Explicit and Implicit Runge-Kutta Methods. *Reliable Computing electronic edition*, 22, July 2016.

[67] Roger Alexander. Diagonally implicit runge-kutta methods for stiff o.d.e.'s. *SIAM Journal on Numerical Analysis*, 14(6):1006–1021, 1977.

[68] Joachim Rang. An analysis of the prothero–robinson example for constructing new adaptive esdirk methods of order 3 and 4. *Applied Numerical Mathematics*, 94:75–87, 2015.

[69] C. F. Curtiss and J. O. Hirschfelder. Integration of stiff equations. *Proceedings of the National Academy of Sciences of the United States of America*, 38(3):235–243, 1952.

[70] Desmond John Higham and Lloyd N. Trefethen. Stiffness of odes. *BIT Numerical Mathematics*, 33:285–303, 1993.

[71] Willem Hundsdorfer. On the construction of splitting methods by stabilizing corrections with runge-kutta pairs, 2017.

[72] Uri M. Ascher and Linda R. Petzold. Computer methods for ordinary differential equations and differential-algebraic equations. 1998.

[73] James H Verner. Numerically optimal runge–kutta pairs with interpolants. *Numerical Algorithms*, 53(2-3):383–396, 2010.

[74] Jim Verner. Jim verner's refuge for runge-kutta pairs. https://www.sfu.ca/~jverner/. Accessed: 2023-05-08.

[75] Christopher Alan Kennedy. *Additive Runge-Kutta schemes for convection-diffusion-reaction equations*. National Aeronautics and Space Administration, Langley Research Center, 2001.

[76] Steven R. White. Density matrix formulation for quantum renormalization groups. *Phys. Rev. Lett.*, 69:2863–2866, Nov 1992.

[77] 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, apr 2013.

[78] Steven R. White. Density-matrix algorithms for quantum renormalization groups. *Phys. Rev. B*, 48:10345–10356, Oct 1993.

[79] Mohamed Ragab Abdelhafez. Quantum optimal control using automatic differentiation. https://knowledge.uchicago.edu/record/2028. Accessed: 2023-05-11.

[80] Quentin Ficheux, Long B. Nguyen, Aaron Somoroff, Haonan Xiong, Konstantin N. Nesterov, Maxim G. Vavilov, and Vladimir E. Manucharyan. Fast logic with slow qubits: Microwave-activated controlled-z gate on low-frequency fluxoniums. *Phys. Rev. X*, 11:021026, May 2021.

[81] 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. *The European Physical Journal D*, 69(12), dec 2015.

[82] U. Boscain, M. Sigalotti, and D. Sugny. Introduction to the pontryagin maximum principle for quantum optimal control. *PRX Quantum*, 2(3), sep 2021.

[83] Wolfram Mathworld. Vector norm. https://mathworld.wolfram.com/VectorNorm.html. Accessed: 2023-05-21.

[84] Jonas Vinther, Johann Bock Severin, Jakob Hallundbæk Schauser, Christian Kragh Jespersen, and Troels Christian Petersen. Københavns universitet: Bachelorstudiet i fysik.

[85] Qiskit. Qiskit swapgate. https://qiskit.org/documentation/stubs/qiskit.circuit.library.SwapGate.html. Accessed: 2023-05-22.

[86] Thomas Propson, Brian E. Jackson, Jens Koch, Zachary Manchester, and David I. Schuster. Robust quantum optimal control with trajectory optimization, 2021.

[87] Yann Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization, 2014.

[88] Aleksandar Botev, Guy Lever, and David Barber. Nesterov's accelerated gradient and momentum as approximations to regularised update descent, 2016.

[89] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[90] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2000.

[91] Jorge J. Moré and David J. Thuente. Line search algorithms with guaranteed sufficient decrease. *ACM Trans. Math. Softw.*, 20(3):286–307, sep 1994.

[92] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[93] Steven Cundiff and Andrew Weiner. Optical arbitrary waveform generation. *Nature Photon.*, 4:760–766, 11 2010.

[94] Re-Bing Wu, Bing Chu, David H. Owens, and Herschel Rabitz. Data-driven gradient algorithm for high-precision quantum control. *Physical Review A*, 97(4), apr 2018.

[95] Keysight. M8199a arbitrary waveform generator. https://www.keysight.com/us/en/product/M8199A/arbitrary-waveform-generator-128-256-gsas.html. Accessed: 2023-05-22.

[96] F. Motzoi, J. M. Gambetta, P. Rebentrost, and F. K. Wilhelm. Simple pulses for elimination of leakage in weakly nonlinear qubits. *Physical Review Letters*, 103(11), sep 2009.

[97] Grégoire Allaire and Charles Dapogny. A linearized approach to worst-case design in parametric and geometric shape optimization. *Mathematical Models and Methods in Applied Sciences*, 24(11):2199–2257, 2014.

[98] Grégoire Allaire. A review of adjoint methods for sensitivity analysis, uncertainty quantification and optimization in numerical codes. *Ingénieurs de l'Automobile*, 836:33–36, July 2015.

[99] 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, 2005.

[100] 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, oct 2011.

[101] Andrew M. Bradley. *PDE-constrained optimization and the adjoint method*. Stanford, October 15, 2019.

[102] M. J. Yedlin and D. van Vorst. Tutorial on the continuous and discrete adjoint state method and basic implementation. https://www.crewes.org//Documents/SlideShows/

2010/CSS201014.pdf, 2010. [CREWES Meeting Talk, 22, no. 14: Accessed 22-May-2023].

[103] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations, 2019.

[104] V Rehbock, KL Teo, and LS Jennings. A computational procedure for suboptimal robust controls. *Dynamics and Control*, 2(4):331–348, 1992.

[105] Rick Groenendijk, Sezer Karaoglu, Theo Gevers, and Thomas Mensink. Multi-loss weighting with coefficient of variations, 2020.

[106] Edwin Barnes, Fernando A. Calderon-Vargas, Wenzheng Dong, Bikun Li, Junkai Zeng, and Fei Zhuang. Dynamically corrected gates from geometric space curves, 2021.

[107] Jonas Vinther. Github video in vinther repository. https://github.com/Vinther901/Quantum-Optimal-Control/blob/main/Production/RobustBySampling/anim.mp4. Accessed: 2023-05-22.

[108] Xuechen Li, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David Duvenaud. Scalable gradients for stochastic differential equations, 2020.

[109] Xian Wu, S. L. Tomarken, N. Anders Petersson, L. A. Martinez, Yaniv J. Rosen, and Jonathan L. DuBois. High-fidelity software-defined quantum logic on a superconducting qudit. *Physical Review Letters*, 125(17), oct 2020.

[110] Zhiwen Zong, Zhenhai Sun, Zhangjingzi Dong, Chongxin Run, Liang Xiang, Ze Zhan, Qianlong Wang, Ying Fei, Yaozu Wu, Wenyan Jin, Cong Xiao, Zhilong Jia, Peng Duan, Jianlan Wu, Yi Yin, and Guoping Guo. Optimization of a controlled- mml:math xmlns:mml="http://www.w3.org/1998/math/MathML" display="inline" overflow="scroll"mml:miz/mml:mi/mml:math gate with data-driven gradient-ascent pulse engineering in a superconducting-qubit system. *Physical Review Applied*, 15(6), jun 2021.

[111] D. Perez-Garcia, F. Verstraete, M. M. Wolf, and J. I. Cirac. Matrix product state representations, 2007.

[112] Michael Kastoryano and Nicola Pancotti. A highly efficient tensor network algorithm for multi-asset fourier options pricing, 2022.

[113] Ivan Oseledets. Tensor-train decomposition. *SIAM J. Scientific Computing*, 33:2295–2317, 01 2011.

[114] Jacob Biamonte and Ville Bergholm. Tensor networks in a nutshell, 2017.

[115] Andrew M. Childs, Yuan Su, Minh C. Tran, Nathan Wiebe, and Shuchen Zhu. Theory of trotter error with commutator scaling. *Physical Review X*, 11(1), feb 2021.

[116] Christopher Rackauckas. Torchdiffeq vs differentialequations.jl (/ diffeqflux.jl) neural ode compatible solver benchmarks. https://gist.github.com/ChrisRackauckas/cc6ac746e2dfd285c28e0584a2bfd320. Accessed: 2023-05-21.

[117] Jonas Vinther. Quantum optimal control repository of vinther. https://github.com/Vinther901/Quantum-Optimal-Control. Accessed: 2023-05-21.

[118] OrdinaryDiffEq.jl. Julia library for odesolvers. https://github.com/SciML/OrdinaryDiffEq.jl/blob/master/src/algorithms.jl. Accessed: 2023-05-08.

[119] J. Werschnik and E. K. U. Gross. Quantum optimal control theory, 2007.

[120] Grégoire Allaire. A review of adjoint methods for sensitivity analysis, uncertainty quantification and optimization in numerical codes. *Ingénieurs de l'Automobile*, 836:33–36, July 2015.

[121] Tobias Wöhrer. Robust neural odes — a second order adjoint sensitivity approach. https://www.benasque.org/2022pde/talks_contr/296_woehrer.pdf. Accessed: 2023-05-22.