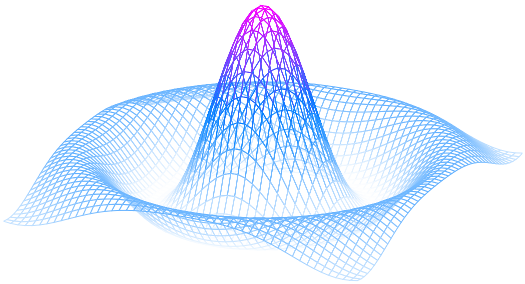
CONTROLLING A SUPERCONDUCTING QUANTUM COMPUTER

DANIEL COHEN HILLEL



Advanced Project in Physics A (20382)

Supervisor: Dr. Serge Rosenblum

Daniel Cohen Hillel: *Controlling a Superconducting Quantum Computer,* Advanced Project in Physics A (20382), © November 2019

ABSTRACT

Quantum computing presents many challenges. Here, we try to tackle one of them: How do we control the quantum computer? What do you need to send to make what you want happen? How do you physically construct the control pulses and what do the pulses even look like? These are all question we are going to try to answer throughout this project.

This manuscript starts with an introduction to quantum computing. Concepts such as the qubit and operations are introduced there.

In the second chapter we introduce the quantum system. We use quantum optics and the Jaynes-Cumming model to model the behavior of the system.

The third chapter is the main subject of this project, quantum optimal control and the **GR**adient **A**scent **P**ulse **E**ngineering (**GRAPE**) algorithm. In this chapter we show how can you use the system characterization we made in chapter 2 to find the pulses you need to send to control the quantum computer as you want.

The fourth and last chapter describes the physical implementation of the transmission of the pulses we found in the previous chapter. We show how the entire system is connected, from the AWG[1] to the actual qubits. We discuss the challenges of creating the pulses and how to solve these problems.

All the codes used in and created for this project, results, and references are available at

https://github.com/DanielCohenHillel/Controlling-a-Superconducting-Quantum-Computer

[1] Arbitrary Waveform Generator

CONTENTS

1 INTRODUCTION

1.1 WHAT IS A QUANTUM COMPUTER?

A classical computer is, essentially, a calculator, not of real numbers but of *binary numbers*. A *binary digit* ("*bit*" from now on) can be in one of two states, usually represented by 0 and 1. We can use *logic gates* to control and manipulate bits to do various calculations. These are the building blocks of the classical computer. With the ability to do calculations on bits, and the ability to store bits in the memory we can construct a computer.

So what is a quantum computer then? Well, if the classical computer uses bits to do calculations, a quantum computer uses *quantum bits* ("*qubits*" from now on) for calculations. A qubit, much the same as a bit, has two states, a 0 state and a 1 state (denoted by $|0\rangle$ and $|1\rangle$ for reasons we'll see later). The principal difference between bits and qubits is that a qubit can be in a *superposition* of the two states. We can use this property to our advantage by manipulating the state of a quantum computer so that desired outcomes interfere constructively, whereas undesired outcomes interfere destructively.

1.2 ALGORITHMS AND FURTHER MOTIVATION

> "Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy."
>
> - Richard Feynman

The possibilities that quantum computation allow are unprecedented. From simulation of drugs for developments of new cures to unbreakable encryption, quantum computing promises a lot.

One of the most famous algorithms in quantum computing is *Shor's algorithm*, a quantum algorithm for factoring large numbers. In classical computing the way to factor a number is to verify if small numbers divide it, one by one. Modern encryption method require you to factor a large number[1]. With classical computers this is a nearly impossible problem, since solving this problem requires a time that scales exponentially with the number of bits. This means that if we increase the size of the number that we need to factor by just just one bit, the time required to factorize the number increases by a factor of two! Shor's algorithm, on the other hand, uses the power of quantum computing to solve this problem in polynomial time![2]. This means that increasing the size of a number to be factorized could quickly become an impossible task for a classical computer, it won't affect a quantum computer as much.

While Shor's algorithm is a great example of the power of quantum computers, and it is also probably the most famous quantum algorithm, it is by no means the

[1] This is called RSA encryption

[2] The actual complexity is more detailed then this but it is meant to show the rough idea

most interesting example. Decrypting messages and breaking the world's cryptography isn't really a good motivator to try to create quantum computers. Let's look at a more useful, optimistic algorithm, *Grover's Algorithm*.

Grover's algorithm is a quantum algorithm that finds the unique input to a black box function that produces a particular output value, using just $o(\sqrt{N})$ evaluations of the function, where N is the size of the function's domain. For a classical algorithm to do this it would take $o(N)$ evaluations. Therefore, Grover's algorithm provides us with a quadratic speedup. Roughly speaking, given function $y = f(x)$ Grover's algorithm finds x when given a specific y. This algorithm could be used to search in databases quadratically faster then with a classical computer.

There are more quantum algorithms that were developed in the last several decades, and even more algorithms that have yet to be developed that might have impactful applications in the future.

1.3 QUBITS AND QUANTUM GATES

Physically, a qubit is a two level quantum system. We call the first level $|0\rangle$ and the second level $|1\rangle$. As we know from quantum mechanics, the qubit could be in a superposition of the two states.

Mathematically, we think of qubits as 2-dimensional vectors, where the first term corresponds to the $|0\rangle$ state and the second term corresponds to the $|1\rangle$ state, so a qubit in a state $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ can be represented as

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

The complex pre-factors of each state are called probability amplitudes, since they are related to the probability of the qubit to be in that state. The probability is given by the absolute value of that state squared

$$P(|i\rangle) = |\langle i|\psi\rangle|^2$$

In the example I just gave, the qubit has a 50% chance to be in the $|0\rangle$ state and a 50% chance to be in the $|1\rangle$ state.

In this world of qubits as vectors, we think of logic gates (*quantum gates*), as unitary matrices. When the qubit goes through a logic gate, the resulting state is obtained by multiplying the initial state by the matrix. Let's look at an example for one of the simplest logic gates we have in classical computing, the NOT gate. A quantum implementation of the NOT gate takes $|0\rangle$ to $|1\rangle$ and $|1\rangle$ to $|0\rangle$). The matrix that achieves this is

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Known as $\hat{\sigma}_x$ (Pauli matrix X). As a simple example to see how this works, if we input $|0\rangle$ into the NOT quantum gate, we get as a result

$$\mathrm{NOT}\,|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$$

As we expected, NOT $|0\rangle$ is $|1\rangle$. There are infinite 1-qubit quantum gates, while there are only four possible one bit gates on a classical computer[3].

The last thing we need to know to understand the basic of quantum computing, is how to represent multiple qubits. If we have several of qubits in our system, we think of all the qubits together as one vector that is the *tensor product*[4] of all the qubits. Let's say we have a $|0\rangle$ and $|1\rangle$ qubits in our system, we represent that by $|01\rangle$ and it is equal to

$$|01\rangle = |0\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

The tensor product of N qubits has 2^N coefficients! This is yet another clue of the power that quantum computers have compared to classical computers. A quantum gate on multiple qubits is thus a 2^N by 2^N square matrix.

Now that we have the basic tools of quantum computing, we can use them to get motivation for the amazing things quantum computers can do

1.4 SUPERCONDUCTING QUANTUM COMPUTERS

The physical implementation of the qubit itself isn't the subject of this project but we still look on how would one implement such a device. A problem we have to face when making a quantum computer is what physical phenomenon would be the qubit. We need some sort of two level system that we can easily measure and manipulate, while also staying coherent[5] and usable. For classical computers we already have this figured out for years, a bit is a voltage on a wire, 1 is when there is voltage on the wire and 0 if there's none, simple. For a quantum computer this is much more complicated, there are many quantum phenomena we can use as our qubit, such as the energy level of an atom, the spin of an electron, the polarization of photons and so on. It is not so obvious what should be the physical realisation of the qubit. This project is about a *superconducting* quantum computer, with superconducting qubits.

Superconducting qubits are microwave circuits in which the cooper-pair condensate effectively behaves as a one-dimensional quantized particle. By inserting Josephson junctions, the circuit can be made nonlinear, allowing us to isolate the qubit from higher-energy levels and treat it as a two level system (instead of the many level system that it really is).

This topic is covered in appendix B, refer there for any additional information, preferably read the appendix after reading chapter 2.

[3] These are: Identity, NOT, always 1, and always 0

[4] Represented as a Kronecker product for the dimensions sake

[5] Coherence is a big subject, you can think of it as a fancy way of saying that the qubit still holds information without being corrupted. This is the main problem facing quantum computing right now

1.5 REFERENCES AND FURTHER READINGS

The main references for this chapter were two great books on quantum information I recommend everyone to read. The first is the well known, *de-facto* book on quantum computation and information. "**Quantum Computation and Quantum Information**" written by Michael Nielsen and Isaac Chuang also known as "Mike and Ike". This book covers everything.

The other book I used was "**Quantum Computing for Computer Scientists**" by Mirco A. Mannucci and Noson S. Yanofsky. It has, in my opinion, clearer explanations on the pure mathematical nature of quantum information, although it is not as comprehensive as "Mike and Ike".

Part I

QUANTUM THEORY

In this part we consider the quantum nature of the problem. We start with a short introduction to quantum optics, the theoretical basis for bosonic quantum computers. Then go on to the main part of the project, quantum optimal control.

2 QUANTUM OPTICS

This chapter introduces concepts in *Quantum Optics*. The implementation of quantum computers we discuss in this manuscript has a qubit which is a two level system[1] interacting with light (both classical and quantum) inside a resonating cavity.

2.1 DIRAC'S METHOD FOR CANONICAL QUANTIZATION

Before we dive into anything new, we'll start by going over the method to quantize any oscillating phenomena introduced by Paul Dirac in his 1925 Ph.D. dissertation.

Any system of which we have a classical description can be quantized following a process known as canonical quantization. A classical system can be described by pairs of *canonically conjugate variables*, (q_j, p_j) satisfying the Hamilton equations

$$\begin{aligned} \dot{q}_j &= \frac{\partial \mathcal{H}}{\partial p_j} \\ \dot{p}_j &= -\frac{\partial \mathcal{H}}{\partial q_j} \end{aligned}$$

q_j is called the *canonical coordinate* and p_j is called the *canonically conjugate momentum to the coordinate* q_j.

To quantize the system we need to replace the dynamical variables (q_j, p_j) with canonically conjugate operators $(\hat{q}_j, \hat{p}_j)$ satisfying the commutation relation

$$[\hat{q}_j, \hat{q}_k] = [\hat{p}_j, \hat{p}_k] = 0 \quad \text{and} \quad [\hat{q}_j, \hat{p}_k] = i\hbar\delta_{jk}$$

The Hamiltonian of the quantum system is obtained by replacing the classical Hamiltonian $E = \mathcal{H}(q_1, p_1, \ldots, q_j, p_j, \ldots)$ with the quantum Hamiltonian $\hat{\mathcal{H}} = \mathcal{H}(\hat{q}_1, \hat{p}_1, \ldots, \hat{q}_j, \hat{p}_j, \ldots)$.

Dirac's method is best used to solve the system of the quantum harmonic oscillator. The Hamiltonian of the system is

$$\hat{\mathcal{H}} = \frac{\hat{p}^2}{2m} + \frac{1}{2}m\omega^2\hat{q}^2$$

To solve it you introduce the annihilation and creation operators (Also called the "ladder operators"), $\hat{a}$ and $\hat{a}^\dagger$ respectively. The ladder operators satisfy the commutation relation $[\hat{a}, \hat{a}^\dagger] = 1$. Since $\hat{a} \neq \hat{a}^\dagger$, these operators are not hermitian and therefore not observables of the system, but every observable of the system can be expressed using them. The operators are

$$\begin{aligned} \hat{a} &= \sqrt{\frac{m\omega}{2\hbar}}(\hat{q} + \frac{i}{m\omega}\hat{p}) \\ \hat{a}^\dagger &= \sqrt{\frac{m\omega}{2\hbar}}(\hat{q} - \frac{i}{m\omega}\hat{p}) \end{aligned}$$

[1] Often I'll use the words "atom" and "qubit" interchangeably. The qubit acts as a two level atom

The Hamiltonian is therefore

$$\hat{\mathcal{H}} = \hbar\omega(\hat{a}^\dagger\hat{a} + \frac{1}{2})$$

The energy eigenstates are denoted $|0\rangle, |1\rangle, |2\rangle, \ldots$ where $|n\rangle$ is the eigen state of the n^{th} energy level, E_n. A set of important relations between the eigenstates the the ladder operators are

$$\hat{a}^\dagger|n\rangle = \sqrt{n+1}\,|n+1\rangle$$
$$\hat{a}\,|n\rangle = \sqrt{n}\,|n-1\rangle$$
$$\hat{a}\,|0\rangle = 0$$

From these relations the ladder operators get their name, since you can think of them as ways to climb up and down the energy "ladder". These eigenstates are called the **number states** or **Fock states** after Vladimir Fock who developed this representation. The number states $|n\rangle$ are also the eigenstates of the *number operator* $\hat{N} = \hat{a}^\dagger\hat{a}$, with eigenvalue n, $\hat{N}\,|n\rangle = n\,|n\rangle$, hence the name.

2.2 THE QUANTIZATION OF THE ELECTROMAGNETIC FIELD

2.2.1 *The Homogeneous Electromagnetic Equation*

Maxwell's equations in free space are

$$\nabla \cdot \mathbf{E} = 0 \tag{1a}$$
$$\nabla \cdot \mathbf{B} = 0 \tag{1b}$$
$$\nabla \times \mathbf{E} = \frac{\partial \mathbf{B}}{\partial t} \tag{1c}$$
$$\nabla \times \mathbf{B} = \mu_0\epsilon_0\frac{\partial \mathbf{E}}{\partial t} \tag{1d}$$

Taking the curl of 1c and 1d yields

$$\nabla \times (\nabla \times \mathbf{E}) = \nabla \times (-\frac{\partial \mathbf{B}}{\partial t}) = -\frac{\partial}{\partial t}(\nabla \times \mathbf{B}) = -\mu_0\epsilon_0\frac{\partial^2 \mathbf{E}}{\partial t^2} \tag{2a}$$
$$\nabla \times (\nabla \times \mathbf{B}) = \nabla \times (\mu_0\epsilon_0\frac{\partial \mathbf{E}}{\partial t}) = \mu_0\epsilon_0\frac{\partial}{\partial t}(\nabla \times \mathbf{E}) = -\mu_0\epsilon_0\frac{\partial^2 \mathbf{B}}{\partial t^2} \tag{2b}$$

Using the vector identity

$$\nabla \times (\nabla \times \mathbf{V}) = \nabla(\nabla \cdot \mathbf{V}) - \nabla^2\mathbf{V} \tag{3}$$

We obtain from 2a and 2b

$$\nabla(\nabla \cdot \mathbf{E}) - \nabla^2\mathbf{E} = -\mu_0\epsilon_0\frac{\partial^2 \mathbf{E}}{\partial t^2} \tag{4a}$$
$$\nabla(\nabla \cdot \mathbf{B}) - \nabla^2\mathbf{B} = -\mu_0\epsilon_0\frac{\partial^2 \mathbf{B}}{\partial t^2} \tag{4b}$$

Using 1a and 1b to cancel the left most term we get

$$\nabla^2\mathbf{E} = \mu_0\epsilon_0\frac{\partial^2 \mathbf{E}}{\partial t^2} \tag{5a}$$
$$\nabla^2\mathbf{B} = \mu_0\epsilon_0\frac{\partial^2 \mathbf{B}}{\partial t^2} \tag{5b}$$

Replacing $v_{ph} = \frac{1}{\sqrt{\mu_0 \epsilon_0}}$ with c since the phase velocity is the speed of light for electromagnetic radiation in vacuum

$$\begin{aligned} \nabla^2 \mathbf{E} &= \frac{1}{c^2}\frac{\partial^2 \mathbf{E}}{\partial t^2} \\ \nabla^2 \mathbf{B} &= \frac{1}{c^2}\frac{\partial^2 \mathbf{B}}{\partial t^2} \end{aligned} \tag{6}$$

These equation are called *the homogeneous electromagnetic wave equations*. We'll pick a polarization, arbitrarily, to be in the x direction. The equations become

$$\begin{aligned} \frac{\partial^2 E_x}{\partial x^2} &= \frac{1}{c_0^2}\frac{\partial^2 E_x}{\partial t^2} \\ \frac{\partial^2 B_y}{\partial y^2} &= \frac{1}{c_0^2}\frac{\partial^2 B_y}{\partial t^2} \end{aligned} \tag{7}$$

2.2.2 *The Single Mode Cavity*

Now that we have the homogeneous electromagnetic field equations at hand we can solve them. We solve 7 using separation of variables,

$$E_x(z,t) = Z(z)T(t)$$

Yielding the solution,[2]

$$\begin{aligned} E_x(z,t) &= \sqrt{\frac{2\omega_c^2}{V\epsilon_0}}q(t)\sin kz \\ B_y(z,t) &= \sqrt{\frac{2\mu_0}{V}}\dot{q}(t)\cos kz \end{aligned} \tag{8}$$

where V is the effective volume of the cavity, q is a time-dependent amplitude with units of length, $k = m\pi/L$ for an integer $m > 0$, and ω_c is the frequency of the mode.

The Hamiltonian of a single mode is hence given by

$$\mathcal{H} = \frac{1}{2}\int \epsilon_0 \mathbf{E}^2 + \frac{\mathbf{B}^2}{\mu_0} dV \tag{9}$$

$$= \frac{1}{2}\int \epsilon_0 E_x^2(z,t) + \frac{B_y^2(z,t)}{\mu_0} dz \tag{10}$$

$$= \frac{1}{2}[\dot{q}^2(t) + \omega_c^2 q^2(t)] \tag{11}$$

[2] We've skipped several of steps going from the differential equation to their solutions, mainly it is not clear how the scalar constant in front of the expression got there, it has to do with the fact that the total energy of the electromagnetic field is given by $\mathcal{H} = \frac{\epsilon_0}{2}\int_V \mathbf{E}^2 + c^2\mathbf{B}^2 d^3r$. We won't go into the calculations because they make it much more complicated and doesn't give much more insights into the physics

Going from dynamical variables to operators, $\hat{q}$ and $\hat{p}$ that satisfy the commutation relation $[\hat{q}, \hat{p}] = i\hbar$, we get

$$\hat{E}_x(z,t) = \sqrt{\frac{2\omega_c^2}{V\epsilon_0}}\hat{q}(t)\sin kz \tag{12}$$

$$\hat{B}_y(z,t) = \sqrt{\frac{2\mu_0}{V}}\hat{p}(t)\cos kz \tag{13}$$

$$\hat{\mathcal{H}} = \frac{1}{2}[\hat{p}^2(t) + \omega_c^2\hat{q}^2(t)] \tag{14}$$

This is the same Hamiltonian as for the harmonic oscillator. **Electromagnetic radiation acts as a harmonic oscillator**.

Let's introduce creation and annihilation operators

$$\hat{a}(t) = \frac{1}{\sqrt{2\hbar\omega_c}}[\omega_c\hat{q}(t) + i\hat{p}(t)]$$
$$\hat{a}^\dagger(t) = \frac{1}{\sqrt{2\hbar\omega_c}}[\omega_c\hat{q}(t) - i\hat{p}(t)]$$

In term of the creation and annihilation operators, the electric and magnetic field operators become

$$\hat{E}_x(z,t) = E_0[\hat{a}(t) + \hat{a}^\dagger(t)]\sin kz \tag{15}$$

$$\hat{B}_y(z,t) = \frac{E_0}{c}[\hat{a}(t) - \hat{a}^\dagger(t)]\cos kz \tag{16}$$

And we can write the Hamiltonian as

$$\hat{\mathcal{H}}_{cavity} = \hbar\omega_c[\hat{a}^\dagger\hat{a} + \frac{1}{2}] \approx \hbar\omega_c\hat{a}^\dagger\hat{a} \tag{17}$$

Ignoring the zero-point energy $\frac{\hbar\omega_c}{2}$.

Since the eigenstates of the quantum harmonic oscillator are the number states $|n\rangle$, they are also the eigenstates of electromagnetic radiation. We can show that the momentum operator of electromagnetic radiation takes the form $\hat{P} = \hbar\mathbf{k}\hat{a}^\dagger\hat{a}$. Where $\mathbf{k}$ is the wave number of the electromagnetic wave. Applying the momentum operator to the number states we see that

$$\hat{P}|n\rangle = \hbar\mathbf{k}\hat{a}^\dagger\hat{a}|n\rangle = n\hbar\mathbf{k}|n\rangle$$

This is an important result, **the $|n\rangle$ state has well defined energy and momentum, same as n particles, each with energy $\hbar\omega$ and momentum $\hbar$k, we call these particles *photons*.** Hopefully you now see why these are called number states, they correspond to the *number of photons in the cavity*.

2.3 THE JAYNES–CUMMINGS MODEL

Our goal is to mathematically model the Hamiltonian of a system of a two-level system, such as an atom, interacting with a single quantized mode of an electromagnetic field inside an optical cavity. First we'll divide the system into 3 parts, The atom, the cavity, and the interaction between them.

2.3.1 *The Hamiltonians*

We separate the Hamiltonian as

$$\mathcal{H} = \mathcal{H}_{atom} + \mathcal{H}_{cavity} + \mathcal{H}_{interaction}$$

We'll now calculate each Hamiltonian separately

CAVITY – We already calculated the Hamiltonian of the cavity and it is given by equation 17 as

$$\boxed{\hat{\mathcal{H}}_{\text{cavity}} = \hbar\omega_c \hat{a}^\dagger \hat{a}} \tag{18}$$

ATOM – The atom is a two-level system, meaning its state is in a general superposition of the ground, $|g\rangle$, and excited, $|e\rangle$, states. We know that $\hat{\mathcal{H}}\,|\psi\rangle = E_\psi\,|\psi\rangle$ for every energy eigenstate, with energy E_ψ. We can use these eigenstates to spectrally decompose the Hamiltonian, $\hat{\mathcal{H}} = \sum_\psi E_\psi\,|\psi\rangle\,\langle\psi|$. In our case, with only ground and excited states, the Hamiltonian is

$$\hat{\mathcal{H}}_{atom} = E_g\,|g\rangle\,\langle g| + E_e\,|e\rangle\,\langle e| \tag{19}$$

Using the vector representation of these states we'll write

$$\begin{aligned}\hat{\mathcal{H}}_{atom} &= E_e \begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix} + E_g \begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix} = \begin{bmatrix}E_e & 0\\ 0 & E_g\end{bmatrix}\\ &= \frac{1}{2}(E_g + E_e)\mathbb{I} + \frac{1}{2}(E_e - E_g)\hat{\sigma}_z\end{aligned}$$

Again, we define the zero point energy so that the first term becomes 0. The energy difference is associated with a frequency ω_a and from the de-broglie relations $E = \hbar\omega$ so $E_e - E_g = \hbar\omega_a$. The atom Hamiltonian is therefore

$$\boxed{\hat{\mathcal{H}}_{atom} = \frac{1}{2}\hbar\omega_a\hat{\sigma}_z} \tag{20}$$

INTERACTION – The atom-cavity coupling comes from the interaction between the atomic dipole, $\hat{\mathbf{d}}$, and the electric field of the cavity mode, $\hat{\mathbf{E}}$. The corresponding Hamiltonian is

$$\hat{\mathcal{H}}_{interaction} = -\hat{\mathbf{d}}\cdot\hat{\mathbf{E}} = -\hat{d}E_0 \sin kz(\hat{a} + \hat{a}^\dagger)$$

Where we assumed that the atomic dipole and electric field are parallel[3]. We'll introduce the atomic transition operators

$$\hat{\sigma}_+ = |e\rangle\,\langle g|\,, \qquad \hat{\sigma}_- = |g\rangle\,\langle e| = \hat{\sigma}_+^\dagger$$

Due to parity selection rules, only the off-diagonal, only the off-diagonal elements of the dipole operator are nonzero so we may write

$$\hat{d} = d\,|e\rangle\,\langle g| + d^*\,|g\rangle\,\langle e| = d\hat{\sigma}_- + d^*\hat{\sigma}_+ = d(\hat{\sigma}_+ + \hat{\sigma}_-)$$

[3] Since the electric field creates the dipole in the direction of the field

Thus, the interaction Hamiltonian is

$$\hat{\mathcal{H}}_{interaction} = \hbar\Omega(\hat{\sigma}_+ + \hat{\sigma}_-)(\hat{a} + \hat{a}^\dagger) = \hbar\Omega(\hat{\sigma}_+\hat{a} + \hat{\sigma}_+\hat{a}^\dagger + \hat{\sigma}_-\hat{a} + \hat{\sigma}_-\hat{a}^\dagger) \tag{21}$$

Where $\Omega = -\frac{d}{\hbar}\sqrt{\frac{\hbar\omega_c}{\epsilon_0 V}}\sin(kz)$ is the amplitude of the interaction. From 17 the cavity ladder operators $\hat{a}$ and $\hat{a}^\dagger$ evolve as

$$\hat{a}(t) = \hat{a}(0)e^{-i\omega_c t}, \qquad \hat{a}^\dagger(t) = \hat{a}^\dagger(0)e^{i\omega_c t} \tag{22}$$

And similarly we can show for the free evolution of the atom

$$\hat{\sigma}_\pm(t) = \hat{\sigma}_\pm(0)e^{\pm i\omega_a t} \tag{23}$$

We can write while approximating $\omega_c \approx \omega_a$

$$\begin{aligned} &\hat{\sigma}_+\hat{a} \sim e^{i(\omega_0-\omega)t} \\ &\hat{\sigma}_-\hat{a}^\dagger \sim e^{-i(\omega_0-\omega)t} \\ &\hat{\sigma}_+\hat{a}^\dagger \sim e^{i(\omega_0+\omega)t} \\ &\hat{\sigma}_-\hat{a} \sim e^{-i(\omega_0+\omega)t} \end{aligned} \tag{24}$$

We can see that the last two term vary much more rapidly than the first two. Furthermore, the last two terms do not conserve energy (They correspond to *[photon addition + atom excitation]* and *[photon reduction + atom relaxation]*). Therefore, we can drop the quickly rotating terms[4], yielding

$$\boxed{\mathcal{H}_{interaction} = \hbar\Omega(\hat{\sigma}_+\hat{a} + \hat{\sigma}_-\hat{a}^\dagger)} \tag{25}$$

Finally, we can write the full JC Hamiltonian

$$\boxed{\hat{\mathcal{H}} = \frac{1}{2}\hbar\omega_0\hat{\sigma}_z + \hbar\omega\hat{a}^\dagger\hat{a} + \hbar\Omega(\hat{\sigma}_+\hat{a} + \hat{\sigma}_-\hat{a}^\dagger)} \tag{26}$$

2.3.2 *Interaction with a Classical Field*

As you might have noticed, we are classical creatures, I'm (fortunately) not in a superposition of being dead and alive at the same time. As classical creatures, if we want to interact with the quantum world we need to do so with classical means, with a classical interface to the quantum world. Back to the Jaynes-Cummings model, what happens when we introduce a classical electromagnetic field (such as the microwave drives that control the superconducting system)?

We can do the full rigorous way to calculate the interactions between the quantized atom and the classical electromagnetic field, but instead, we can use a "hack"[5] to make the calculations much simpler, since we already calculated the interaction between the atom and the *quantized* electromagnetic field. The Hamiltonian of the full quantum interaction, as given in equation 2.3.1 is

$$\mathcal{H}_{interaction} = \hbar\Omega(\hat{\sigma}_+ + \hat{\sigma}_-)(\hat{a} + \hat{a}^\dagger)$$

[4] This is called the *Rotating Wave Approximation* or RWA in short

[5] The "hack" is that, like we can turn a classical equation into a quantum one, by replacing dynamical variables with operators, we can do the same thing but in reverse. Taking a quantum expression and turning it into a classical one, by replacing operators with dynamical variables

We can go from the quantized field to the classical field by simply replacing the field operators with dynamical variables representing the strength of the field, $\hat{a} \rightarrow a$ and $\hat{a}^\dagger \rightarrow a^*$. We can also replace $\hat{\sigma}_+ + \hat{\sigma}_-$ with the Pauli matrix $\hat{\sigma}_x$ and get

$$\mathcal{H}_{classical} = \hbar\Omega a \cdot \hat{\sigma}_x \tag{27}$$

The transition from the middle term to the right term is allowed since we know that the Pauli matrices are observable quantities, hence are hermitian conjugates of themselves. a is the amplitude of the classical field, later we'll replace it with $\epsilon_I(t)$ and $\epsilon_Q(t)$ for the drive fields since the classical field is not constant and it is, in fact, what we control in the system, but I'm getting ahead of myself, more on this will be shown in the optimal control chapter 3.

2.3.3 *Effective Hamiltonian in the Dispersive Limit*

As we've seen in appendix A, when an atom interacts with a classical electromagnetic field, it oscillates between higher and lower energy states. These oscillations are the so called ***Rabi Oscillations*** and are an important result in the atom-photon interaction theory. There is one catch though, if the electromagnetic field doesn't have **exactly** the same energy as the energy difference between the two levels of the atom, the atom will never reach fully the higher energy state (see figure 20). The difference between the energy of the electromagnetic field, and the atom energy difference is called the detuning, $\Delta = \omega_{field} - \omega_{atom} = \frac{E_{field} - E_{atom}}{\hbar}$. The dispersive limit occurs when the detuning is very large compared to the frequencies (in terms of the variables used in appendix A, the limit is for $\Delta \gg \Omega_0$).

We are now going to consider the energies of the states $|g, n\rangle$ and $|e, n+1\rangle$, for any n. The energies are the eigenvalues of the Hamiltonian[6]. We can write this Hamiltonian as

$$\hat{\mathcal{H}}_n = \frac{\hbar}{2}(\Delta\hat{\sigma}_z + \Omega_n\hat{\sigma}_y) + \hbar\omega_0(n + \frac{1}{2})\hat{I}$$

Where $\Omega_n = \Omega_0\sqrt{n+1}$.

The eigenvalues of this operator are

$$E_n^\pm = \hbar\omega_0(n + \frac{1}{2}) \pm \frac{\hbar}{2}\sqrt{\Delta^2 + \Omega_n^2}$$

Where the ± is $+$ for the ground state of the atom and $-$ for it's excited state. For $\Delta \gg \Omega_n$ we can approximate the square root as a Taylor series, and the overall expression is

$$E_n^\pm = \hbar\omega_0(n + \frac{1}{2}) \pm (\frac{\Delta}{2} + \frac{\Omega_n^2}{\Delta}) = \hbar\omega_0(n + \frac{1}{2}) \pm (\frac{\Delta}{2} + \frac{(n+1)\Omega_0^2}{\Delta})$$

We can replace the number n with the operator $\hat{n} = \hat{a}^\dagger\hat{a}$. Moreover, the operator $\hat{\sigma}_z$ can be used to "replace" the ±, since it's eigenvalue is $+1$ for the ground state and -1 for the excited state. The overall Hamiltonian is of the form

$$\hat{\mathcal{H}}_{\text{eff}} = \hbar\omega_0\hat{a}^\dagger\hat{a} + \frac{\Delta^2 + \Omega_0^2}{2\Delta}\hat{\sigma}_z + \frac{\Omega_0^2}{\Delta}\hat{a}^\dagger\hat{a}\hat{\sigma}_z \tag{28}$$

[6] Sometimes referred to as the *time independent schrödinger equation* $\hat{\mathcal{H}}|\psi\rangle = E_n|\psi\rangle$, although this is an inaccurate name

The physical interpretation of this new expression is that the frequency of the cavity changes by $\chi = 2\frac{\Omega^2}{\Delta}$ depending on the state of the atom. Or equivalently, the atom transition frequency changes by χ for each additional photon in the cavity.

2.4 VISUALIZING QUANTUM STATES

As you'll soon find out in the optimal control chapter, we're going to develop a system that is some sort of a *black box*. We can't really control directly how it works, we can only make indirect changes to try to get everything to work[7].

This is why we want to get as much insight about the system as we can, and understand that insight in a way that allows us to create intelligent changes to the system according to the data we gather. Visualization of quantum states give us a better understanding of the system performance, and allows us to better diagnose errors. There are many more ways to visualize quantum information than I show here,but the methods I will present are sufficient for our purposes.

2.4.1 *Population Graphs*

The simplest way we can visualize the quantum system is using *population graphs*. In a population graph, you plot the probability that the quantum system is in each of its possible eigenstates. For a time-dependent unitary operator, $\hat{U}(t)$, acting on an initial state $|\psi\rangle$, the population of the eigenstates $|\phi_n\rangle$ over time is given by

$$P_n(t) = \left|\langle\phi_n|\,\hat{U}(t)\,|\psi\rangle\right|^2$$

Although they are simple, population graphs are a really powerful, simple tool you can use to visualize the system.

2.4.2 *The Bloch Sphere*

As it turns out, you can represent any qubit in the form[8]

$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\frac{\theta}{2}\,|1\rangle$$

This representation describes a sphere with polar coordinate θ and azimuthal coordinate ϕ. Since this is the case, it's very useful to visualize a qubit state as some point on a sphere, the *Bloch Sphere*. An example of the representation of a state using the Bloch sphere is shown in Figure 1

[7] This is the nature of numerical optimization problems

[8] You can find a proof of this in the "Mike and Ike" book

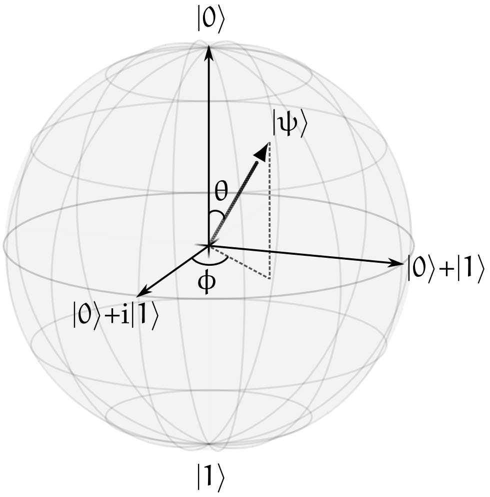


Figure 1: Bloch sphere representation of a qubit

2.4.3 *Wigner Function*

The *Wigner function* is the *distribution* in the *phase space* of a continuous variable quantum system[9]. For classical systems, phase space distributions correspond to a classical probability distribution. However, the Wigner function the distribution also involves quantum uncertainty. Wigner functions can take on negative values in small regions, and are therefore quasi-probability distributions. This is allowed, since only areas larger than $\hbar$ are ever allowed to be measured, according to the Heisenberg uncertainty principle.

The Wigner function of a particle with wave function $\psi(q)$, is defined as

$$W(q,p) := \frac{1}{\pi\hbar}\int_{-\infty}^{\infty} \psi^*(q+y)\psi(q-y)e^{\frac{2ipy}{\hbar}}dy$$

An interesting example to use of the Wigner function is looking at the Wigner function of a Fock state. The Wigner distribution of a Fock state could be calculated directly from definition and the resulting expression is[10]

$$W_n = \frac{2}{\pi}(-1)^n L_n[4(q^2+p^2)]e^{-2(q^2+p^2)}$$

More importantly, we can look at a heat map of the Wigner function, as a nice visual aid to understand the state. An example of the $|4\rangle$ state Wigner distribution is shown in figure 2

[9] Such as position and momentum

[10] proof of which is left as an exercise to the reader ;)

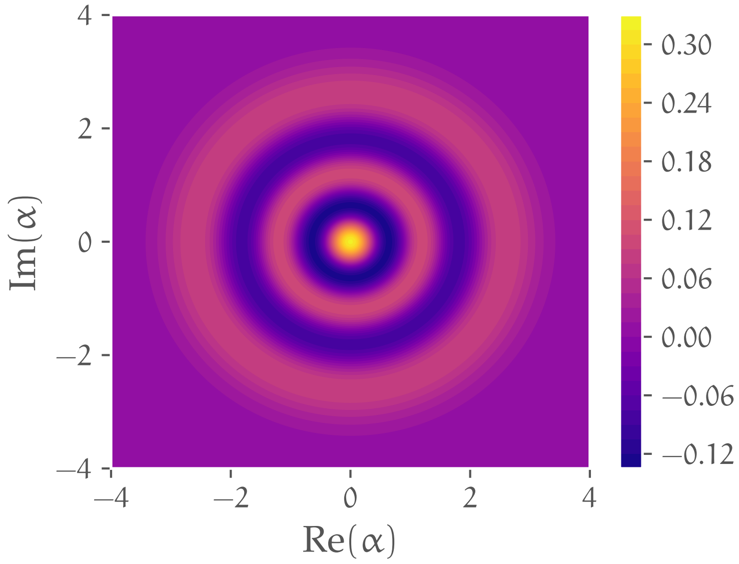


Figure 2: Wigner distribution of the $|4\rangle$ Fock state. $\alpha = q + ip$

2.5 REFERENCES AND FURTHER READINGS

The standard introductory book on this subject (which is also the one I used for reference) is "**Introductory Quantum Optics**" by Christopher C. Gerry and Peter L. Knight.

Another great resource is the series of online lectures taught by Prof. Alain Aspect from École Polytechnique.

3 OPTIMAL CONTROL

In this chapter, we tackle the problem of *Quantum Optimal Control*. We control our quantum system by sending some (classical)electromagnetic pulse into a cavity. An obvious question arises: what pulse shape needs to be sent to achieve a particular operation on our quantum system?

It turns out that this questions does not have an easy answer. In the rest of the chapter we'll try to give an answer using an algorithm called ***GRAPE***.

3.1 GRAPE

Although in some cases we can calculate the desired pulse analytically[1], most of the time this isn't an option. Often, we need to use numeric means to find an optimal pulse. To find the desired pulse numerically, we can model our system on a classical computer and simulate what happens when you send a pulse. Using the results of the simulation, we try to change the pulse until we get the desired effect.

So what's GRAPE then? GRAPE in an acronym for ***GR**adient **A**scent **P**ulse **E**ngineering*. When we model a quantum system, in our case, a qubit interacting with a cavity, we can model the physical information about the system using the Hamiltonian of the system. We can split the Hamiltonian into two. There's the Hamiltonian that nature gives us, of how a qubit behaves and how it interacts with its surrounding. And there's the Hamiltonian we control, using electromagnetic control pulses. We can send any possible pulse shape. To account for that fact, we treat the pulse as a function of time, as a step-wise constant function. The pulse is, essentially, just an array of many variables, and we want to find the values that give the desired result. This is a classical case of an optimization problem. To optimize the values, we set a cost function that tells us how "good" the current pulse is.

In principle, we could use a brute-force search and check every possible pulse until we find one that minimizes the cost function. This method is inefficient. A better way to optimize the function is by calculating also the *gradient* of the cost function, and using an optimization algorithm[2], to find the minimum of the function efficiently.

[1] As seen in appendix A

[2] The **L**imited memory **B**royden–**F**letcher–**G**oldfarb–**S**hanno (L-BFGS-B) algorithm (the **B** is since it could handle simple box constraints) in our case. It's a quasi-newton iterative method for nonlinear optimization problems. We "feed" into it the cost function and gradient

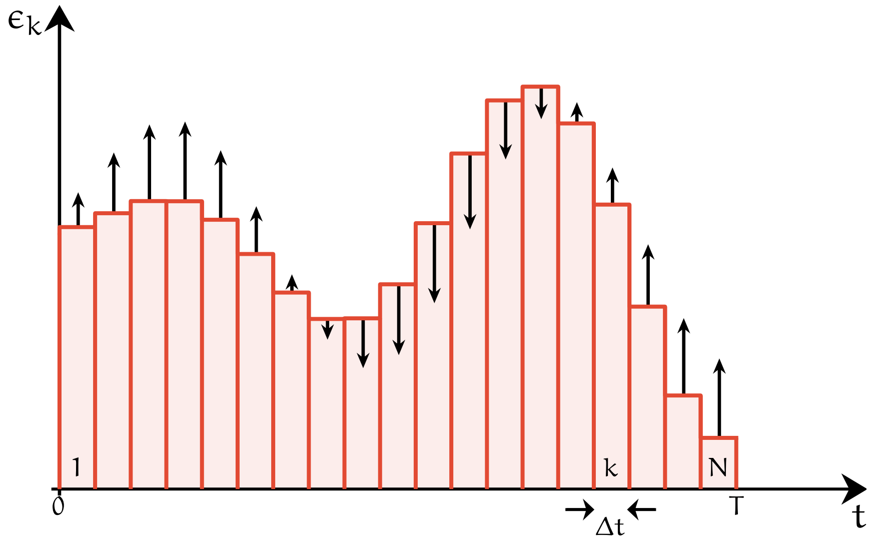


Figure 3: Example of a step-wise constant function as an array of numbers

3.2 THE COST FUNCTION

When our desired operation is to prepare our quantum system in a predetermined state, a good figure of merit of how close our system is to the desired state is given by the **fidelity**. The fidelity is a measure of the "closeness" of two quantum states, The fidelity is 0 when the two states are orthogonal, and 1 if they are identical. The fidelity is given by the magnitude of the *overlap* between the two states squared[3].

$$\mathcal{F}(\psi_1, \psi_2) = |\langle\psi_1|\psi_2\rangle|^2 \tag{29}$$

The *infidelity*, $1 - \mathcal{F}$ between the two states, can be conveniently used as the cost function in our optimization procedure[4]. Given an initial state and a target state, along with the Hamiltonian and pulse information, we want to be able to calculate the fidelity for any given pulse. To do so, we need to solve Schrödinger's equation for the pulse

$$i\hbar\frac{d}{dt}|\Psi(t)\rangle = \hat{\mathcal{H}}|\Psi(t)\rangle \tag{30}$$

In the previous chapter we characterized the Hamiltonian of the system (equations (26) and (28)),

$$\hat{\mathcal{H}}(t) = \hat{\mathcal{H}}_0 + \sum_k \epsilon_k(t)\hat{\mathcal{H}}_k \tag{31}$$

where H_0 is the (time-independent) Hamiltonian of the system without the drives (given, for example, from the Jaynes-Cummings model). Each $\epsilon_k(t)$ is the amplitude as a function of time of the control drive pulses, and each H_k is the (time-independent)Hamiltonian describing the interaction between the control pulse and the rest of the system. We call these Hamiltonians the *drive hamiltonians*. Our goal with GRAPE is to find optimal $\epsilon_k(t)$.

[3] Assuming both states are pure states and not mixed states
[4] Since we want to maximize the fidelity, we want to minimize the infidelity

On each constant step of the amplitude functions $\epsilon_k(t)$, the entire Hamiltonian is constant. Luckily, the solution of the Schrödinger equation for a constant Hamiltonian is easily solved by

$$\hat{U}(t) = e^{-\frac{i}{\hbar}\int_{T_0}^{T_1}\hat{\mathcal{H}}(t)dt} \tag{32}$$

If we choose T_0 and T_1 as the end points of a step of the Hamiltonian, the total Hamiltonian of the system is constant so the integral is just a simple multiplication by $T_1 - T_0$ which we'll write as $\delta t = T_1 - T_0$. The solution becomes

$$\hat{U}(t) = e^{-\frac{i\cdot\delta t}{\hbar}\hat{\mathcal{H}}(t)} \tag{33}$$

In order to calculate the solution over the entire pulse we need to solve for the first step, then find the solution by the end of that time-step, and use it as the initial condition for the next time step, repeating for each time step. The solution until the N^{th} time step is simply the product of the previous solutions for each time step

$$\hat{U}_N(\epsilon(t)) = \prod_{k=1}^{N}\hat{U}_k(\epsilon(t)) \tag{34}$$

With $\hat{U}(\epsilon)$ in hand, we can calculate the evolution of the state over the entire pulse

$$|\Psi_{final}\rangle = \hat{U}(\epsilon)\,|\Psi_{initial}\rangle \tag{35}$$

This way, if we want to calculate the fidelity after applying the drives, we can simply calculate the fidelity between the target state and the final, calculated state

$$\mathcal{F}(\epsilon(t)) = \mathcal{F}(\Psi_{target}, \Psi_{final}) = \left|\langle\Psi_{target}|\,\hat{U}_N\,|\Psi_{initial}\rangle\right|^2 \tag{36}$$

As mentioned, if we want to optimize the cost function efficiently we'll need to calculate the gradient of the cost function along with the cost function itself.

3.3 THE GRADIENT

For simplicity, we'll first derive the gradient of the *overlap*

$$c = \langle\Psi_{target}|\Psi_{final}\rangle = \langle\Psi_{target}|\,\hat{U}\,|\Psi_{initial}\rangle \tag{37}$$

We can then get the fidelity by noting that $\mathcal{F} = |c|^2$. We want to differentiate the overlap by each control parameter. To do so, note that $\hat{U}$ is defined as:

$$\hat{U} = \hat{U}_N\hat{U}_{n-1}...\hat{U}_2\hat{U}_1$$

And when differentiating by a control parameter only one $\hat{U}_k$ is affected,

$$\frac{\partial c}{\partial \epsilon_k} = \langle\Psi_{target}|\,\hat{U}_N\hat{U}_{N-1}...\hat{U}_{k+1}\frac{\partial\hat{U}}{\partial\epsilon_k}\hat{U}_{k-1}...\hat{U}_2\hat{U}_1\,|\Psi_{initial}\rangle$$

We can write that for a constant Hamiltonian (from Schrödinger's equation)

$$\hat{U}_k = e^{-\frac{i\cdot\delta t}{\hbar}\hat{\mathcal{H}}(t)}$$

And approximate the derivative $\frac{\partial \hat{U}_k}{\partial \epsilon_k}$ in the limit of small δt by writing

$$\frac{\partial \hat{U}_k}{\partial \epsilon_k} \approx -\frac{i \cdot \delta t}{\hbar} \frac{\partial H}{\partial \epsilon_k} \cdot e^{-\frac{i \cdot \delta t}{\hbar} \hat{\mathcal{H}}(t)} = -\frac{i \cdot \delta t}{\hbar} \frac{\partial H}{\partial \epsilon_k} \hat{U}_k$$

Theoretically, we have everything we need to calculate the gradient, but it's still rather complex computationally ($o(N^2)$ complexity). A different method can be used to reduce the computational overhead.

The derivative of the cost function by a control parameter of the pulse has become

$$\frac{dc}{d\epsilon_k} = -\frac{i \cdot \delta t}{\hbar} \underbrace{\langle \Psi_{target} | \hat{U}_N \hat{U}_{N-1} ... \hat{U}_{k+1}}_{\left\langle \psi_{bwd}^{(k+1)} \right|} \frac{dH}{d\epsilon_k} \underbrace{\hat{U}_k ... \hat{U}_2 \hat{U}_1 | \Psi_{initial} \rangle}_{\left| \psi_{fwd}^{(k)} \right\rangle} \tag{38}$$

Where we defined two arrays, $\langle \psi_{bwd} |$ and $| \psi_{fwd} \rangle$, the multiplication components before and after the derivative of H

$$\frac{\partial c}{\partial \epsilon_k} = -\frac{i \cdot \delta t}{\hbar} \left\langle \psi_{bwd}^{(k+1)} \middle| \frac{\partial H}{\partial \epsilon_k} \middle| \psi_{fwd}^{(k)} \right\rangle \tag{39}$$

We can see from 38 that

$$\left| \psi_{fwd}^{(k)} \right\rangle = \begin{cases} | \psi_{init} \rangle & k = 0 \\ \hat{U}_k \left| \psi_{fwd}^{(k-1)} \right\rangle & otherwise \end{cases}$$

$$\left| \psi_{bwd}^{(k)} \right\rangle = \begin{cases} | \psi_{targ} \rangle & k = N+1 \\ \hat{U}_k^{\dagger} \left| \psi_{bwd}^{(k+1)} \right\rangle & otherwise \end{cases}$$

Now all we need is to do $2N$ calculations in the beginning (N for bwd and N for fwd), and then calculating the actual gradient using equation 39. This improves the computation complexity from $o(N^2)$ to $o(N)$, while the memory complexity stays $o(N)$.

It's important to note that c is not the fidelity, but the overlap. We can get the fidelity from c by

$$\mathcal{F} = |c|^2$$

since c might be complex this derivative is a bit less trivial than it might seem. We can write $c(\vec{\epsilon})$ as $a(\vec{\epsilon}) + b(\vec{\epsilon})i$, where $a, b \in R$ and we get that

$$\frac{\partial \mathcal{F}}{\partial \epsilon_k} = \frac{\partial |c|^2}{\partial \epsilon_k} = \frac{\partial |a + bi|^2}{\partial \epsilon_k} = \frac{\partial (a^2 + b^2)}{\partial \epsilon_k} = 2(a\frac{\partial a}{\partial \epsilon_k} + b\frac{\partial b}{\partial \epsilon_k})$$

We can notice that $c(\frac{\partial c}{\partial \epsilon_k})^* = a\frac{\partial a}{\partial \epsilon_k} + b\frac{\partial b}{\partial \epsilon_k} + (ab - \frac{\partial a}{\partial \epsilon_k}\frac{\partial b}{\partial \epsilon_k})i$. More importantly, we can see that the real part of that expression is exactly what we need. Putting it all into one formula we get

$$\frac{\partial \mathcal{F}}{\partial \epsilon_k} = 2 \cdot \text{Re}\left\{ c(\frac{\partial c}{\partial \epsilon_k})^* \right\} \tag{40}$$

Now all you need is to plug 39 and 37 into 40 and you got your gradient :)

Let's do a little test now to see that everything is working well. The simplest pulse you can send is the pulse that takes the qubit from being in state $|0\rangle$ to state $|1\rangle$, where the transition frequency is set to 1Ghz (hence the period is 1ns). We discussed this situation in appendix A so we know how the solution should look like. Running our GRAPE code with some random initial pulse we get

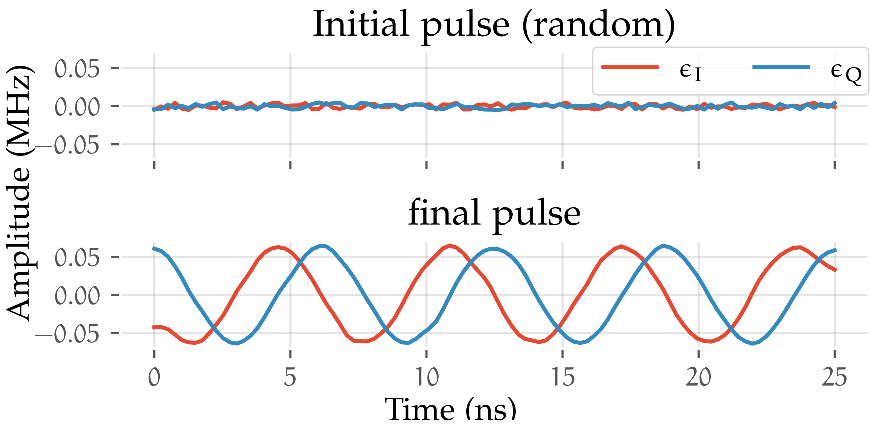


Figure 4: Pulses solution for $|0\rangle \rightarrow |1\rangle$. Each color is a different microwave control pulse of the system, I and Q. They are the real and imaginary parts of the calculated wave in appendix A

Amazing! From some random initial pulse we got sinusoidal waves with 1Ghz frequency, just as predicted in appendix A.

Before we get too excited, there are a couple of things weird with this pulse. The first, more obvious problem, is that although the waves are sin and cos as expected, they're still pretty jagged-y, there is some randomness on top of the wave and it's not as smooth as we expected. This is since small random changes don't really change the final result[5]. In addition, high-frequency components generated by the computer do not have an effect in reality. This is due to the limited bandwidth of our pulse generator and RF circuitry.

Another problem, that is not immediately obvious, we can only see if we look at the population graph of the qubit over the duration of the pulse. We expect the graph to start at 1 and end at 0 for $|0\rangle$ and start at 0 and end at 1 for $|1\rangle$. Let's look at that graph for the initial random pulse and for the optimized pulse

[5] The noise cancels itself out

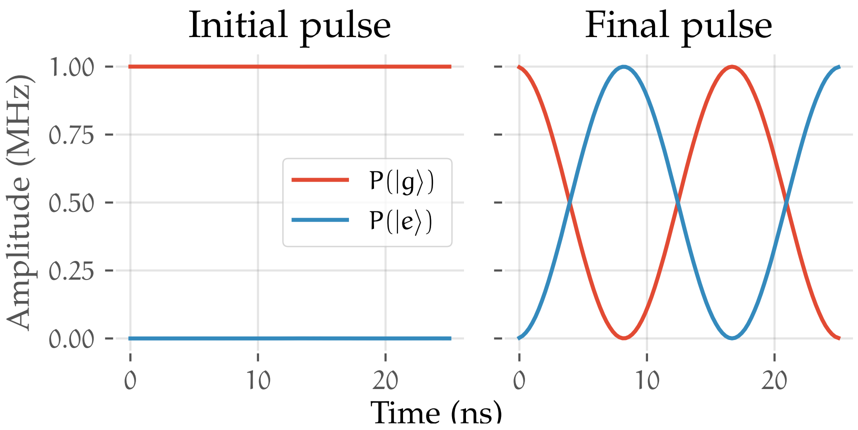


Figure 5: Population of qubit levels over pulse duration. Before the optimization, the state of the qubit (population of ground and excited states) almost did not change at all. After the optimization, the qubit goes from state $|0\rangle$ to $|1\rangle$ to $|0\rangle$ to $|1\rangle$, doing some unnecessary back and forth between the states

As expected, the initial random pulse doesn't change the pulse almost at all. The optimized pulse on the other hand is problematic. The population of $|0\rangle$ for example, goes from 0 to 1 and then goes back to 0 to start over again. Ideally, the population will change from 0 to 1 (and vice-versa) smoothly, only once.

This happens because the amplitude of the optimized pulse is too big. As we defined the optimization, the algorithm doesn't care that the population acts weirdly in the middle of the pulse as long as it ends at the desired state.

To solve these problems (and more we'll talk about), we introduce *constraints* to the algorithm.

3.4 CONSTRAINTS

Since instruments have physical limitations, for example, on the maximum amplitude of a pulse, we must add constraints on the computer calculations to not exceed these limitations.

We define a set of constraints on the solution $g_i \geqslant 0$, and associate a Lagrange multiplier λ_i to each constraint[6].
Our goal is to minimize

$$1 - \mathcal{F}(\vec{\epsilon}) + \sum_i \lambda_i g_i(\vec{\epsilon})$$

Let's add a constraint to each of the most problematic physical limitations.

3.4.1 *Limiting the Amplitude*

This is the most obvious physical limitation. We can't generate pulses with infinite energy, so we have to restrict it. There are two ways we can do so, the first is to create a hard cut-off amplitude. No matter what, the amplitude will never go above this amplitude. We usually want this cut-off to be the maximal output

[6] In an ideal optimized pulse, $g_i = 0$

amplitude of our pulse generator. But normally we don't want our generator to work at its absolute limit[7], so we can add also a soft amplitude maximum by "rewarding" the cost function to stay at a lower amplitude. Let's see how we would implement such a thing, starting with the hard cut-off.

Instead of controlling and changing the amplitude ($\vec{\epsilon}$) directly, we'll introduce a variable $\vec{x}$ and relate them as

$$\vec{\epsilon} = \epsilon_{max} \tanh \vec{x}$$

As you probably guessed, ϵ_{max} is the maximum amplitude of the pulse. Since the optimization algorithm can only change $\vec{x}$, the amplitude of the pulse will always be between $-\epsilon_{max}$ and ϵ_{max}. Unfortunately, this changes the gradient of the cost function since we now want the derivative with respect to $\vec{x}$ instead of $\vec{\epsilon}$. We can relate the two

$$\frac{\partial \mathcal{F}}{\partial \vec{x}} = \frac{\partial \mathcal{F}}{\partial \vec{\epsilon}} \frac{\partial \vec{\epsilon}}{\partial \vec{x}} = \frac{\epsilon_{max}}{\cosh^2 \vec{x}} \frac{\partial \mathcal{F}}{\partial \vec{\epsilon}}$$

We can use the derivative $\frac{\partial \mathcal{F}}{\partial \vec{\epsilon}}$ we got from 40 and simply calculate $\frac{\epsilon_{max}}{\cosh^2 \vec{x}}$ and we again have the gradient.[8]

For the soft amplitude penalty all we want is that *bigger amplitudes ⇒ bigger cost function*. Since our algorithm seeks to minimize the cost function, this will lead to the overall amplitude being smaller. The way we do so is simple, we can define a constraint g_{amp} that sums all the amplitudes of the steps of the pulse, so

$$g_{amp} = \sum_k |\epsilon_k|^2$$

Still, since it is added to our cost function we need to find the gradient of the penalty as well. In this case it's rather simple since it's a basic parabola

$$\frac{\partial g_{amp}}{\partial \epsilon_k} = 2\epsilon_k$$

and now we have all we need in order to add this penalty to our cost function.[9]

3.4.2 *Limiting Bandwidth and Slope*

The next limitation is the maximum frequency our AWG can create because the device can't change the voltage instantaneously. Again, like we had with the amplitude limit, there are 2 types of limits we can make, hard and soft. Let's start with the hard limit.

We have some frequency ω_{max} which is the maximum frequency that our AWG can generate. To make sure that our simulation doesn't produce such a pulse we can go from time space to the frequency space with a Fourier transform

$$\vec{\epsilon} = (\mathrm{DFT})^{-1} \vec{x}$$

[7] Not only that it might damage the device, with stronger pulses the non-linear optical effects increase

[8] Since tanh(x) is a positive monotonic transformation it preserves the locations of the maxima of the function. This means that in practice we can just ignore the new derivative

[9] We can also create non-linear soft amplitude constraint. With such constraint we can set a soft maximum we want to stay well below of. We won't show how to do so

The numerical optimization algorithm controls $\vec{x}$ which is in the frequency space. Now if we want to limit the frequency we can simply set to 0 any frequency that is above our maximum frequency

$$x(\omega > \omega_{max}) = 0$$

The gradient of the new cost function is simply the Fourier transform of x

$$\frac{\partial \vec{x}}{\partial \epsilon_k} = (\mathsf{DFT})\frac{\partial \vec{\epsilon}}{\partial \epsilon_k}$$

And we know $\frac{\partial \vec{\epsilon}}{\partial \epsilon_k}$ from previous sections. It's important to note that the hard cut-off of the amplitude and the hard cut-off of the frequency do not work together since one is in time domain and one is in frequency domain. This is not much of a problem since we can compensate with the soft limits that do work well together (since they require adding to the cost function instead of changing coordinates). In my simulations I use the amplitude hard limit instead of the frequency one since it is easier to work with.

For the soft limits, we're limiting the slope (derivative) of the pulses and not the frequency directly. The slope of a step function is simply $\epsilon_{k+1} - \epsilon_k$, we want to limit the size of the slope so we'll look at the expression $|\epsilon_{k+1} - \epsilon_k|^2$ instead. Summing all the slopes (to get an overall slope size of the entire pulse) we get the expression[10]

$$g_{slope} = \sum_{k=0}^{N-1} |\epsilon_{k+1} - \epsilon_k|^2 \tag{41}$$

Unlike the amplitude, since the slope of the boundaries is not well defined we'll have the edges defined differently than the center of the pulse. The gradient of g in the center is a simple derivative, notice that each ϵ_k appears only twice in the sum

$$\frac{\partial g_{slope}}{\partial \epsilon_k} = 4\epsilon_k - 2(\epsilon_{k+1} + \epsilon_{k-1})$$

It's nice to see that the expression looks like how'd you numerically estimate the second derivative, since the gradient of the slope (which is the derivative) is the second derivative. Now we need to define the gradient at the edges, you can see that the derivative of ϵ_k depends on his neighbors on both sides. Since the first and last element of the pulse don't have 2 neighbors they are treated a little differently. each of the edges appears only once in the sum 41 unlike the others that appear twice, we can simply take the derivative of that one term and get

$$\frac{\partial g_{slope}}{\partial \epsilon_0} = 2(\epsilon_1 - \epsilon_0)$$
$$\frac{\partial g_{slope}}{\partial \epsilon_N} = 2(\epsilon_N - \epsilon_{N-1})$$

Now, before we continue, we'll add another small constraint that will also solve the problem of the slope at the boundaries. It might seem weird at first, but we

[10] Note that we have a problem at the edges since the slope of the end points is not well defined, we'll fix this problem later but for now we just ignore the last point $k = N$

want to pulse to zero-out at the edges. This is since our AWG device can't immediately start a pulse with some amplitude, it can't get from 0 to that amplitude instantaneously (for the same reason we limit the slope in the first place). This could be achieved by simply setting the first and last steps of the pulse and their gradient to 0.

$$\epsilon_0 = \epsilon_N = \frac{\partial\, \text{Cost}}{\partial \epsilon_0} = \frac{\partial\, \text{Cost}}{\partial \epsilon_N} = 0$$

This solves the problem we were trying to solve we were having with the slope at the boundaries, since the gradient is 0 at the edges and does not depend on its neighbors.

Now that we have both amplitude constraint and bandwidth constraint we can use them to get a nice, smooth solution, that any wave generator would be happy to produce

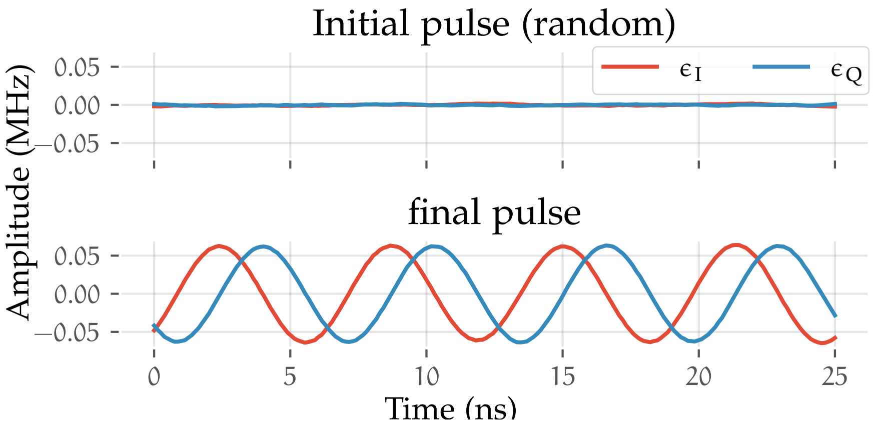


Figure 6: Control pulses before and after GRAPE optimization with amplitude and bandwidth constraints

The pulse is exactly what we expect it to be. More importantly, if we look at the population graph of the levels, it does exactly what we want, go from state $|0\rangle$ to state $|1\rangle$ without going back and forth.

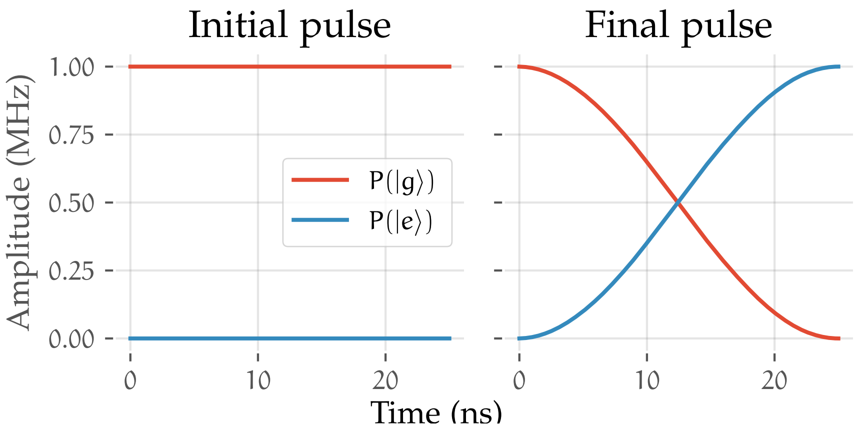


Figure 7: Population of qubit levels over pulse duration. Before the optimization, the state of the qubit (population of ground and excited states) almost did not change at all. After the optimization, the qubit goes from state $|0\rangle$ to $|1\rangle$

Another interesting visualisation of the success of this pulse is the path of the qubit on the Bloch sphere over time (see the last section in chapter 2). Plotting the populations on the Bloch sphere we get

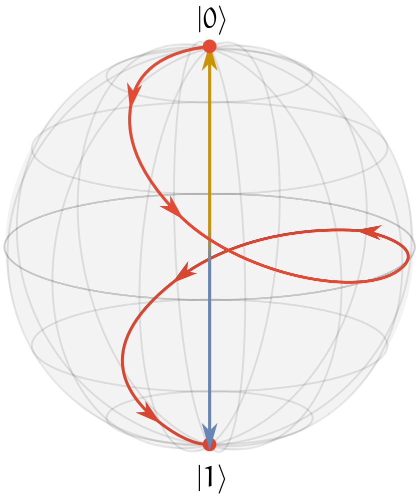


Figure 8: Path of the qubit along the Bloch sphere. The qubit goes from state $|0\rangle$ to state $|1\rangle$, preforming one loop on the sphere, the arrows represent the initial state and the target state, and the points represent the path of the state of the qubit

As you can see, the path of the qubit isn't a straight line, but some loop, completing a full rotation around the $\hat{z}$ axis. This is explained by the fact that the base Hamiltonian of the qubit is $\omega\hat{\sigma}_z$, where $\hat{\sigma}_z$ is the Pauli matrix Z. This matrix corresponds to rotation of the Bloch sphere around the $\hat{z}$ axis. This is why we can think of the entire Bloch sphere as always rotating with frequency ω_0 around the $\hat{z}$ axis, this is why a "straight" path is actually one that does one loop around the $\hat{z}$ axis.

3.4.3 *Limiting Pulse Duration*

As much as I wouldn't mind waiting a few nanoseconds longer for the qubit operation to end, the qubit itself isn't as patient as me. A state-of-the-art qubit would last, at most, few hundred microseconds (and that's a very optimistic estimation). We simply don't have the time to wait for the operation to end if we want to run some complicated quantum circuit. This is why we want to add a constraint on the *duration* of the pulse. This way, if we set the total duration of the pulse to longer than the shortest possible pulse, it would find the shorter pulse and the rest of the pulse will change nothing (see figure 9)

The constraint is fairly straight forward, add a penalty for any time a fidelity of 1 isn't achieved. Put into an equation we get

$$g_{dur} = \sum_{i=0}^{N-1} (1 - \mathcal{F}_i)$$

Where $\mathcal{F}_i$ is the fidelity at time step i.

We can rather simply calculate the fidelity at any given time since they are already calculated in order to find the fidelity in the last time step. We can simply modify the loop that calculates the final state into giving the fidelity at each time step and sum the results.

Luckily for us, the calculation of the gradient is also pretty simple. The gradient of the fidelity at each time step is calculated the same as the gradient of the fidelity we calculated in the beginning of the chapter[11]

$$\frac{\partial g_{dur}}{\partial \epsilon_k} = -\sum_{i=k}^{N-1} \frac{\partial \mathcal{F}_i}{\partial \epsilon_k}$$

The pulse duration constraint works nicely to complete the other constraints. Without this constraint, the pulse will "try" to use all it's time to get the result we desire. When running the algorithm without the constraint we can get problems if the duration we gave to the pulse is too long or too short. With this constraint on, we can simply give the algorithm a duration that we know for sure is more then the minimum required time, and the algorithm will simply use the minimum time it needs, and no more. On the other hand, if we didn't have the amplitude (and bandwidth) constraints, the algorithm might find that it's best to just give a super-strong pulse for a tiny amount of time, but that's not physically possible as we discussed. This is why we can think of the constraints working together to "box in" the pulse into an ideal "size".

If we run GRAPE now, with all the constraints together, and look at the population of all the level over the duration of the pulse, we get[12]

[11] note that ϵ_k only appears in the expression for $\mathcal{F}_i$, if $i > k$, so the sum starts at $i = k$

[12] This is a rather extreme case where the total duration of the pulse is around three times longer then the minimum duration. This was done purposefully to demonstrate the point.

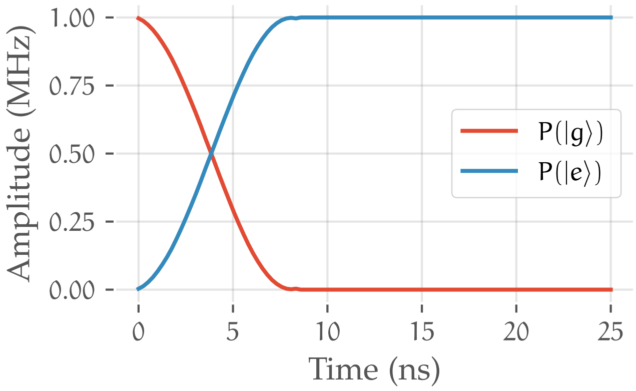


Figure 9: Level population of each state as a function of time over the duration of the pulse.

As expected, we get exactly what we want, the pulse uses the least amount of time that it needs and then stop. This way, if we pick a long duration for the pulse, instead of the pulse trying to fill the entire time at a very low amplitude, or do several of loops before arriving at the target, the qubit simply takes exactly the amount of time that it needs to get to the desired state under all of the constraints and then stops.

3.5 IMPLEMENTING QUBIT OPERATIONS WITH GRAPE

3.5.1 *DRAG - Imperfect Qubits*

When we did all of our calculation on the qubit we didn't include one detail, it's really hard to create a true two-level system. In the way our qubits are implemented, there are actually more than two levels. It's not a two level system but we treat it as one. Since the higher levels are off-resonance, we can often just ignore them. However, there is some probability of the higher levels getting populated by high-frequency components of our pulses. The shorter our pulses are, the more these higher levels excitation occur. We will prevent those so-called leakage errors by using **D**erivative **R**emoval via **A**diabatic **G**ate (DRAG) pulses. We can generate those pulses by including more than two levels in our algorithm and changing the Hamiltonian a little bit so it accounts for the off-resonance higher levels.

Before we continue to implement DRAG, let's see if the third level really is that much of a problem. We run a simulation of GRAPE just as we did before but this time with three levels instead of two, and the third level should start and end at 0 population. We get after running GRAPE

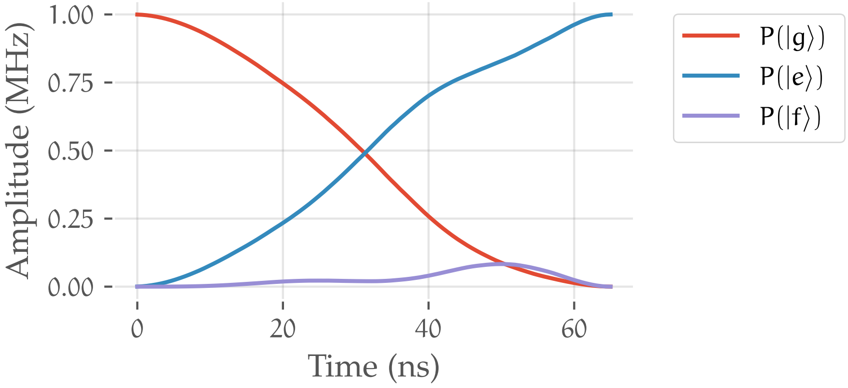


Figure 10: Level population of the 3-level "qubit" over the duration of the pulse calculated by the GRAPE algorithm we have so far

Well yes, the forbidden level did start and end at 0 population[13], but in the middle the qubit really became a 3 level system with the population of the forbidden level being really dominant around $t = 50$ns! We can't simply replace the qubit with a 3 level system and make the target of the third level always 0 and call it a day. **We can't treat higher levels as just more "qubit" levels**, they are unwanted and we need to give them a penalty so the probability of being in a higher level would be always almost zero and change only a tiny bit. There are many ways we could implement such a penalty, the most obvious way is by simply making the probability to be in an higher level into a penalty, summing over all time we get (We'll call the third level of the qubit $|f\rangle$ to not be confused with the $|3\rangle$ Fock state (photon number state))[14]

$$g_{\textit{forbidden}} = \sum_{i=0}^{N-1} \left| \left\langle f \middle| \psi_{fwd}^{(i)} \right\rangle \right|^2$$

We already have $\psi_{fwd}^{(i)}$ that we calculated earlier, so for so good.

Now moving to to complex part of DRAG, the gradient. Let's again define the overlap

$$c_f = \sum_{i=0}^{N-1} \left\langle f \middle| \psi_{fwd}^{(i)} \right\rangle$$

now to calculate the gradient we'll derive over ϵ_k

$$\frac{\partial c_f}{\partial \epsilon_k} = \frac{\partial}{\partial \epsilon_k} \sum_{i=0}^{N-1} \left\langle f \middle| \psi_{fwd}^{(i)} \right\rangle = \sum_{i=0}^{N-1} \frac{\partial}{\partial \epsilon_k} \left\langle f \middle| \psi_{fwd}^{(i)} \right\rangle$$

recall that $\psi_{fwd}^{(i)} = \hat{U}_i \cdot \hat{U}_{i-1} \cdot ... \cdot \hat{U}_1 \left|\psi_{initial}\right\rangle$, $\hat{U}_k$ only appears for $i > k$, so we can start the sum from $i = k$. We'll also expand $\psi_{fwd}^{(i)}$ into what it is and get

$$\frac{\partial c_f}{\partial \epsilon_k} = \sum_{i=k}^{N-1} \frac{\partial}{\partial \epsilon_k} \langle f| \hat{U}_i \cdot ... \cdot \hat{U}_0 \left|\psi_{initial}\right\rangle$$

[13] If you decrease the anharmonicity you would also have final forbidden level population, which is a more serious problem

[14] If we wanted to accounted for higher levels we can sum over the sum for each level

the only element that depends on ϵ_k is $\hat{U}_k$, so we can rearrange the equation as

$$\begin{aligned}\frac{\partial c_f}{\partial \epsilon_k} &= \sum_{i=k}^{N-1} \langle f|\, \hat{U}_i \cdot ... \cdot \frac{\partial \hat{U}_k}{\partial \epsilon_k} \cdot ... \cdot \hat{U}_0 \,|\psi_{initial}\rangle \\ &= \sum_{i=k}^{N-1} \langle f|\, \hat{U}_i \cdot ... \cdot i \cdot \delta t \frac{\partial H_k}{\partial \epsilon_k} \hat{U}_k \cdot ... \cdot \hat{U}_0 \,|\psi_{initial}\rangle \\ &= i \cdot \delta t \sum_{i=k}^{N-1} \langle f|\, \hat{U}_i \cdot ... \cdot \hat{U}_{k+1} \cdot \frac{\partial H_k}{\partial \epsilon_k} \left|\psi^{(k)}_{fwd}\right\rangle\end{aligned}$$

Now just to keep everything simple and maintainable, we'll define

$$\left\langle \phi^{(i,k)}_{bwd} \right| = \langle f|\, \hat{U}_i \cdot ... \cdot \hat{U}_{k+1}$$

The equation for the overlap now becomes

$$\boxed{\frac{\partial c_f}{\partial \epsilon_k} = i \cdot \delta t \sum_{i=k}^{N-1} \left\langle \phi^{(i,k)}_{bwd} \right| \frac{\partial H_k}{\partial \epsilon_k} \left|\psi^{(k)}_{fwd}\right\rangle}$$

Now we got all we need to calculate the penalty of the occupying the higher level and it's gradient. This isn't a perfect solution though, for N time steps we need to do $o(N^2)$ calculations to get $\left\langle \phi^{(i,k)}_{bwd} \right|$. This slows down the calculation considerably[15] and there is a lot of overhead in the way we calculated $\left\langle \phi^{(i,k)}_{bwd} \right|$. We can use a smarter way to calculate it.

Consider a function, very similar to $\langle \psi_{bwd}|$ we had earlier (in fact, it's the same function minus multiplying by the target state on the left)

$$\psi^{(k)}_{bwd} = \hat{U}_N \hat{U}_{N-1} ... \hat{U}_{k+2} \hat{U}_{k+1}$$

taking the inverse of the resulting matrix we get

$$(\psi^{(i)}_{bwd})^{-1} = \hat{U}^{-1}_{i+1} \hat{U}^{-1}_{i+2} ... \hat{U}^{-1}_{N-1} \hat{U}^{-1}_{N}$$

by multiplying the two matrices we get (defining their product as ϕ_{bwd})

$$\phi^{(i,k)}_{bwd} = (\psi^{(i)}_{bwd})^{-1} (\psi^{(k)}_{bwd}) = (\hat{U}^{-1}_{i+1} \cdot ... \cdot \hat{U}^{-1}_{N})(\hat{U}_N \cdot ... \cdot \hat{U}_{k+1}) = \hat{U}_i \cdot ... \cdot \hat{U}_{k+1}$$

This is exactly what we wanted! from this we'll define

$$\left\langle \phi^{(i,k)}_{bwd} \right| = \langle f|\, \phi^{(i,k)}_{bwd}$$

remember that ψ_{bwd} was already calculated from the gradient calculation, taking the inverse of ψ_{bwd} isn't affected by how many time steps there are, also the multiplications between ψ_{bwd}, $(\psi_{bwd})^{-1}$ and $\langle f|$ isn't dependent on the amount of time steps, so the entire calculation is $o(N)$ complexity.

Let's run now the algorithm and get some results

[15] it makes to calculation run around 100 times slower, pretty bad considering it's just a penalty

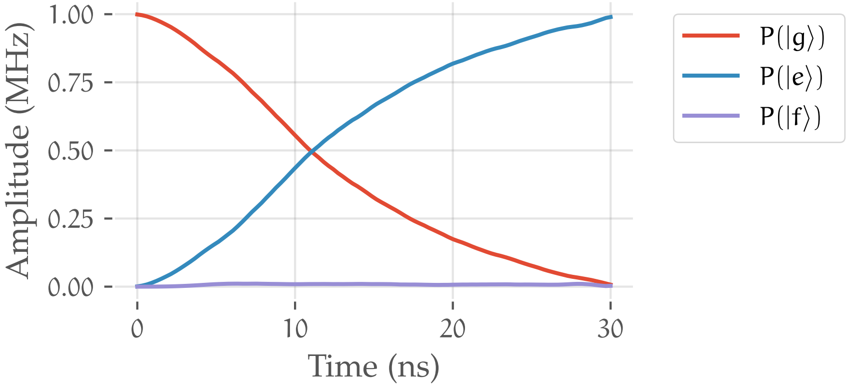


Figure 11: Level population of the "qubit" over the duration of the pulse that was found by the GRAPE algorithm, with all the penalties turned on, including the DRAG penalty.

Nice! state $|0\rangle$ goes directly to 1, $|1\rangle$ goes directly to 0 and the forbidden level is barley changed throughout the pulse (the fidelity gotten from this pulse is around 99.9%, so pretty good).

I think that we talked enough about the qubit for now, let's move the the other half of the system, the cavity.

3.6 IMPLEMENTING CAVITY OPERATIONS

3.6.1 *Limiting the photon number*

"Hilbert space is a big place."
- Carlton Caves

Here's the thing about the cavity levels, there are infinite amount of them. We can make an assumption that the cavity only has N levels, but it is still possible that something happens in the higher levels that may affect the physical result. We want to make sure that everything interesting is contained in the N levels that we have.

This is quite similar to what we did in the previous section with DRAG, we want to put a penalty on the higher levels. Still, there are two main differences. The first, is that there are much more than one or two extra levels, and as we've seen, the method we used in the previous uses very heavy computation and we can't do it for so many levels. The other difference is that we care less about if some higher level is occupied for a part of the pulse. In the cavity there are higher levels and they're all likely to be but **the reason we're limiting the cavity levels is for computing reasons, not physical ones**. Unlike the cavity, we really want the qubit dynamics to involve only two levels, so it makes sense that we'll use a different penalty to limit the photon number.

The idea is this, we'll define n_{ph} as the highest level we want the cavity to have. Now let's calculate what will happen if the cavity will have another level, for a total of $n_{ph}+1$. Ideally, nothing will change, the new level should start at 0 probability and end at 0 probability with no change in between. If there is a change, will add a penalty to the pulse, so in the next iteration there will be less of a change. We can do so for any level higher than one, so instead of using only $n_{ph}+1$ we'll some over $n_{ph}+k$ for reasonable amount of k's.

We'll define $\mathcal{F}_{n_{ph}+k}$ as the fidelity if there were $n_{ph}+k$ levels. Putting the idea into a formula we get that the new cost function is given by

$$\text{Cost} = \sum_{k=0}^{N} \mathcal{F}_{n_{ph}+k}(\vec{\epsilon}) - \sum_{i} \lambda_i g_i(\vec{\epsilon})$$

We can double enforce the penalty if we add a constraint making sure there is no change in the fidelity for different levels

$$g_{ph} = \sum_{k_1 \neq k_2} (\mathcal{F}_{n_{ph}+k_1} - \mathcal{F}_{n_{ph}+k_2})^2$$

The gradient of which is simply the gradient of which is simply

$$\frac{\partial g_{ph}}{\partial \epsilon_k} = 2 \sum_{k_1 \neq k_2} [(\mathcal{F}_{n_{ph}+k_1} - \mathcal{F}_{n_{ph}+k_2})(\frac{\partial \mathcal{F}_{n_{ph}+k_1}}{\partial \epsilon_k} - \frac{\partial \mathcal{F}_{n_{ph}+k_2}}{\partial \epsilon_k})]$$

and everything in this expression was previously calculated (they're simply the derivatives of the fidelity and the fidelity itself).

It's important to note that while it might be tempting to leave n_{ph} at a small value so there will be less to calculate (the size of the matrices grows with n_{ph}^2), there is good reason to use a high values of n_{ph}. Bigger n_{ph} oscillate at higher frequency (since the cavity is simply a harmonic oscillator), so it's possible to use shorter pulses, and since keeping qubit alive is really a major problem, keeping the pulses short is important to be able to accomplish the most with the time we have with the qubit before it dies.

Lets see what happens if we run the transmon-cavity code. We'll only look at the resulting population graph after the optimization. I warn you that the graph is a bit cluttered. You shouldn't look at any specific details or any specific curve, I didn't put a legend explaining what each curve is. We'll look at it then discuss

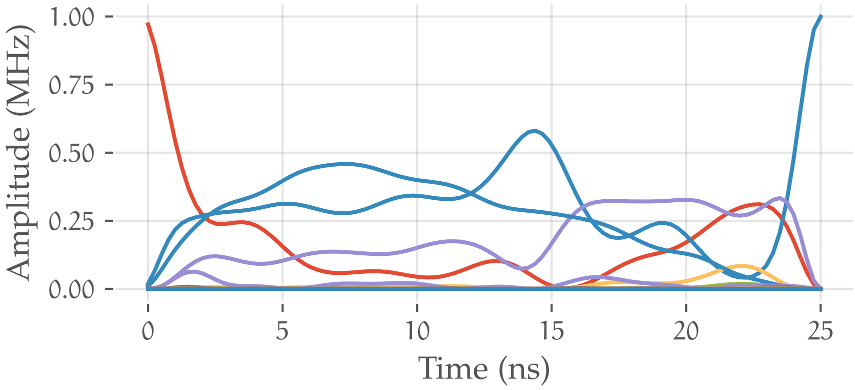


Figure 12: Transmon-cavity state population over the duration of the pulse that does the transformation $|g\rangle \otimes |0\rangle$ (Blue) $\longrightarrow$ $|g\rangle \otimes |1\rangle$ (Red).

After you see this graph, you'd think that the constraint didn't work correctly. After all, the transformation is from level $|0\rangle$ to level $|1\rangle$, but there were many other levels that got occupied during the middle of the pulse. This is a valid guess, but there is an explanation why the higher level would get occupied. The reason is that you can't really create number states *directly* in a cavity. You can only create coherent states *directly*. To create number state you need to make the coherent states interfere in a way that creates a number state.

So why then should this graph show it was successful then? Well, I ran the algorithm with 50 cavity levels (!), this means that there are actually 100 curves in that graph (50 of the cavity times 2 of the qubit). If you'd try to count the number of curves that you see you'll probably count 7-8 curves and not all of the 100, this is since most of the curves stay at zero population. This is exactly what we want, the higher levels don't affect the physics of the system, if we add more levels (like in the real world where there are infinite levels) the pulse would still give the same desired result.

3.7 finding a good initial guess

Although the GRAPE algorithm is the one responsible to find to optimal control pulse, we still need to give it some initial guess and the algorithm does the rest. You might think that this isn't much of a problem since theoretically any initial guess should arrive at a desired result. The problem is that often the algorithm gets stuck and can't find a result. This could be caused by a number of reasons, the main two are when the constraints are too strong and when the initial guess is not good enough.

When the constraints are too strong, the algorithm might prefer optimizing them instead of the fidelity and what we get is a pulse that achieves horrible fidelity but within the constraints. The solution is simply weakening the constraints (choosing a smaller λ for the constraint).

The other, harder to solve problem is when the initial guess of the pulse is problematic. For example, if you choose the initial guess to be the most obvious initial guess you can make, constant 0, the algorithm will probably stop after one iteration changing nothing. This is because the gradient of a constant 0 pulse is actually zero, so the optimization algorithm thinks it's in an optimized minimum when in fact, it couldn't be more wrong.

The other simple initial guess you might think of using is a random pulse. After all, we don't know what is the desired pulse so picking a random pulse is as good as any other. The problem with a random pulse is that it is not a smooth function, so the algorithm might find it difficult to get a smooth solution.

There are two approaches we can take to get a good initial pulse. The first approach assumes nothing about the system, it's good since it's really general and can be used in any case, but might be not ideal in some cases. The second approach is when we know roughly how the solution should look like, we can use some pulse that is similar to what we expect and GRAPE will get the actual pulse from that.

In the first approach, we want GRAPE to do most of the work, but not get stuck by some weird problem of the initial pulse. We want a guess that is close to 0, pretty random, but not so much that it would be hard to smoothen. We can get such a pulse by doing a convolution between a random pulse and a Gaussian window

Unlike the first approach, the second approach could look really different for different examples but I'll give some general guidelines you can use the get a good initial pulse. Let's say, for example, you have a 3-level system where the third level isn't wanted (such as in the DRAG example). If you give an initial guess like the one in the first approach, the third level will still be excited by that pulse, and it might be hard for the algorithm to fix this. What you might do in this scenario, is to start with an initial guess that you know excites only the first and second levels of the system but not the third. This is easy since you know the transition frequencies of the atom, from that you know the frequency that excites each level. What you can do is some random pulse that has frequency around the first-second levels frequency difference. So if you look in the frequency space, what you see is some Gaussian distribution around the first levels frequency with some random noise on it.

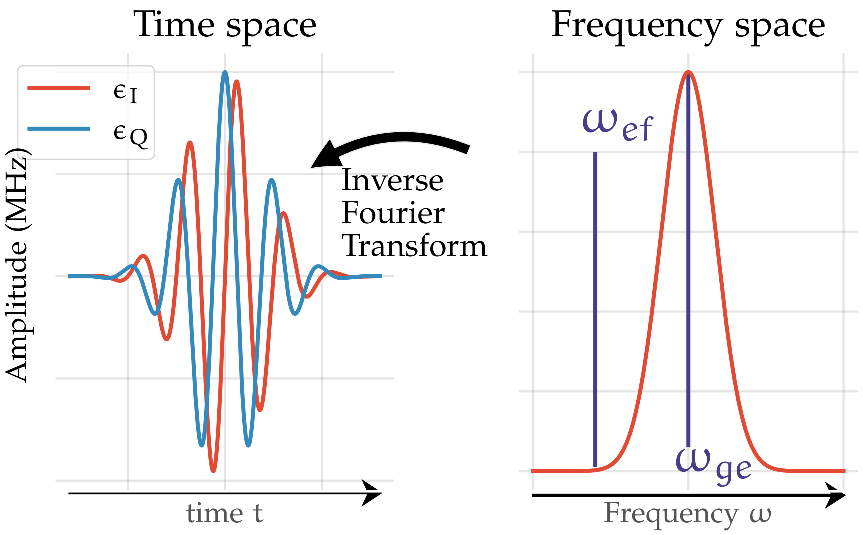


Figure 13: Example of an engineered an initial pulse for system with known characteristics. Normally you would also add some random noise

3.8 FROM STATES TO GATES

Until now we've discussed how to find pulses that take our system from one state to another. This is useful for initializing the quantum system in a desired state. However, the operation is not well defined if we start in a state other than the one the pulse was designed for. In contrast, a numerically optimized quantum gate must perform a well defined unitary transformation on arbitrary states.

Here's the thing, turns out, you can change the algorithm just a little bit and get a GRAPE algorithm that gives you back the optimal pulses that realizes a desired *quantum gate*, instead of just taking you from one state to another.

To make such gate GRAPE, instead of optimizing for one state transformation, you optimize for initial states that constitute a basis for the entire state space. This way, since quantum operations are linear, a unique transformation on the basis of the state space is a unique transformation on the entire system. The implementation of gate GRAPE is outside the scope of this project.

3.9 REFERENCES AND FURTHER READINGS

The best reference I've found on this subject is by far the 2019 dissertation of Philip Reinhold from Yale, "**Controlling Error-Correctable Bosonic Qubits**", especially the forth chapter. Another great resource is the documentation of the python library "**QuTiP**", it's documentation has some really nice explanations.

Part II

EXPERIMENTAL METHODS

Now that we understand the theory, we can go on to the experiments. In this section we see some experimental problems in the control of a quantum system and how to solve them.

4 CONTROLLING A SUPERCONDUCTING QUANTUM COMPUTER

4.1 OVERVIEW

Before jumping into the specifics of how'd we control the quantum computer, let's start with a general overview and show how everything is connected.

This is a diagram that shows how the system is connected, from the pulse generator to the cavity.

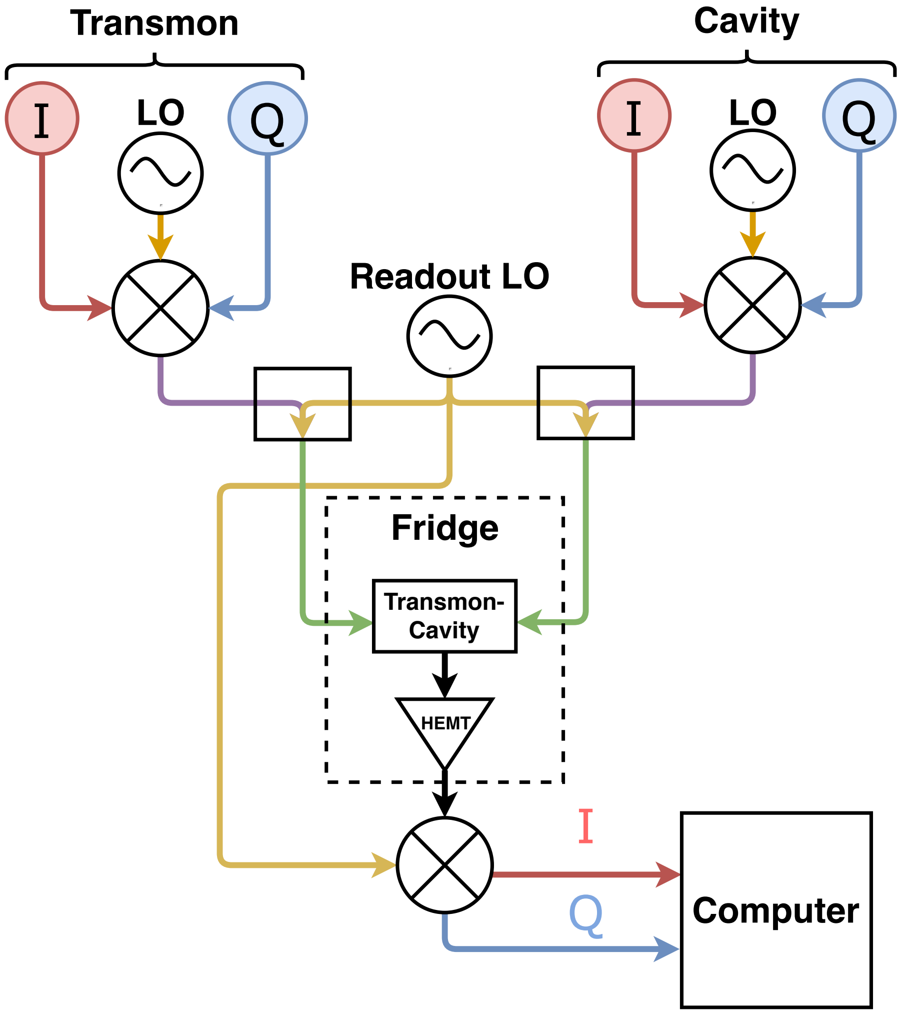


Figure 14: Diagram of the system

The I and Q signals are generated in the AWG, the LO frequency comes from a frequency generator and so does the Read Out signal. **H**igh **E**lectron **M**obility **T**ransistor (HEMT) is a low-noise cryogenic amplifier.

4.2 GENERATING THE PULSES

4.2.1 *The AWG*

An **A**rbitrary **W**aveform **G**enerator (AWG) is a device that we use to generate the pulses we calculated with GRAPE in the previous chapter. To control the qubit we need to send RF signals, typically ranging from 4GHz to 10GHz. Such signals can be generated by RF signal generators (also called LO's - local oscillators). In contrast, the AWG sends out slowly varying envelopes, usually with a bandwidth of a few hundred MHz. To bridge this gap, we will mix a fixed-frequency signal from the LO with a time-varying signal from the AWG. In particular, this will also enable us to vary the frequency of the LO in real-time with the AWG. Our goal in the next section will be to achieve this so called single-sideband modulation of the LO signal.

We also want to be able to un-mix the measurement result to get back only the interesting parts of the pulse. The device we'll use for this task is the *IQ-Mixer*, but before we can get into the IQ-Mixer, we need to understand how a *regular* mixer works.

4.2.2 *The Mixer*

The mixer has 2 inputs and one output. When you enter 2 pulses as an input, you get their product as the output (inputting for example cos(t) and cos(2t) will result in cos(t)cos(2t) at the output). We draw a mixer in a diagram as

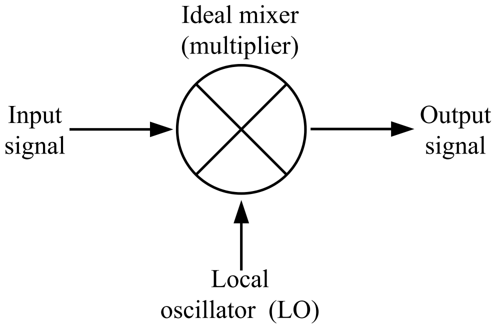


Figure 15: Ideal mixer in a diagram

The mixer circuit is *non-linear*. The non-linearity could be achieved with non-linear components, such as diods.

4.2.3 *The IQ-Mixer*

We've seen what's a *regular* (and *ideal*) mixer is, but how can we use it for the desired effect? remember, we want to input a high frequency and a lower frequency and we want the output to be a wave with a frequency that is the sum of the 2 frequencies. To do so, we can consider the following diagram

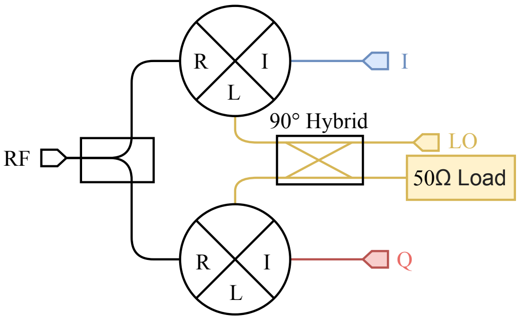


Figure 16: The IQ mixer

Where the 90° hybrid in the diagram is a *90° hybrid coupler*. This device splits the signal into 2 signals at a 90°phase difference, hence the name. The square near the *RF* sign simply adds the 2 waves.

As we can see, the IQ mixer has 3 inputs, In-phase (*I*), Quadrature (*Q*), hence the name, and *LO*. We can also see that there's only one output, *RF* (although you can reverse the roles of the input and the output).

How can we use this IQ mixer to add frequencies? Let's consider the following inputs[1]

$$\begin{aligned} \mathrm{I} &\longrightarrow \cos\left(\omega_{IQ}t\right) \\ \mathrm{Q} &\longrightarrow \sin\left(\omega_{IQ}t\right) \\ \mathrm{LO} &\longrightarrow \sin(\omega_{LO}t) \end{aligned}$$

In this case, the input into the top mixer will be *I* and a *LO*, which is $\cos\left(\omega_{IQ}t\right)\sin(\omega_{LO}t)$. Similarly, the input into the bottom mixer will be *Q* and 90°phase of *LO*, which is $\sin\left(\omega_{IQ}t\right)\cos(\omega_{LO}t)$.

The total output (in *RF*) will be the sum of the two waves

$$\mathrm{RF} = \cos\left(\omega_{IQ}t\right)\sin(\omega_{LO}t) + \sin\left(\omega_{IQ}t\right)\cos(\omega_{LO}t)$$

So we get

$$\boxed{\mathrm{RF} = \sin\left((\omega_{LO} + \omega_{IQ})t\right)} \tag{42}$$

Perfect! this is exactly what we wanted, the output is a wave with frequency that is the sum of the input frequencies. Only one problem, this simple scheme turns out not to work in practice : (.

4.2.4 *Theory VS Reality*

If we use a spectrum analyzer and view what frequencies are in the final wave we get the following picture

[1] You can flip I and Q and get subtraction instead of addition

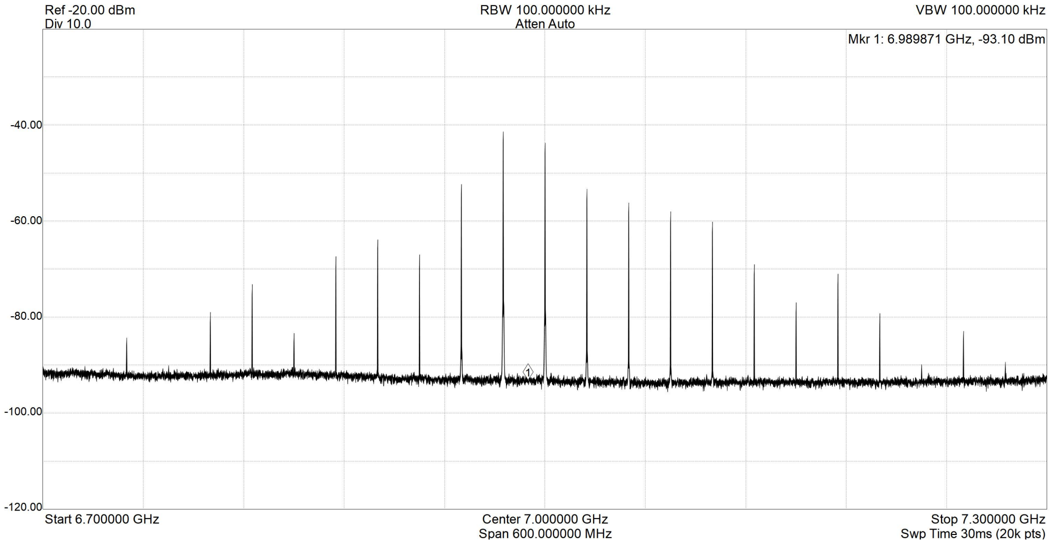


Figure 17: Full Spectrum Without Any Corrections

Zooming in around the LO frequency we see

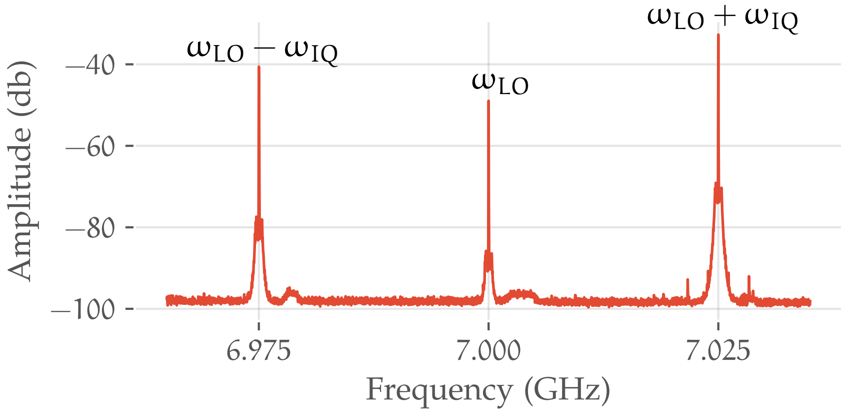


Figure 18: Spectrum Around the LO Frequency

What is the problem? We've proved mathematically that it should work, so why doesn't it? The problem is that we can't just assume the waves to come and go with the same phase, the waves travel through the wire at some speed so if we input into two different wires, two waves that are at the same phase, at the other side of the wire they might come at different phases because of differences in wire length, resistance, etc... So what can we do about it? You could try to make identical parts and make everything just perfect but even slight deviation will cause the system no to work properly, a better solution is to input more complex waves and have some parameters to play with so we can simply find the right parameters for the system.

We can analyze the frequency space of the output of out IQ mixer and we can see two types of it not working correctly

- Leakage at the LO frequency
- Leakage at the sideband ($\omega_{LO} - \omega_{IQ}$)

We can solve the first type of Leakage, at the LO frequency, by adding DC offsets to the input frequencies (We'll prove this mathematically later), and we can solve the second type of leakage by adding phase offsets to the waves (We'll prove this mathematically later).

Before we can solve the problem, we need to understand what's causing it. As we explained earlier, a phase is created in the wires that connect everything.

LEAKAGE AT $\omega_{LO} - \omega_{IQ}$ – Let's consider now inputting into the IQ mixer the same waves but with the phases that were created in the transmission wires instead of what we had earlier

$$\begin{aligned} I &\longrightarrow \cos\big(\omega_{IQ}t + \varphi_I\big) \\ Q &\longrightarrow \sin\big(\omega_{IQ}t + \varphi_Q\big) \\ LO &\longrightarrow \sin(\omega_{LO}t + \varphi_{LO}) \end{aligned}$$

Using the same calculation we did before, we get that

$$\begin{aligned} RF &= I \cdot LO + Q \cdot LO(90^\circ) \\ &= \cos\big(\omega_{IQ}t + \varphi_I\big) \cdot \sin(\omega_{LO}t + \varphi_{LO}) \\ &+ \sin\big(\omega_{IQ}t + \varphi_Q\big) \cdot \cos(\omega_{LO}t + \varphi_{LO}) \end{aligned}$$

After some algebra we get

$$\begin{aligned} RF =& \cos\left(\frac{\varphi_Q - \varphi_I}{2}\right) \sin\left((\omega_{IQ} + \omega_{LO})t + \frac{\varphi_Q + \varphi_I}{2} + \varphi_{LO}\right) \\ &+ \sin\left(\frac{\varphi_Q - \varphi_I}{2}\right) \cos\left((\omega_{IQ} - \omega_{LO})t + \frac{\varphi_Q + \varphi_I}{2} - \varphi_{LO}\right) \end{aligned}$$

This expression is quiet scarier than the one we got earlier... More than that, we get two frequencies instead of one, we've got an unwanted frequency at $\omega_{LO} - \omega_{IQ}$ and the only way to make it disappear is if the phases are equal, $\varphi_I = \varphi_Q$. Also the final wave as a phase of φ_{LO}, this isn't really a problem and if we define our starting point differently we can set φ_{LO} to 0.

LO FREQUENCY LEAKAGE – Another type of leakage we've observed is leakage at the LO frequency, it makes sense that some of the original wave will go through the mixer untouched. To fix that leakage, we'll need to somehow change the I and Q waves to cancel it out. The simplest way to do so is to add DC offsets to the inputs [2]

$$\begin{aligned} I &\longrightarrow \cos\big(\omega_{IQ}t + \varphi_I\big) + \epsilon_I \\ Q &\longrightarrow \sin\big(\omega_{IQ}t + \varphi_Q\big) + \epsilon_Q \\ LO &\longrightarrow \sin(\omega_{LO}t) \end{aligned}$$

[2] I've removed the phase on the LO wave since we've seen it doesn't really change anything

We've seen this story before... the calculation of the RF wave is the same so we'll skip the calculation. The end result is

$$\begin{aligned}\mathsf{RF} =&\cos\left(\frac{\varphi_Q-\varphi_I}{2}\right)\sin\left((\omega_{IQ}+\omega_{LO})t+\frac{\varphi_Q+\varphi_I}{2}\right)\\ &+\sin\left(\frac{\varphi_Q-\varphi_I}{2}\right)\cos\left((\omega_{IQ}-\omega_{LO})t+\frac{\varphi_Q+\varphi_I}{2}\right)\\ &+\epsilon_I\sin(\omega_{LO}t)+\epsilon_Q\cos(\omega_{LO}t)\\ =&\mathsf{RF}_{old}+\epsilon_I\sin(\omega_{LO}t)+\epsilon_Q\cos(\omega_{LO}t)\end{aligned}$$

where RF_{old} is the RF wave before adding the DC offsets.

What we get is the same wave, but now we can play with the LO frequency at the output. Later we'll change the DC offsets so that they will cancel to LO frequency leakage to minimize it.

Now that we have all of our "knobs" we can change and play with, we can start using them to minimize the leakages.

4.2.5 *Finding Optimal Constants*

As we've seen in the previous section, there are 4 variables we can "play" with to get the best variables for our system, as long as we don't change the system, these variables stay the same. What we want to do now is to actually find them. Our system is connected like so

We have the Quantum Machine[3] that generates the I and Q inputs that go into the mixer (and also to an oscilloscope for debugging). There's the frequency generator[4] that is connected to the mixer and generates 7GHz wave, and there's the frequency spectrum analyzer[5] that is connected to the computer.

Let's first attack the leakage at the LO frequency. For now we'll have an LO frequency of 7GHz that we want to change by 25MHz (The same variables work for all frequencies this is just as an example)

LO FREQUENCY LEAKAGE As proven in section 4.2.4, to minimize this kind of leakage all we need is to play with the DC offsets of the IQ inputs. To do so, we first need to define what we want to minimize, which in this case is simply the power of the frequency at 7GHz, we can measure that power with our spectrum analyzer, we'll call that our *cost function*. We have a 2-dimensional variable space, we need to find where in this space the cost function is at a minimum. To do so, we'll start by using a brute force method to find the general location of the minimum in the variable space, since brute force is very inefficient we can't really use it to find the exact location of the minimum so we start by only doing a low precision brute force and then use a different optimization algorithm to find the exact location of the minimum. We'll use the `scipy.optimize.fmin` as the algorithm for precise minimum location.

SIDEBAND LEAKAGE Now that we've minimized the LO frequency leakage, we want to minimize the sideband leakage. We do that by changing the phases of the

[3] this is the heart of the system, for now we'll use it to make the MHz waves with different phases, DC offsets and frequencies

[4] KeySight N5173B

[5] SignalHound USB SA-124B

IQ waves from the quantum machine, it's important that changing the phases doesn't change the LO leakage and luckily for us, as proven in section 4.2.4 that's whats happening. to change the phases we don't simply specify the phases, we use the correction matrix of the Quantum Machine. This time our cost function is the frequency at $\omega_{LO} - \omega_{IQ}$, we can do the same as we did in the LO leakage and use first a brute force optimization to find the general location of the minimum and the use the fmin algorithm to find the exact location of the minimum of the cost function (this time in the scale-angle variable space). You can see the result here

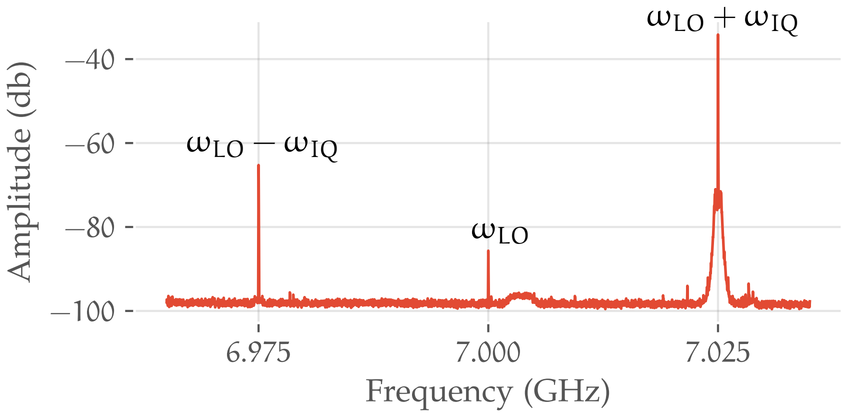


Figure 19: Spectrum Around the LO Frequency of the optimized IQ mixer

The other frequencies aren't completely reduced because nothing is perfect, but ir's pretty close and good enough for this demonstration.

4.3 REFERENCES AND FURTHER READINGS

A great book on couplers and mixers I used while writing this chapter is "**Microwave Engineering**" by David M. Pozar. It provides a much more in depth look on the subject.

Another great resource is the **Marki Microwave RF & Microwave knowledge base** at
`https://www.markimicrowave.com/engineering/`. Which provides useful introduction to many topics in microwave engineering.

5 FUTURE WORK AND CONCLUSIONS

We conclude this project with the knowledge and tools to create and control quantum systems, from understanding them theoretically to connecting and calibrating the devices that actually send the control pulses, to creating the control pulses to create arbitrary quantum states and operations.

As with anything in life, this project must to come to an end at some point, each subject we discuss opens a rabbit hole we'll never see the end of.

A natural next step for this project would be to implement the gate GRAPE. explain exactly how the gate GRAPE from section 3.8 works, and implement it in code. The gate GRAPE is the missing piece needed to actually creating quantum circuits, and it opens many possibilities for quantum computation and information.

Another, pretty obvious, continuation to this project would be, actually implementing it, physically, in an experiment. This project is (almost) entirely theoretical and numerical, for all you know I was lying to you the hole time. It would be nice to actually check GRAPE on an actual quantum system. Unfortunately, at this point the Quantum Circuits Laboratory is not yet fully built and there is no fridge to cool the quantum system in.

Another worthwhile extension of this project is to develop numerically optimized operations for multiqubit systems. This shouldn't be that difficult since the qubit-qubit (or atom-atom) interaction aren't that different from the qubit-cavity interaction we already have, and our optimal control code accepts quantum states in any Hilbert space. Evidently, the main limitation would be the exponential time required for the numerics as the number of qubits grows. Luckily, in an actual quantum computer we will only want to perform local operations on a few qubits, while ignoring the rest.

Part III

APPENDIX

Funnily enough, I actually find the appendices to be more interesting than the project itself.

A ANALYTICAL CALCULATIONS OF OPTIMAL PULSES

Throughout chapter 3 we embarked on a journey finding optimal pulses with numerical methods, but it's important to note that in some specific cases we can calculate the solution analytically. This has more uses than for mathematical beauty, we can use these cases as test cases to debug our GRAPE algorithm.

We're going to start with everyone's favourite, Shrödinger's equation[1]

$$\dot{\psi} = -i\hat{\mathcal{H}}\psi$$

The qubit is in a general superposition of the ground and excited states

$$\psi = C_g(t)\left|g\right\rangle + C_e(t)\left|e\right\rangle$$

where C_g and C_e are the probability amplitudes of the ground and excited states.

Shrödinger's equation now becomes

$$\dot{C}_g(t)\left|g\right\rangle + \dot{C}_e(t)\left|e\right\rangle = -i\hat{\mathcal{H}}_{atom}(C_g(t)\left|g\right\rangle + C_e(t)\left|e\right\rangle) \tag{43}$$

The Hamiltonian of an atom interacting with a classical field (ignoring the cavity) was derived section 2.3.2 and is given from equation 27. We can write it as

$$\hat{\mathcal{H}}_{atom} = \omega_0 \hat{a}^\dagger \hat{a} + \Omega(t)\hat{\sigma}_x = \omega_0 \left|e\right\rangle\left\langle e\right| + \Omega(t)(\left|g\right\rangle\left\langle e\right| + \left|e\right\rangle\left\langle g\right|)$$

Where $\Omega(t)$ is the electromagnetic field amplitude.

Replacing $\mathcal{H}_{atom}$ in equation (43) with the expression we have for it the equation becomes

$$\dot{C}_g(t)\left|g\right\rangle + \dot{C}_e(t)\left|e\right\rangle = -i(\ \omega_0\left|e\right\rangle\left\langle e\right| + \Omega(t)(\left|g\right\rangle\left\langle e\right| + \left|e\right\rangle\left\langle g\right|)\ ) \cdot (C_g(t)\left|g\right\rangle + C_e(t)\left|e\right\rangle)$$

Some algebra magic later (remembering that $\{\left|g\right\rangle, \left|e\right\rangle\}$ constitutes an orthonormal basis, so $\left\langle e|e\right\rangle = \left\langle g|g\right\rangle = 1$ and $\left\langle g|e\right\rangle = \left\langle e|g\right\rangle = 0$)

$$\dot{C}_g(t)\left|g\right\rangle + \dot{C}_e(t)\left|e\right\rangle = -i\omega_0 C_e\left|e\right\rangle - i\Omega(t)(C_e\left|g\right\rangle + C_g\left|e\right\rangle)$$

We can left multiply this equation once with $\left\langle g\right|$ and once with $\left\langle e\right|$, getting the 2D system of differential equations

$$\begin{aligned} \left\langle g\right| \quad &\to \quad \dot{C}_g(t) = -i\Omega(t)C_e(t) \\ \left\langle e\right| \quad &\to \quad \dot{C}_e(t) = -i\omega_0 C_e(t) - i\Omega(t)C_g(t) \end{aligned}$$

Instead of looking at an arbitrary pulse $\Omega(t)$, we can look at a sinusoidal pulse of the form $\Omega(t) = \Omega_0 e^{i\omega t}$ where ω is the frequency of the pulse. The general equations now become

$$\begin{aligned} \dot{C}_g(t) &= -i\Omega_0 e^{i\omega t} C_e(t) \\ \dot{C}_e(t) &= -i\omega_0 C_e(t) - i\Omega_0 e^{i\omega t} C_g(t) \end{aligned}$$

[1] Planck's reduced constant is set to 1, $\hbar = 1$

It's comfortable to make the unitary transformation

$$\alpha(t) \rightarrow C_g(t) \quad \beta(t) \rightarrow e^{i\omega t} C_e(t)$$

and after substituting and a bit of algebra get the linear system of equations

$$\dot{\alpha} = i\Omega_0 \beta$$
$$\dot{\beta} = i\Omega_0 \alpha + i\Delta\beta$$

where $\Delta = \omega - \omega_0$ and is known as the detuning. Deriving the second equation over time and substituting $\dot{\alpha}$ with it's known expression we get the following differential equation for β

$$\ddot{\beta} - i\Delta\dot{\beta} + \Omega_0^2 \beta = 0$$

We can find the solution by "guessing" a solution of the form $\beta(t) = e^{At}$ and when plugging that into the equation we get the quadratic

$$A^2 - i\Delta \cdot A + \frac{\Omega_0^2}{4} = 0$$

Assuming my high school math teacher wasn't lying to me, the solutions to this equation are

$$A_{\pm} = \frac{i\Delta \pm i\sqrt{\Delta^2 + 4\Omega_0^2}}{2}$$

We'll define the parameter

$$\Omega = \sqrt{\Delta^2 + 4\Omega_0^2}$$

The two solutions we found constitute a basis of solution for the linear equation, so the general solution is of the form

$$\beta(t) = e^{i\frac{\Delta t}{2}} (C_1 e^{i\frac{\Omega t}{2}} + C_2 e^{-i\frac{\Omega t}{2}})$$

We are looking for solutions that start at the ground state, this gives us an initial condition

$$\beta(0) = 0 \quad \Rightarrow \quad C_1 + C_2 = 0$$
$$\alpha(0) = 1 \quad \Rightarrow \quad \dot{\beta}(0) = i\Omega_0 = i\Omega C_1$$

Plugging in the calculated coefficients and the solution becomes

$$\beta(t) = 2i\frac{\Omega_0}{\Omega} e^{i\frac{\Delta t}{2}} \sin\left(\frac{\Omega}{2} t\right)$$

Remember that β is the population of the excited state with the addition of a phase, the phase doesn't change the probabilities so

$$P_e(t) = |C_e|^2 = |\beta|^2 = \frac{\Omega_0^2}{\Omega^2}(1 - \cos(\Omega t))$$

where $P_e(t)$ is the probability for the atom to be in an excited state at time t.

The result we got, where the atom oscillates between the ground and excited states is called **Rabi Oscillations**. Figure 20 plots this result, once with zero detuning (and therefore $\Omega = \Omega_0$) and once with non zero detuning (and therefore $\Omega > \Omega_0$)

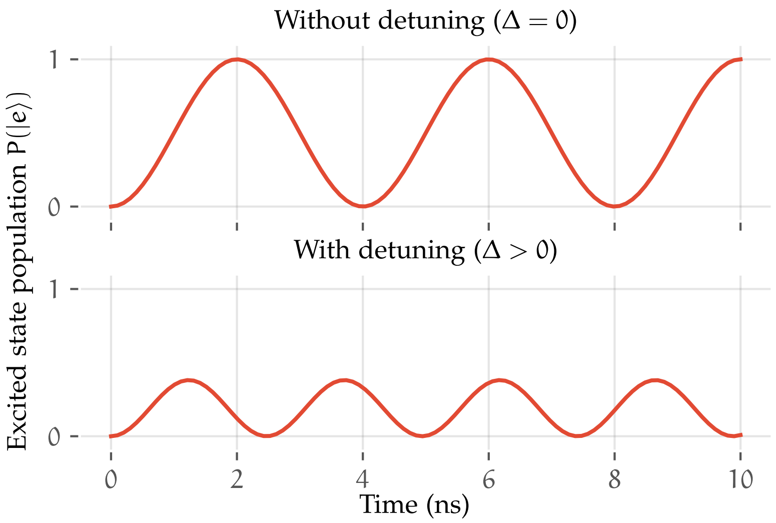


Figure 20: Rabi oscillations of atom in a classical electromagnetic field. The top graph is without detuning $\Delta = 0$ and the bottom graph is with detuning $\Delta \neq 0$

You can see that with zero detuning, the atom oscillates fully between ground and excited states. With non-zero detuning, the atom doesn't fully reach the excited state, it goes only part of the way and then goes back down. This means that if you want to get to the excited states you need to send a pulse with exactly the same frequency as the atom.

We can take everything we got so far and construct a pulse that will take the ground state into the excited state. First, the pulse must have the same frequency as the atom $\omega = \omega_0$. If the total duration of the pulse is T, then the excited state is fully populated at time T

$$P_e(T) = 1 = (1 - \cos(\Omega_0 T))$$

And the solutions are

$$\Omega_0 = \frac{\pi k}{T} \quad \text{for} \quad k = 1, 2, 3, \ldots$$

Where k corresponds to the amount of oscillations between the ground and excited states. Ideally the atom would go directly to the excited state without oscillating between the states, so we'll set $k = 1$.

Putting it all together, for a two level atom with frequency ω_0, and a pulse with duration T, the pulse you need to send to get from the ground state to the excited state is

$$\boxed{\Omega(t) = \frac{\pi}{T} e^{i\omega_0 t}}$$

Where the real and imaginary parts correspond the sin and cos waves we send to the atom (or I and Q pulses if you prefer). Writing them explicitly we get

$$\boxed{\Omega_I(t) = \frac{\pi}{T} \sin(\omega_0 t)} \quad \text{and} \quad \boxed{\Omega_Q(t) = \frac{\pi}{T} \cos(\omega_0 t)}$$

We can actually check using the simulation we made in chapter 3 and see that yes! These pulses lead to the qubit going from the ground to excited states, amazing!

There is actually a more general result about *π-pulses* we can see from here, we won't prove it but we can see it works. Any pulse that satisfies

$$\int_0^T \Omega(t)dt = \pi$$

Would take the ground state atom to the excited state. You can use a something like a Gaussian or any other weird pulse that satisfies this condition, and they will all work. We can see that indeed, the pulse we derived does satisfy it.

There are actually many more examples we can find analytical solutions for. There are even analytical solution for the 3-level system and DRAG. We won't touch on them here, they are much more complicated, although they are definitely possible to calculate analytically.

B QUANTIZING ELECTRIC CIRCUIT AND THE JOSEPHSON JUNCTION

Throughout the project, I mentioned several times that the physical implementation of the qubit is not the subject of the project and ignored it. It would be a crime no to at least give a simple explanation of the implementation of the qubit, especially since we dedicated and entire section (section 3.5.1) to the problem with our physical implementation. In this appendix I will try to give a simple explanation of the qubit as a quantum LC circuit, using the tools we got when we quantized the electromagnetic field.

This appendix is here for two reasons. The first we already mentioned, to gain some insights on how the qubit is implemented physically. The second, less obvious reason, is to really drive home the point that Dirac's method for Canonical quantization can be applied to any oscillating phenomena. From mechanical oscillator, to the electromagnetic field and even an LC circuit, as we'll see shortly.

B.1 QUANTIZING THE LC CIRCUIT

The LC circuit is constructed by connecting a capacitor with capacitance C to a coil with inductance L, hence the name, *LC*. Drawing it in a diagram is as shown

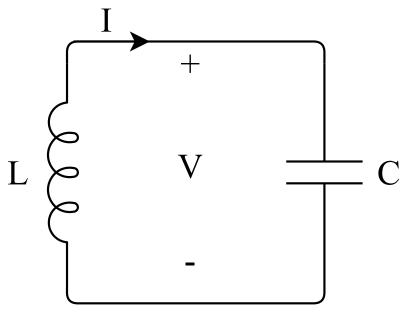


Figure 21: The LC circuit

To begin the quantization we first need choose a pair of canonically conjugate variables to represent the state of the system. The most obvious pair of variables would be the voltage V and the current I, but turns out they are not canonically conjugate variables. Instead, we'll choose the charge of the capacitor q and the inductor's magnetic flux φ to be the variables.

Let's check that they are indeed canonically conjugate. The energy (and therefore, the Hamiltonian) of the capacitor's electric field is given by

$$\mathcal{H}_C = \frac{q^2}{2C}$$

and the energy of the magnetic field stored in the inductor is given by

$$\mathcal{H}_L = \frac{\varphi^2}{2L}$$

Now we can sum these two donations to the Hamiltonian and get that the Hamiltonian of the LC circuit is

$$\mathcal{H}_{LC}(q,\varphi) = \frac{\varphi^2}{2L} + \frac{q^2}{2C}$$

The Hamilton equations in terms of q and φ are

$$\dot{q} = \frac{\partial \mathcal{H}_{LC}}{\partial \varphi} = \frac{\varphi}{L} = I \tag{44}$$

$$\dot{\varphi} = -\frac{\partial \mathcal{H}_{LC}}{\partial q} = -\frac{q}{C} = -V \tag{45}$$

These equation are correct since 44 is the definition of current and 45 is the definition of potential. Therefore q and φ are canonically conjugate variables and we can go on and quantize them.

To quantize these variables we simply need to replace them with operators

$$q \to \hat{q}$$
$$\varphi \to \hat{\varphi}$$

satisfying the commutation relation

$$[\hat{q}, \hat{\varphi}] = i\hbar$$

as always with quantum harmonic oscillator we'll define annihilation and creation operators

$$\hat{a} = \frac{\hat{q}}{q_0} + i\frac{\hat{\varphi}}{\varphi_0}$$
$$\hat{a}^\dagger = \frac{\hat{q}}{q_0} - i\frac{\hat{\varphi}}{\varphi_0}$$

where

$$q_0^2 = 2\hbar\sqrt{\frac{C}{L}}$$
$$\varphi_0^2 = 2\hbar\sqrt{\frac{L}{C}}$$

After some algebra magic we can get the expression of the Hamiltonian

$$\hat{\mathcal{H}}_{LC} = \hbar\omega(\hat{a}^\dagger\hat{a} + \frac{1}{2}) \quad \text{with} \quad \omega = \frac{1}{\sqrt{LC}}$$

This is the good old expression for the Hamiltonian of a quantum harmonic oscillator and we can treat it as we did with any other quantum harmonic oscillator.

B.2 ARTIFICIAL ATOMS - THE JOSEPHSON EFFECT

The problem with the simple quantum LC circuit is that it's energy levels are *linear*. By linear, I mean that the difference in energy between two neighboring levels is the same for every level, $E_{n+1} - E_n = \hbar\omega$ for all n.[1] This is a problem since we

[1] Unlike an actual atom where the energy difference decreases at higher levels

want to treat it as a two level system. If we send a pulse with $\hbar\omega$ energy, we want it to affect only the lower two levels, but since the energy difference between all the levels is the same, sending such a pulse would also excite the higher levels uncontrollably.

We want to introduce *un-linearity* to the levels, so still $E_1 - E_0 = \hbar\omega_{01}$ but $E_2 - E_1 = \hbar\omega_{12}$, $E_3 - E_2 = \hbar\omega_{23}$ and so on, where $\omega_{n\ n+1} \neq \omega_{01}$. This way if the system is only populated at one of the lower two levels and we send a pulse with energy $\hbar\omega$, the only levels affected are the lower two and not the higher ones[2].

To create these a-linearities we can use the ***Josephson Junction***. A Josephson junction is simply a cut in the wire, a very (very) thin cut, the two pieces of wire are around 10nm apart. The Josephson junction replaces that inductor in the LC circuit. In a circuit diagram it's drawn as a little x, and the circuit diagram of the LC circuit with it is

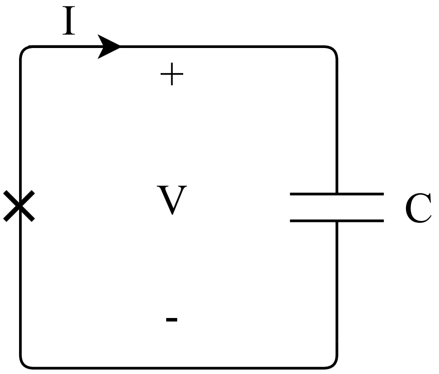


Figure 22: The LC circuit with a Josephson junction

And the effect of adding a Josephson junction is replacing the Hamiltonian of the inductor $\frac{\hat{\varphi}^2}{2L}$ with $E_j \cos(\frac{2e}{\hbar}\hat{\varphi})$.

$$\hat{\mathcal{H}}_{\text{Josephson}} = \frac{\hat{q}^2}{2C} + E_j \cos\left(\frac{2e}{\hbar}\hat{\varphi}\right)$$

This does exactly what we want, for the first level we can approximate the cosine as a parabola and get $E_1 - E_0 = \hbar\omega_{01}$ but since the cosine diverges from the parabola after that the difference between the energy levels lessens and lessens between each pair of higher levels. Here's a visual representation of the energy level of the LC circuit with the Josephson junction

[2] This is obviously an ideal case, as we seen in section 3.5.1 with DRAG

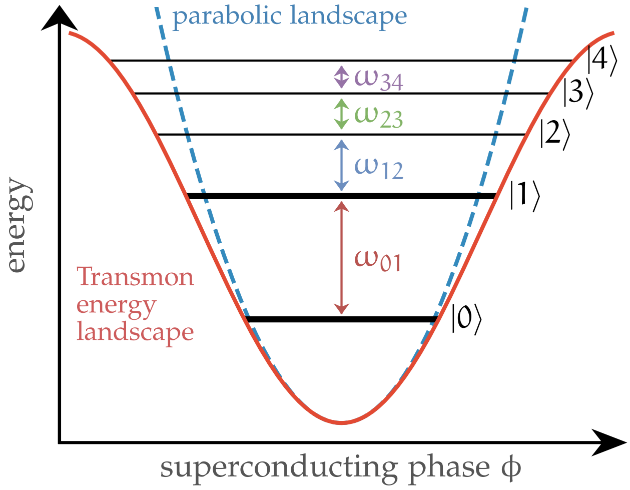


Figure 23: Potential of the circuit as a function of the superconducting phase, and the non-linear energy levels of the Josephson junction

We can treat the Josephson junction as a potential barrier, and calculate the wave function on both sides of the wire.

We'll call the wave function on one side of the junction ψ_1 and on the other side ψ_2, their dynamics are determined by the coupled shrödinger equations:

$$i\hbar\frac{\partial\psi_1}{\partial t} = \mu_1\psi_1 + K\psi_2$$
$$i\hbar\frac{\partial\psi_2}{\partial t} = \mu_2\psi_2 + K\psi_1$$

Where K is the coupling across the barrier and $\mu_{1/2}$ are the lowest energy states of each wave function. We'll "guess" solutions of the form

$$\psi_{1/2} = \sqrt{n_{1/2}}e^{i\theta_{1/2}}$$

Yielding the two equations

$$\hbar\frac{\partial n_1}{\partial t} = -\hbar\frac{\partial n_2}{\partial t} = 2K\sqrt{n_1 n_2}\sin(\theta_2 - \theta_1)$$
$$-\hbar\frac{\partial(\theta_2 - \theta_1)}{\partial t} = \mu_2 - \mu_1$$

$n_{1/2}$ are the density of cooper pairs, and by definition of the current, the current is $\frac{\partial n_1}{\partial t}$. When voltage is applied the energy levels shift as $\mu_2 - \mu_1 = 2eV$. Finally to make the equations shorter we'll write $I_0 = 2K\sqrt{n_1 n_2}$ and $\delta = \theta_2 - \theta_1$. The equations now become

$$I = I_0 \sin\delta$$
$$\frac{\partial\delta}{\partial t} = \frac{2eV}{\hbar}$$

We can now calculate the electric power (which is the derivative of the energy) from the classical equation

$$\frac{\partial E}{\partial t} = P = IV = (I_0\sin\delta)\cdot(\frac{\hbar}{2e}\frac{\partial\delta}{\partial t}) = -\frac{I_0\hbar}{2e}\frac{\partial\cos\delta}{\partial t}$$

Integrating over time, we get that the expression for the energy is

$$E = -\frac{I_0 \hbar}{2e} \cos \delta$$

If we define $E_j = -\frac{I_0 \hbar}{2e}$, replace δ with $\hat{\varphi}$, and replace the energy with the Hamiltonian, we get

$$\hat{\mathcal{H}}_{\text{Josephson}} = E_j \cos \hat{\varphi}$$

just as we've written earlier.

This is not the best treatment of the Josephson junction but it's a rough idea of the explanation. This expression gives us the a-linearities we want to get to implement the qubit.

C ANOTHER METHOD: COMPUTATIONAL GRAPHS

In the optimal control section, in essence, we try to minimize the cost of many many variables, this is a similar problem that to teaching neural networks and machine learning, we might be able to borrow some tricks and method they use to solve our problem.

Instead of thinking of a cost function as just a function, we can treat it as a *computational graph*. We'll discuss briefly about what's a computational graph and how can we implement GRAPE using one. We won't go in depth as we did in the chapter on optimal control, it's only a general overview meant to show an alternative method. Let's start by defining a computational graph[1].

C.1 WHAT ARE COMPUTATIONAL GRAPHS

A computational graph consists of nodes and connection between them, a node can be one of one of three things

- **Operations** (red), the operation takes a list of numbers from other nodes and outputs another list of numbers (doesn't need to be of same size)
- **Parameters** (orange), these are, as the name suggests, the parameters of the graph and can be used by the operations.
- **Variables** (blue), these are the variables that of the graph, and they too can be used by the operators of the graph

The graph starts as the variables, goes through some operations that use parameters and gives out some resualt. Let's take a look at a simple example

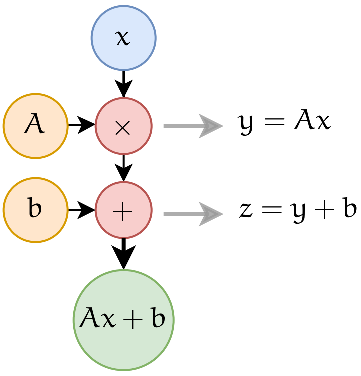


Figure 24: Example of a basic computational graph

Here, x is the variable, A and b are the parameters, and y and z are the operators. The function that this graph represents is $y = a \cdot e^x$. For now it properly seems overkill to use the graph to represent that simple of a function but in the next

[1] Of course, graph theory is an entire field of mathematics by itself and we can't really give a rigorous definition, but more of an intuitive explanation

two sections you'll see the magic of the computational graph, in terms of thinking about the gradient.

C.2 BACK-PROPAGATION

This is the magic of the computational graph, calculating the gradient of the cost function by back-propagating the derivatives to the initial variables. The idea is simple, instead of trying to directly calculating the gradient, we calculate the derivative of node by the previous node, and relate the cost function to the variables by the chain rule. What does that mean, let's show an example.

Let's say we have a simple computational graph that looks like

$$x \to L_1 \to L_2 \to L_3 \to y$$

and we want to calculate the derivative, $\frac{\partial y}{\partial x}$. Instead of calculating the derivative directly, we can back-propagate the derivative as

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial L_3} \cdot \frac{\partial L_3}{\partial L_2} \cdot \frac{\partial L_2}{\partial L_1} \cdot \frac{\partial L_1}{\partial x} \cdot \frac{\partial y}{\partial x}$$

luckily for us each operation is very simple so each derivative is simple as well.

C.3 GRAPE IN A COMPUTATIONAL GRAPH

Now we can finally implement grape as a computational graph, we'll consider the simplest case of GRAPE, no constraints, only the original definition (see equations 36, 34 and 31)

$$F(\vec{\epsilon}) = \left| \langle \psi_{target} | \prod_{i=0}^{N} e^{-i \cdot \delta t (H_0 + \epsilon_k H_d)} | \psi_{initial} \rangle \right|^2$$

In this case, $\vec{\epsilon}$ is the variable, H_0, $-i \cdot \delta t$, $\langle \psi_{target}|$ and $|\psi_{initial}\rangle$ are the parameters. They relate through the operators and everything can be displayed as in figure. 25[2]

The layers are as follows

$$L_1^k = -i\delta t H_d \epsilon_k$$
$$L_2^k = -i\delta t H_0 + L_1^k$$
$$L_3^k = e^{L_2^k}$$
$$L_4 = \prod_{k=0}^{N} L_3^k$$
$$C = \langle \psi_{target} | L_4 | \psi_{initial} \rangle$$

And the derivatives are calculated rather easily as

[2] Simplifying for 3 time steps, to add more time steps is pretty straight forward

$$
\begin{aligned}
\frac{\partial L_1^k}{\partial \epsilon_k} &= -i\delta t H_d \\
\frac{\partial L_2^k}{\partial L_1^k} &= 1 \\
\frac{\partial L_3^k}{\partial L_2^k} &= e^{L_2^k} \\
\frac{\partial L_4}{\partial L_3^k} &= \prod_{i \neq k} L_3^i \\
&\downarrow \\
\frac{\partial C}{\partial \epsilon_k} &= \langle \psi_{target}| \frac{\partial L_4}{\partial L_3^k} \frac{\partial L_3^k}{\partial L_2^k} \frac{\partial L_2^k}{\partial L_1^k} \frac{\partial L_1^k}{\partial \epsilon_k} |\psi_{init}\rangle
\end{aligned}
$$

It's important to note that $\frac{\partial L_1^i}{\partial \epsilon_j} = 0$ for $i \neq j$ so we didn't refer to those derivative (This is true for the derivative between any two layers).

We now everything we need to implement GRAPE as a computational graph, and we can use a library, such as google's tensorflow, to find the optimal pulse.

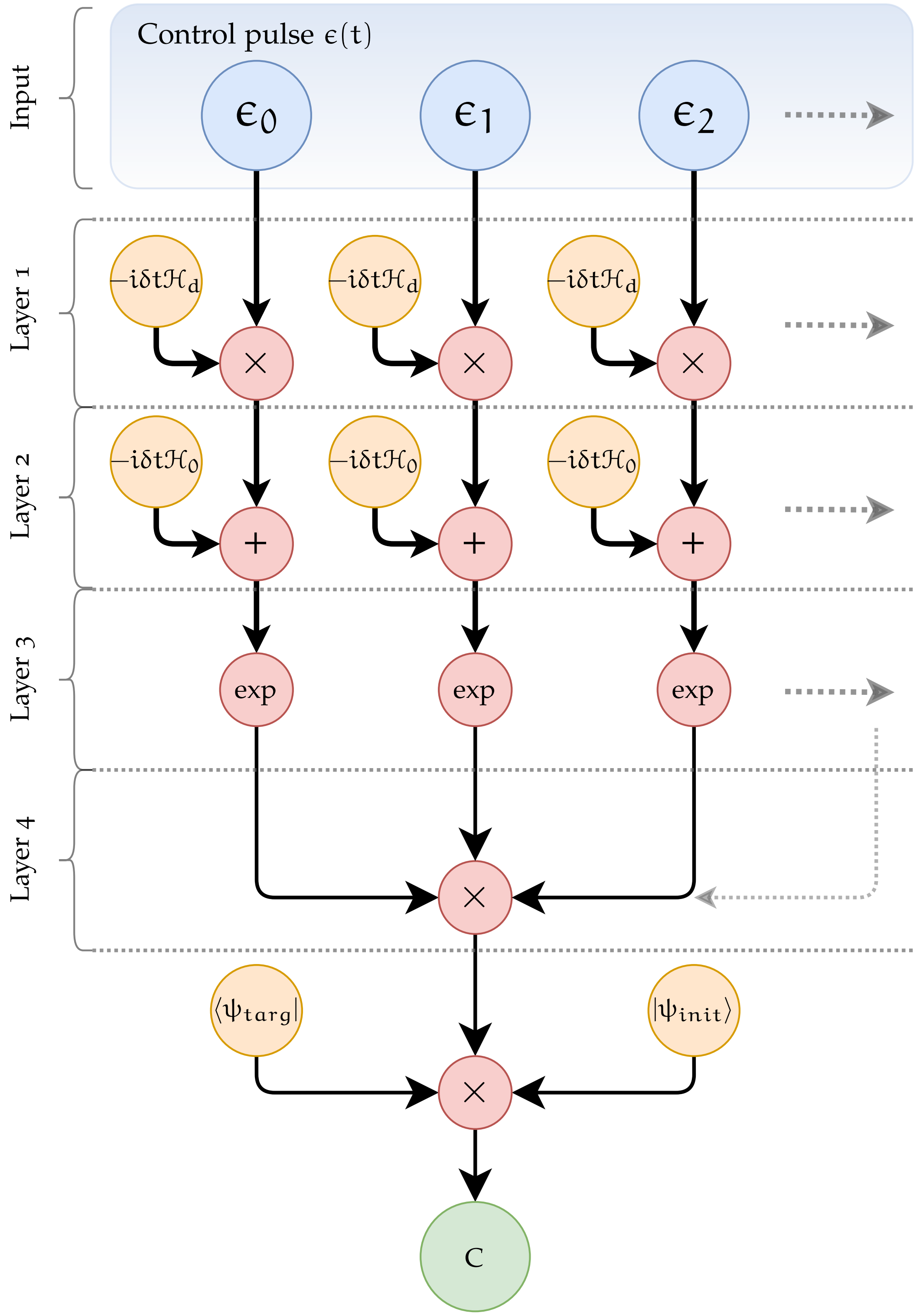


Figure 25: Diagram of the computational graph implementation of GRAPE