Abstract

Controlling Error-Correctable Bosonic Qubits

Philip Reinhold

2019

The harmonic oscillator is a ubiquitous system in physics, describing a wide range of phenomena, both classically and quantum mechanically. While oscillators are relatively straightforward to control classically, they present much more of a challenge in the quantum realm where such systems, modeled as Bosonic modes, have many more degrees of freedom. Controlling Bosonic modes is a crucial task in light of proposals to use these systems to encode quantum information in a way that is protected from noise and dissipation. In this thesis a variety of approaches to controlling such systems are discussed, particularly in the superconducting microwave domain with cavity resonators. In the first part, an experiment demonstrates how a simple dispersively coupled auxiliary system results in universal control, and therefore allows the synthesis of arbitrary manipulations of the system. This approach is employed to create and manipulate states that constitute an error-correctable qubit. The main drawback of this approach is the way in which errors and decoherence present in the auxiliary system are inherited by the oscillator. In the second part, I show how these effects can be suppressed using Hamiltonian engineering to produce a simple form of first-order "fault-tolerance." This approach allows us to demonstrate versions of cavity measurements and manipulations that are protected from dominant error mechanisms.

Controlling Error-Correctable Bosonic Qubits

A Dissertation
Presented to the Faculty of the Graduate School
of
Yale University
in Candidacy for the Degree of
Doctor of Philosophy

by
Philip Reinhold

Dissertation Director: Robert J. Schoelkopf

December 2019

Contents

List of Figures

Introduction **1**

Controlling cavities **9**

Doing more with less: error correction with harmonic oscillators **29**

Numerical quantum optimal control **53**

Meet the samples **73**

Universal control of a cat-code qubit **97**

Venturing forth in frequency space: sideband drives **116**

A Fault-Tolerant Parity Measurement **125**

Fault-tolerant SNAP **147**

Conclusion and Prospects **167**

Tomographies, large and small **200**

Taming the Black Mamba: Structure and software for the Yngwie FPGA quantum controller **211**

List of Tables

Meet the samples **73**

Universal control of a cat-code qubit **97**

A Fault-Tolerant Parity Measurement **125**

Taming the Black Mamba: Structure and software for the Yngwie FPGA quantum controller **211**

List of Abbreviations and Symbols

Abbreviations

ADC	analog-digital converter, page 211
AWG	Arbitrary waveform generator, page 54
BBQ	Black Box Quantization, page 81
BCH	Baker-Campbell-Hausdorff expansion of $e^{\boldsymbol{A}}\boldsymbol{B}e^{-\boldsymbol{A}}$, page 186
CQED	cavity quantum electrodynamics, page 13
cQED	circuit quantum electrodynamics, page 13
DAC	digital-analog converter, page 211
DFT	Discrete Fourier Transform, page 65
DSL	domain specific language, page 227
FPGA	field-programmable gate array, page 211
FT	fault tolerant, page 126
GKP	Gottesman-Kitaev-Preskill cavity encoding, page 51
GRAPE	Gradient ascent pulse engineering, page 54
HEMT	High electron mobility transistor, page 94
iRB	Interleaved randomized benchmarking, page 111
JPC	Josephson Parametric Converter, page 93
QEC	Quantum error correction, page 97
RB	Randomized benchmarking, page 111
SNAP	selective number-dependent arbitrary phase operation, page 23
SPAM	State preparation and measurement. Usually the errors associated with these processes, page 109
SSB	single-sideband modulation. A method of mixing two signals together to produce only the sum or difference frequencies, rather than both., page 213

Greek Characters

α_T The transmon anharmonicity $\omega_{ef} - \omega_{ge}$, page 79

$\mathbf{\Pi}$ The photon number parity operator, page 18

$\mathbf{\Pi}_{\mathsf{FT}}$ The fault-tolerant parity measurement protocol using the $|g\rangle$ and $|f\rangle$ ancilla states, as well as a χ_{fe}-cancelling drive, page 139

$\mathbf{\Pi}_{gf}$ A parity measurement protocol using the $|g\rangle$ and $|f\rangle$ ancilla states, page 139

$\mathbf{\Pi}_{ge}$ A parity measurement protocol using the $|g\rangle$ and $|e\rangle$ ancilla states, page 139

χ The strength of the dispersive interaction between a transmon and cavity. Equivalent to χ_e., page 15

χ' The higher-order cavity-transmon dispersive shift via $\frac{\chi'}{2}(\boldsymbol{a}^\dagger)^2\boldsymbol{a}^2\boldsymbol{b}^\dagger\boldsymbol{b}$, page 86

χ_{RO} The readout-transmon dispersive interaction strength, page 82

χ_e The cavity-transmon dispersive shift associated with the excited state $|e\rangle$ via $\chi_e\boldsymbol{a}^\dagger\boldsymbol{a}\,|e\rangle\langle e|$. Equivalent to χ., page 86

χ_f The cavity-transmon dispersive shift associated with the excited state $|f\rangle$ via $\chi_f\boldsymbol{a}^\dagger\boldsymbol{a}\,|f\rangle\langle f|$, page 86

χ_{fe} $\chi_f - \chi_e$, page 131

χ_{fg} $\chi_f - \chi_g = \chi_f$, page 143

χ_{eg} $\chi_e - \chi_g = \chi_e$, page 131

Δ A frequency difference, page 14

$\hat{\sigma}_\pm$ Pauli raising or lowering operators, page 13

$\hat{\sigma}_{x,y,z}$ Pauli X/Y/Z operators, page 13

κ The cavity loss rate, page 74

κ_{ext} The loss rate induced by (usually intentional) coupling to a transmission line, page 83

κ_{int} The loss rate induced by (usually unintentional) coupling to internal degrees of freedom, such as phonons, two level systems, and quasiparticles, page 83

$|\alpha\rangle$ Coherent state with amplitude α, page 12

$|\mathcal{C}_\alpha^\pm\rangle$ The even or odd 2-legged Cat state of amplitude α, page 22

$\Omega(t)$ The qubit/transmon driving profile, page 16

ω_{RO} The readout resonant frequency, page 82

ω_c The storage cavity frequency, page 13

ω_q	The (Jaynes-Cummings) qubit resonant frequency, page 13
ω_{ef}	The transmon $\|e\rangle \leftrightarrow \|f\rangle$ transition frequency, page 99
ω_{ge}	The transmon $\|g\rangle \leftrightarrow \|e\rangle$ transition frequency, page 79
$\varepsilon(t)$	Cavity drive profile, page 11

Latin Characters

$\boldsymbol{a}, \boldsymbol{a}^\dagger$	Cavity annihilation and creation operators, page 11
$\boldsymbol{b}, \boldsymbol{b}^\dagger$	Transmon annihilation and creation operators, page 86
$\boldsymbol{C}_\phi$	Entangling conditional phase operation, page 16
$\boldsymbol{D}_\alpha$	Displacement operator (by an amount α), page 12
$\boldsymbol{I}_c$	Identity operation on the cavity, page 16
$\boldsymbol{I}_q$	Identity operation on the qubit, page 16
$\boldsymbol{P}_{\text{even/odd}}$	The projector onto the even or odd photon number parity subspace, page 18
$\boldsymbol{r}, \boldsymbol{r}^\dagger$	Readout annihilation and creation operators, page 86
$\boldsymbol{R}_\phi(\theta)$	Qubit rotation by an angle θ around axis given by $\cos(\phi)\hat{x} + \sin(\phi)\hat{y}$, page 16
$\boldsymbol{S}(\vec{\theta})$	SNAP operation with phases θ_k, page 23
$\|C_\alpha^\pm\rangle$	The even/odd cat state with amplitude α, page 38
$\|e\rangle$	The transmon first excited state, page 16
$\|f\rangle$	The transmon second excited state, page 16
$\|g\rangle$	The transmon ground state, page 16
$\|h\rangle$	The transmon third excited state, page 16
$\mathcal{F}$	Fidelity, either of a state or process, page 109
$\mathcal{N}$	A normalization constant, page 38
$\mathcal{T}$	The time-ordering operator, page 12
$\mathscr{F}$	The Fourier transform operator, taking a time domain representation to a frequency domain representation, page 71
$D[\boldsymbol{A}]$	The dissipator Liouvillian for jump operator $\boldsymbol{A}$, page 35
E_C	Cooper pair box/Transmon charging energy, page 78
E_J	Josephson energy determining the hopping rate of cooper pairs across the tunnel barrier, page 78

g	The rate associated with a photon exchange term, page 14
K	Cavity self-Kerr anharmonicity, page 86
Q	The quality factor $Q = \omega/\kappa$, page 74
T_1^c	Cavity energy decay rate, page 86
T_1^{ef}	Transmon $f \to e$ relaxation rate, page 86
T_1^{ge}	Transmon $e \to g$ relaxation rate, page 86
T_2^c	Cavity coherence time, page 135
T_ϕ	Transmon dephasing rate, page 86
$W_\alpha(\boldsymbol{\rho})$	The Wigner function associated with the quantum state $\boldsymbol{\rho}$, page 20

Acknowledgements

Working on the fourth floor of Becton is every day a humbling and inspiring experience, and its not because of the architecture. I'd like to express my gratitude to everyone who helped me along the way in this journey.

Firstly, I will be forever indebted to ROBERT SCHOELKOPF for allowing me to be a part of his team. Rob has created an environment that allows science (and scientists) to flourish. In addition to sound advice about microwave hygiene, ground loops, and experimental best practices, Rob has taught me about the importance of crafting a message, and always keeping an eye on the bigger picture. Although I did not have as many conversations with MICHEL DEVORET as others, each time I did, I walked away with new understanding. It was his insistence on careful precision that disabused me of many mistaken beliefs. I am grateful to LIANG JIANG for many fruitful discussions. From this experimentalist's perspective, Liang is the perfect theorist collaborator. He understands the state of the art well enough to be able to pinpoint immediately those ideas which are exactly the right combination of interesting yet achievable.

After the faculty on my committee, I am most indebted to my postdoc collaborators with whom I had the pleasure to work side by side, facing the joys and hardships of constructing experiments and interpreting data. I can thank REINIER HEERES in particular for saving my scientific career at a time when I was struggling by giving me a direction and a project that worked well with my skill set. Working with Reinier taught me how qubits worked in practice, and how to think about designing experiments to probe their properties. His encouragement and enthusiasm took the FPGA software project from a hidden corner on my hard drive to tens of projects throughout the lab. I have had the pleasure of working very closely with SERGE

ROSENBLUM for the past few years, and I believe it has been one of the most productive parnterships I have ever had. Serge, a true scientist, is ceaseless in his drive to find intuitive explanations. Serge understands the importance of making our work accessible, and working with him has made me a better communicator.

I am thankful to everybody on the fourth-floor who I've had an opportunity to work or discuss science with over the past six years. I'd like to thank those members of the previous generation welcomed me to the lab and brought me onboard. JACOB BLUMOFF was the first person I met arriving at Yale. Jacob, along with KEVIN CHOU, taught me the essential tools of the trade, and were my first line of defense against confusion. The lab would have descended into total chaos if it was not for the watchful eyes of CHRIS AXLINE and ANIRUDH NARLA, who, in addition to being skilled physicists, made sure that negligence was never allowed to accumulate. Alongside them I must also thank LUIGI FRUNZIO, whose influence kept my fabrication experience as pain free as any graduate student could hope. NISSIM OFEK underwent a heroic effort in both architecting and implementing the FPGA controller. It's not a task that any one person could be expected to do, but by sheer force of will he succeeded, and created something wonderful.

I'd like to thank rest of those in my cohort who have shared with me in the PhD experience. EVAN ZALYS-GELLER is a nearly infinite font of spontaneously generating, wildly creative ideas, whose discussions taught me more engineering concepts than I knew existed. KYLE SERNIAK has been the social core of the lab in my time here, and his friendliness belies his deep understanding of fabrication and superconductivity. I've particularly enjoyed my interactions with CHRIS WANG and SAL ELDER. Chris's ability to juggle many responsibilities has kept the lab running smoothly in recent years, and Sal's probing questions have deepened my understanding of physics I had never thought to question. I've also learned a great deal about quantum circuits in their many forms from discussions with NICK FRATTINI. My experience would not have been the same without those who joined me at the lunch carts nearly daily, in particular CHAN U LEI, VIJAY JAIN, LEV KRAYZMAN, JACOB CURTIS, SUHAS GANJAM and ZHIXIN WANG with whom I've had an enormous volume of wide-ranging conversations.

I would be remiss to omit mention of DAVID SCHUSTER, who started me down this path, mentoring me as an undergraduate, and inspiring me to pursue quantum information studies. Dave has a way of presenting ambitious projects as imminently achievable, and he is one of the most creative scientists I know.

Finally I'd like to thank my family and friends for encouraging me through these many years of uncertainty. My mother and father, CHRISTY REYNOLDS and LARRY REINHOLD, as well as my partner PAMELA SOTO, have held steadfast their confidence in my ability, through my periods of self-doubt. Without their support this thesis would not have been possible.

Chapter 1

Introduction

The "second quantum revolution" which has been progressing over the past three decades is characterized by a transformation in the way we view quantum theory. From our initial perspective, quantum mechanics seemed largely to be a nuisance; very important, if one wished to get things right, but still a *nag*. It seemed to place limitations on what ham-fisted creatures as ourselves could know about the world at once. It told us that we could never observe the world as neutral outsiders, but that we necessarily influence the world by our act of observation. It claimed that the world was not even definite, until experiment and observation forced its hand. At the same time, our ability to manipulate and experience these unusual effects seemed confined to the realm of subatomic particles, only accessible via the careful interpretation of laboratory measurements.

Nevertheless, using the understanding of the rules of quantum physics has led us to the development countless technologies—the laser, semiconductors, medical imaging, and many others—leading to real material improvements in our control over the natural world and in human well-being. However, in all of these cases, the quantum behavior is hidden by the coarse-graining of many subsystems into a single ensemble. Only recently have we gained access to the realm of single quantum systems which we can manipulate and measure individually, as opposed to in collective ensembles.

This development has allowed our perspective on quantum mechanics to shift. The promise of quantum technology, is that we are blessed, rather than cursed, to live in a world that is fundamentally quantum, if only we can begin *harnessing* and *engineering* the uniquely quantum mechanical effects nature exhibits. We can use the knowledge limitations quantum mechanics provides to implement unbreakable encryption protocols. We can engineer quantum states whose sensitivity to disturbance allows us to learn and measure the world faster and more accurately than would be otherwise possible. And most tempting of all is the promise of a quantum computer. That we could take the most crucial, revolutionary, and transformative invention in the entire twentieth century, and radically improve its capabilities goes a long way in explaining why this application has held such a central position in the dreams of quantum technology prognosticators.

The story of quantum computing began with exactly the sort of perspective reversal that characterizes the second quantum revolution, following an observation by Feynman: quantum mechanics is *hard* to simulate on a computer. As the system being simulated gets bigger, the system of equations that must be solved gets *exponentially* larger. People who have thought very carefully about this do not see any way to remove this exponential scaling from the problem. From one perspective this just seems like a bad thing, a limit on our ability to predict and calculate the world around us. The counterpoint, however, is that we *can* know how quantum systems behave, since they exist in reality! We simply must construct a quantum system, and then observe it. It seems like nature is doing this exponentially difficult calculation somewhere, and we simply lack the tools to tap into, and control, this natural computational power. This led early researchers to consider what a computer constructed out of quantum mechanical parts would look like, and what its capabilities would be. Such a device could certainly solve the problem of quantum simulation, but the discovery of Shor's algorithm for integer factoring (Shor, 1994) proved that the utility of such a device would extend beyond simulation. Since then a torrent of additional quantum algorithms have been proposed, extending the practical reach of quantum computation beyond breaking the security of widely used cryptosystems.

The theoretical promise of quantum computing was vast and tantalizing. It was not at all obvious, however, that such a device would be possible to construct. Classical computers are *robust* by design. If you take a modern computer, you can expose it to all sorts of radiation, heat, physical motion, magnetic fields, etc. and within a reasonable margin the computation performed will not be affected. In contrast, *any* interaction between a quantum computer and its environment will affect its state[1]. Perhaps this effect would be small, but even small errors have the potential to accumulate and destroy the computation. In some sense, the problem is related to the special status of measurement in quantum theory. Classically, measurement is exactly the type of interaction that (at least potentially) does not affect the state of the measured system. Quantum mechanically, measurement is not a harmless act. Our inability to measure without affecting the measured object inhibits our ability to construct robust quantum systems. It seemed, then, that the task of quantum computing rests on two individually impossible requirements, which had the further complication of being mutually incompatible: perfect isolation of the quantum system from its environment and perfect control and manipulation of the quantum system (Unruh, 1995). Quantum computing seemed to be ruled out by the same issues that plagued analog computing, a platform that could theoretically outperform digital Turing machines, but in practice could not overcome the limitations of real-world noise and imperfections.

The subsequent development of quantum error correction (Shor, 1995) and fault-tolerant quantum computation (DiVincenzo and Shor, 1996; Preskill, 1997) responded to those criticisms. These methods showed that, under certain reasonable sounding assumptions, quantum computing in the presence of noise was not physically impossible, but merely staggeringly difficult. If one could reduce the noise per operation below a certain threshold value then one could aggregate many noisy, imperfect quantum systems together in order to create a less error-prone quantum system. If one could repeat this process, aggregating together many aggregates, then the combined system could have even lower error, and so on. As a result, arbitrarily good performance could be obtained despite the presence of noise. While there are many ways to achieve the same result, early proposals for devices that could run Shor's method to factor a 430 bit number would require resources equivalent to 10^6 physical qubits, each operating with an error rate per operation of 10^{-6} (Steane, 1998). At this time, the most advanced platform for quantum computing was liquid state nuclear magnetic resonance (NMR), where the weakly coupled nuclear spin degree of freedom formed a naturally (relatively) isolated quantum bit,

[1] at least any interaction which couples to a degree of freedom used in the computation

and one could perform 3 qubit algorithms, with gate errors around 5% (Cory et al., 1998). Either experiment had a long way to go to meet the grandiose demands of the theory, or theory had a long way to go to come up with a proposal that could be implemented in practice.

Since then, many developments have brought us closer to the dream of a useful quantum computer. A proliferation of physical implementations has been proposed and developed, from linear optics (Knill et al., 2001), to spins trapped by crystal defects or within quantum dots (Loss and DiVincenzo, 1998), to ions suspended in vacuum (Steane, 1997), to superconducting circuits (Devoret, 1997). Each approach has strengths and weaknesses, and none has clearly and convincingly set itself apart as uniquely capable of addressing all of the challenges involved in creating a scalable quantum machine. Alternate error correction architectures have been developed that have raised the threshold error rate, and introduced models where physical qubits need only local interactions. But practical error correction, which *actually enhances* a quantum machine's computation ability, is still a work in progress. Quantum machines are getting bigger, but still too slowly in order to hope for an error corrected quantum computer soon. For this reason, there has been much recent interest in the capabilities of a noisy intermediate scale quantum (NISQ) machine (Preskill, 2018). Such a machine, which lacks error correction, may still be able to perform useful tasks that a classical computer could not. But this is far from certain, and while such machines are a necessary stepping stone, they will not be transformative in the same way as fully fault-tolerant general purpose computers.

We can look for avenues for progress towards that ultimate goal in two different ways, each of which is going to be necessary in the long term. We can make slow and steady progress improving the construction and operation of the types of devices we already possess: improving the materials we use, reducing spurious interactions, finding faster ways to complete operations, and the like. Such difficult, and at times tedious, engineering is the backbone of our conviction that we will one day reach the summit. But we can make progress also by looking for shortcuts, new ways to arrange or join the tools we have, to circumvent, rather than surmount, some of the obstacles we see in our way. This sort of lateral thinking is best represented by the cat-code method of quantum error correction (section 3.6) which allows us to simplify and reduce the hardware overhead requirements of traditional quantum error correction. These sorts of reimaginings of what a quantum information processor might look like are going to be critical for keeping the field alive and invigorated as we continue to slog through the difficult engineering tasks ahead of us.

1.1 Outline of thesis

In this work, we take as a given the motivation to pursue quantum information processing devices, and the background that entails (see section 1.2). We will begin straight away in chapter 2 with an overview of the central problem of this thesis: quantum control of the state of light contained within a cavity resonator. These devices are ubiquitous, easy to fabricate, and can be highly coherent when compared with other superconducting elements. Yet they face difficulties in the crucial aspect of control. In isolation, these systems are cursed by linearity, which renders their outputs trivially related to their inputs, and therefore unsuitable for computation, as they can be simulated with complexity polynomial in their size. However, by using these systems in conjunction with non-linear elements, we can control them, and expose their potential quantum information processing capacity.

One of the promising opportunities a quantum controlled cavity presents is the possibility to do quantum error correction. In chapter 3, we will see how it is possible to do error correction with only a single component, and introduce the setting and motivation for the coming experimental work. Our first experimental goal will be to perform operations on encoded states in a cavity. In order to get there, we will first set up some necessary background. First in chapter 4 we return to the issue of control, this time with an eye towards how control protocols for quantum systems can be developed with the aid of numerical optimization methods running *in silico*. These methods have a storied past, finding application in many systems, surprising many with the ease in which carefully designed protocols can be replaced with a naïve search. Next, in chapter 5, we discuss the hardware with which we can perform experiments on superconducting cavities interacting with transmon qubits. This chapter serves as an overview of the state of the art in superconducting cavities, and tries to motivate some of the design choices that were made in constructing the experimental samples used.

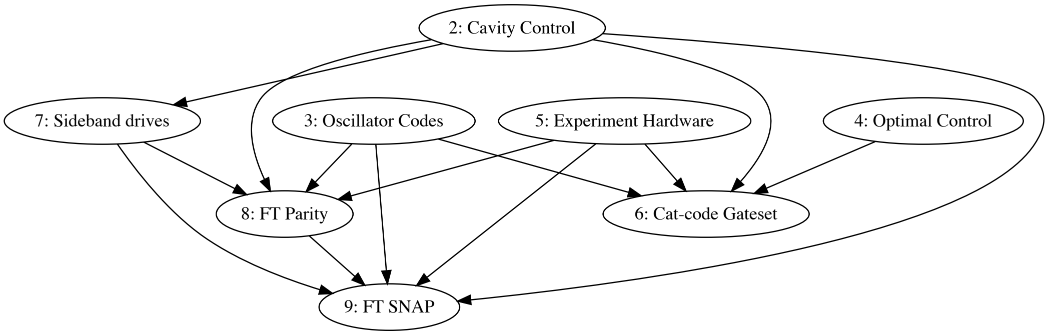


Figure 1.1: **Chapter dependency graph.** Chapters 6, 8, and 9 contain the primary experimental results. Arrows indicate which chapters are suggested prerequisites for each other.

With the prerequisites of experimental design out of the way, we are free to move on in chapter 6 to the first main result, which is the application of numerically optimized control sequences to the subject of cavity control, and more specifically control of a cat encoded qubit. This chapter will cover the results shown in "Implementing a universal gate set on a logical qubit encoded in an oscillator" (Heeres et al., 2017).

In order to introduce the final two experiments, it is worth covering the subject of sideband transitions more carefully (chapter 7). Understanding this class of operations, which extends the picture associated with the dispersive coupling model discussed in chapter 2, is crucial for understanding nearly all of the results in the Yale quantum computing groups in the past few years. Finally, we get to the fault-tolerance results, which try to address the main limitation of the cavity control method shown in chapter 6. Specifically, we address the issue of transmon decoherence events being transformed into cavity decoherence events by the action of the coupling. This phenomenon, which allows the types and frequencies of errors to be dictated by the relatively noisy transmon, severely limits the usefulness of cat-code error correction. The interaction that mediates this error conduction is a fundamental, and necessary component of our method of manipulating the cavity state. In order to circumvent this problem, we develop a surgical modification of the interaction Hamiltonian between the transmon and cavity using detuned sideband drives, which is designed to prevent the spreading of errors between the components. This allows us, in chapter 8, to develop a "fault-tolerant" parity measurement, including the results shown in “Fault-tolerant detection of a quantum error” (Rosenblum et al., 2018b). Using the same general set of tools, and with a bit more sophistication, we can extend these results to a (non-universal) set of cavity operations (Reinhold et al., in preparation) in chapter 9.

1.2 Background and suggested reading

This thesis is not intended to be either comprehensive or self-contained. It leans heavily on the work of others who have come before me. Instead of duplicating this previous effort here, we will instead assume a level of familiarity with foundational concepts. Here we would like to make these assumptions explicit, and give pedagogical references which explain (better than I could) the necessary background.

We assume familiarity with quantum mechanics firstly, of course, for which the standard texts Shankar (2011) and Griffiths (2004) I have found more than adequate. One should be comfortable with the concepts of Hilbert space, projective quantum measurement, the Schrödinger equation, Heisenberg and interaction pictures. For considerations involving open systems, including density matrices, POVMs, the Lindblad equation, and superoperators, Carmichael (1993) is helpful, as are the (quite comprehensive) notes by Steck (2007).

Secondly, we will not spend any more pages motivating or explaining basic concepts of quantum information and computation, such as defining qubits, quantum gates, or quantum algorithms. For this “Mike and Ike” (Nielsen and Chuang, 2011) is required reading. In addition, there are many high-quality treatments of quantum information to be found in the theses of previous Yale students (Chow, 2010; Reed, 2013; Blumoff, 2017; Chou, 2018).

Some familiarity with very basic quantum optics concepts is also assumed, including the concept of raising and lowering operators as well as that of Fock (photon-number) states. While the level of knowledge required here does not extend much beyond what can be found in Griffiths (2004), it can be supplemented with the treatments found in Walls and Milburn (1995) and Scully and Zubairy (1997). A more focused discussion relevant to the work presented here can be found in Vlastakis (2015).

Finally, we will not give a pedagogical explanation of the foundational concepts of Cavity QED (CQED) and its superconducting cousin, Circuit QED (cQED). The fantastic Haroche and Raimond (2006) is a complete treatment of the former. The latter was first exposited in Schuster (2007) and has received treatments in many other theses (Bishop, 2010; Chow, 2010; Reed, 2013; Chou, 2018; Blumoff, 2017; Brecht, 2017; Axline, 2018).

Chapter 2

Controlling cavities

Harmonic oscillators are one of the first entities that one encounters when learning about quantum mechanics. Their quantum mechanical description demonstrates the effects of quantization on a system which is ubiquitous in all sorts of physics. It gives rise to a simple energy level structure, yet the states described by this structure bear little resemblance to what is classically familiar. It is a fertile ground for exploring the correspondence principle linking the classical and quantum worlds. From the standard introduction to quantum oscillators, however, it is not clear how one might perform experiments which explore many of the quantum mechanical aspects of such a system, beyond the uncertainty relation. For instance, if quantum mechanics predicts a certain wavefunction for an energy eigenstate, how can one prepare such a state, and measure that wavefunction? How can we measure properties other than position and momentum? How can we encode quantum information?

In this chapter, I will explore the methods and limitations of control in simple harmonic oscillator systems, of which electromagnetic states in microwave cavities form a subset.[1] I start in section 2.1 by addressing the issue of linearity, defining linear systems in terms of their representation using ladder operators, and showing the limited scope of control in this regime. Next, in section 2.2 I will discuss how this control can be expanded by a controllable resonant coupling to a two-level system, in the Jaynes-Cummings model. The non-linearity associated

[1] We can see the direct analogy between quantum mechanical oscillators and cavity modes via the quantization of the electromagnetic field (Steck, 2007, §8.3).

with the two-level system is the resource used to prepare new types of cavity states. From there, in section 2.3, we consider the off-resonant, or dispersively coupled regime. Here the tools at our disposal are subtler, and require some cleverness in their assembly to produce useful operations, but I give some examples of operations which can be constructed from a "dispersive toolkit." Next in section 2.4 we consider how to formalize the notion of universal control, that is, the idea of a general process or recipe, which takes an arbitrary quantum-mechanically allowed transformation of the cavity, and produces a means for realizing that transformation with available controls. The "SNAP protocol" allows for this by giving a sort of template pulse sequence, constructed out of the dispersive toolkit, for which every cavity transformation is approximated by filling out this template with the correct numbers and parameters. This reduces the problem of realizing control to one of searching within a limited parameter space. Finally, in section 2.5 we will see that this "template" formulation, while theoretically useful, imposes a large overhead, which can be alleviated by considering arbitrary control pulses, represented as general functions of time. The techniques necessary to find these control pulses will be explored in chapter 4. This is enough to get us through the first set of experimental results. We will revisit the subject of cavity control using more exotic sideband drives, whose frequencies are far from any mode's resonance, in chapter 7.

2.1 The limits of control in linear cavities

A superconducting cavity resonator is a simple object to describe: it consists of a box, whose walls define electromagnetic boundary conditions, which give rise to a set of non-interacting bosonic modes. Each of these modes has a resonance frequency, and has a simple harmonic oscillator Hamiltonian (setting $\hbar = 1$ from now on):

$$\boldsymbol{H} = \sum_k \omega_k \boldsymbol{a}_k^\dagger \boldsymbol{a}_k. \tag{2.1}$$

If the resonator is coupled to the outside world, for instance with pins connected to a drive line, we can often treat the incident fields classically ($\varepsilon(t)$), and find the driven Hamiltonian:

$$\boldsymbol{H} = \sum_k \omega_k \boldsymbol{a}_k^\dagger \boldsymbol{a}_k + \varepsilon(t)\Omega_k \left(\boldsymbol{a}_k + \boldsymbol{a}_k^\dagger\right) \tag{2.2}$$

We will now show that the evolution of such a system is “trivial” in the sense that it can only produce a displacement of the modes. Restricting ourselves to a single mode (with frequency ω) for now, but without loss of generality, we first go into a rotating frame to remove the detuning term $\omega \boldsymbol{a}^\dagger \boldsymbol{a}$. Frame changes are performed by first specifying a unitary operation $\boldsymbol{U}(t)$ and making the following replacements (See appendix A.1)

$$|\psi\rangle \to |\tilde{\psi}\rangle = \boldsymbol{U}\,|\psi\rangle \tag{2.3}$$

$$\boldsymbol{H} \to \widetilde{\boldsymbol{H}} = \boldsymbol{U}\boldsymbol{H}\boldsymbol{U}^\dagger - i\boldsymbol{U}\dot{\boldsymbol{U}}^\dagger \tag{2.4}$$

In this case, we have $\boldsymbol{U} = \exp\left(-i\omega \boldsymbol{a}^\dagger \boldsymbol{a} t\right)$. For this transformation, we will frequently make use of the relation (see appendix A.3)

$$\boldsymbol{U} f(\boldsymbol{a}, \boldsymbol{a}^\dagger)\boldsymbol{U}^\dagger = f(\boldsymbol{a}e^{i\omega t}, \boldsymbol{a}^\dagger e^{-i\omega t}) \tag{2.5}$$

Applying equation 2.4 and 2.5 to the single-mode version of equation 2.2 results in

$$\boldsymbol{H} \to \widetilde{\boldsymbol{H}} = \varepsilon(t)\left(\boldsymbol{a}e^{i\omega t} + \text{h.c.}\right) \tag{2.6}$$

We can “solve” $\widetilde{\boldsymbol{H}}$ using the Magnus expansion (Magnus, 1954). The evolution of the system in the rotating frame is given by the unitary:

$$\widetilde{\boldsymbol{U}}(t_0, t) = \mathcal{T}\exp\left(-i\int_{t_0}^{t} \mathrm{d}\tau\, \boldsymbol{H}(\tau)\right) = \exp\left\{\sum_k \boldsymbol{\Omega}_k\right\} \tag{2.7}$$

The terms in the expansion ($\boldsymbol{\Omega}_k$) are formed from increasingly nested commutators. Here, setting $\boldsymbol{H}_i \equiv \widetilde{\boldsymbol{H}}(\tau_i)$:

$$\begin{aligned}
\boldsymbol{\Omega}_1 &= -i \int_{t_0}^{t} \mathrm{d}\tau_1 \, \boldsymbol{H}_1 \\
\boldsymbol{\Omega}_2 &= \frac{-1}{2} \int_{t_0}^{t} \mathrm{d}\tau_1 \int_{t_0}^{\tau_1} \mathrm{d}\tau_2 \, [\boldsymbol{H}_1, \boldsymbol{H}_2] \\
\boldsymbol{\Omega}_3 &= \frac{i}{6} \int_{t_0}^{t} \mathrm{d}\tau_1 \int_{t_0}^{\tau_1} \mathrm{d}\tau_2 \int_{t_0}^{\tau_2} \mathrm{d}\tau_3 \, [\boldsymbol{H}_1, [\boldsymbol{H}_2, \boldsymbol{H}_3]] + [\boldsymbol{H}_3, [\boldsymbol{H}_2, \boldsymbol{H}_1]] \\
&\vdots
\end{aligned}$$

Since $\left[\widetilde{\boldsymbol{H}}(\tau_1), \widetilde{\boldsymbol{H}}(\tau_2)\right]$ is a complex scalar quantity (as follows from $\left[\boldsymbol{a}, \boldsymbol{a}^\dagger\right] = 1$),the Magnus series terminates after the second level. The second level itself gives only a global phase which we will ignore for now, and thus the evolution is given by

$$\boldsymbol{U}(t_0, t) = \exp\left(-i \int_{t_0}^{t} \mathrm{d}\tau \, \boldsymbol{H}(\tau)\right) \tag{2.8}$$

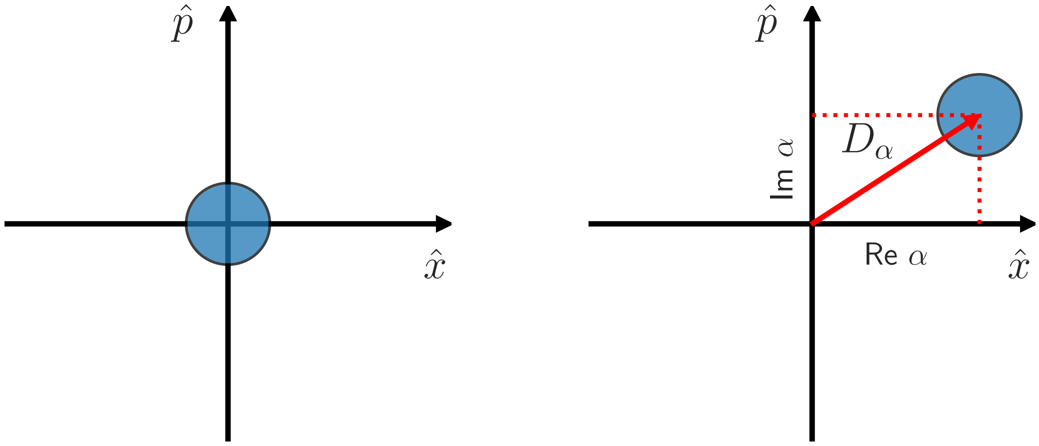
$$= \exp\left(\alpha \boldsymbol{a}^\dagger - \alpha^* \boldsymbol{a}\right) \equiv \boldsymbol{D}_\alpha \tag{2.9}$$

Where $\alpha = \int_{t_0}^{t} \mathrm{d}\tau \, \varepsilon(t) e^{i\omega t}$. The displacement, $\boldsymbol{D}_\alpha$, is one of the fundamental concepts in quantum optics. Displacements acting on the vacuum produce coherent states

$$|\alpha\rangle = \boldsymbol{D}_\alpha |0\rangle = e^{-|\alpha|^2/2} \sum_n \frac{\alpha^n}{\sqrt{n!}} |n\rangle \tag{2.10}$$

Coherent states can be thought of as “nearly classical” states. They have (expected values) of both position and momentum[2] ($\mathrm{Re}(\alpha)$ and $\mathrm{Im}(\alpha)$ respectively) while also maintaining the necessary (but minimal) requisite uncertainty in these properties required for consistency with Heisenberg uncertainty. However, because all evolution is reducible to displacements, the only states which can be produced are coherent states. The system described in equation 2.2 does not allow for the production of non-classical states of light, e.g. photon-number states. It can be shown (see Lloyd and Braunstein (1999), or appendix B) that a Hamiltonian must have

[2] These quantities correspond to position and momentum (in some units) for a mechanical oscillator. For an electromagnetic oscillator these correspond to electric and magnetic field (in some units). See section 5.1 for more discussion.



(a) The ground state in phase space

(b) A displaced state

Figure 2.1: **Displacement in phase space**. We can picture the state of the cavity in *phase space* whose coordinates can be thought of as "position" and "momentum," in analogy with a mechanical oscillator, and with some appropriate scaling factors. **(a)**. In the ground state, the cavity state is centered on the origin, but is spread out over an area as required by Heisenberg uncertainty. **(b)** A displaced state $|\alpha\rangle$ has the same spread in phase space distribution, but is centered about a new point with coordinates given by the real and imaginary parts of α. This state is produced from the ground state by the displacement operator $\boldsymbol{D}_\alpha$

terms of the form $(\boldsymbol{a}^\dagger)^M \boldsymbol{a}^N$ with $M + N \geq 3$ in order to be able to create arbitrary states or operations. While such a term would be useful, it is not at all obvious how to create such non-linearities directly without disrupting many of the properties of the cavity which we wish to exploit, such as its high coherence, and simple idling evolution.

2.2 Control in the Jaynes-Cummings model

An alternative to directly controlling the cavity with internal non-linearity is to couple the cavity to an external system with its own non-linearity. Any finite dimensional system possesses the required non-linearity, so a natural model to consider is the Jaynes-Cummings model of an oscillator coupled to a two-level qubit (Jaynes and Cummings, 1962):

$$\boldsymbol{H} = \omega_c \boldsymbol{a}^\dagger \boldsymbol{a} + \frac{\omega_q}{2}\boldsymbol{\sigma}_z + g\left(\boldsymbol{a}^\dagger \boldsymbol{\sigma}_- + \boldsymbol{a}\boldsymbol{\sigma}_+\right). \tag{2.11}$$

This model is the foundation of the field of cavity quantum electrodynamics (CQED) and by analogy the superconducting circuit equivalent, circuit quantum electrodynamics (cQED). It is natural to again go into the rotating frame using equation 2.4 as before, with the transformations $\boldsymbol{U}_1 = \exp\left(i\omega_c \boldsymbol{a}^\dagger \boldsymbol{a} t\right)$ followed by $\boldsymbol{U}_2 = \exp\left(i\omega_q \boldsymbol{\sigma}_z t/2\right)$. This results in

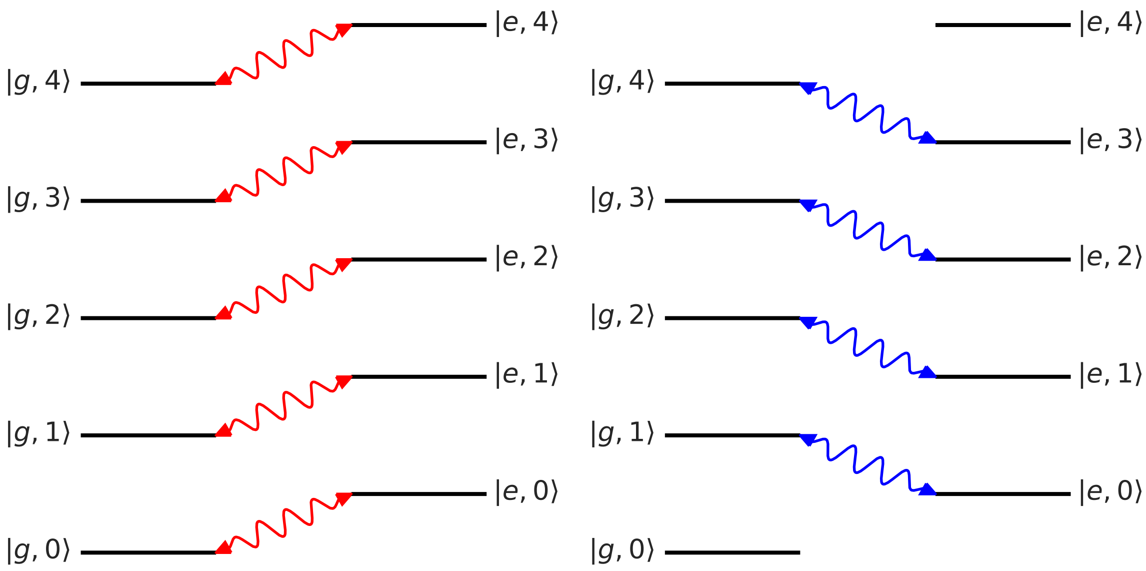
$$\widetilde{\boldsymbol{H}} = g\left(\boldsymbol{a}^\dagger \boldsymbol{\sigma}_- e^{-i\Delta t} + \text{h.c.}\right), \tag{2.12}$$

where $\Delta \equiv \omega_c - \omega_q$.

Let us see how, assuming control of the qubit frequency ω_q, such an interaction can be used to create photon number states in the cavity. First note that it is possible to create "photon number states" in the qubit, when $|\Delta| = |\omega_c - \omega_q|$ is large, simply by applying a drive. Since the form of the drive is $\boldsymbol{H}_d = \frac{\Omega(t)}{2}\boldsymbol{\sigma}_x$, applying any pulse with the appropriate area ($\int \mathrm{d}t\, \Omega(t) = \pi$) results in the unitary operation $\boldsymbol{\sigma}_x$, which produces an exact excited state in the qubit. One can use the qubit's ability to inject single excitations, along with the excitation-conserving Jaynes-Cummings interaction (equation 2.12) in order to produce photon number states in the cavity, as shown in figure 2.2. This method was used to create some of the first non-classical states in cavity resonators, such as Fock states (Krause et al., 1989) and cat states (Meystre et al., 1990). It was then shown that it was theoretically possible to construct control sequences which create *arbitrary* cavity states (Vogel et al., 1993; Law and Eberly, 1996). These protocols have been experimentally demonstrated using superconducting cavities and flux-tunable transmon qubits (Hofheinz et al., 2009).

2.3 Control in the dispersive regime

There are several reasons why one might prefer to avoid relying upon frequency tuning, as is required in order to toggle the interaction in equation 2.12 in order to perform operations. To begin with, such control will always increase the complexity of implementation of the device, but more worrying is the impact such a control mechanism has on the qubit coherence properties, as the qubit frequency becomes sensitive to fluctuations in the controlling field, and thus dephases at a faster rate. In the case of fixed frequency, and large detuning ($\Delta \gg g$) we can approximate the Jaynes-Cummings model with a dispersive model. Treating the interaction term as a perturbation, and going to second order in perturbation theory, we end up with the



(a) exciting the qubit

(b) swapping the qubit and cavity

Figure 2.2: **Photon number creation using resonant Jaynes-Cummings interactions**. One can see how to prepare photon number states in the Jaynes-Cummings model by "climbing the ladder." There are two steps involved. **(a)** One can insert a single excitation into the system by driving the qubit. Its non-linearity assures that exactly one excitation is added. **(b)** The excitation is swapped from the qubit to the cavity by bringing the qubit into resonance with the cavity for a fixed amount of time. If these two steps are alternated, one can bring the cavity to higher and higher photon number levels (Law and Eberly, 1996; Hofheinz et al., 2009).

new model

$$\boldsymbol{H} = \omega_c \boldsymbol{a}^\dagger \boldsymbol{a} + \frac{\omega_q}{2}\boldsymbol{\sigma}_z + \frac{\chi}{2}\boldsymbol{a}^\dagger \boldsymbol{a}\boldsymbol{\sigma}_z, \tag{2.13}$$

with $\chi = 2g^2/\Delta$. This is the correct form only for an actual two-level system. For a multi-level system like a transmon (section 5.2) this is only an approximation valid when the detuning to the ge transition is small compared with the detuning to other transitions. See appendix A.5 for a general treatment. In our preferred rotating frame equation 2.13 becomes simply

$$\boldsymbol{H} = \chi \boldsymbol{a}^\dagger \boldsymbol{a} \left|e\right\rangle\!\left\langle e\right| \tag{2.14}$$

where $|g\rangle$ and $|e\rangle$ are the ground and excited states of the qubit. There are two complementary pictures we can adopt when thinking about this Hamiltonian: Either there is a χ frequency shift of the cavity when the qubit is in the excited state, or there is a χ frequency shift of the qubit per photon in the cavity. When we attempt to control this system, we again introduce drives:

$$\boldsymbol{H} = \chi \boldsymbol{a}^\dagger \boldsymbol{a} \, |e\rangle\langle e| + (\Omega(t)\boldsymbol{\sigma}_- + \varepsilon(t)\boldsymbol{a} + \text{h.c.}) \tag{2.15}$$

There are some "obvious" ways we can exploit this Hamiltonian to perform operations, which taken together, form a "toolkit" for manipulating cQED systems (Vlastakis, 2015). The five most important members of this toolkit are:

1. unselective cavity displacements: $\boldsymbol{D}_\alpha$
2. unselective qubit rotations about axis $\cos(\phi)\boldsymbol{\sigma}_x + \sin(\phi)\boldsymbol{\sigma}_y$: $\boldsymbol{R}_\phi(\theta)$
3. entangling conditional phase: $\boldsymbol{C}_\phi = \exp\left(i\phi \boldsymbol{a}^\dagger \boldsymbol{a} \, |e\rangle\langle e|\right)$
4. selective cavity displacements $\boldsymbol{D}_\alpha \, |g\rangle\langle g| + \boldsymbol{I}_c \, |e\rangle\langle e|$
5. selective qubit rotations: $|0\rangle\langle 0| \, \boldsymbol{R}_\phi(\theta) + (\boldsymbol{I}_c - |0\rangle\langle 0|) \, \boldsymbol{I}_q$

To obtain the first two, we note that, given a large enough driving field ($\Omega \gg \chi$ or $\varepsilon \gg \chi$), we can simply ignore the effect of the interaction and produce the unselective variants of these operations. The entangling phase can be prodcued by simply waiting, i.e. $\Omega = \varepsilon = 0$. If there is a superposition state in both the qubit and the cavity[3], the two systems will become entangled. In general we can make gates of the form $\boldsymbol{C}_\phi = \exp\left(i\phi \boldsymbol{a}^\dagger \boldsymbol{a} \, |e\rangle \langle e|\right)$ by evolving under the Hamiltonian 2.14 for a time $t = \phi/\chi$. For instance, over a time $t = \pi/\chi$, we can use $e^{i\phi \boldsymbol{a}^\dagger \boldsymbol{a}} \, |\alpha\rangle = \left|e^{i\phi}\alpha\right\rangle$ to show

$$(|g\rangle + |e\rangle) \, |\alpha\rangle \to |g\rangle \, |\alpha\rangle + |e\rangle \, |{-\alpha}\rangle \tag{2.16}$$

We can obtain the final operations by re-introducing a drive term, but now weakly. If a drive pulse is sufficiently long, and hence its bandwidth is sufficiently narrow (BW $\ll \chi$) then the

[3] A displaced state $|\alpha\rangle$ will do, since it is a superposition of photon number states

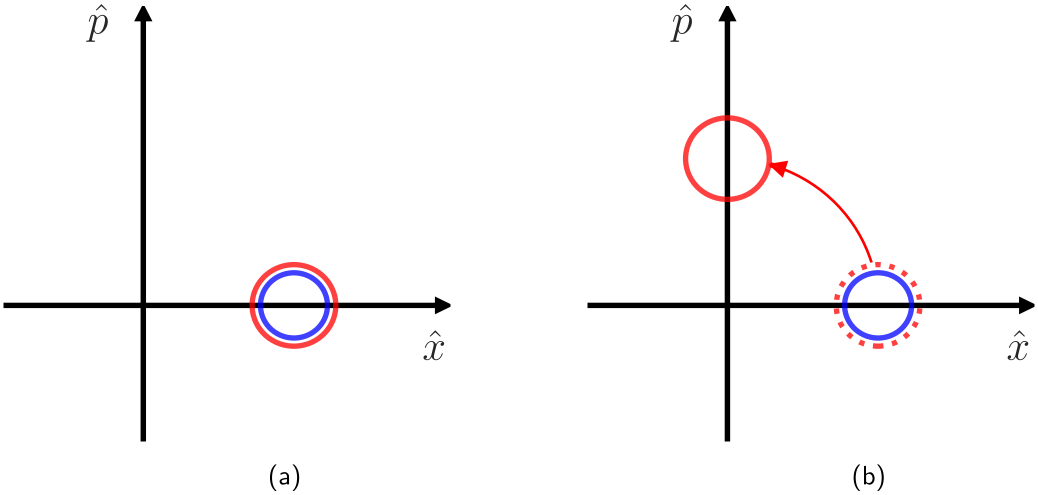


Figure 2.3: **Phase-space rotation conditional on qubit state**. We can represent the state of an oscillator entangled with a qubit by showing the oscillator state conditioned on a given qubit state. Here we show the oscillator state conditioned on $|g\rangle$ as a blue outline in phase space, while the state conditioned on $|e\rangle$ is red. (a) shows an unentangled coherent state, $|g, \alpha\rangle + |e, \alpha\rangle$. (b) shows how, under the action of the $\chi \boldsymbol{a}^\dagger \boldsymbol{a} \, |e\rangle\langle e|$ interaction, the cavity becomes entangled with the qubit by rotating only the $|e\rangle$ picture with respect to the $|g\rangle$ picture.

drive can be made *selective* with respect to the state of the undriven system. For instance, a narrow drive, centered around zero-frequency in this rotating frame, applied to the transmon, will induce Rabi oscillations if and only if the cavity contains zero photons. This is a photon-number selective qubit drive. By detuning this drive by a frequency $n\chi$, for some integer n, this drive can become selective on any number of photons, resulting in the operation:

$$\boldsymbol{R}_\phi^{(n)}(\theta) = |n\rangle\langle n| \, \boldsymbol{R}_\phi(\theta) + (\boldsymbol{I}_c - |n\rangle\langle n|) \, \boldsymbol{I}_q. \tag{2.17}$$

Conversely, a drive applied to the cavity with such a narrow bandwidth will induce a displacement of the cavity, if and only if the qubit is in the ground state.

2.3.1 Applications: Parity map

One of the most useful applications of this toolkit for our purposes is the photon number parity mapping operation. Photon number parity is a property of the cavity which is defined to be $+1$ when the number of photons is even, and -1 when the number of photons is odd. We can represent this as an observable operator ($\boldsymbol{\Pi}$) using regular ladder operators:

$$\boldsymbol{\Pi} \equiv e^{i\pi \boldsymbol{a}^\dagger \boldsymbol{a}} = (-1)^{\boldsymbol{a}^\dagger \boldsymbol{a}} = \boldsymbol{P}_{\text{even}} - \boldsymbol{P}_{\text{odd}}. \tag{2.18}$$

Here we have defined the projectors onto the even and odd photon number parity subspaces:

$$\boldsymbol{P}_{\text{even}} = \sum_{k\,\text{even}} |k\rangle\langle k| \qquad \boldsymbol{P}_{\text{odd}} = \sum_{k\,\text{odd}} |k\rangle\langle k|$$

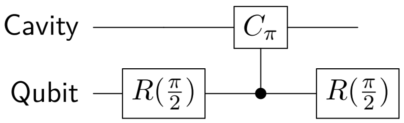
The parity map operation copies the bit of information corresponding to the photon number parity from to the state of the qubit. The linchpin of this operation is the entangling conditional phase operation. We can see how they are related

$$\begin{aligned} \boldsymbol{C}_\pi &= \boldsymbol{I}_c\,|g\rangle\langle g| + \boldsymbol{\Pi}\,|e\rangle\langle e| && (2.19)\\ &= (\boldsymbol{P}_{\text{even}} + \boldsymbol{P}_{\text{odd}})\,|g\rangle\langle g| + (\boldsymbol{P}_{\text{even}} - \boldsymbol{P}_{\text{odd}})\,|e\rangle\langle e| && (2.20)\\ &= \boldsymbol{P}_{\text{even}}\boldsymbol{I}_q + \boldsymbol{P}_{\text{odd}}\boldsymbol{\sigma}_z && (2.21) \end{aligned}$$

The entangling conditional phase, with an angle of π, flips the *phase* of the qubit if and only if there are an odd number of photons (see figure 2.4). In order to make this into a parity map, we need to flip the (computational basis) state ($\boldsymbol{\sigma}_x$ or $\boldsymbol{\sigma}_y$) rather than the phase ($\boldsymbol{\sigma}_z$). This can be achieved by sandwiching the entangling operation by qubit $\pi/2$ rotations

$$\boldsymbol{R}(\pi/2)\boldsymbol{C}_\pi\boldsymbol{R}(-\pi/2) = \boldsymbol{P}_{\text{even}}\boldsymbol{I}_q + \boldsymbol{P}_{\text{odd}}\boldsymbol{\sigma}_y \tag{2.22}$$

We can represent this operation in a more traditional circuit diagram form, like so:



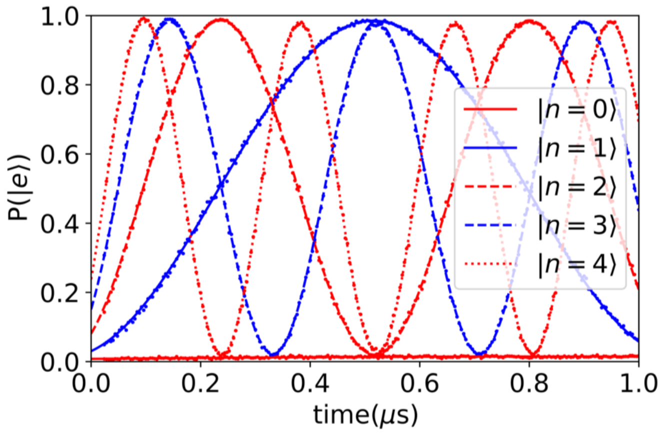


Figure 2.4: **Experimental plot of qubit Ramsey spectroscopy vs. photon number.** At $t \approx 0.53\,\mu$s the even and odd peaks coalesce on $|g\rangle$ and $|e\rangle$ respectively, indicating that this wait time produces a parity map. The very slight asymmetry of the curves is a result of the higher order dispersive shift, which makes the transmon frequency shift not *exactly* linear in the photon number. This data was measured on sample 3 (table 5.1)

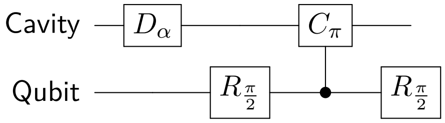
2.3.2 Applications: Wigner tomography

One of the most significant uses of the parity map operation is that it allows us to easily characterize the quantum state of the cavity, via a process known as Wigner tomography. First, let us recall what quantum state tomography is, and how it works on qubits. In tomography, we have some process which generates a quantum state, and our goal is to learn what that state is, by performing measurements. Because measurements are destructive, we need to use many copies of the state. A general d dimensional quantum system has $d^2 - 1$ degrees of freedom in its density matrix, and therefore we need to measure the expectation value of $d^2 - 1$ linearly independent operators in order to reconstruct the state. In a qubit, we can measure the expectation value of 3 operators, typically $\boldsymbol{\sigma}_x$, $\boldsymbol{\sigma}_y$ and $\boldsymbol{\sigma}_z$.

In cavity quantum state tomography, we have an infinite dimensional system, so we do not want to proceed by simply listing some set of operators. We would rather have a family of operators generated in some simple way, where subsets of these operators can accurately represent quantum states which we are reasonably likely to encounter in the lab. One way of generating such a family of operators is via displacements. If we can measure a given operator $\boldsymbol{X}$, then we can also easily measure a new operator $\boldsymbol{D}_{-\alpha}\boldsymbol{X}\boldsymbol{D}_{\alpha}$ by first displacing the cavity by an amount α, and then measuring $\boldsymbol{X}$. Almost any operator $\boldsymbol{X}$ will suffice in order to generate a tomographically complete set of measurements, but it turns out that parity ($\boldsymbol{X} = \boldsymbol{\Pi}$) is a particularly effective choice which yields the Wigner function(Cahill and Glauber, 1969, eq. 4.12):

$$W_\alpha(\boldsymbol{\rho}) = \frac{2}{\pi} x \left\langle \boldsymbol{D}_{-\alpha} \boldsymbol{\Pi} \boldsymbol{D}_\alpha \right\rangle_{\boldsymbol{\rho}} \tag{2.23}$$

We can also represent this measurement in circuit form by concatenating a cavity displacement with a cavity parity measurement:



The effectiveness of the Wigner function in characterizing the cavity state derives from its relationship with the marginal distributions.

$$\int \mathrm{d}p \, W_{x+ip}(\boldsymbol{\rho}) = \langle x|\boldsymbol{\rho}|x\rangle \tag{2.24}$$

$$\int \mathrm{d}x \, W_{x+ip}(\boldsymbol{\rho}) = \langle p|\boldsymbol{\rho}|p\rangle \tag{2.25}$$

Here $|x\rangle$ and $|p\rangle$ are eigenstates of $\boldsymbol{a} + \boldsymbol{a}^\dagger$ and $i(\boldsymbol{a} - \boldsymbol{a}^\dagger)$, respectively. In this repsect it resembles a joint probability distribution on position and momentum space. But *unlike* a probability density function, the Wigner function can be negative (figure 2.5). Negativity of the Wigner function is the signature of uniquely quantum mechanical states, which cannot be thought of classically, and is equivalent to the notion of *contextuality* (Spekkens, 2008), which states that the results of a quantum measurement depend on what other measurements one might be trying to make.

Since the Wigner function is linearly related to the density matrix, it is a matter of simple inversion to reconstruct the density matrix. The viability of this inversion is given by the condition number of the operator relating the two. It is well conditioned for a sufficiently dense sampling of the complex plane. See appendix C.1.2 for more details on how we perform state reconstruction with Wigner tomograms.

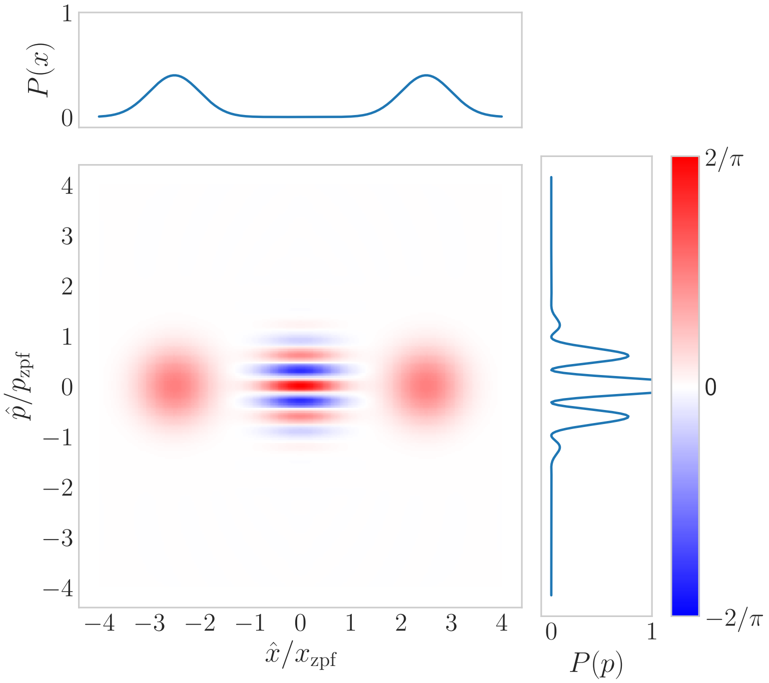


Figure 2.5: **Example Wigner function.** The Wigner function for the cat state $\frac{1}{N}(|\alpha\rangle + |-\alpha\rangle)$. The function integrates to give the probability density for measurements in the position or momentum bases. Unlike a joint probability density, the Wigner function can be negative, as seen in the blue parts of the fringes between the two coherent states. x_{zpf} and p_{zpf} are the zero-point fluctuations of the quadratures, which for mechanical oscillators are $x_{\text{zpf}} = \sqrt{\frac{\hbar}{2m\omega}}$ and $p_{\text{zpf}} = \sqrt{\frac{\hbar m\omega}{2}}$.

2.3.3 Applications: qcMAP

The toolbox gives us a way of characterizing states of the cavity. What we need next is an interesting state to characterize. We can easily generate coherent states $|\alpha\rangle$, but the Wigner function of such a state is just a shifted version of the Wigner function of the vacuum. One of the core features of quantum mechanics is superposition, so we might hope to construct superpositions of coherent states as an example of a non-trivial, non-classical state.

$$|\mathcal{C}_\alpha^\pm\rangle \propto |\alpha\rangle \pm |-\alpha\rangle \tag{2.26}$$

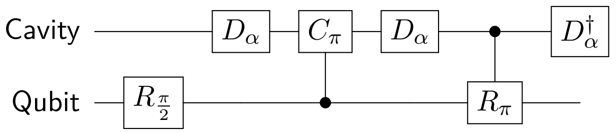
The "qcMAP" protocol, derived first by Leghtas et al. (2013a) and shown experimentally by Vlastakis et al. (2013). Assuming we start in the coherent state $|\alpha\rangle$, we can put the qubit into superposition of $|g\rangle$ and $|e\rangle$ using $\pi/2$ rotation. Then if we again employ the conditional phase space rotation C_π we find we get part of the way there:

$$|g, \alpha\rangle + |e, \alpha\rangle \xrightarrow{\boldsymbol{C}_\pi} |g, \alpha\rangle + |e, -\alpha\rangle$$

This is almost what we want, but the cavity is still entangled with the transmon. This can be dealt with using a clever combination of displacements and conditional qubit rotations.

$$\begin{aligned} |g, \alpha\rangle + |e, -\alpha\rangle &\xrightarrow{\boldsymbol{D}_\alpha} |g, 2\alpha\rangle + |e, 0\rangle \\ &\xrightarrow{\boldsymbol{R}_\pi^{(0)}} |g\rangle \left(|2\alpha\rangle + |0\rangle\right) \\ &\xrightarrow{\boldsymbol{D}_{-\alpha}} |g\rangle \left(|\alpha\rangle + |-\alpha\rangle\right) \end{aligned}$$

We can summarize this set of operations in the gate notation:



2.4 SNAP and universality

While it is clear from the preceding sections that the cQED toolkit enables the construction of a wide variety of operations, it is also quite "ad hoc". While the construction of certain operations is quite obvious, or can be worked out simply on pen and paper, it lacks a general recipe for going from a desired operation to a concrete circuit.

The first question which must be answered is that of the possibility of such a recipe. Is there always a circuit which corresponds to a given operation? What we would like is something resembling the Solovay-Kitaev theorem (Nielsen and Chuang, 2011) which gives a recipe for producing approximations of any single-qubit operation from an "instruction set" of fixed single-qubit gates.

This was achieved by Krastanov et al. (2015) which gave a construction for how to achieve arbitrary operations on dispersively coupled cQED systems using a set of two operations: displacements and *selective number-dependent arbitrary phase* (SNAP) operations. SNAP operations allow an arbitrary set of relative phases to be applied to different photon number states, and can be represented with the form

$$\boldsymbol{S}(\vec{\theta}) = \sum_k e^{i\theta_k} \left|k\middle\rangle\middle\langle k\right| . \tag{2.27}$$

The key to implementing such an operation is the combination of two concepts, the previously discussed selective qubit rotations, and the notion of a "geometric phase."

Geometric phases were first described by Berry (1984) in the context of explaining adiabatic deformations of a Hamiltonian which traversed a loop in parameter space. Being an adiabatic transformation, eigenstates are mapped onto themselves, but superpositions of eigenstates can acquire relative phases which are dependent on the path traversed. Specifically, the phases are equal to the integral over the interior of the path of a quantity defined on called the Berry curvature. This was later generalized as a property of loops traversed within Hilbert space by Aharonov and Anandan (1987). There are two salient examples of this phenomenon for our purposes, one for cavities and one for qubits. In cavities, for instance, one can imagine traversing a loop by performing the following displacements, tracing out a parallelogram in phase space:

$$\boldsymbol{D}_\alpha \boldsymbol{D}_\beta \boldsymbol{D}_{-\alpha} \boldsymbol{D}_{-\beta} = e^{2i\,\mathsf{Im}\,\alpha\beta^*} \tag{2.28}$$

This equality can be checked using the relation

$$\boldsymbol{D}_\alpha f(\boldsymbol{a}, \boldsymbol{a}^\dagger) \boldsymbol{D}_{-\alpha} = f(\boldsymbol{a} + \alpha, \boldsymbol{a}^\dagger + \alpha^*). \tag{2.29}$$

The resulting phase, it is easy to check, is given by the area enclosed by the loop, and this is a general result which applies to loops other than parallelograms. A similar result can be

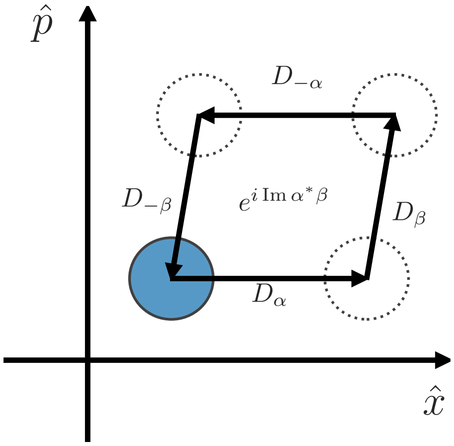


Figure 2.6: **Geometric phase from cyclic displacements**. By performing a set of displacements which traverses a fixed loop, the system acquires a total phase which is proportional to the area of the enclosed loop. Here the area of the parallelogram is Im $\alpha^*\beta$. This phase is global and unobservable by itself, but can be made physically relevant by performing the displacements in a selective, entangling way.

obtained for qubits,

$$\boldsymbol{R}_{\phi}(-\pi)\boldsymbol{R}_0(\pi) = e^{i\phi\boldsymbol{\sigma}_z} = e^{i\phi}\,|g\rangle\langle g| + e^{-i\phi}\,|e\rangle\langle e| \tag{2.30}$$

We can see that the phase is proportional to the area enclosed by the trajectory on the Bloch sphere. Note that, in the context of a single cavity, or a qubit starting in the ground state, there is no way to observe this phase, as it is a global phase. However, when the path taken depends on the state of another system, this phase can become relative. If we replace the qubit rotations in equation 2.30 with photon number selective rotations from 2.17 we obtain

$$\boldsymbol{R}^{(n)}_{\phi}(-\pi)\boldsymbol{R}^{(n)}_0(\pi) = e^{i\phi}\,|n,g\rangle\langle n,g| + e^{-i\phi}\,|n,e\rangle\langle n,e| + (\boldsymbol{I}_c - |n\rangle\langle n|)\,\boldsymbol{I}_q \tag{2.31}$$

We can obtain the SNAP operation (2.27) by chaining several of these operations together (figure 2.8).

$$\boldsymbol{R}^{(N)}_{\theta_N}(-\pi)\boldsymbol{R}^{(N)}_0(\pi)\cdots\boldsymbol{R}^{(0)}_{\theta_0}(-\pi)\boldsymbol{R}^{(0)}_0(\pi) = \sum_{k=0}^{N} e^{i\theta_k}\,|k,g\rangle\langle k,g| + e^{-i\theta_k}\,|k,e\rangle\langle k,e| \tag{2.32}$$

$$= \boldsymbol{S}(\vec{\theta})\,|g\rangle\langle g| + \boldsymbol{S}(-\vec{\theta})\,|e\rangle\langle e| \tag{2.33}$$

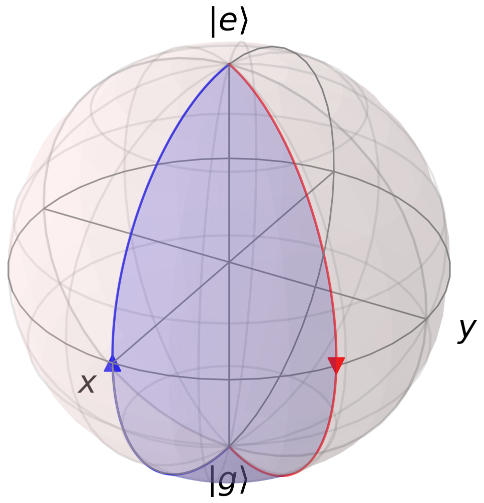


Figure 2.7: **Geometric phase on the Bloch sphere**. By performing a set of rotations which bring the energy eigenstates back to themselves, we effectively perform a rotation around the z axis, with angle given by the enclosed area. Unlike the cavity case, the phase is state-specific. This imparts one phase to the state $|g\rangle$, and the opposite phase to the state $|e\rangle$.

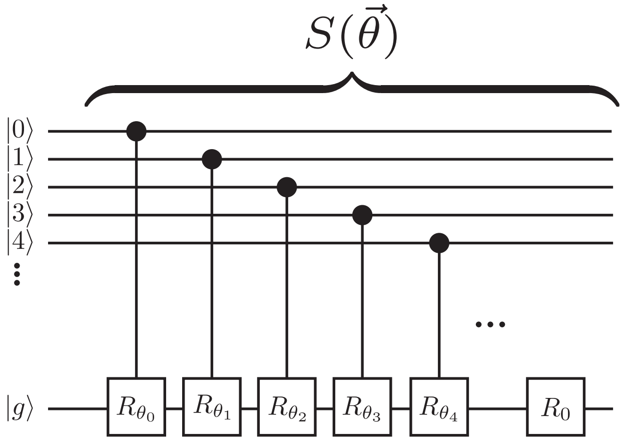


Figure 2.8: **Circuit depiction of SNAP.** In this figure, the upper set of lines represent the photon number *states* which compose the cavity, and the bottom line represents the qubit. The SNAP protocol consists of a set of conditional qubit flips (here all of the rotations are by an angle pi, i.e. $R_\theta = \cos\theta\boldsymbol{\sigma}_x + \sin\theta\boldsymbol{\sigma}_y$

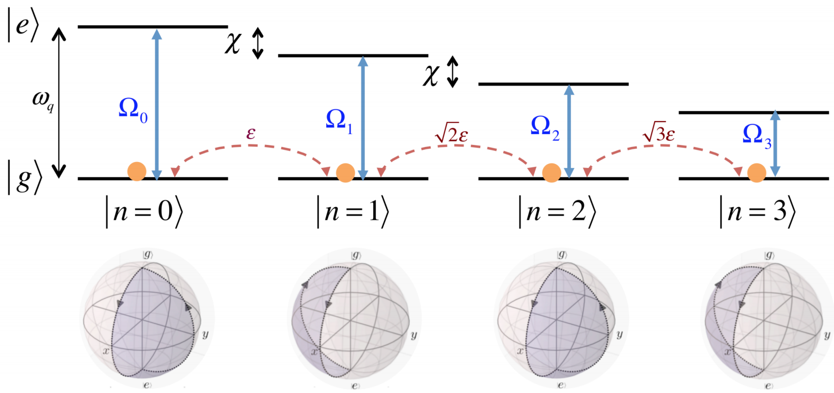


Figure 2.9: **SNAP energy level diagram.** The SNAP protocol consists of simultaneous drives on the χ-separated transitions $|g, n\rangle \leftrightarrow |e, n\rangle$ for all relevant n. The different enclosed areas on the $|n\rangle$ conditional Bloch spheres gives a phase to the relevant photon number state. The progressive sum of the different qubit drives produces a pulse shape shown at the bottom. In order to have complete control of the cavity, a photon number changing drive, such as the displacement, indicated in orange, must be added. Figure adapted from Krastanov et al. (2015)

We see that, assuming the transmon starts in the ground state, the correct SNAP operation is performed on the cavity. While this construction seems awkward and unwieldy at first glance, this protocol can be drastically simplified by realizing that the various selective pulses on different photon number states can be performed simultaneously, as shown in figure 2.9.

SNAP is not a true addition to the cQED toolbox, but rather a re-arrangement, with an eye towards cavity control. It has the important property that, assuming the qubit begins in the ground state, the qubit remains unentangled with the cavity at the end of the operation. Krastanov et al. (2015) showed that any operation on the cavity could be approximated by an alternating sequence of SNAP operations and displacements, following an argument similar to that in appendix B. One can approximate a target unitary operation $\boldsymbol{U}$ as

$$\boldsymbol{U} \approx \boldsymbol{D}_{\alpha_1} \boldsymbol{S}(\vec{\theta}_1) \boldsymbol{D}_{\alpha_2} \boldsymbol{S}(\vec{\theta}_2) \cdots \boldsymbol{D}_{\alpha_N} \boldsymbol{S}(\vec{\theta}_N) \tag{2.34}$$

In order to find the parameters, $\{\boldsymbol{\theta}_k\}$ and $\{\alpha_k\}$, in the construction 2.34, one must turn to a numerical optimization, using a computer to evaluate the fidelity with which some proposed parameter set approximates the desired operation, and modifying the parameters to increase that fidelity. This continues in successive rounds until a desired fidelity is reached.

2.5 Optimal control

The construction 2.34 has a serious downside when it comes to practical application however. While some operations can be constructed using a small number of gates, for instance the construction of Fock state $|1\rangle$ as shown in Heeres et al. (2015), in general an operation on n photons requires $O(n^2)$ gates. We can reduce this cost by noting that the form of 2.34 was motivated primarily by theoretical convenience in terms of analysis, rather than optimality in terms of performance. Specifically, we can see that the construction consists of alternating sections of driving the cavity and driving the qubit. Additionally, the qubit drive is designed to leave the qubit unentangled from the cavity. In this case we have a clear intuitive model for the system dynamics in each stage. However, there is no technical reason why we cannot drive both the qubit and cavity at the same time. While the dynamics are more difficult to reason about without resorting to brute numerical integration, it is clear that the fastest version of an operation will not arbitrarily enforce the alternating form seen in the 2.34 construction. The implication is that we should choose driving fields from a much broader set of possibilities. A drive should be acceptable so long as we can predict in simulation what the result of such a driving field will be. We will show how the combination of a differentiable quantum simulation algorithm and gradient descent optimization methods results in the ability to numerically identify suitable driving fields in chapter 4, and demonstrate the efficacy of these techniques as applied to a particular problem in chapter 6.

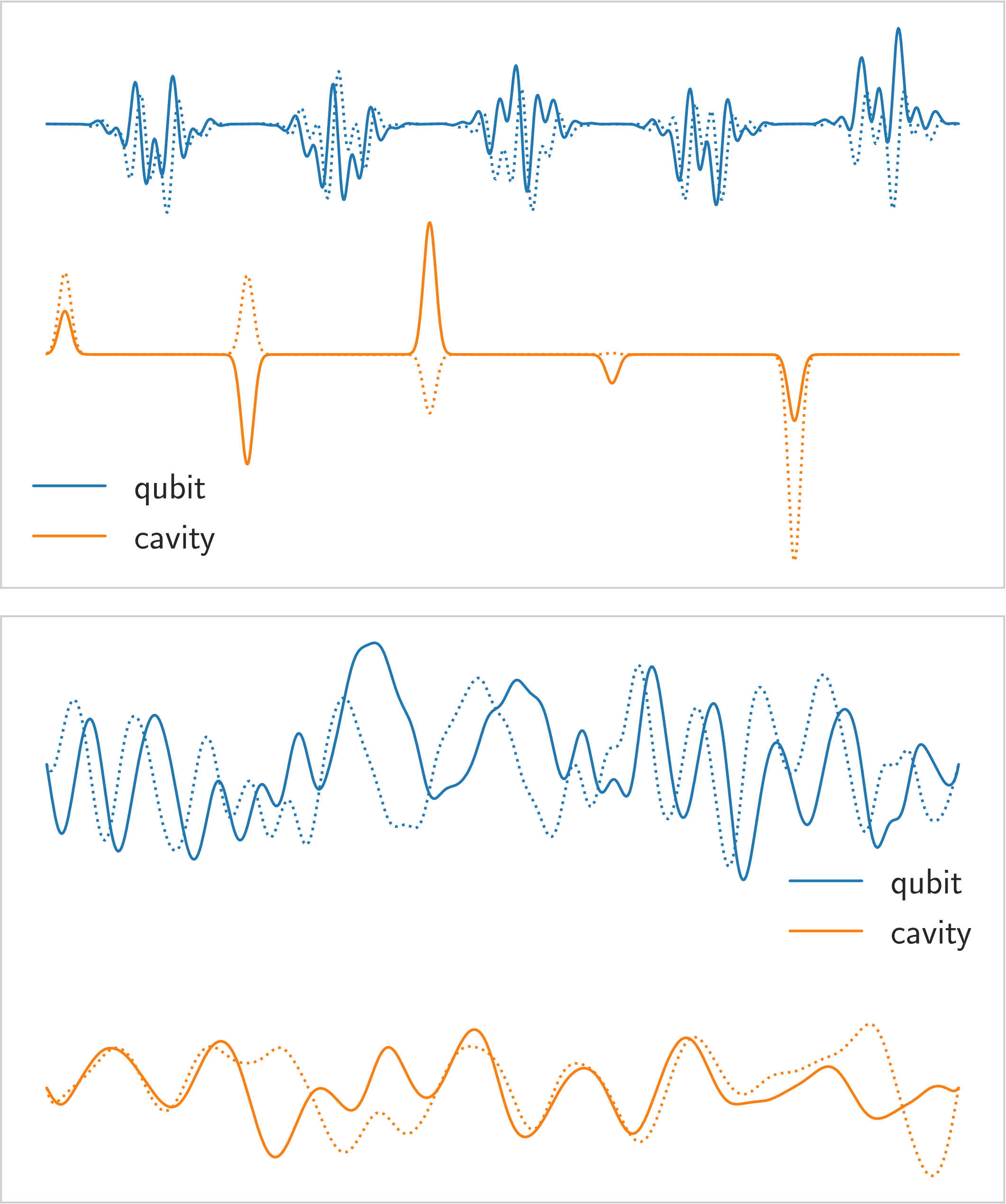


Figure 2.10: **Comparing SNAP and optimal control sequences**. This is a (not-to-scale) schematic comparing the structure of the SNAP and optimal control protocols in the time domain. The SNAP protocol (top) for universal cavity control designates alternating periods of driving the transmon and the cavity. The short cavity pulses produce unconditional displacements, and the transmon driving (each step of time $O(1/\chi)$) produces relative phases on the different photon numbers (equation 2.27). Alternating these components allows any operation to be well approximated in a number of steps related to the number of photons involved. In comparison, the optimal control method (bottom) has no such restriction preventing the simultaneity of transmon and cavity driving. This particular driving field comes from the experiment in chapter 6, and produces a Hadamard operation on a cat encoded qubit. This method of control can in general be much shorter in length than the SNAP protocol. Note that this pulse is still limited in the sense of having intentionally restricted amplitude and bandwidth.

Chapter 3

Doing more with less: error correction with harmonic oscillators

In quantum computation, information is no longer binary, or even discrete, but parameterized by continuously varying amplitudes. Even very small disturbances to these amplitudes can accumulate and inevitably destroy the result of a computation. The first critics of quantum computation pointed to this aspect, and suggested that quantum computing might only be a mathematical curiosity, no more useful than analog computers, which might beat classical digital computers at solving some problems in an idealized setting, but cannot scale due to the noise and imperfections that pervade the real world. The development of the first quantum error correction protocols (Shor, 1995) demonstrated that this is a flawed analogy. While quantum states are "analog" entities, the errors which can occur are discrete, or more accurately, *can be discretized* by the act of measurement. Error correction is the foundation of useful quantum computation, and without it quantum mechanics is too fragile an edifice for the construction of useful machines.

In this chapter, I will introduce our approach to performing error correction, by encoding information in harmonic oscillators. I will begin in section 3.1 by explaining the generic mathematical framework underpinning quantum error correcting codes in all types of systems. Next, in 3.2 we see how this framework is applied in "typical" two-level system based error correcting codes, as well as some of the issues which make implementing this approach a daunting task. In section 3.4, I will explain how moving from two-level systems to cavities changes the underlying error model. Subsequently, in 3.5, we see how the error model and error correction criteria combine to guide our development of oscillator-based encodings. This leads us to consider "cat codes," encodings based on superpositions of coherent states (3.6) as well as other types of codes (3.7).

3.1 Error correction criteria

There are many quantum error correcting codes, but at the heart of all of them, is the satisfaction of the error correction criteria, first put forward by Knill and Laflamme (1997). In this model, we imagine we encode quantum information by preparing a "logical" state $|\psi_L\rangle$ which is a superposition of "code words" $\{|k_L\rangle\}$:

$$|\psi_L\rangle = \sum_k c_k |k_L\rangle \tag{3.1}$$

This state is then processed by a noisy quantum channel, denoted in the Kraus operator-sum representation

$$\boldsymbol{\rho}_{\mathsf{in}} = |\psi_L\rangle\langle\psi_L| \mapsto \boldsymbol{\rho}_{\mathsf{out}} = \sum_k \boldsymbol{E}_k |\psi_L\rangle\langle\psi_L| \boldsymbol{E}_k^\dagger \tag{3.2}$$

The criteria state that the encoded information can be exactly recovered from $\boldsymbol{\rho}_{\mathsf{out}}$ if and only if the following conditions are met for all i and j in the dimension of the code space:

$$\langle j_L| \boldsymbol{E}_b^\dagger \boldsymbol{E}_a |i_L\rangle = C_{ab}\delta_{ij}. \tag{3.3}$$

These criteria can be separated into two parts. The δ_{ij} indicates that the action of the error operators cannot make different code words overlap with each other, which would certainly scramble the encoded information. The C_{ab} part (a matrix independent of i and j), indicates that the rate of occurrence of the errors must be the same for all code words. We can represent this in operator notation, as in Nielsen and Chuang (2011), as

$$\boldsymbol{P}_L \boldsymbol{E}_b^\dagger \boldsymbol{E}_a \boldsymbol{P}_L = C_{ab}\boldsymbol{P}_L, \tag{3.4}$$

where $\boldsymbol{P}_L = \sum_k |k_L\rangle\langle k_L|$ is the projector on the logical space. In the case of a logical qubit, with two code words $|0_L\rangle$ and $|1_L\rangle$ the criteria reduce to the equations

$$\langle 0_L| \boldsymbol{E}_a \boldsymbol{E}_b |1_L\rangle = 0 \tag{3.5}$$

$$\langle 0_L| \boldsymbol{E}_a \boldsymbol{E}_b |0_L\rangle = \langle 1_L| \boldsymbol{E}_a \boldsymbol{E}_b |1_L\rangle \tag{3.6}$$

3.2 Error models for qubits

In general, a quantum channel acting on a d dimensional system has $d^2(d-1)^2$ parameters[1]. This gives 12 parameters for a channel on qubits. This gives a very large space of possibilities for how errors can creep into the system. However, we can collect this entire continuum of error models under a single umbrella via *error discretization*. We can achieve this using two facts. First, it is easy to check, if an error correcting code $\{|0_L\rangle, |1_L\rangle\}$ satisfies the criteria 3.3 for a set of errors $\{\boldsymbol{E}_k\}$, then it also corrects for a set of errors which is linearly related, i.e. $\{\boldsymbol{F}_j = m_{jk}\boldsymbol{E}_k\}$, for some set of coefficients m. Second, any single-qubit error operator can be represented in the Pauli basis, $\boldsymbol{E} = c_I\boldsymbol{I} + c_x\boldsymbol{\sigma}_x + c_y\boldsymbol{\sigma}_y + c_z\boldsymbol{\sigma}_z$. Therefore, it is possible to correct any single qubit error model using a code which targets only single qubit Pauli errors. To simplify even further, we can imagine that the rate for each Pauli channel is identical, which results in the standard "depolarizing channel" model of qubit errors.[2]

$$\boldsymbol{\rho} \rightarrow \frac{p}{2}\boldsymbol{I} + (1-p)\boldsymbol{\rho} \tag{3.7}$$

This is equivalent to a set of errors given by $\{(1-\frac{3p}{4})\boldsymbol{I}, \frac{p}{4}\boldsymbol{\sigma}_x, \frac{p}{4}\boldsymbol{\sigma}_y, \frac{p}{4}\boldsymbol{\sigma}_z\}$. It should be clear that there can be no error correcting code satisfying equation 3.3 when our system consists of a single qubit undergoing depolarizing noise. When we consider multiple qubits undergoing depolarizing noise, we take the error operators to be any Pauli operator acting on any individual qubit. This is only approximately correct, since there are error operators corresponding to errors

[1] This can be derived by starting with the d^4 parameters of a hermiticity-preserving linear map on d^2-dimensional operators, and subtracting the d^2 constraints imposed by the trace preservation.

[2] This approximation can be justified via a process of "Clifford twirling" which replaces any error channel with a depolarizing one. See section C.4.

on multiple qubits. We can justify omitting these terms when p is small, since they will have rates scaling like $O(p^k)$, with $k \geq 2$. Some simple math shows how many qubits we need. If we have n qubits there are $3n + 1$ error operators, 3 Paulis per qubit, plus the identity corresponding to the no-error case. In order for each error to be uniquely identifiable, we must have a two-dimensional subspace corresponding to each error operator, for a total dimension of $2(3n + 1)$.[3] is not strictly require The dimension of an n qubit system is 2^n, so we must have $2^n \geq 2(3n + 1)$, which is only possible when $n \geq 5$.

There are many examples of codes which have been found to satisfy these properties. Almost all of these codes are elegantly described by the "stabilizer code" framework presented in Gottesman (1997). The first of these, and the easiest to understand, was the nine qubit Shor (1995) code, which concatenates three-qubit bit-flip and phase-flip encodings. The smallest, a five qubit code, was found by Laflamme et al. (1996). However, the most elegant, for a variety of reasons, is the seven qubit encoding by Steane (1996). While correction of single-qubit errors is an attractive place to start, without orders of magnitude reduction in physical qubit error rates, codes which allow for multiple errors are needed. The "obvious" approach is concatenation, which stacks error correction protocols, so that the logical qubits of one layer form the physical qubits of the layer above. While this is the easiest approach to describe on paper, topological codes are an attractive alternative because they can be realized using only *local* interactions between qubits in a lattice (see figure 3.1).

3.3 Making error correction work in practice

There have been experimental implementations of the several of these codes, in trapped ions Chiaverini et al. (2004); Nigg et al. (2014), nitrogen vacancy centers Cramer et al. (2016), and superconducting circuits Reed et al. (2012); Kelly et al. (2015); Ristè et al. (2015); Córcoles et al. (2015). However, none of these demonstrations had the desired effect of producing a higher quality qubit. Whether measuring the lifetime or the gate fidelity, the encoded qubits

[3] This property of uniquely identifying every error makes a code "non-degenerate," and is not strictly required by equations 3.3. However, it can be shown that no smaller ($n \leq 4$) degenerate quantum codes exist which correct for single qubit errors.

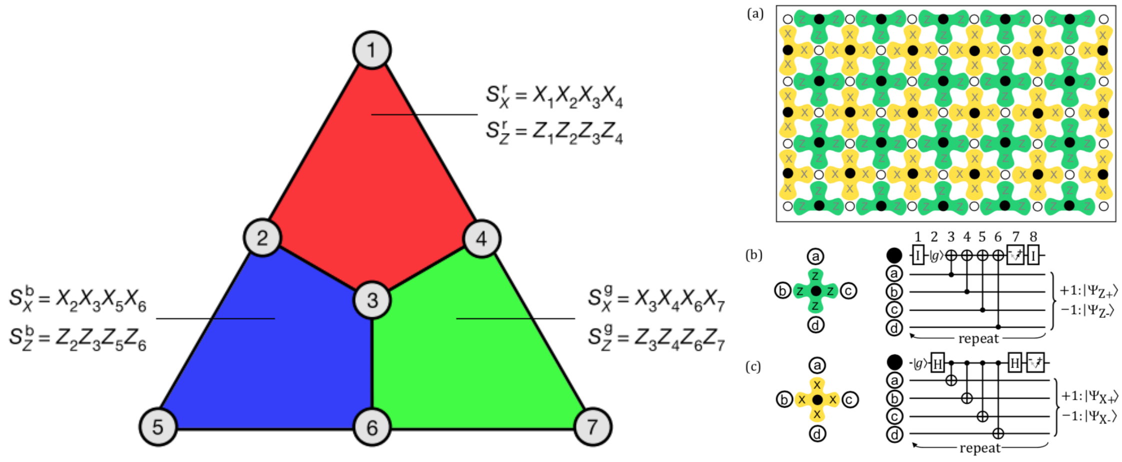


Figure 3.1: **Steane code & Surface code** The Steane code (Steane, 1996)) is the smallest example of a Calderbank, Rains, Shor, and Sloane (1997) (CSS) code which corrects for any single qubit error. It consists of 7 physical qubits, and requires the measurement of 6 stabilizer syndromes, each of weight 4 (left). It is also the smallest example of the "color code" (Bombín, 2015) which is a type of *topological* error correcting code. The most common topological error correcting code is the surface code (right). These codes can be extended by tiling of a primitive unit cell, and are designed to tolerate correlated errors so long as the errors remain "local." Figures adapted from Campbell et al. (2012) and Fowler et al. (2012).

end up being worse than the physical qubits, unless errors were deliberately introduced. Why is this the case? Gate fidelity is a separate beast, which requires a discussion of the issues of fault-tolerance, which we shall delay until chapter 8. If we focus solely on the simpler case of storage error rates, the first issue one encounters is that of *overhead*. The mere act of increasing the size of the system, say from one qubit to n qubits, has increased the rate of error occurrence by a factor of n. Now, even adding a single layer of error correction, for an error rate of γ and a time of δt, we will go from a physical qubit error probability of $p = \gamma\delta t$ to a logical qubit error probability $p_L = \binom{n}{2}p^2$. Therefore, unless $p < 1/\binom{n}{2}$, the final error rate will be worse. Under this consideration, one wants to minimize δt, that is perform correction as often as possible, to avoid double error events. However, the problem is not this simple. In practice one must actually detect and correct the error using some protocol.[4] This protocol generally involves introducing an ancilla qubit, interacting this ancilla with many system qubits, and finally performing a measurement of the ancilla. Each of these steps can have imperfections

[4] In many types of codes, as in the cat-code to be described, it is actually possible to get away without correction, at least until the final stage of decoding.

which add back into the overall error rate, in a way which is independent of the time. Adding all of these per-round errors into one quantity $p_{\text{per-round}}$, the total effective error rate is

$$\gamma_L \equiv \frac{p_L}{\delta t} \approx \binom{n}{2}\gamma^2\delta t + \frac{p_{\text{per-round}}}{\delta t}. \tag{3.8}$$

If we choose δt to minimize this quantity γ_L one finds

$$\delta t = \sqrt{\frac{p_{\text{per-round}}}{\binom{n}{2}\gamma^2}} \tag{3.9}$$

$$\gamma_L = 2\gamma\sqrt{\binom{n}{2}p_{\text{per-round}}} \tag{3.10}$$

Now we find improvement only when $\gamma_L < \gamma$ or equivalently,

$$2\sqrt{\binom{n}{2}p_{\text{per-round}}} < 1 \tag{3.11}$$

From this perspective, we see the fundamental issue is both fighting the overhead (here $\binom{n}{2}$) and the error associated with the detection and correction ($p_{\text{per-round}}$).

3.4 Error models for cavities

In principle, given the infinite dimension of the harmonic oscillator Hilbert space, there could be an infinite number of error operators to be accounted for. However, in practice it is often the case that, when compared with most two level systems, there are actually *fewer* relevant errors to consider. This comes down mostly to the fact that cavities' resonant frequencies are defined almost entirely by their geometry and the speed of light. The stability of these properties leads to stability of the cavity resonant frequencies, and thus essentially eliminates intrinsic dephasing noise. In contrast, most two level systems have frequencies which depend on many fluctuating parameters, such as magnetic or electric fields.

This leaves energy decay. Resonators contain energy very well, but never perfectly. Energy always leaks from a resonator into its environment. This energy loss can be modelled using a

Lindblad-Markov master equation with jump operator $\boldsymbol{a}$ (see appendix A.6):

$$\dot{\boldsymbol{\rho}} = i[\boldsymbol{H}, \boldsymbol{\rho}] + \kappa D[\boldsymbol{a}](\boldsymbol{\rho}), \tag{3.12}$$

where the dissipator Liouvillian is defined as

$$D[\boldsymbol{a}](\boldsymbol{\rho}) \equiv \boldsymbol{a}\boldsymbol{\rho}\boldsymbol{a}^\dagger - \frac{1}{2}\left\{\boldsymbol{a}^\dagger\boldsymbol{a}, \boldsymbol{\rho}\right\}. \tag{3.13}$$

If we integrate this equation for a time δt, we can write the effective quantum channel in the operator sum notation:

$$\boldsymbol{\rho} \to \sum_{n=0} \boldsymbol{E}_n \boldsymbol{\rho} \boldsymbol{E}_n^\dagger \tag{3.14}$$

where the Kraus operator $\boldsymbol{E}_n$ corresponds to the loss of n photons (Michael et al., 2016)

$$\boldsymbol{E}_n = \sqrt{\frac{(1 - e^{-\kappa\delta t})^n}{n!}} e^{-\frac{\kappa\delta t}{2}\boldsymbol{a}^\dagger\boldsymbol{a}} \boldsymbol{a}^n \tag{3.15}$$

The part involving $e^{-\frac{\kappa\delta t}{2}\boldsymbol{a}^\dagger\boldsymbol{a}}$ gives the amplitude damping component, which we will come back to in section 3.6.2. In the limit of $\kappa\delta t \ll 1$, we can make the approximation

$$\boldsymbol{E}_0 \approx \boldsymbol{I} \tag{3.16}$$

$$\boldsymbol{E}_1 \approx \sqrt{\kappa\delta t}\boldsymbol{a} \tag{3.17}$$

$$\boldsymbol{E}_{n>1} = O(\kappa\delta t) \tag{3.18}$$

3.5 Error correction in damped harmonic oscillators

Let us take a look at the error correction criteria 3.5 and 3.6 again from the perspective of the damped harmonic oscillator error set, $\{\boldsymbol{I}, \boldsymbol{a}\}$. In this case, there are eight independent equations which constitute the error correction criteria, and these equations all have reasonable interpretations, which one can keep in mind while considering potential codes. To begin with, our code words must be orthogonal

$$\langle 0_L | 1_L \rangle = 0 \tag{3.19}$$

Photon loss must not make $|1_L\rangle$ look like $|0_L\rangle$ or vice versa.

$$\langle 0_L| \boldsymbol{a} |1_L\rangle = 0 \tag{3.20}$$

$$\langle 0_L| \boldsymbol{a}^\dagger |1_L\rangle = 0 \tag{3.21}$$

Our states must remain orthogonal after photon loss.

$$\langle 0_L| \boldsymbol{a}^\dagger \boldsymbol{a} |1_L\rangle = 0 \tag{3.22}$$

Our states are normalized.

$$\langle 0_L|0_L\rangle = \langle 1_L|1_L\rangle \tag{3.23}$$

Photon loss must not take $|+_L\rangle \propto |0_L\rangle + |1_L\rangle$ to $|-_L\rangle \propto |0_L\rangle - |1_L\rangle$, or vice versa.[5]

$$\langle 0_L| \boldsymbol{a} |0_L\rangle = \langle 1_L| \boldsymbol{a} |1_L\rangle \tag{3.24}$$

$$\langle 0_L| \boldsymbol{a}^\dagger |0_L\rangle = \langle 1_L| \boldsymbol{a}^\dagger |1_L\rangle \tag{3.25}$$

The states must have equal average photon number, and thus equal probability of decay occurring.

$$\langle 0_L| \boldsymbol{a}^\dagger \boldsymbol{a} |0_L\rangle = \langle 1_L| \boldsymbol{a}^\dagger \boldsymbol{a} |1_L\rangle \tag{3.26}$$

While not strictly necessary, one convenient way of satisfying equations 3.24 and 3.25 is to impose the constraint that

$$\langle 0_L| \boldsymbol{a} |0_L\rangle = \langle 1_L| \boldsymbol{a} |1_L\rangle = 0 \tag{3.27}$$

This can be interpreted as saying that we should look for codes where the "error space" formed by the span of $\boldsymbol{a}|0_L\rangle$ and $\boldsymbol{a}|1_L\rangle$ should be completely orthogonal to the uncorrupted logical space. In all of the (functional) codes we will analyze here, this additional constraint will be satisfied.

[5] We can see the equivalence of these two statements by looking at $\langle +_L| \boldsymbol{a} |-_L\rangle = 0$ and making use of 3.20 and 3.21.

3.6 Cat codes

Given that, for short times, the single dominant error channel for a damped harmonic oscillator is the application of the photon annihilation operator $\boldsymbol{a}$, one is naturally led to consider coherent states, when considering building codes for this channel. This is because coherent states are eigenstates of $\boldsymbol{a}$:

$$\begin{aligned} \boldsymbol{a}\left|\alpha\right\rangle &= \boldsymbol{D}_\alpha \boldsymbol{D}_{-\alpha} \boldsymbol{a} \boldsymbol{D}_\alpha \left|0\right\rangle && (3.28) \\ &= \boldsymbol{D}_\alpha (\boldsymbol{a} + \alpha) \left|0\right\rangle && (3.29) \\ &= \alpha \left|\alpha\right\rangle && (3.30) \end{aligned}$$

Is this sufficient then? Can we protect information by encoding into coherent states? Let's see what happens when we do so, with $\left|0_L\right\rangle = \left|\alpha\right\rangle$ and $\left|1_L\right\rangle = \left|\beta\right\rangle$, and with the (approximate) error operators $\{\boldsymbol{I}, \boldsymbol{a}\}$. The error correction orthogonality criteria, 3.5, becomes 4 equations:

$$\langle\alpha|\beta\rangle = e^{-|\alpha-\beta|^2/2} \overset{?}{=} 0$$

$$\left\langle\alpha\right| \boldsymbol{a} \left|\beta\right\rangle = \beta e^{-|\alpha-\beta|^2/2} \overset{?}{=} 0$$

$$\left\langle\alpha\right| \boldsymbol{a}^\dagger \left|\beta\right\rangle = \alpha^* e^{-|\alpha-\beta|^2/2} \overset{?}{=} 0$$

$$\left\langle\alpha\right| \boldsymbol{a}^\dagger \boldsymbol{a} \left|\beta\right\rangle = \alpha^* \beta e^{-|\alpha-\beta|^2/2} \overset{?}{=} 0$$

While these equations can never be exactly satisfied, they can be arbitrarily close to being satisfied by taking α and β to be sufficiently far apart. So far so good! Our basis states remain orthogonal under the action of the errors. We proceed to the next criteria, 3.6, where we again get four equations.

$$\langle\alpha|\alpha\rangle = 1 \overset{?}{=} 1 = \langle\beta|\beta\rangle$$

$$\left\langle\alpha\right| \boldsymbol{a} \left|\alpha\right\rangle = \alpha \overset{?}{=} \beta = \left\langle\beta\right| \boldsymbol{a} \left|\beta\right\rangle$$

$$\left\langle\alpha\right| \boldsymbol{a}^\dagger \left|\alpha\right\rangle = \alpha^* \overset{?}{=} \beta^* = \left\langle\beta\right| \boldsymbol{a}^\dagger \left|\beta\right\rangle$$

$$\langle\alpha|\, \boldsymbol{a}^\dagger \boldsymbol{a}\, |\alpha\rangle = |\alpha|^2 \overset{?}{=} |\beta|^2 = \langle\beta|\, \boldsymbol{a}^\dagger \boldsymbol{a}\, |\beta\rangle$$

The first is of course satisfied, and we can satisfy the last by choosing $\beta = e^i\phi\alpha$. However, we cannot satisfy middle two. There are several ways we can interpret this failure. The most clear, however, is to consider the logical superposition states $|\pm_L\rangle = |0_L\rangle \pm |1_L\rangle$:

$$\langle -_L|\, \boldsymbol{a}\, |+_L\rangle = (\langle\alpha| - \langle\beta|)\, \boldsymbol{a}\, (|\alpha\rangle + |\beta\rangle) \tag{3.31}$$

$$\approx \alpha - \beta \tag{3.32}$$

We see that the action of the error takes $|+_L\rangle$ to at least some part $|-_L\rangle$. This means that, while the errors may not induce confusion in the $|0/1_L\rangle$ basis, it will corrupt superposition states.

We can solve this problem by recalling that equations 3.20, 3.21 and 3.27 suggest that we make the "error space" completely orthogonal to the "logical space." We can do this by noting that the action of the error changes the photon number parity: if we have an even number of photons, and then we lose a photon, we now have an odd number of photons. This suggests we encode our information into states of definite parity, in which case the error space is guaranteed to be orthogonal to the logical space. While coherent states do not have definite parity (with the exception of the vacuum), we can easily construct such states with a sparse representation in the coherent state basis ($\mathcal{N}$ is a normalization constant):

$$\frac{1}{\mathcal{N}}\left(|\alpha\rangle \pm |{-\alpha}\rangle\right) = \frac{1}{\mathcal{N}}\sum_n \left(\frac{\alpha^n}{\sqrt{n!}} \pm \frac{(-\alpha)^n}{\sqrt{n!}}\right)|n\rangle \tag{3.33}$$

$$= \frac{1}{\mathcal{N}}\sum_n \frac{\alpha^n}{\sqrt{n!}}(1 \pm (-1)^n)\,|n\rangle \tag{3.34}$$

$$= \frac{1}{\mathcal{N}}\sum_{n\ \text{even/odd}} \frac{2\alpha^n}{\sqrt{n!}}\,|n\rangle \tag{3.35}$$

$$\equiv |C_\alpha^\pm\rangle \tag{3.36}$$

These are the so-called cat states, theoretically described by Leghtas et al. (2013b) and Mirrahimi et al. (2014), and produced experimentally by Vlastakis et al. (2013). They are named by analogy to Schrödinger's thought experiment, where we imagine that, in the limit of large amplitudes α, the state is a superposition of what would otherwise be considered "classical" states.

If we want to choose logical basis states using cat states such that the parity is well defined in the logical subspace, we must keep the sign fixed (we choose even states for reasons discussed in section 3.6.1) and differentiate the states by choice of α. Maximizing symmetry leads to the following encoding[6]

$$|0_L\rangle = \left|C_\alpha^+\right\rangle \tag{3.37}$$

$$|1_L\rangle = \left|C_{i\alpha}^+\right\rangle \tag{3.38}$$

This encoding satisfies our demand that the photon number parity be well defined and even. We know then consequentially, that under the action of photon loss ($\boldsymbol{a}$) the parity will remain definite, but change from even to odd.

$$\begin{aligned}
\boldsymbol{a}\,|0_L\rangle &= \boldsymbol{a}\left|C_\alpha^+\right\rangle & (3.39)\\
&= \frac{1}{\mathcal{N}}\left(\boldsymbol{a}\,|\alpha\rangle + \boldsymbol{a}\,|{-\alpha}\rangle\right) & (3.40)\\
&= \frac{1}{\mathcal{N}}\left(\alpha\,|\alpha\rangle - \alpha\,|{-\alpha}\rangle\right) & (3.41)\\
&= \alpha\left|C_\alpha^-\right\rangle \equiv \alpha\,|1_E\rangle & (3.42)\\
\boldsymbol{a}\,|1_L\rangle &= \boldsymbol{a}\left|C_{i\alpha}^+\right\rangle & (3.43)\\
&= i\alpha\left|C_{i\alpha}^-\right\rangle \equiv i\alpha\,|1_E\rangle & (3.44)
\end{aligned}$$

We see then, that more than just going to the odd parity subspace, the action of photon loss keeps us in the odd parity *cat states*, which form the "error space" $\{|0_E\rangle\,,|1_E\rangle\}$. It is instructive

[6] This is not the final definition we will be using, which is rather equations 3.56 and 3.57.

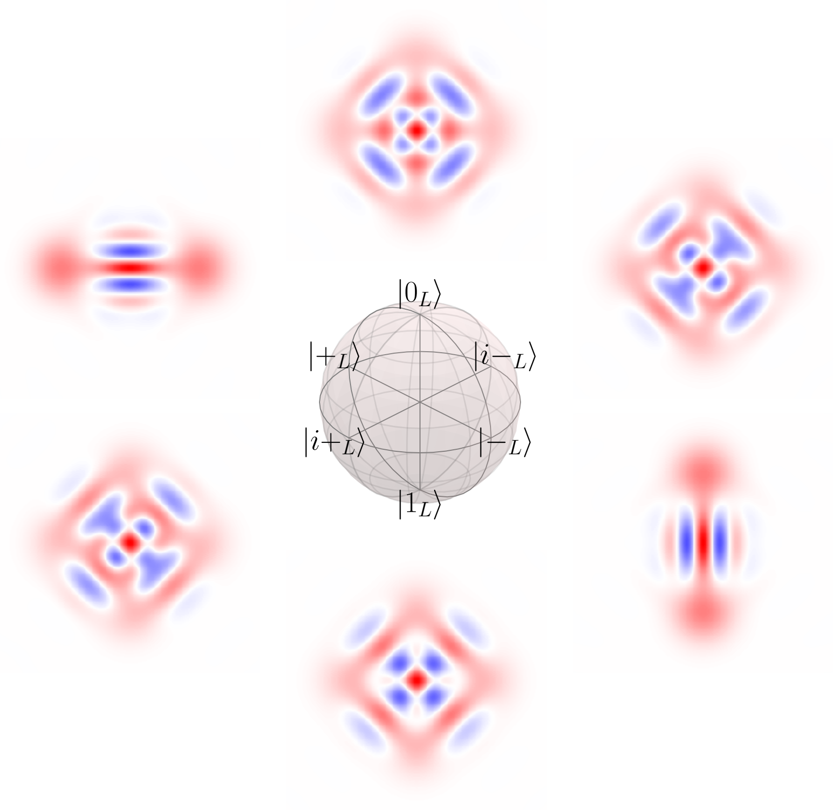


Figure 3.2: **Cat code logical Bloch sphere.** Wigner functions for the six stabilizer states of the logical Bloch sphere using cat states of size $\alpha = \sqrt{3}$. This follows definitions 3.56 and 3.57, and is the same size cat as is used in the experimental implementation of chapter 6.

to see what happens when we lose a second photons after the first:

$$\boldsymbol{a}\,|0_E\rangle = \boldsymbol{a}\,|C_\alpha^-\rangle \tag{3.45}$$

$$= \frac{1}{\mathcal{N}}\left(\boldsymbol{a}\,|\alpha\rangle - \boldsymbol{a}\,|-\alpha\rangle\right) \tag{3.46}$$

$$= \frac{1}{\mathcal{N}}\left(\alpha\,|\alpha\rangle + \alpha\,|-\alpha\rangle\right) \tag{3.47}$$

$$= \alpha\left|C_\alpha^+\right\rangle \equiv \alpha\,|0_L\rangle \tag{3.48}$$

$$\boldsymbol{a}\,|1_E\rangle = \boldsymbol{a}\left|C_{i\alpha}^-\right\rangle \tag{3.49}$$

$$= i\alpha\left|C_{i\alpha}^+\right\rangle \equiv i\alpha\,|1_L\rangle \tag{3.50}$$

We see then that photon loss returns us from the error space back to the logical space. We do not return without incident, however.

$$\boldsymbol{a}^2 \left|0_L\right\rangle = \alpha^2 \left|0_L\right\rangle \tag{3.51}$$

$$\boldsymbol{a}^2 \left|1_L\right\rangle = -\alpha^2 \left|1_L\right\rangle \tag{3.52}$$

We can remove the factor of α^2 via normalization, but the relative sign between $\left|0_L\right\rangle$ and $\left|1_L\right\rangle$ is an $\boldsymbol{\sigma}_z$ operation on the logical qubit which switches $\left|+_L\right\rangle$ and $\left|-_L\right\rangle$. In fact, photon loss forms a *4-cycle* on the cat codes

$$\boldsymbol{a}^4 \left|0_L\right\rangle = \alpha^4 \left|0_L\right\rangle \tag{3.53}$$

$$\boldsymbol{a}^4 \left|1_L\right\rangle = \alpha^4 \left|1_L\right\rangle \tag{3.54}$$

However, so long as we know the number of photons lost modulo 4, we know what position we occupy within the 4-cycle and correspondingly how to recover our quantum information (figure 3.3).

From the perspective of a quantum memory application, where the only goal is to store and recover quantum information, the only thing we must do is count photon jumps, in a way which does not learn information about the encoded qubit, which is possible by measuring specifically the photon number parity. By repeatedly measuring the photon number parity, we can infer that, between two parity measurements where the parity remains the same, no photons were lost, and between parity measurements which differ, a single photon was lost. This is a valid inference only in the limit where the probability of multiple photons being lost is negligible. The extent to which this is not negligible represents a failure mode of the encoding.

Over a period of time δt given some uniform photon loss rate κ, and assuming the population of cavity is kept fixed at $\bar{n}$, the number of photons lost should follow a Poisson distribution with rate parameter $\bar{n}\kappa\delta t$, i.e. $p(n) = (\bar{n}\kappa\delta t)^n e^{-\kappa\delta\bar{n}t}/n! \approx (\kappa\delta t)^n/n!$ in the limit of small $\kappa\delta t$. The action of the error correction can be seen as eliminating the error associated with single photon loss, pushing the error rates from first to second order, i.e. $O(\bar{n}\kappa\delta t)$ to $O((\bar{n}\kappa\delta t)^2)$.

In order for cat code error correction to be practical to use in real systems, there are several

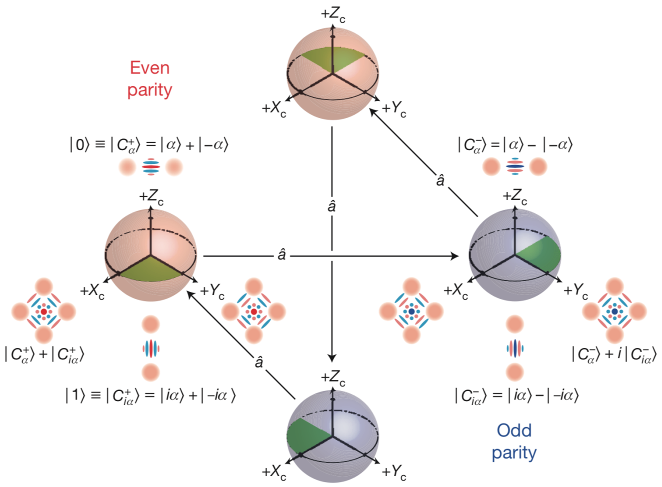

$+Z_c$
Even parity
$+X_c$
$+Y_c$
$|0\rangle \equiv |C_\alpha^+\rangle = |\alpha\rangle + |-\alpha\rangle$
$|C_\alpha^-\rangle = |\alpha\rangle - |-\alpha\rangle$
$\hat{a}$
$|C_\alpha^+\rangle + |C_{i\alpha}^+\rangle$
$|C_\alpha^-\rangle + i\,|C_{i\alpha}^-\rangle$
$|1\rangle \equiv |C_{i\alpha}^+\rangle = |i\alpha\rangle + |-i\alpha\rangle$
$|C_{i\alpha}^-\rangle = |i\alpha\rangle - |-i\alpha\rangle$
Odd parity

Figure 3.3: **The 4 cycle of the cat code under photon loss.** The logical subspace (red Bloch spheres) has definite (even) photon number parity. Losing a photon brings us from the logical space to the error space (blue Bloch spheres) which has odd photon number parity. If we lose additional photons, we are taken back to the logical space. However, this process of losing two photons does not behave as the identity, but rather amounts to the application of a logical Pauli operator corresponding to the axis associated with the two-legged cat states (in this figure, $\boldsymbol{\sigma}_z$, but for the code words in equations 3.56 and 3.57 it is $\boldsymbol{\sigma}_x$) This figurew was adapted from Ofek et al. (2016).

issues we must address. First is the possibility of preparing such states in the first place, an obvious prerequisite. Next is the detection and correction of photon loss in a way which does not "learn too much" and destroy the encoded quantum information. Additionally we must consider other types of errors, both of the coherent control type as well as the dissipative noisy type, and make sure these errors are either negligible or suppressed. Finally, if we wish to go beyond information storage, and do actual computation, we will need ways of manipulating the cat states, preferably in a way which preserves the structure of the errors.

We will address the preparation and manipulation problems head on in chapter 6, since these issues are tied closely to the particular control scheme employed, specifically using far detuned, dispersively coupled transmon qubits to allow for universal control of cavity states. We can discuss the issue of error detection and correction at a much higher level of generality.

3.6.1 Choosing α

Cat codes are not a single code, but are instead a family of codes parameterized by the coherent state amplitude α. As α becomes bigger, the average photon number $\bar{n}$ increases quadratically, as $|\alpha|^2$. The rate of photon loss is proportional to $\bar{n}$, and therefore increasing α can increase the error rate via undetected double photon loss events. However, α cannot be too small either, because of the non-orthogonality of coherent states.

$$\langle\alpha|\beta\rangle = \langle 0|\beta - \alpha\rangle = e^{-|\alpha-\beta|^2/2} \tag{3.55}$$

While this non-orthogonality is the root source of the problem, it is possible to formulate exactly orthogonal code words for any given value of α by defining our basis states in terms of the "four-legged cats" instead of the "two-legged cats":

$$|0_L\rangle = \frac{1}{\mathcal{N}_+}\left(|\alpha\rangle + |-\alpha\rangle + |i\alpha\rangle + |-i\alpha\rangle\right) \tag{3.56}$$

$$|1_L\rangle = \frac{1}{\mathcal{N}_-}\left(|\alpha\rangle + |-\alpha\rangle - |i\alpha\rangle - |-i\alpha\rangle\right) \tag{3.57}$$

Here $\mathcal{N}_\pm \neq 2$ because of the finite overlap of of $|C_\alpha^+\rangle$ and $\left|C_{i\alpha}^+\right\rangle$. However, $|0_L\rangle$ and $|1_L\rangle$ *are* exactly orthogonal, as can be confirmed via examining their Fock basis representations:

$$|0_L\rangle = \frac{4}{\mathcal{N}_+}\sum_n \frac{\alpha^{4n}}{\sqrt{4n!}}|4n\rangle \tag{3.58}$$

$$|1_L\rangle = \frac{4}{\mathcal{N}_+}\sum_n \frac{\alpha^{4n+2}}{\sqrt{(4n+2)!}}|4n+2\rangle \tag{3.59}$$

These states have definite and distinct "super-parity," defined by the second-least significant bit of the binary expansion of the photon number. This is orthogonal, no matter the value of α, so long as $\alpha > 0$. It is instructive to consider the limit of infinitesimal α, which results in the following encoding:

$$|0_L\rangle = |0\rangle \tag{3.60}$$

$$|1_L\rangle = |2\rangle \tag{3.61}$$

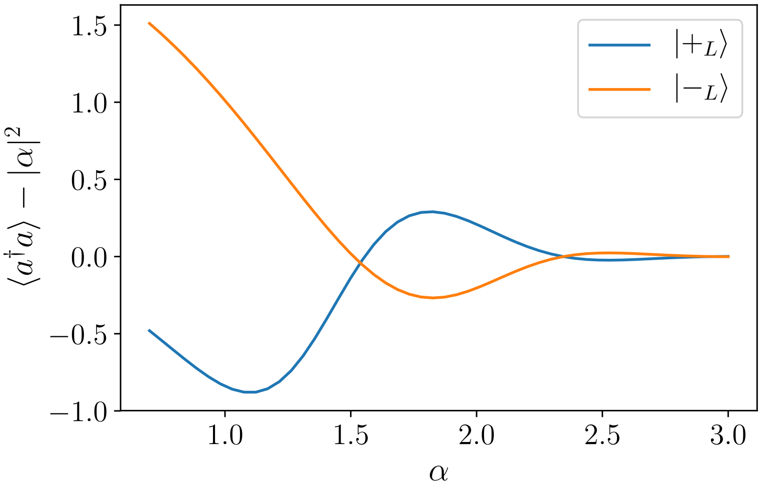


Figure 3.4: **Expected photon number in cat code vs α.** The photon number difference vanishes at $\alpha \approx$ 1.54, 2.34 and 2.94. This can make a difference if the cat code is used for only a single round of error correction. In a more long-term error correction setting, we will need the code to work equally well in both the even- and odd-parity subspaces. Additionally, if no mechanism is provided to re-inject photons, α will decline over time (section 3.6.2)

This is obviously not a valid error correction code. While we can certainly detect the presence of an error here, by measuring the state $|1\rangle$, once we do this, there is no way of recovering the encoded information. For general small values of α, the problem is the difference in photon number between the two logical states, violating the error correction criteria 3.26. This can be seen from figure 3.4, which shows this deviation of $\bar{n}$ from $|\alpha|^2$ as a function of α. There are particular values of α at which the difference vanishes. These points are the optimal point for the even encoding. A general examination of these issues involved with selecting α is given by Li et al. (2017).

3.6.2 “No-jump” errors and autonomous stabilization

While monitoring the parity degree of freedom can track the occurrence of photons escaping the cavity, there is no point in doing so indefinitely. There are a finite number of photons in the cavity, and after a sufficiently long time, *all* of them leak out, and we will inevitably find ourselves in the vacuum state. This would seem to set a time limit on our ability to perform error correction. Let's see how this manifests dynamically by first considering the evolution of a coherent state under photon loss. Using the Lindblad equation for a damped harmonic oscillator (appendix A.6), we have

$$\partial_t \boldsymbol{\rho} = \kappa \left(\boldsymbol{a}\boldsymbol{\rho}\boldsymbol{a}^\dagger - \frac{1}{2}\{\boldsymbol{a}^\dagger \boldsymbol{a}, \boldsymbol{\rho}\} \right). \tag{3.62}$$

There are two parts to this evolution, the "jump" component $\boldsymbol{a}\boldsymbol{\rho}\boldsymbol{a}^\dagger$, corresponding to the loss of a a photon, and the "no jump" component $\{\boldsymbol{a}^\dagger\boldsymbol{a}, \boldsymbol{\rho}\}$, corresponding to the backaction of the effective weak measurement photon-loss induces. A coherent state is an eigenstate of photon loss, so it turns out that we can actually ignore the effect of photon jumps, and focus on the backaction.[7] This is effectively a non-Hermitian Hamiltonian evolution (Dalibard et al., 1993), which being time-independent, has a matrix exponential solution. We can then show that a coherent state remains a coherent state:

$$\begin{aligned} |\psi(t+\delta t)\rangle &= e^{-\frac{\kappa}{2}\delta t \boldsymbol{a}^\dagger \boldsymbol{a}} |\alpha\rangle & (3.63) \\ &= \sum_k e^{-\frac{\kappa}{2}\delta t k} \frac{\alpha^k}{k!} |k\rangle & (3.64) \\ &= \left|\alpha e^{-\frac{\kappa}{2}\delta t}\right\rangle. & (3.65) \end{aligned}$$

A coherent state therefore collapses in amplitude, exponentially, with a timescale of $\frac{2}{\kappa}$. In cat states, similar behavior will occur. While we cannot ignore the effect of photon loss in the same way (indeed this is the purpose of the error correction) the amplitude of the cat components will inevitably shrink in the same way if not counteracted. So long as the coherent states $|\alpha\rangle$ and $|i\alpha\rangle$ remain sufficiently orthogonal, the induced error is manageable, but this cannot remain the case over arbitrarily large timescales, without exponentially large cat state amplitudes. One needs a mechanism for re-introducing photons to the cat code. We note that it is possible to *stabilize* a coherent state against the effect of photon loss by providing a drive. This can be checked by doing the algebra in the full master equation, but it can be more easily seen by starting from the driven non-Hermitian Hamiltonian $\boldsymbol{H} = \epsilon \boldsymbol{a} + \epsilon^* \boldsymbol{a}^\dagger - i\frac{\kappa}{2}\boldsymbol{a}^\dagger \boldsymbol{a}$, and seeing that we can make the drive vanish by going to a displaced frame of amplitude $\frac{2i\epsilon}{\kappa}$. This process is

[7] The photon loss component is not zero, but rather proportional to $\boldsymbol{\rho}$. This term compensates for the loss of normalization we see when only focusing on the backation.

completely analogous to the process of driving with a detuning, as described in appendix A.4.

It turns out that a similar mechanism can be employed to stabilize cat states. The problem of using a drive like $\epsilon \boldsymbol{a} +$ h.c. is that it necessarily changes parity and thus would immediately mix the even and odd subspaces. What is needed is actually a parity-conserving drive of the form $\epsilon \boldsymbol{a}^2 +$ h.c.. This type of drive was implemented using sideband drives (see chapter 7) and was shown by Leghtas et al. (2015) to stabilize a cat state in the cavity. More accurately, it stabilizes not just a cat state, but any state in the manifold spanned by $|\alpha\rangle$ and $|-\alpha\rangle$. However, this is not quite enough to stabilize all of the cat-code states spanned by 3.56 and 3.57. For this purpose, we need to go one level higher, and drive *four* photons at a time, with a drive of the form $\epsilon \boldsymbol{a}^4 +$ h.c. (Mirrahimi et al., 2014). Implementing such a drive is a difficult, demanding task, but progress has recently been made toward this goal (Mundhada et al., 2018). In addition, it is possible to use optimal control pulses (as discussed in chapter 6) to perform re-inflation in a stroboscopic manner, rather than a continuous one.

3.7 Alternate cavity encodings

3.7.1 Binomial codes

Cat codes are built from coherent states, and are best integrated with other components geared toward coherent states, such as those found in the dispersive toolbox described in section 2.3. However, there are occasions and reasons to prefer states which are sparsely described in the photon number basis. For this reason, we often turn to the so-called "binomial codes," described by Michael et al. (2016). These codes are closely related to the cat code, but are constructed in such a way that they are supported by a finite number of photon number states. There is an entire family of such codes which can be constructed to protect against any number of photon loss ($\boldsymbol{a}$), photon gain ($\boldsymbol{a}^\dagger$) or dephasing events ($\boldsymbol{a}^\dagger \boldsymbol{a}$). The best known version, however, is the one which mimics the original cat code (colloquially known as the "kitten code").

$$|0_L\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |4\rangle\right) \tag{3.66}$$

$$|1_L\rangle = |2\rangle \tag{3.67}$$

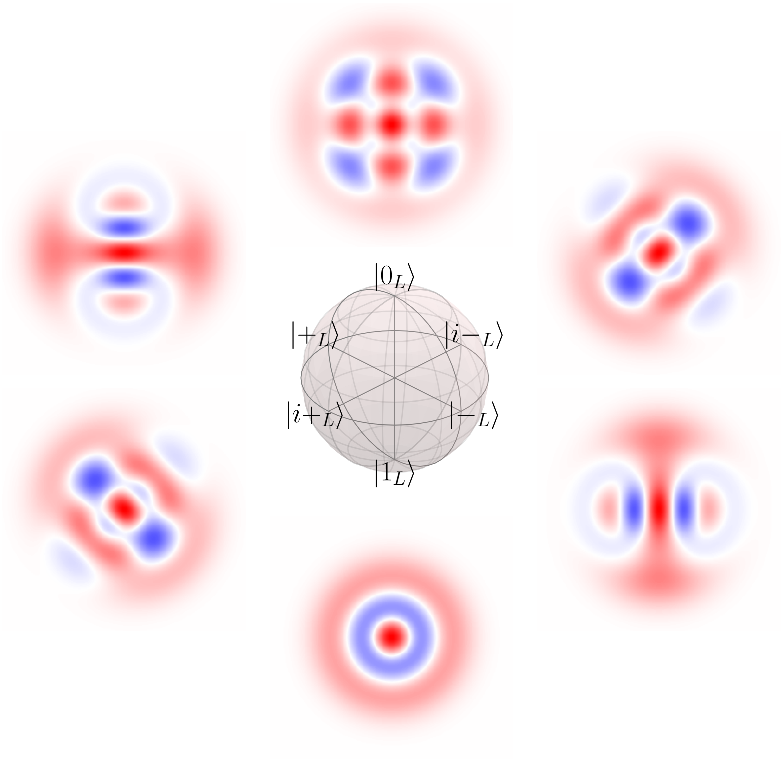


Figure 3.5: **Binomial code logical Bloch sphere.** Wigner functions for the six stabilizer states of the logical Bloch sphere using the lowest order binomial encoding (equation 3.66). These states clearly resemble the cat-code states (figure 3.2), but are supported by a finite number of Fock states.

This is in some sense the smallest functioning version of the cat code (although not the smallest functioning code, see section 3.7.3). In practice it operates in much the same way, requiring continual monitoring of the parity degree of freedom to catch photon jumps as they occur. Unlike the cat code, however, photon loss events must be immediately corrected, rather than simply tracked, as the resulting error space

$$|0_E\rangle = |3\rangle \tag{3.68}$$

$$|1_E\rangle = |1\rangle \, , \tag{3.69}$$

is not a valid error correcting code at all. Additionally, the issue of no-jump errors are more severe, as any decrease in $\bar{n}$ takes us to an (at least partially) uncorrectable space. However, because these code words involve such a small number of photons, different methods can be

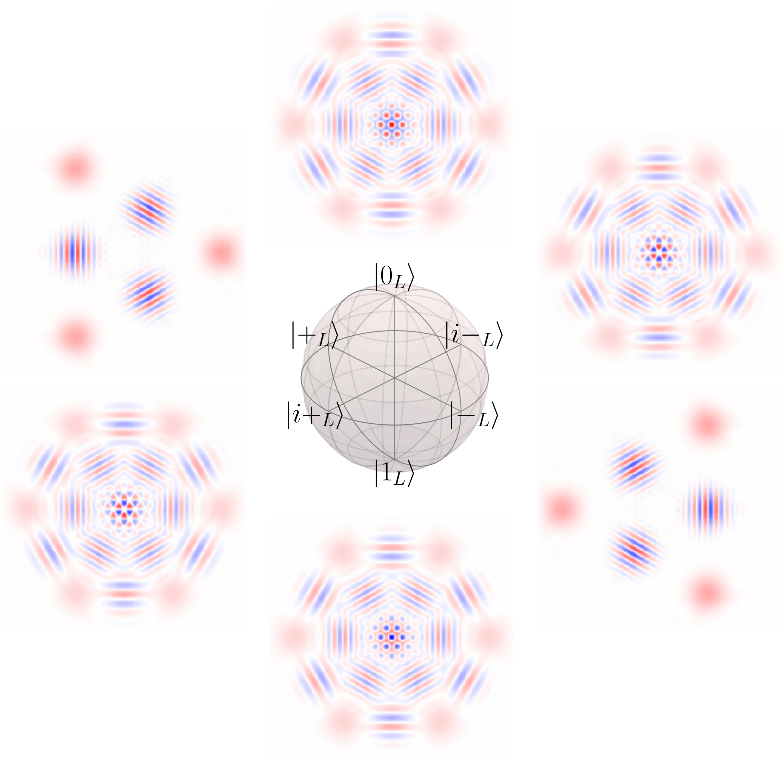


Figure 3.6: **Six-legged cat code logical Bloch sphere.** Stabilizer states of a generalized cat code, which corrects up to two photon loss events. Created from equation 3.70, with $d = 3$ and $\alpha = 4$.

used to address and manipulate them, as demonstrated in Rosenblum et al. (2018a).

3.7.2 Cat code generalizations

The cat code as presented consists of superpositions of 4 equally spaced coherent states. We can easily generalize to $2d$ equally spaced coherent states, for any value of d (Li et al., 2017).

$$|0_L\rangle = \frac{1}{\mathcal{N}} \sum_{k=0}^{2d-1} \left| e^{i\pi k/d} \alpha \right\rangle \tag{3.70}$$

$$|1_L\rangle = \frac{1}{\mathcal{N}} \sum_{k=0}^{2d-1} (-)^k \left| e^{i\pi k/d} \alpha \right\rangle \tag{3.71}$$

For $2d$ coherent states in the code words, we can correct up to $d - 1$ photon loss errors, essentially by defining code words which occupy only Fock state levels $|n\rangle$ with $n = 0$ mod $2d$ or $n = d$ mod $2d$ Fock state levels. We can see one example of these generalized codes in figure 3.6, where in order to protect against two photon loss events, we can promote the four-legged cat to a six-legged cat. This maintains a spacing of 3 photons between logical states, ensuring that single or double photon loss events remain in orthogonal spaces to the code space.

3.7.3 Numerically optimized codes

A more exotic class of codes can be found by a numerical optimization. The code words produced by this procedure are not known to be generated by any generic analytic representation. In order to find logical states $|\psi_i\rangle$ for $i \in \{0, 1\}$ which allow for the correction of error operators E_k for $E_k \in \{\boldsymbol{I}, \boldsymbol{a}\}$, one can consider the various inner products formed from the range of the error operators acting on the logical states

$$f_{ijkl} = \langle\psi_i| E_k^\dagger E_l |\psi_j\rangle . \tag{3.72}$$

The degree of violation of the error correction criteria $f_{ijkl} = \delta_{i,j} c_{k,l}$ is summarized by the cost function

$$c_1 = \sum_{k,l} |f_{00ij} - f_{11ij}|^2 + |f_{01ij}|^2. \tag{3.73}$$

In order to prefer lower occupation, the penalty

$$c_2 = \lambda_{\bar{n}} \sum_i \langle\psi_i| \boldsymbol{a}^\dagger \boldsymbol{a} |\psi_i\rangle , \tag{3.74}$$

is introduced with $\lambda_{\bar{n}} = 10^{-3}$. An optimization is considered successful if the value c_1 goes to zero as we relax $\lambda_{\bar{n}} \to 0$. Code words are produced by numerically optimizing over complex unit vectors the total cost:

$$\underset{\psi_0, \psi_1 \in \mathbb{C}^d}{\text{minimize}} \; c_1 + c_2. \tag{3.75}$$

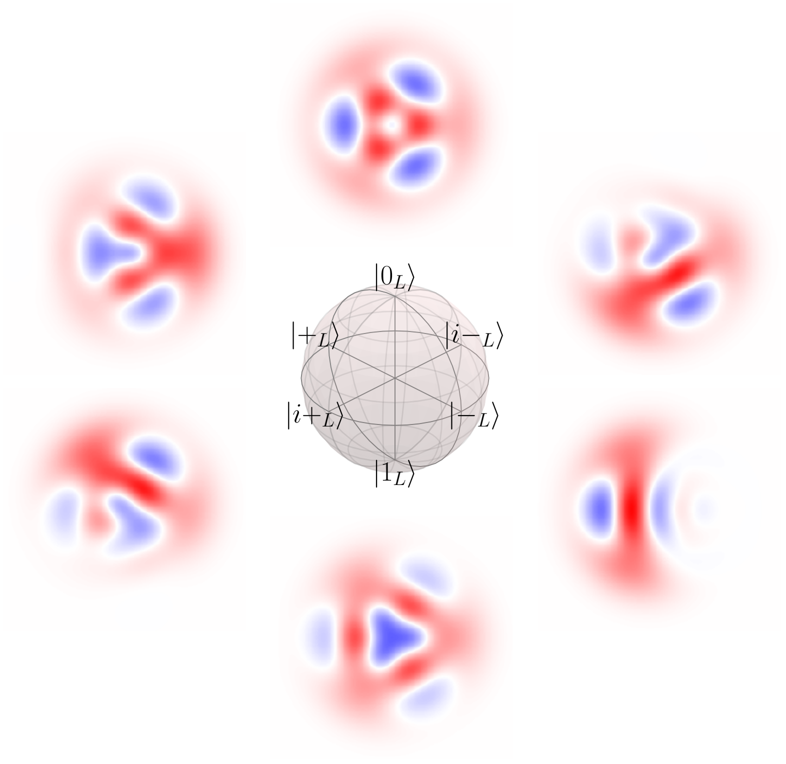


Figure 3.7: **Numerically optimized $\sqrt{17}$ code logical Bloch sphere.** Wigner functions for the six stabilizer states of the numerically identified "$\sqrt{17}$" code (equation 3.76)

Several of these codes have analytic expressions which have been found.[8] One such code is the "$\sqrt{17}$" code (Michael et al., 2016)

$$|0_L\rangle = \frac{1}{\sqrt{6}}\left(\sqrt{7-\sqrt{17}}\,|0\rangle + \sqrt{\sqrt{17}-1}\,|3\rangle\right) \tag{3.76}$$

$$|1_L\rangle = \frac{1}{\sqrt{6}}\left(\sqrt{9-\sqrt{17}}\,|1\rangle - \sqrt{\sqrt{17}-3}\,|4\rangle\right) \tag{3.77}$$

The states of this code word are visualized in figure 3.7. It is interesting to note that the error syndrome for this code is *not* photon number parity. This set of states manages to be smaller than the binomial code (3.66) by enforcing condition 3.20 in a more subtle way. Instead of ensuring that $|0_L\rangle$ and $\boldsymbol{a}\,|1_L\rangle$ have disjoint support in the photon number basis, the orthogonality relies on the relative signs of the amplitudes. In fact, the use of negative signs in the relative amplitudes of the photon number states is the only known defining feature of

[8] These expressions have been found essentially by performing a reverse lookup from the floating-point representation

these codes.

3.7.4 GKP codes

The oldest of the bosonic encodings is the code named after its authors, Gottesman, Kitaev, and Preskill (2001), termed the GKP, or "grid" code. The motivation for this code comes from considering a completely different model for the errors that the system undergoes. Instead of thinking about photons leaking into the environment, one considers that the basic control mechanism of an oscillator is displacement. What happens if there is noise on this control line? The answer is small random displacements. The GKP code is designed so that, if one performs an unknown displacement on the system which is within a small enough magnitude, then one can measure some properties of the system which reveals this displacement, without revealing any information about the encoded qubit. The properties which one needs to measure are $(\boldsymbol{a} + \boldsymbol{a}^\dagger) \bmod \sqrt{2\pi}$ and $i(\boldsymbol{a} - \boldsymbol{a}^\dagger) \bmod \sqrt{2\pi}$. Equivalently, one can measure the eigenvalues of the displacements $D_{\sqrt{2\pi}}$ and $D_{i\sqrt{2\pi}}$. In fact, measuring these two values completely determines the state of the oscillator, up to the encoded degree of freedom. That means that one can prepare GKP states simply by measuring the error syndromes. If one did this perfectly, the result would be seen in the Wigner function as an infinite grid of points in phase space. It suffices in practice to use finite-precision measurements, which results in a grid tapered by a Gaussian envelope (figure 3.8). One could perform these measurements in cQED settings using phase-estimation protocols on the conditional displacement operator (Terhal and Weigand, 2016). One might think that the performance of such codes on the photon loss model would be inferior to that of codes which were explicitly designed to handle photon loss, rather than the more general class of errors. However, numerical investigations of photon loss over finite times, a more realistic error model was analyzed in (Albert et al., 2018), which showed that GKP codes have an advantage in certain parameter regimes. More recently it was shown that A variant of the traditional GKP code which uses a hexagonal, rather than square, lattice can be shown to be "optimal" code for photon loss under average photon number occupation constraints (Noh et al., 2019). The ability to correct photon loss events in a code which corrects for displacements, comes from the ability to map photon loss into a linear combination of small

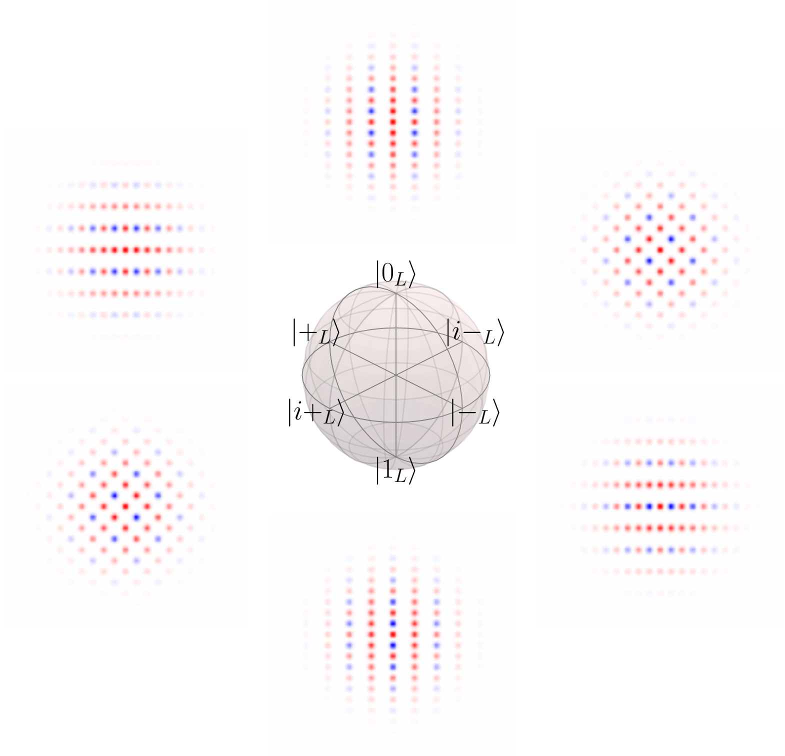


Figure 3.8: Wigner functions for the six stabilizer states of the logical Bloch sphere using GKP code words having envelope parameter $\Delta = 0.25$ (see definition from Albert et al. (2018)).

displacements

$$\boldsymbol{a} = \lim_{\epsilon \to 0} \frac{1}{4\epsilon} \left(\boldsymbol{D}_{\epsilon} + i\boldsymbol{D}_{i\epsilon} - \boldsymbol{D}_{-\epsilon} - i\boldsymbol{D}_{-i\epsilon} \right) \tag{3.78}$$

Chapter 4

Numerical quantum optimal control

In many fields of science we are often faced with two related problems: analysis and synthesis. The former starts from a set of assumptions about a system regarding its structure and dynamics, and proceeds to calculate some effects, how it will behave, how it transforms inputs to outputs, etc. The latter starts with desired behavior or desired transformation function, and proceeds to construct a system or set of assumptions that would produce such effects. In quantum mechanics the analysis problem is very well known. Typically all one needs to do is write down the appropriate equation, whether that be one of Schrödinger, Lindblad or Langevin form, and integrate in order to predict the system dynamics.

What is the "quantum synthesis" problem then? There are many candidates depending on which parts of the system one considers fixed and which one considers mutable. Here we are going to consider the synthesis of *control fields*. We assume we have a quantum system, whose Hamiltonian can be written in the following form:

$$\boldsymbol{H}\left(\vec{\epsilon}(t)\right) = \boldsymbol{H}_0 + \sum_k \epsilon_k(t)\boldsymbol{H}_k \tag{4.1}$$

The component we shall "synthesize" is the time-dependent control field $\vec{\epsilon}(t)$. This is a problem tailored to the tools we have available. Since the introduction of the arbitrary waveform generator (AWG) as a control instrument, we have the ability to implement arbitrary drive fields in order to control our systems, limited only by the amplitude and bandwidth the AWG provides. The question of *optimal control* has become salient, given the unlimited range of possibilities in terms of feasible control. In order to establish "optimality" we will need a specific method for evaluating the desirability of any given instance of the control field, i.e. a function $f(\vec{\epsilon}(t))$ that we seek to maximize.

As has been shown by the rapid rise and development of neural network machine learning algorithms, even an extremely large number of parameters can be efficiently optimized so long as two conditions hold: there exists an efficient means of calculating gradients of the target cost function with respect to the parameters, and that sub-optimal local minima are sufficiently unlikely, or at least are approximately equivalent to the true global minimum. With quantum optimal control we face a similar problem when considering the space of all possible control fields, since the number of parameters representing a technically feasible control field can be quite large. However, because there are efficient methods of computing the gradient, and because the quantum control problem in the appropriate limit contains few "traps," it is in practice tractable (Rabitz et al., 2005).

This gradient based approach to pulse optimization was first identified by Khaneja et al. (2005) and called gradient ascent pulse engineering (GRAPE). It was immediately apparent that this method had many possible applications and extensions which could be achieved by modifying the cost function while maintaining differentiability. We will discuss the basic GRAPE problem definition as well as some variants which make the algorithm a flexible tool for designing control sequences. Initially derived in an NMR context, GRAPE, and related methods, have found use in a wide variety of quantum systems and applications (Dolde et al., 2014; Anderson et al., 2015). Since GRAPE crucially depends on the model of the system, its successful application is powerful evidence that the Hamiltonian used accurately captures the system dynamics over a broad range of driving conditions.

4.1 Defining the problem

The simplest problem that can be tackled by GRAPE is that of state transfer, i.e. the operation should take some specified initial state $|\psi_{\text{init}}\rangle$ to a corresponding target state $|\psi_{\text{targ}}\rangle$. We need a differentiable notion of how close any control pulse $\vec{\epsilon}$ is to achieving this goal. For this purpose we use the fidelity, and set our task to be one of maximizing this fidelity:

$$\underset{\epsilon(t)}{\text{maximize}} \;\; \mathcal{F}\left(\epsilon(t)\right) \tag{4.2}$$

$$\mathcal{F}(\epsilon(t)) = \left|\langle\psi_{\text{targ}}|\, \boldsymbol{U}(T, \epsilon(t))\, |\psi_{\text{init}}\rangle\right|^2 \tag{4.3}$$

where the unitary U defined by the waveforms $\vec{\epsilon}(t)$ is given by the time-ordered exponential of the Hamiltonian up to some final time T,

$$\boldsymbol{U}(T, \epsilon(t)) = \mathcal{T} \exp\left(-\int_0^T \mathrm{d}t\, \boldsymbol{H}\left(\epsilon(t)\right)\right). \tag{4.4}$$

To make the problem numerically tractable, $\epsilon(t)$ is represented as a piecewise constant function with $N = T/\delta t$ steps of length δt. We denote the vector of parameters describing this piecewise constant function as $\vec{\epsilon}$.

$$\boldsymbol{U}\left(\vec{\epsilon}\right) = \boldsymbol{U}_N \boldsymbol{U}_{N-1} \cdots \boldsymbol{U}_2 \boldsymbol{U}_1 \tag{4.5}$$

$$\boldsymbol{U}_k = \exp\left(\frac{i\delta t}{\hbar} \boldsymbol{H}(\vec{\epsilon}(k\delta t)\right) \tag{4.6}$$

The time step δt can be set to the corresponding time resolution of the AWG, or can be set in accordance with the needed bandwidth (see figure 4.1).

4.2 Calculating the gradient

The calculation of the fidelity itself is a moderately computationally intensive feat. The easiest way to measure the gradient of a function is via the finite difference method, which is to simply apply a small perturbation in every direction in control space, and measure the change in the fidelity. For N control parameters, this would require N evaluations of the fidelity. For a realistic simulation of a long pulse, N could be extremely large, numbering thousands of parameters.

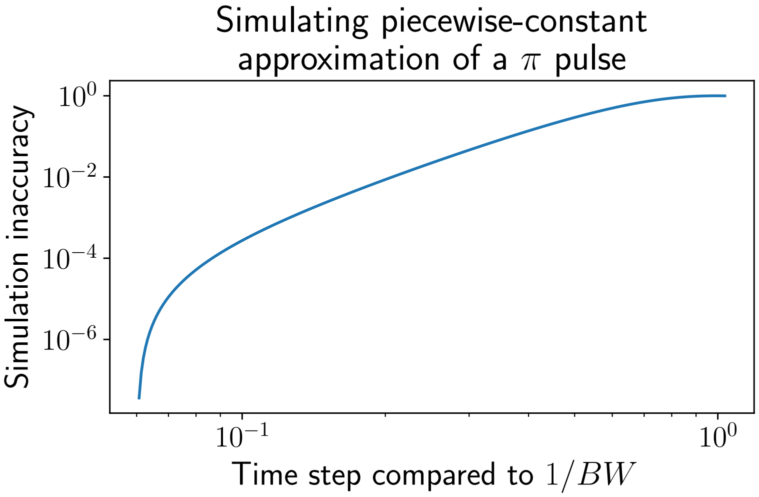


Figure 4.1: **Inaccuracy introduced by piecewise-constant approximation**. We consider the effects of replacing a smooth pulse with a piecewise constant approximation, in this case a Gaussian π pulse with detuning $5/\sigma$. The error introduced by this approximation is plotted versus the choice of time step δt. The result indicates that one should have roughly an order of magnitude separation between the time step and the desired pulse bandwidth in order to avoid errors introduced by filtering, interpolation or finite output bandwidth.

This overhead is preventable if we use a smarter approach to calculating the gradient.

The operation fidelity will typically break into one or multiple computations of overlap integrals between some propagated initial state:

$$c = \langle \psi_{\text{targ}} | \psi_{\text{final}} \rangle \tag{4.7}$$

$$= \langle \psi_{\text{targ}} | \boldsymbol{U}_N \cdots \boldsymbol{U}_1 | \psi_{\text{init}} \rangle \tag{4.8}$$

Where each of the $\boldsymbol{U}$ are propagators corresponding to a time slice of width δt where the Hamiltonian is time independent, and thus can be calculated via simple matrix exponentiation.

$$\boldsymbol{U}_k = \exp\left(\frac{i\delta t}{\hbar}\boldsymbol{H}(\epsilon_k)\right) \tag{4.9}$$

The derivative with respect to any particular control parameter, say ϵ_k, can be found by simply differentiating the relevant term in equation 4.8 which yields the following expression:[1]

$$\partial_{\epsilon_k} c = \langle \psi_{\text{targ}} | \boldsymbol{U}_N \cdots \boldsymbol{U}_{k+1} \left(\partial_{\epsilon_k} \boldsymbol{U}_k \right) \boldsymbol{U}_{k-1} \cdots \boldsymbol{U}_1 | \psi_{\text{init}} \rangle \tag{4.10}$$

In the limit of small δt, the derivative of the time step propagator can be approximated with the following:

$$\partial_{\epsilon_k} \boldsymbol{U}_k \approx \frac{i\delta t}{\hbar} \left(\partial_{\epsilon_k} \boldsymbol{H} \right) \boldsymbol{U}_k, \tag{4.11}$$

but in practice we can also use an exact form of the derivative, which is derived in appendix A.7 as well as by Najfeld and Havel (1995). If we naïvely calculate the cost of calculating $\partial_{\epsilon_k} c$ and multiply by the number of independent components (that is indices k), we get our N^2 cost as was the case with a finite-differences approach. However, by examining the expression, we see that we can get all components together much more cheaply by caching some intermediate results (Khaneja et al., 2005). Plugging this back into 4.10, we can see that the calculation can be broken into three components:

$$\partial_{\epsilon_k} c = \frac{i\delta t}{\hbar} \underbrace{\langle \psi_{\text{targ}} | \boldsymbol{U}_N \cdots \boldsymbol{U}_{k+1}}_{\left\langle \psi_{\text{bwd}}^{(k+1)} \right|} \left(\partial_{\epsilon_k} \boldsymbol{H}(\epsilon_k) \right) \underbrace{\boldsymbol{U}_k \cdots \boldsymbol{U}_1 | \psi_{\text{init}} \rangle}_{\left| \psi_{\text{fwd}}^{(k)} \right\rangle}, \tag{4.12}$$

that is to say a matrix element calculation on the derivative of the Hamiltonian $\partial_{\epsilon_k} \boldsymbol{H}$. This calculation involving two states: the forward-propagated initial state $\left| \psi_{\text{fwd}}^{(k)} \right\rangle$ as well as the reverse propagated target state $\left| \psi_{\text{bwd}}^{(k+1)} \right\rangle$. We can compute and store in memory each of these trajectories using the following rules:

$$\left| \psi_{\text{fwd}}^{(k)} \right\rangle = \begin{cases} |\psi_{\text{init}}\rangle & k = 0 \\ \boldsymbol{U}_k \left| \psi_{\text{fwd}}^{(k-1)} \right\rangle & \text{otherwise} \end{cases} \tag{4.13}$$

$$\left| \psi_{\text{bwd}}^{(k)} \right\rangle = \begin{cases} |\psi_{\text{targ}}\rangle & k = N+1 \\ \boldsymbol{U}_k^\dagger \left| \psi_{\text{bwd}}^{(k+1)} \right\rangle & \text{otherwise} \end{cases} \tag{4.14}$$

[1] We treat the case of a single control field, for the purposes of simplicity of notation. The generalization to multiple control fields is straightforward.

In terms of the basic operation of matrix exponential vector multiplication[2], it takes N calls to evaluate the overlap c and only $2N$ calls in order to get the entire gradient as well! This is a drastic improvement over the N^2 steps required for a naïve approach.

Once we have a method of computing the cost function (which consists of the fidelities along with other pulse dependent penalty terms which will be discussed in detail in the following sections) the next step is to choose an algorithm for actually performing the function minimization. Specifically, the algorithm must use the knowledge of the function value and gradient in order to propose new points in control space to evaluate. Luckily there has been quite a bit of work done in this area, which allows for use of off-the-shelf function minimization routines. There are two main classes of minimization routines: Line-search methods and trust-region methods (Nocedal and Wright, 2000). In line search methods, one alternates between picking a direction in parameter space, radiating from our current point, and subsequently performing a 1-d minimization protocol to find the minimum along this line. In trust region methods, we develop a (usually quadratic) model of the cost function, assume its validity within a "trust region" consisting of a ball of some radius centered on our current point, and move to the model's predicted minimum within this trust region.

The simplest method is basic gradient descent, which is a line search method where the direction chosen is simply the gradient at the point. However, this is rarely the best choice. The Newton method chooses directions using not only the gradient, but also the Hessian matrix of second derivatives of the function. However, we need not invoke the cost of calculating the Hessian (an inherently $O(N^2)$ operation simply by the size of the Hessian). We can get much of the benefit of the Newton method using so-called quasi-Newton methods, which seek to build a model of the Hessian using knowledge of the history of the gradient. Out of these algorithms, the L-BFGS method (Byrd et al., 1995) distinguishes itself by never explicitly constructing the Hessian, which would be memory intensive, but rather only keeps a long enough history of the gradient in order to evaluate approximate Hessian-vector products. Detailed comparisons of these quasi-newton algorithms suggest that BFGS and its limited memory variant are the most performant on GRAPE style pulse optimization problems (de Fouquieres et al., 2011). There

[2] This operation is called `expm_multiply` in the `scipy` linear algebra library, and can be done quicker than separately computing the matrix exponential and doing matrix-vector multiplication.

are compelling reasons to believe that even better convergence can be achieved by computing not only the gradient but the Jacobian of the induced propagator $\nabla_{\vec{\epsilon}} \boldsymbol{U}(\vec{\epsilon})$ in combination with a trust-region approach (de Fouquieres, 2012), although my attempts at replicating this performance enhancement were unsuccessful.

4.3 Cost function variations

We can consider many variations on the cost function defined in equation 4.3. The most obvious of which is to consider the action of the operation not on a single state, but on multiple states. There are two primary ways we can synthesize the effect on multiple states, coherently (i.e. enforcing a relative phase between state transfers)

$$\mathcal{F}(\vec{\epsilon}) = \left| \sum_k \left\langle \psi^{(k)}_{\textsf{targ}} \right| \boldsymbol{U}(\vec{\epsilon}) \left| \psi^{(k)}_{\textsf{init}} \right\rangle \right|^2 , \tag{4.15}$$

or incoherently (i.e. caring only of the probability for each state to end up in the correct target state)

$$\mathcal{F}(\vec{\epsilon}) = \sum_k \left| \left\langle \psi^{(k)}_{\textsf{targ}} \right| \boldsymbol{U}(\vec{\epsilon}) \left| \psi^{(k)}_{\textsf{init}} \right\rangle \right|^2 . \tag{4.16}$$

In the limit where the number of states equals the system dimension, the coherent multi-state fidelity (4.15) is equivalent to the unitary fidelity:

$$\mathcal{F}(\vec{\epsilon}) = \left| \mathrm{Tr}\left\{ \boldsymbol{U}(\vec{\epsilon}) \boldsymbol{U}^{\dagger}_{\textsf{targ}} \right\} \right|^2 \tag{4.17}$$

Optimizing a full unitary operation is sensible for qubits, or systems of a few qubits, but in the cavity manipulations to be performed in the following sections, this is not usually a sensible operation, since doing operations involving states at the border of the truncation (i.e. the maximum photon number state considered in the simulation), would typically require a larger truncation to simulate accurately (driving $|n_{\textsf{ph}}\rangle$ will almost always in part bring us partially to $|n_{\textsf{ph}} + 1\rangle$).

4.3.1 Open system GRAPE

We can generalize the problem in a different way, by considering the effect of dissipation and decoherence. In this case we replace the Schrödinger equation

$$\partial_t \left|\psi\right\rangle = i\boldsymbol{H}(t) \left|\psi\right\rangle, \tag{4.18}$$

with the Liouville master equation

$$\partial_t \boldsymbol{\rho} = \boldsymbol{L}(\boldsymbol{\rho}), \tag{4.19}$$

where the time evolution is generated by the Liouvillian

$$\begin{aligned} \boldsymbol{L}(\boldsymbol{\rho}) &= i[\boldsymbol{H}, \boldsymbol{\rho}] + \sum_k D[\boldsymbol{A}_k](\boldsymbol{\rho}) && (4.20) \\ &= i[\boldsymbol{H}, \boldsymbol{\rho}] + \sum_k \boldsymbol{A}_k \boldsymbol{\rho} \boldsymbol{A}_k^\dagger - \frac{1}{2}\left(\boldsymbol{A}_k^\dagger \boldsymbol{A}_k \boldsymbol{\rho} + \boldsymbol{\rho} \boldsymbol{A}_k^\dagger \boldsymbol{A}_k\right) && (4.21) \end{aligned}$$

The Liouvillian can be considered itself to be a simple matrix via *vectorization*, which uses the isomorphism between the space of $d \times d$ matrices $\mathbb{C}^{d\times d}$ and the product space $\mathbb{C}^d \otimes \mathbb{C}^d \simeq \mathbb{C}^{d^2}$.

$$\boldsymbol{\rho} \in \mathbb{C}^{d\times d} \rightarrow |\rho\rangle\!\rangle \in \mathbb{C}^d \otimes \mathbb{C}^d \tag{4.22}$$

$$\boldsymbol{\rho}_{ij} = \langle\!\langle i, j|\rho\rangle\!\rangle \tag{4.23}$$

$$(\boldsymbol{A}\boldsymbol{\rho}\boldsymbol{B})_{ij} = \sum_{k,l} \boldsymbol{A}_{ik}\boldsymbol{\rho}_{kl}\boldsymbol{B}_{lj} \tag{4.24}$$

$$\boldsymbol{A}\boldsymbol{\rho}\boldsymbol{B} \rightarrow \boldsymbol{A} \otimes \boldsymbol{B}^T |\rho\rangle\!\rangle \tag{4.25}$$

Putting this together allows us to rewrite the Liouvillian in explicit matrix form:

$$\boldsymbol{L} \rightarrow i\left(\boldsymbol{H} \otimes \mathbb{I} - \mathbb{I} \otimes \boldsymbol{H}^T\right) + \sum_k \boldsymbol{A}_k \otimes \boldsymbol{A}_k^* - \frac{1}{2}\left(\boldsymbol{A}_k^\dagger \boldsymbol{A} \otimes \mathbb{I} + \mathbb{I} \otimes \boldsymbol{A}_k^\dagger \boldsymbol{A}_k\right) \tag{4.26}$$

The point is that we can cast the Liouville equation 4.19 in exactly the same form as the Schrödinger equation 4.18

$$\partial_t |\rho\rangle\!\rangle = \boldsymbol{L}|\rho\rangle\!\rangle. \tag{4.27}$$

The major differences introduced here are first that the Hilbert space has transformed from $\mathbb{C}^d$ to the larger $\mathbb{C}^{d^2}$, and second that the generator of evolution is no longer skew-Hermitian, and therefore algorithms that rely on the ability to diagonalize the Hamiltonian by a unitary transformation will fail. With these caveats aside however, we can compute the evolution under the master equation as well as its gradient with respect to the control parameters as in the lossless case.

4.3.2 Robust control

Often times, the control system one is dealing with may have uncertainties in its parameters. These uncertainties can arise either because it is difficult to characterize these parameters accurately, or because these parameters are not stable in time, but rather fluctuate. This uncertainty leads to model inaccuracy, and therefore infidelity in the final operation. However, we know it is possible to construct protocols that are robust to variations in parameters. Most famous of these is the "spin echo" experiment first discovered by Hahn (1950), in which a spin's Bloch vector under an uncertain precession frequency, can be refocused at time t by the application of a π pulse at time $t/2$. This technique has been extended to a variety of applications under the heading of "dynamical decoupling" (Viola et al., 1999). While these techniques are powerful and general, numerical optimization is more general still, and requires fewer assumptions and mathematical finesse.

One can define a robustness metric for a pulse in several ways. If this metric is differentiable, and efficiently computable, then we can attempt to perform gradient descent. While the global optimum may be unattainable, we can always at least seek something locally optimal. If the Hamiltonian depends on some parameter θ, we may attempt to become insensitive to this parameter by minimizing $\partial_\theta \mathcal{F}$. However, a more pragmatic approach is to sample. Randomly draw M values θ_k from the predicted distribution of θ, and optimize simultaneously for all values:

$$\underset{\epsilon(t)}{\text{maximize}} \sum_k^M \mathcal{F}\left(\epsilon(t), \theta_k\right) \tag{4.28}$$

The computational overhead is the factor of M. Random sampling is preferable to drawing the points uniformly, as the imposed structure that comes from a regular spacing of θ_k can be exploited by the algorithm to produce results that seem robust on the sample set, but are not valid over the entire region.

4.3.3 Gauge degrees of freedom

Sometimes, rather than add constraints or additional objectives to our problem, we would prefer to remove unnecessary specifications, to allow the pulse more freedom, and in turn, hopefully let it be done in the shortest time possible. To take a concrete example, because of the nature of operating in a rotating frame, the difference between a cavity or qubit operation along one quadrature or another is only a matter of changing the phase of our pulses. To put it another way, we can implement phase space rotations ($\boldsymbol{\sigma}_z$ for qubits, $\boldsymbol{a}^\dagger \boldsymbol{a}$ more generally) "in software" simply by adjusting the phase of all subsequent pulses (McKay et al., 2017). This means that we can make our pulse optimization problem easier by specifying the desired final state only up to a final phase space rotation. We can formalize this as follows:

$$\underset{\vec{\epsilon},\theta}{\text{maximize}} \quad \left| \langle \psi_{\text{targ}} | \, e^{i\theta \boldsymbol{a}^\dagger \boldsymbol{a}} \boldsymbol{U}(\vec{\epsilon}) \, | \psi_{\text{init}} \rangle \right|^2 \tag{4.29}$$

We introduce the additional parameter θ, which gives the optimization algorithm another degree of freedom, by which to adjust the target to match the implementation, rather than the other way around. More generally, we can specify a list of "gauge degrees of freedom," $\{\boldsymbol{A}_k\}$, and perform the following maximization:

$$\underset{\vec{\epsilon}(t),\vec{\theta}}{\text{maximize}} \quad \left| \langle \psi_{\text{targ}} | \, e^{i \sum_k \theta_k \boldsymbol{A}_k} \boldsymbol{U}(\vec{\epsilon}(t)) \, | \psi_{\text{init}} \rangle \right|^2 \tag{4.30}$$

Another useful class of degrees of freedom is a "subsystem" degree of freedom. In this case we have a Hilbert space with a Kronecker product structure $\mathcal{H} = \mathcal{H}_A \otimes \mathcal{H}_B$, and we only care about the state in one part, say in $\mathcal{H}_A$. In this case we want to allow the target state in $\mathcal{H}_B$ to be whatever makes the task the easiest. To do so, we can use the set of gauge operators

$\{\boldsymbol{I}_A \otimes |i_B\rangle\langle j_B|\}_{i,j\in[\dim \mathcal{H}_B]}$.[3] This set of operators allows the optimization to set whatever targets states in subsystem B it would like to make its optimization easiest. This can be useful when, for instance, we are performing a measurement of some property of the cavity using the transmon, and have no intention of using the cavity state after the measurement.

4.4 Constraints and penalties

The optimization problem defined by equation 4.2 is generally underdetermined, i.e. there are many solutions $\vec{\epsilon}(t)$ which achieve equally high fidelities. Therefore, we can add additional terms to the optimization cost function, such that the resulting solution optimizes against several other desiderata. For a set of constraints on the solution $\{g_i \geq 0\}$, where ideally $g_i\left(\boldsymbol{\epsilon}(t)\right) = 0$, we can associate a Lagrange multiplier λ_i, and modify our optimization to read:

$$\underset{\vec{\epsilon}}{\text{maximize}}\;\; \mathcal{F}\left(\vec{\epsilon}\right) - \sum_i \lambda_i g_i\left(\vec{\epsilon}\right) \tag{4.31}$$

The values λ_i are chosen by trial-and-error, set to be just large enough that the violation of the constraint upon termination is within acceptable levels. If the λ_i are set too large, it is possible to distort the control landscape significantly enough to introduce inescapable local optima, preventing the optimization from succeeding. In this case, it can be desirable to perform the optimization first with a lower value of λ_i, and only ratcheting it up when the correct "basin" has been found (Riviello et al., 2015).

4.4.1 Limiting the pulse amplitude

There are several reasons why we might want to limit the pulse amplitude. The first is that, without introducing additional amplifiers, the output power of our AWG is limited, and thus to be feasible, we need $\epsilon(t) \leq \epsilon_{\text{max}}$ for all t. In other words, we would like a *hard cut-off*. We can achieve this cutoff in one of two ways. We can either employ an optimization algorithm which naturally allows for such constraints, or we can use an alternate parameterization to represent

[3] It can be convenient to work with the Hermitian operator basis instead, substituting $(|i_B\rangle\langle j_B|, |j_B\rangle\langle i_B|) \rightarrow (|i_B\rangle\langle j_B| + \text{h.c.}, i(|i_B\rangle\langle j_B| - \text{h.c.}))$.

the pulse. The former depends on the availability of a suitable algorithm implementing inequality constraints[4]. The latter can be implemented by changing the optimization problem:

$$\underset{\vec{x}}{\text{maximize}} \;\; \mathcal{F}\left(\vec{\epsilon}(\vec{x})\right), \tag{4.32}$$

where the pulse $\vec{\epsilon}$ is constructed from the parameters $\vec{x}$ as

$$\epsilon_k = \epsilon_{\text{max}} \tanh(x_k). \tag{4.33}$$

In order to make this formulation compatible with gradient descent, we need to transform $\nabla_{\vec{\epsilon}}\mathcal{F}$ into $\nabla_{\vec{x}}\mathcal{F}$

$$\partial_{x_k}\mathcal{F} = \frac{\epsilon_{\text{max}}\mathcal{F}}{\cosh(x_k^2)}\partial_{\epsilon_k} \tag{4.34}$$

. We may also wish to prefer to stay well below the AWG maximum output power for the majority of the pulse for other reasons. For instance, the active components in the lines, such as mixers and amplifiers, each of which have non-linearities, which become more relevant, and difficult to model, at higher powers. While we can try to compensate for this behavior (see section 4.8), we can also try to minimize its relevance by using lower powers. In this case we can use a penalty realizing a "soft" cut-off relative to some lower amplitude $\epsilon_{\text{softmax}}$:

$$g_{\text{amp,nonlin}}(\vec{\epsilon}) = \sum_k \left(e^{|\epsilon_k|^2/|\epsilon_{\text{softmax}}|^2} - 1\right) \tag{4.35}$$

Finally, because each pulse induces a heat load proportional to its integrated power, we might wish to minimize this undesired side effect. To address this, a simple penalty will suffice:

$$g_{\text{amp,lin}}\left(\epsilon(t)\right) = \sum_k |\epsilon_k|^2 \tag{4.36}$$

4.4.2 Limiting the bandwidth

As with amplitude there are a variety of reasons one might want to constrain the bandwidth of a pulse. We again must consider the maximum available bandwidth of the AWG, the model

[4] The `L-BFGS-B` method supplied by `scipy.optimize` supports these types of constraints

uncertainty at higher detunings from resonance, as well as nonlinearities and dispersion of the transfer function. We can similarly take a variety of approaches to constraining the bandwidth, from a hard cutoff, to a soft cutoff, to a linear frequency-dependent penalty. To implement the hard cutoff, we reparameterize as in equation 4.32, however we write the pulse this time as the inverse discrete Fourier transform (DFT) of the parameters (Motzoi et al., 2011),

$$\vec{\epsilon} = (\mathsf{DFT})^{-1}\,\vec{x} \tag{4.37}$$

We can constrain certain components of the frequency representation, corresponding to frequencies above the cutoff ω_{max} to be identically zero in this representation,[5]

$$|\omega_k| > \omega_{\mathsf{max}} \Rightarrow x_k = 0 \tag{4.38}$$

We must similarly propagate the derivative calculation through this reparameterization:

$$\vec{\nabla}_{\vec{x}} = (\mathsf{DFT})\,\vec{\nabla}_{\vec{\epsilon}} \tag{4.39}$$

We can allow for penalty formulations as well, by penalizing, in either a linear or non-linear way, the magnitude of the derivative of the pulse. Since the pulse is piecewise constant, we replace the derivative with the difference between adjacent points.

$$g_{\mathsf{deriv,lin}} = \sum_k |\epsilon_{k+1} - \epsilon_k|^2 \tag{4.40}$$

$$g_{\mathsf{deriv,nonlin}} = \sum_k e^{|\epsilon_{k+1} - \epsilon_k|^2/\delta\epsilon^2_{\mathsf{softmax}}} - 1 \tag{4.41}$$

The choice between these two options comes down to whether or not there exists a relevant frequency scale $\epsilon_{\mathsf{softmax}}$, or whether it is a general preference for lower bandwidth pulses.

[5] The hard cutoff in frequency and the hard cutoff in amplitude are incompatible with each other. It is not known to us whether there exists a representation which guarantees a hard cutoff in both frequency and amplitude.

4.5 Limiting the intermediate photon number

Since computer memory is finite, we are forced to choose a photon number truncation n_{ph} such that the operator $\boldsymbol{a}$ becomes a $n_{\text{ph}} \times n_{\text{ph}}$ matrix. When we do this, we are in effect replacing our infinite-dimensional oscillator with a finite-dimensional qudit. This replacement is only valid if all of the system dynamics relevant for the desired state transfers occurs within the $\{|0\rangle, \ldots, |n_{\text{ph}} - 1\rangle\}$ subspace. For generic applied drives this is not the case. In order to enforce this property, we modify the optimization problem to find a solution which operates identically under several different values of n_{ph}. Writing the fidelity as computed with a truncation n_{ph} as $\mathcal{F}_{n_{\text{ph}}}$, we have:

$$\underset{\tilde{\boldsymbol{\epsilon}}(\omega)}{\text{maximize}} \quad \left(\sum_k \mathcal{F}_{n_{\text{ph}}+k}\left(\boldsymbol{\epsilon}(t)\right)\right) - \left(\sum_i \lambda_i g_i\left(\boldsymbol{\epsilon}(t)\right)\right) \tag{4.42}$$

To enforce that the behavior is identical in the different truncations, we add the penalty term

$$g_{\text{discrepancy}}\left(\boldsymbol{\epsilon}(\boldsymbol{t})\right) = \sum_{k_1 \neq k_2} \left(\mathcal{F}_{n_{\text{ph}}+k_1}\left(\boldsymbol{\epsilon}(t)\right) - \mathcal{F}_{n_{\text{ph}}+k_2}\left(\boldsymbol{\epsilon}(t)\right)\right)^2 \tag{4.43}$$

The choice of n_{ph} determines the maximum photon number population that can be populated during the pulse, and figures in determining the minimum time necessary for the operation (faster pulses can be achieved with higher n_{ph}).

A more recently developed, and more direct, method is to add a penalty term for *any* occupation of the final photon state ($|n_{\text{ph}} - 1\rangle$) in the truncated Hilbert space at any time:

$$g_{\text{trajectory}} = \sum_{k=1}^{N} \left|\left\langle n_{\text{ph}} - 1 \middle| \psi_{\text{fwd}}^{(k)} \right\rangle\right|^2 \tag{4.44}$$

A naïve analysis of the complexity of calculating the gradient for this penalty seems to yield a scaling behavior of N^2, as it involves the computation of N terms (each k) where the k-th term has complexity of $k = O(N)$, as it mirrors the computation of the fidelity itself (equation 4.8). However, as was shown in Leung et al. (2017), by clever application of the backpropagation method, it is possible to design an approach with $O(N)$ scaling.

4.6 Troubleshooting optimization convergence

One of the first difficulties one might encounter when attempting to prepare optimal control pulses is difficulties in convergence. This could manifest as a failure of the search algorithm to find an improvement on the initial guess, or convergence to the zero-amplitude identity sequence, or perhaps convergence to a trivial operation (such as a single mode rotation or displacement). In this case there are several things one should check for.

1. Check that the time given $T = N\delta t$ is appropriate, and is specified in units that are consistent with the units specifying the Hamiltonian. For instance, if the Hamiltonian is specified in GHz, then the time step should be in units of ns.

2. Ensure constraints are not too stringent. If the optimization is failing, a good first step is to completely remove all constraints and penalties, and make sure that the algorithm works in this context before re-introducing them.

3. Ensure that the starting guess is sufficiently "random." If the initial guess is too close to a special point, such as the identity operation, the gradient can become vanishingly small, below machine precision. To overcome this, increase the amplitude of one's initial guess.

4. Ensure that the algorithm is being patient enough. Gradient based search algorithms usually have termination conditions specified in terms of the norm of the gradient. It is often necessary to lower the gradient norm threshold for termination to ensure that it does not give up [6].

5. Ensure that you are being patient enough. It can often appear that an algorithm is stuck, as it quickly finds some way to produce partial fidelity and then seems to peter out. It may take many iterations of low fidelity gain steps in order to find the correct "direction" in control space to take. Once this direction is found, then the fidelity gain can increase rapidly, resulting in a characteristic "double-sigmoid" curve shape for fidelity-vs-iteration.

[6] `gtol` in `scipy.minimize`

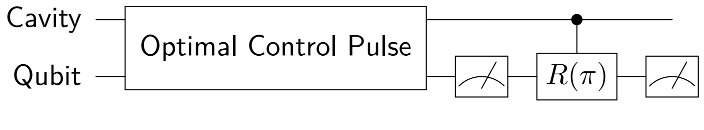
4.7 What if it doesn't work?: Debugging optimal control pulses

A frequent objection to the use of optimal control pulses is that they seem impossible to debug. This is in contrast with constructive approaches, where each component of the pulse sequence should effect a known operation, which can be tested in isolation. Additionally, each component typically contains only a few parameters, each of which can be empirically tuned. When the combined operation fails to perform, we have steps to take in order to try to identify what is going wrong. Optimal control pulses, on the other hand, have no straightforward decomposition into component parts, and have far too many parameters to optimize empirically. However, a methodical approach can still be employed to systematically eliminate potential problems.

It is helpful to consider the task of making optimal control pulses not as a search for any one particular operation, but as the construction of a *system* which maps desired operations to instantiated pulses. Only when this system is approximately correct should one move on to the particular operation of interest. In order to go from scratch to an approximately correct optimal control system, one should begin by designing pulses which are as *simple as possible to verify*. This allows one to quickly check whether a change to the system improved performance or not. For instance, in the transmon-cavity system, a simple diagnostic pulse is one which produces a single photon state:

$$|0\rangle \otimes |g\rangle \longrightarrow |1\rangle \otimes |g\rangle \tag{4.45}$$

The verification experiment corresponding to this operation is a pair of measurements: do we end the operation in the ground state and, given that we do, does a selective pulse on the transmon detuned by χ bring us to the excited state $|e\rangle$?



With such a sequence we can easily check the fidelity of the operation, keeping in mind the

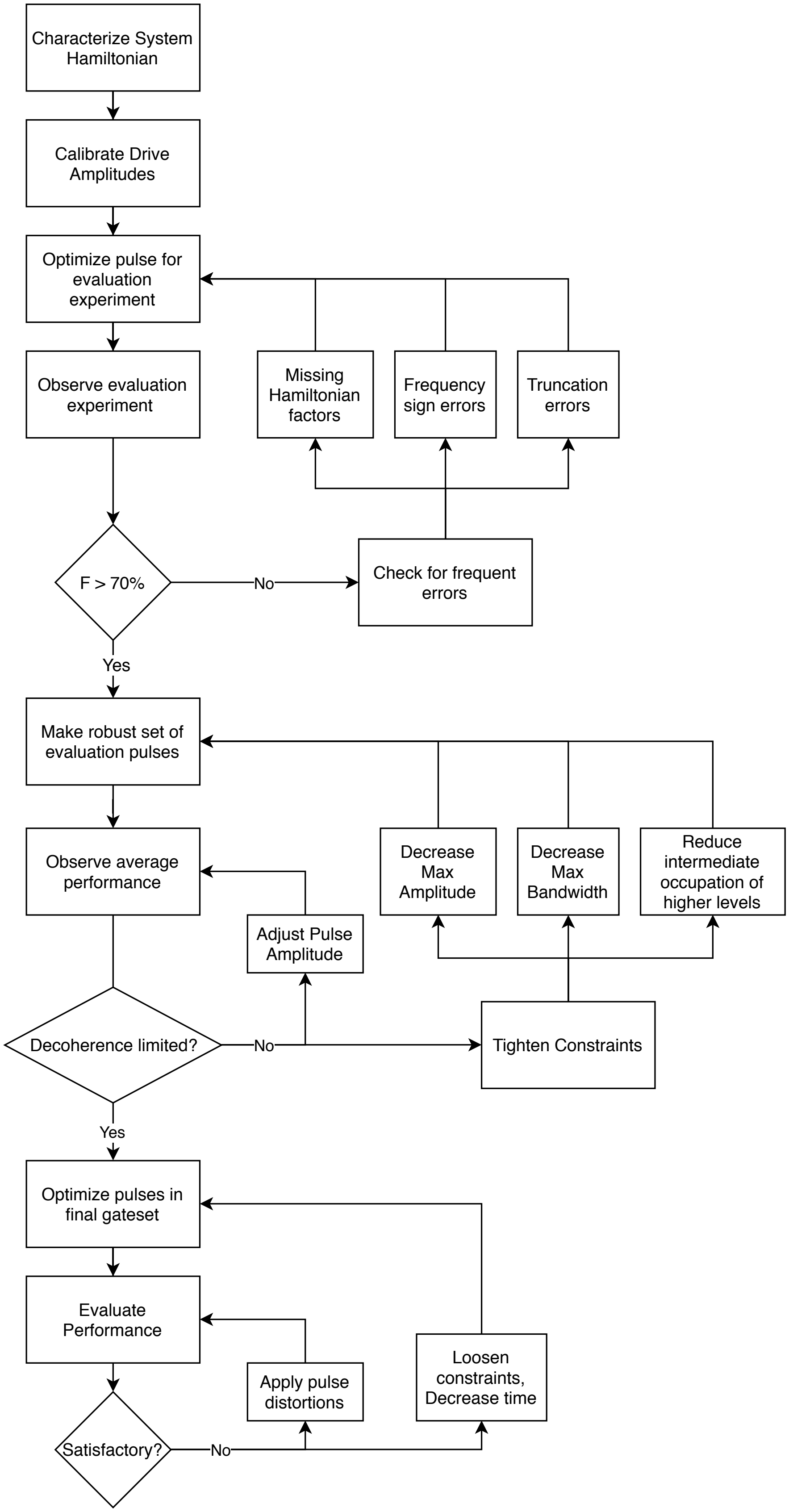

Characterize System Hamiltonian
Calibrate Drive Amplitudes
Optimize pulse for evaluation experiment
Observe evaluation experiment
Missing Hamiltonian factors
Frequency sign errors
Truncation errors
F > 70%
No
Check for frequent errors
Yes
Make robust set of evaluation pulses
Observe average performance
Decrease Max Amplitude
Decrease Max Bandwidth
Reduce intermediate occupation of higher levels
Adjust Pulse Amplitude
Decoherence limited?
No
Tighten Constraints
Yes
Optimize pulses in final gateset
Evaluate Performance
Apply pulse distortions
Loosen constraints, Decrease time
Satisfactory?
No

Figure 4.2: Flowchart for bringing an optimal control system online.

fidelity of the readout and selective pulses used to characterize it. Our goal is to proceed in steps, the first of which is to reach the point where the system is in the vicinity of the desired operating point, where we can gain increasing trust that modifications which improve our observed evaluation metric are truly bringing us closer to the optimal point.

Often times, even with a fairly accurate starting characterization of the Hamiltonian, the fidelity at first will be nil, as there can be major errors in the implementation of the system. Here are a list of common errors one can check for at this stage:

- Inconsistent use of factors of 2π in the definition of the Hamiltonian. In the `pygrape` software package, Hamiltonians should be specified with 2π.

- Incorrect relationship between the AWG amplitude and the effective driving rate. This can be checked by simulating a simple pulse such as a displacement or qubit rotation.

- Sign error in the definition of the quadrature drive, resulting in frequency inversion i.e. should the drive term be $+i(\boldsymbol{a} - \boldsymbol{a}^\dagger)$ or $-i(\boldsymbol{a} - \boldsymbol{a}^\dagger)$? The answer depends on the conventions used in realizing the pulse in the control electronics. This can be checked by negating the sign of the quadrature drive, and seeing if it performs better in practice.

- Use by the pulse of non-linearities which are artifacts of the Hilbert space truncation. This can be checked by simulating in a larger Hilbert space and verifying that the fidelity is practically unchanged.

After performing checks for these basic types of errors, the next most crucial step is to identify the right set of pulse constraints. The motivation behind constraining the pulse is to attempt to strike the right balance between the underconstrained regime, where coherent control errors dominate, and the overconstrained regime, where excessive pulse length results in decoherence dominated errors. If pulses fail to operate “as expected” it can be a sign that the constraints are not large enough. If pulses are taking too long, it can be a sign that the constraints are too stringent.

4.8 Closed-loop optimization methods

Finally, there are modifications to the pulse which can be implemented and tested empirically, without the need to re-optimize via GRAPE. The intention behind these modifications is to account for an imperfect delivery of the control pulse to the system. There are two main categories of errors which can be addressed via these methods. The first is distortion due to reflections and impedance mismatches resulting in frequency dependent transmission between the AWG and device port. The second is non-linearity of the active components, such as mixers and amplifiers, resulting in compression. We can model the first process, which takes our pulse, as played by the AWG $f(t)$ to the drive experienced by the system $\epsilon(t)$, as multiplication in the frequency domain by the transfer function $G(\omega)$:

$$\epsilon(t) = \mathscr{F}^{-1}\left[G(\omega) \cdot \mathscr{F}\left[f(t)\right](\omega)\right](t) \tag{4.46}$$

If we had complete knowledge of $G(\omega)$, we could instead play the pre-distorted pulse $\tilde{f}(t)$ at the AWG level:

$$\tilde{f}(t) = \mathscr{F}^{-1}\left[G(\omega)^{-1} \cdot \mathscr{F}\left[f(t)\right](\omega)\right](t) \tag{4.47}$$

The pre-distortion would in this case cancel with the line distortion to produce a pulse for which $\epsilon(t) = f(t)$. However, in practice, we often do not have a good way of measuring this transfer function directly. Instead we can assume that $G(\omega)^{-1}$ is well approximated by a low-order polynomial in both amplitude and phase:

$$G(\omega)^{-1} \approx \left(\sum_k b_k \omega^k\right) e^{i\sum_k c_k \omega^k} \tag{4.48}$$

Similarly, regarding compression, we would have the following type of relationship between the f and ϵ

$$\epsilon(t) = z(f(t)), \tag{4.49}$$

which could be inverted given perfect knowledge of z:

$$\tilde{f}(t) = z^{-1}(\epsilon(t)). \tag{4.50}$$

Instead, we can model z^{-1} as a low order polynomial

$$z^{-1}(x) \approx \sum_k d_k x^k \tag{4.51}$$

This gives us a handful of parameters $\{b_k\}$, $\{c_k\}$, and $\{d_k\}$, with which to vary the pulse shape. This is a large reduction over the hundreds or thousands of parameters which comprise the pulse in its full piecewise-constant representation. With this lower dimensional space in hand, we can perform a closed-loop optimization which considers the empirical performance of any given pulse.

Chapter 5

Meet the samples

The experiments which will be described in chapters 6, 8 and 9 were all performed with a similar experimental apparatus, with a few refinements realized in between. As seen in figure 5.1, the system is comprised of three principal components, each of which corresponds to an electromagnetic mode in our final quantum system. These are the storage cavity, formed by the walls of the aluminum enclosure, the transmon qubit, defined by a capacitively shunted Josephson junction, and the readout oscillator, simply formed by a length of open-terminated transmission line. In this chapter, I will discuss each of these components in turn, explaining the reasons for adopting these components, and focusing on the methodology for choosing their parameters.

5.1 The seamless storage cavity

In order to motivate the design of the cavity which acts as our storage resonator, which contains our encoded logical qubit, I will briefly describe the considerations which go into designing high quality factor cavities.

Any enclosed “box” formed from good conductors can be described by a discrete set of electromagnetic modes, which describe fields occurring inside of the box, and which do not couple to the outside world, at least not without “puncturing” the box. Each of these modes has a particular electric and magnetic field profile within the box. Energy contained by a mode oscillates between electric and magnetic components at a rate given by the mode’s character-

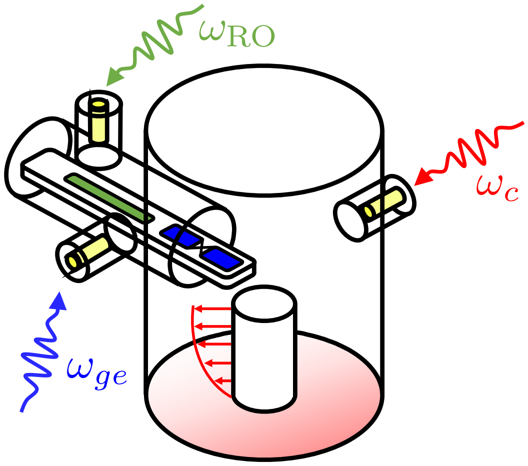


Figure 5.1: **Cartoon schematic of the cavity-transmon system.** A $\lambda/4$ coax post cavity resonator is coupled to a transmon and readout resonator on a sapphire substrate. Input couplers close to the transmon and cavity deliver the respective time-dependent microwave control fields $\epsilon_T(t)$, $\epsilon_c(t)$ and $\epsilon_{\text{RO}}(t)$.

istic resonance frequency, ω. The particular details of the geometry of the box determine its electromagnetic modes. In particular, there is a mode of lowest frequency, and we are typically concerned only with a few of the lowest frequency modes.

If the box was formed entirely from perfect conductor and vacuum, this would be the entire story, but since this can never be exactly true, there is also the possibility of dissipation. The energy contained within a mode can transfer from the mode to other, auxiliary degrees of freedom, such as phonons, plasmons, or other quasiparticles in the conductor, spins or dangling bonds in the imperfect layers of dielectric which coat the conductor surface, or out through the apertures which are inevitable in an experimental device which by necessity has some coupling to the outside world. We can try to simplify the vast space of possible detailed mechanisms for energy loss by categorizing them into three groups. The first is *conductor loss*, which is associated with current which is forced through an imperfect conductor. The second is *dielectric loss*, which arises from electric fields coupling to dipoles in the dielectric which have their own intrinsic loss mechanisms. The final is *radiative loss*, in which energy transfers from the box modes, to propagating external modes. The sum total of theses loss mechanisms induces a characteristic loss rate κ. We can calculate the number of oscillations per decay time as the quality factor $Q = \omega/\kappa$.

If we are interested in making the loss rate κ as small as possible, there are broadly two approaches we can take. We can either change the *materials* from which we construct the box, or we can change the *geometry* of the box. We will have to pursue both avenues in order to reach the highest possible quality factors. We begin with the choice of conductor, where superconductors are the obvious choice. However, while superconductors have no resistance to static, DC currents, they do have resistance to oscillating currents, especially when that frequency approaches the superconducting gap, so conductor loss cannot be neglected. If bulk conductor loss was the only concern, we could simply choose the superconductor with the smallest residual resistivity, but the choice of metal also dictates the choice of oxide which covers the surface of the metal. The properties of this oxide are not just a property of the metal, but also the conditions of its formation, which can be manipulated by removing native oxide, and allowing it to regrow under controlled conditions. For these reasons, (superconductor, high-quality oxide) high purity (99.999%) aluminum has been our material of choice for constructing cavity resonators, although similar qualities have been achieved in Niobium as well. In an attempt to improve the quality of the oxide, which would have normally developed during the machining process in a dirty environment, a chemical etch is used to remove the existing oxide and allow for a cleaner oxide to redevelop. However it has been found that extremely long etches, removing material far beyond the oxide layer has been shown to improve quality even more, and therefore the underlying mechanism behind the improvement remains obscure (Reagor, 2015).

The geometry itself plays a very important role in determining the loss rate. While these calculations are difficult and tedious to perform exactly, some rules of thumb apply. In general, the more compact the geometry, the more the cavity is susceptible to the surface effects, both in terms of current and dielectric, and therefore the higher the effective loss. One particularly pernicious aspect of the geometry is the presence of *seams*, which are regions where two separate pieces of conductor are mechanically affixed to one another. These are necessities of manufacturing with a subtractive process[1] and are entirely undesirable by themselves. Seams introduce a localized region of relatively high resistivity when compared with the superconductor

[1] Additive processes, such as 3D printing and laser sintering have the potential to bypass this issue, but bring their own challenges, especially regarding material purity and cleanliness (Creedon et al., 2016)

which surrounds it. The mere presence of a seam is not a problem however, but rather the current which is forced to flow through it. Therefore the location of the seam can be crucial. For instance, modes with symmetry can have planes bisecting them across which no current flows. Choosing to put a seam in this place will ideally induce no loss, although in practice the symmetry is broken by various imperfections, partially reintroducing some seam associated loss.

In order to overcome the issues associated with seam, one can turn to the magic of under-cutoff waveguides. When a machining tool such as a drill bit removes material from a piece of metal, it necessarily leaves an opening to the outside world. This opening forms a waveguide, i.e. a single-conductor transmission channel for electromagnetic signals. Unlike two-conductor transmission lines, such as twisted-pair or coaxial transmission lines, waveguides do not allow signals of arbitrary frequency to pass. They are in effect high-pass filters, with a cutoff frequency that scales inversely with the radius of the aperture. For a circular waveguide of radius r this cutoff is[2]

$$f_c \approx \frac{2.405c}{2\pi r} \tag{5.1}$$

where the 2.405 factor is calculated from the roots of the first Bessel function (Pozar, 2011). If we can create a mode of resonance frequency f_0 using only apertures small enough that the cutoff frequency f_c is larger than f_0, then signals from the mode at one end of the waveguide cannot reach the seam which will necessarily be present at the other end. Now coming up with designs for mode geometries which can be machined in this way requires a fair bit of ingenuity, but it is possible as demonstrated by "post cavity" design of Reagor et al. (2016). This is the design for all of the experiments performed in this thesis, as well as most of the experiments involving 3D cavities performed recently at Yale. However, there are other designs which are currently in development using the same methodology, in order to produce seamless *multi-mode* cavities and other variations (Naik et al., 2017).

In the post cavity design, we form a section of coaxial transmission line, with one end

[2] There are actually many modes of circular waveguides which can be excited. The lowest frequency mode is the TE11 mode, which has a cutoff constant of 1.841. However, the symmetries of the post cavity we are coupling to means there is zero nominal coupling to the TE11 mode, and therefore we move to the next lowest frequency mode, the TM01 mode.

shorted and the other end open, as can be seen in figure 5.1. The frequency of the mode with post length l is approximately $f_0 \approx \frac{c}{4l}$. The cutoff frequency of the hole needed to create this post by machining is determined by the outer radius of the coaxial transmission line r. Therefore, in order to be protected, we need

$$\frac{2.405c}{2\pi r} > \frac{c}{4l} \Rightarrow \frac{l}{r} > \frac{2\pi}{4 \times 2.405} \approx 0.653 \tag{5.2}$$

Therefore the aspect ratio needed is actually not too bad. Of course the post must be narrower than the outer radius, and we should be significantly below the cutoff, but we can achieve negligible seam participation with aspect ratios of 2-3 in reasonable designs.

We can summarize our cavity design methodology by referring to figure 5.2. We choose the cavity frequency which gives us the post length $\textcircled{1}$ as $\frac{2\pi c}{4\omega_c}$. Note this is only approximate, and must be adjusted for in a finite element simulation which can account for all of the stray capacitances and edge effects. For instance, our cavities have a target frequency of 4.5 GHz, which would naïvely prescribe a post length of 1.67 cm, but given other details of our geometry, we find we need a length of 1.50 cm. The resonance frequency also determines the maximum aperture radius $\textcircled{2}$ as $\frac{2.405c}{\omega_c}$. We should choose the actual radius r as far below this maximum as possible. In our case, the maximum radius was 2.55 cm, and the actual radius was 0.53 cm. We can then specify the allowable level of participation in the seam in order to choose the waveguide length $\textcircled{3}$. The current induced as a function of distance along the waveguide (z) vanishes exponentially as $e^{-\beta z}$ where $\beta = \sqrt{\left(\frac{2.405}{r}\right)^2 - \left(\frac{\omega}{c}\right)^2}$. The energy density goes as the square of current and thus vanishes as $e^{-2\beta z}$. In order to suppress the seam participation by a factor of x, the waveguide should be at least of distance $\frac{\ln x}{2\beta}$. In our case we had $1/\beta \approx 2.2\,\mathrm{mm}$. The depth of 25 mm gives us a factor of $e^{-2(25/3)} \approx 10^{-10}$ reduction in participation. See the work done by Brecht (2017) for a more rigorous treatment of calculating seam losses.

5.2 The antenna transmon

The 3D transmon qubit consists of two antenna-like aluminum pads connected by a single Josephson junction. It is fabricated on a piece of sapphire 430 microns thick, which it shares

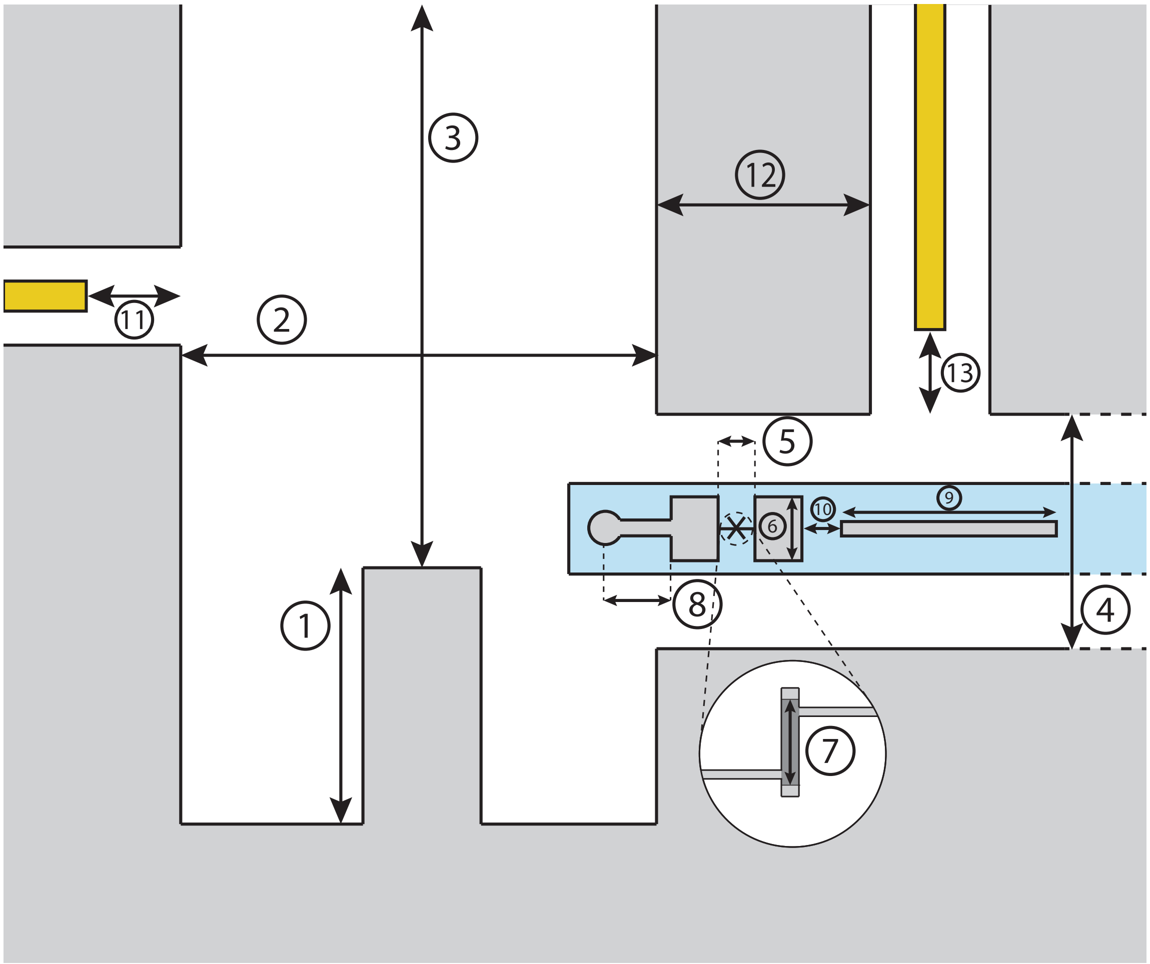


Figure 5.2: **Schematic of cavity and transmon critical physical dimensions**. This cartoon shows the critical dimensions of the geometry of the cavity-transmon system. The circled numbers are referenced in the text.

with the readout resonator. This qubit is topologically identical with a "Cooper pair box," (Nakamura et al., 1999) which is described using two primary parameters, as can be seen in the Hamiltonian (Schuster, 2007):

$$\boldsymbol{H} = 4E_C(\boldsymbol{N} - N_g)^2 + \frac{E_J}{2}\sum_n |n\rangle\langle n+1| + \text{h.c.} \tag{5.3}$$

The term N_g describes an offset voltage ($N_g = V_gC/e$) applied between the two pads either intentionally by a gate, or unintentionally by a noisy environment. First we have the charging energy E_C which is the energy associated with moving moving a single electron from one pad to the other. It is determined by the capacitance between the pads via $E_C = e^2/2C$. Second we have the Josephson energy E_J, which is the tunnel rate, set by the junction area and thickness. It is linearly related to the area and is vanishes exponentially with the thickness.[3] We can infer E_J at room temperature by measuring the resistance across the junction (R_N), using the Ambegoakar-Baratoff relationship: (Ambegaokar and Baratoff, 1963):

$$E_J = \frac{\hbar\Delta}{4e^2 R_N} \tanh \frac{\Delta}{2k_B T}. \tag{5.4}$$

The so called *transmon*[4] limit of the Cooper pair box Hamiltonian occurs when $E_J \gg E_C$. In this limit the sensitivity to offset gate voltage vanishes $\partial_{N_g}\omega_{ge} \to 0$, (Koch et al., 2007). We can relate the junction Hamiltonian 5.3 to a diagonalized picture, where we consider the transmon to be an anharmonic oscillator (essentially, a particle in a cosine potential rather than a quadratic potential), which for the first few energy levels can be represented as

$$\boldsymbol{H} = \omega_{ge}\boldsymbol{b}^\dagger\boldsymbol{b} + \frac{\alpha_T}{2}(\boldsymbol{b}^\dagger)^2\boldsymbol{b}^2. \tag{5.5}$$

We can relate the parameters of 5.3 and 5.5 in the transmon limit:

$$\omega_{ge} \approx \sqrt{8E_J E_C}/\hbar \tag{5.6}$$

$$\alpha_T \approx E_C/\hbar \tag{5.7}$$

Therefore, we have the following prescription for designing a transmon. Given the desired anharmonicity, we can determine the needed capacitance $C = e^2/(2\alpha_T\hbar)$ (for 120 MHz this is about 1 pF). Then we can specify a value for ω_{ge}[5] which sets our desired Josephson energy as $E_J \approx \omega^2/8\alpha_T$.

Now we need a way of translating C and E_J into geometry. There is obviously no unique way of performing this translation, and therefore our description of how to proceed here is as much art as science. When it comes to the geometry of the capacitance there are two major

[3] For this reason, it is especially important to control the growth of the junction oxide, as small variations can produce large swings of E_J

[4] The name "transmon" derives from "transmission line shunted plasma oscillation." The "plasma oscillation" part of this is clear: the excitations of the system are not definite charge states, but rather vibrations of the charge plasma back and forth across the junction. The "transmission line shunted" part remains obscure.

[5] this must be at least a factor of ~ 20 larger than α_T in order to stay in the transmon regime

concerns we have. First is the mechanical stability, and susceptibility to vibrations. Second is the quality of the dielectric. Let's address these concerns one at a time.

If the capacitance is prone to variations induced by mechanical instability, this can lead to an unstable transmon frequency, and therefore dephasing. The main source of mechanical instability is the mechanism holding the sapphire substrate on which the transmon is printed. Movement of the chip within the cavity can vary the distance between the transmon pads and the sidewalls, and therefore change the capacitance. The closer the pads are to the walls, the more important this capacitance is, and the larger the change in capacitance for a given variation in distance. Under this consideration there are two changes we can prescribe. First, we can increase the distance of the transmon from any side walls present, for instance, by increasing the radius of the cylindrical cutout in which it is housed. Second, we may bring the transmon pads closer together, increasing the part of the capacitance which is direct, rather than mediated by the sidewalls, and making the electric field profile tighter.

One major component of energy loss is imperfect dielectric, with defects and disorder which can absorb excitations from the transmon. By volume the vast majority of dielectric present in our system is one of two types: vacuum and sapphire. The vacuum can be considered perfect and lossless, while the sapphire is merely very high quality. While these parts store the majority of the electric energy, they are not necessarily the most important contribution to dielectric loss. The small amounts of dielectric on the interface between two components is often the most disordered and lossy, by many orders of magnitude, and therefore can be a dominant source of loss, despite having such a small volume. Each type of interface can contribute, including metal-vacuum, substrate-vacuum, and metal-substrate. Different geometries will have differing relative participation in the surface, and therefore different dielectric induced loss rates. For instance, a very narrow gap between the pads will confine much of the electric field near the sapphire surface, increasing the substrate-vacuum surface participation. For this reason, larger, more spread out designs, which decrease the ratio of surface to bulk, have a tendency to have higher lifetimes (Paik et al., 2011).

With these considerations in mind we can write down a schematic procedure for generating the transmon geometry features as seen in figure 5.2. We will first choose the transmon enclosure radius ④ as large as possible without allowing field from the cavity mode to propagate to the seam associated with this opening, i.e. placing the cutoff frequency above the storage frequency. In our case this was about 0.2 cm. Given that radius, we choose the pad distance ⑤ as large as possible such that the field will still be primarily directed from one pad to the other, without being mediated by the side walls of the enclosure. One might also be inclined to reduce this distance for the sake of keeping the transmon pads relatively compact. In our case, the distance was kept fixed at 150 micron. Next we take the desired anharmonicity to determine the length and width of the pads ⑥. The capacitance is roughly linear in the width and logarithmic in the length, so the width is the primary variable. Next take the desired transmon frequency, calculate the desired E_J, and use this to determine the Junction area ⑦. While a first principles calculation of the needed area is in principle possible, in practice we use calibration measurements to determine the relationship between junction size and normal-state resistance R_N, and then this calibration lets us pick the desired size via equation 5.4. Finally, we can determine the coupling χ between the transmon and storage mode using both the positioning of the transmon, and a "coupler" bulb ⑧. We can scale the coupling by increasing the insertion of the bulb into the cavity. The precise value of χ is determined from finite-element simulation of the entire geometry via *black box quantization* (BBQ) (Nigg et al., 2012). A thorough description of how we use BBQ in practice can be found in Blumoff (2017).

5.3 The stripline readout

In the original version of the post cavity device (Reagor et al., 2016), the readout mode was a second post cavity. It was soon realized, however, that this was not necessarily economical. A readout mode has different requirements than a storage mode. A readout mode is intentionally overcoupled to a transmission line which carries away the readout signal. This means that any loss rate due to internal mechanisms (such as seam or dielectric losses) which is significantly smaller than the loss rate due to the transmission line has a negligible effect. As a result, we prefer to use the "stripline" design (Axline et al., 2016) which has several benefits. First it is much more compact, taking up a much smaller volume than a post cavity. Second it is lithographically defined, allowing greater precision in its geometry, and therefore in its frequency and transmon coupling. Finally, it can be more easily changed, i.e. if a different frequency or coupling rate is desired, a new chip can be manufactured, as opposed to a new enclosure.

There are only two design steps required. First is choosing the readout frequency. This is usually done on the basis of the frequency of available quantum limited amplifiers. In our case, we had a JPC with maximum performance around 9.2 GHz, and therefore we aimed for this as our readout resonant frequency. This choice determines the length of the readout ⑨, which as a $\lambda/2$ mode, is set to $\frac{\pi c}{\sqrt{\epsilon_{\text{eff}}}\omega_{\text{RO}}}$. The effective dielectric is somewhere between 1 (vacuum) and ≈ 10 (sapphire) given by the exact geometry, and in our case is about 3.37. Finally, we must choose our desired interaction strength χ_{RO}, which in turn determines the spacing between the transmon and readout ⑩. The precise mapping between spacing and χ_{RO} is determined via BBQ, as was done with the storage-transmon χ.

5.4 Coupling pins

The final crucial piece of sample geometry is our means of communicating with it. For this purpose we have introduced coupling pins, which can be thought of as coax transmission lines which connect to our control electronics at one end, transitioning to an under-cutoff circular waveguide at the other end. The length of this under-cutoff waveguide attenuates the coupling between the pin and the mode at the other side.

Ultimately we must choose the length of under-cutoff waveguide (or equivalently, the length of pin to insert into a predetermined length of waveguide) to determine the coupling rate κ_{ext}. There are two tasks ahead of us: first determining the desired value of κ_{ext} for each mode to each coupler, and determining the geometry required to achieve that desired coupling rate. Our answer for the desired coupling rate is simplest for the readout cavity. We wish to maximize the "readout rate" for any given amplitude, which for a perfect acquisition chain, is equal to the dephasing rate (equation 8.18, with $\Gamma \to \kappa_{\text{ext}}$). This is maximized when $\chi_{\text{RO}} = \kappa_{\text{ext,RO}}$. For the coupling rates to the transmon and storage modes, we want these to be non-zero for the purposes of driving the system, but would prefer them to be as small as possible to avoid the induced energy decay rate. Typically, if we have some other loss rate κ_{int} from other uncontrolled sources, e.g. dielectric loss, we can choose a value $\kappa_{\text{ext}} \ll \kappa_{\text{int}}$, where our ultimate energy decay rate should not be significantly affected by the presence of the coupler.[6]

In order to compute what geometry is required to implement a desired coupling rate, we could in principle do an analytic calculation involving under-cutoff waveguide, and aperture or dipole coupled cavities (Pozar, 2011, §7.6), but more practical is performing a finite-element simulation. We have used the ANSYS electromagnetic simulation package in its "eigenmode" method of operation. A matched (50Ω) resistor is added to the distant edge of the pin couplers to simulate the effect of an arbitrarily long transmission line which allows outgoing radiation. The eigenmode analysis produces resonant frequencies as well as quality factors which can be converted to coupling rates. If the only source of loss in the simulation is the resistor corresponding to a pin of interest, then the simulated loss rate is the pin's associated coupling rate.

For the cavity pin, which by virtue of its position couples only to the storage cavity, we can simply choose a depth ⑪ such that the storage cavity lifetime is unaffected. However, for the readout pin, we will necessarily have some coupling to both the transmon and readout modes. Since we want the readout coupling high and the transmon coupling to be low, the salient metric is the ratio of $\kappa_{\text{ext},T}/\kappa_{\text{ext,RO}}$. We can choose the pin's position along the length of the chip enclosure ⑫ in order to maximize this ratio. This can actually be a bit non-intuitive, as the maximum ratio may not be as far from the transmon as possible (Blumoff, 2017, §3.6.2). In our geometry and frequency range, it was possible to find a location with a coupling ratio of approximately 10^3. This means that for a $500\ \mathrm{ns}$ readout lifetime, we limit the transmon lifetime to 500μs, an acceptable limit at our current stage of technological development. If we wished to develop a sample with a higher coupling ratio we would need to turn to the method of *Purcell filtering*, developed by Reed et al. (2010). One can find a treatment of this method which is more amenable to the stripline readout in Axline (2018), §4.3. Given that we have minimized the pin impact on transmon lifetime, we can set the length ⑬ to achieve a desired

[6] Interestingly, this implies that as our systems' intrinsic coherence parameters become better and better, we will need to weaken our coupling more and more, necessitating larger and larger drive amplitudes.

readout time.

Parameter Name	Hamiltonian Term	Quoted quantity	Device 1 Heeres et al. (2017) Chapter 6	Device 2 Rosenblum et al. (2018a) Chapter 8	Device 3 Chapter 9
Transmon frequency	$\omega_{ge}\boldsymbol{b}^\dagger\boldsymbol{b}$	$\omega_{ge}/2\pi$	5.66 GHz	6.5 GHz	4.2 GHz
Oscillator frequency	$\omega_c\boldsymbol{a}^\dagger\boldsymbol{a}$	$\omega_c/2\pi$	4.5 GHz	4.5 GHz	4.5 GHz
Readout frequency	$\omega_{\mathsf{RO}}\boldsymbol{r}^\dagger\boldsymbol{r}$	$\omega_{\mathsf{RO}}/2\pi$	9.33 GHz	9.33 GHz	9.33 GHz
Dispersive shift ($\lvert e\rangle$)	$\chi_e\boldsymbol{a}^\dagger\boldsymbol{a}\,\lvert e\rangle\langle e\rvert$	$\chi_e/2\pi$	-2194 ± 3 kHz	-93 kHz	-900 kHz
Dispersive shift ($\lvert f\rangle$)	$\chi_f\boldsymbol{a}^\dagger\boldsymbol{a}\,\lvert f\rangle\langle f\rvert$	$\chi_f/2\pi$	Not Measured	-236 kHz	1.2 MHz
Transmon anharmonicity	$\frac{\alpha}{2}(\boldsymbol{b}^\dagger)^2\boldsymbol{b}^2$	$\alpha_T/2\pi$	-236 MHz	-210 MHz	-137 MHz
Oscillator anharmonicity	$\frac{K}{2}(\boldsymbol{a}^\dagger)^2\boldsymbol{a}^2$	$K/2\pi$	-3.7 ± 0.1 kHz	-10 kHz	-2.2 kHz
Second order dispersive shift	$\frac{\chi'}{2}(\boldsymbol{a}^\dagger)^2\boldsymbol{a}^2\boldsymbol{b}^\dagger\boldsymbol{b}$	$\chi'/2\pi$	-19.0 ± 0.4 kHz	Not Measured	Not Measured
Transmon $e \to g$ relaxation	$\frac{1}{T_1^{ge}}D[\lvert g\rangle\langle e\rvert]$	T_1^{ge}	$170 \pm 10\mu s$	$25\mu s$	$50\mu s$
Transmon $f \to e$ relaxation	$\frac{1}{T_1^{ef}}D[\lvert e\rangle\langle f\rvert]$	T_1^{ef}	Not measured	$23\mu s$	$50\mu s$
Transmon dephasing	$\frac{1}{T_\phi}D[\boldsymbol{b}^\dagger\boldsymbol{b}]$	T_ϕ	$43 \pm 5\mu s$	$17\mu s$	$81\mu s$
Oscillator relaxation	$\frac{1}{T_1^c}D[\boldsymbol{a}]$	T_{cav}	2.7 ± 0.1 ms	1.07 ms	1.0 ms
Transmon thermal population	$\frac{\bar{n}}{T_1^{ge}}D[\lvert e\rangle\langle g\rvert]$	$\bar{n}$	$\bar{n} \approx .05$	$\bar{n} \approx .03$	$\bar{n} \approx .003$

Table 5.1: **Comparison of System Parameters for experiments performed in this thesis**

5.5 Putting it all together

5.5.1 Fabrication

The fabrication procedure is relatively standard, and well documented in other sources, particularly Reagor (2015). I will summarize the process for completeness. The cavity enclosure (figure 5.3) is machined from a block of high-purity (99.999%) aluminum. It is subjected to two sequential two-hour chemical etches using "aluminum etch A," a combination of nitric and phosphoric acid. The cavity design should account for the removal of approximately 150 microns of material uniformly throughout.

The transmon is fabricated on a 430 micron thick 2" wafer of sapphire by a liftoff process. A MMA/PMMA bilayer is spun on the surface of the wafer, followed by deposition of a thin gold anti-charging layer using a sputterer. Then a pattern is written via electron beam lithography (figure 5.4). The junction is defined using the bridge-free method (Lecocq et al., 2011), which has been shown to allow faster quasiparticle relaxation times than the Dolan bridge method.

After writing, the gold layer is removed by a 15 second potassium iodide (KI) etch. The e-beam resist is developed by a two-minute immersion in a temperature-controlled IPA/Water bath which has been shown to perform favorably in comparison with MIBK (Yasin et al., 2002). After development, the sample is subjected to an Ar/O2 plasma to clean the exposed sapphire surface, and then undergoes two cycles of aluminum deposition, each followed by controlled oxidation. After deposition, the excess metal is lifted off by a 2 hour (typically overnight) soak in NMP kept at 50° C. After liftoff, a thick photo-resist (SC1827) is applied for the purpose of protecting the sample surface during dicing. The chip is diced, and the samples are cleaned using NMP, acetone, and methanol. The chip is held in place by a clamp and inserted into the cavity trench resulting in a final geometry similar to that seen in figure 5.5.

5.5.2 Attenuators and filtering

No less crucial than the sample itself is the apparatus we connect it to. To begin with, we must thermalize the sample, and we have two objectives in this regard. First, we must bring the sample temperature low enough to reach the superconducting regime, which for

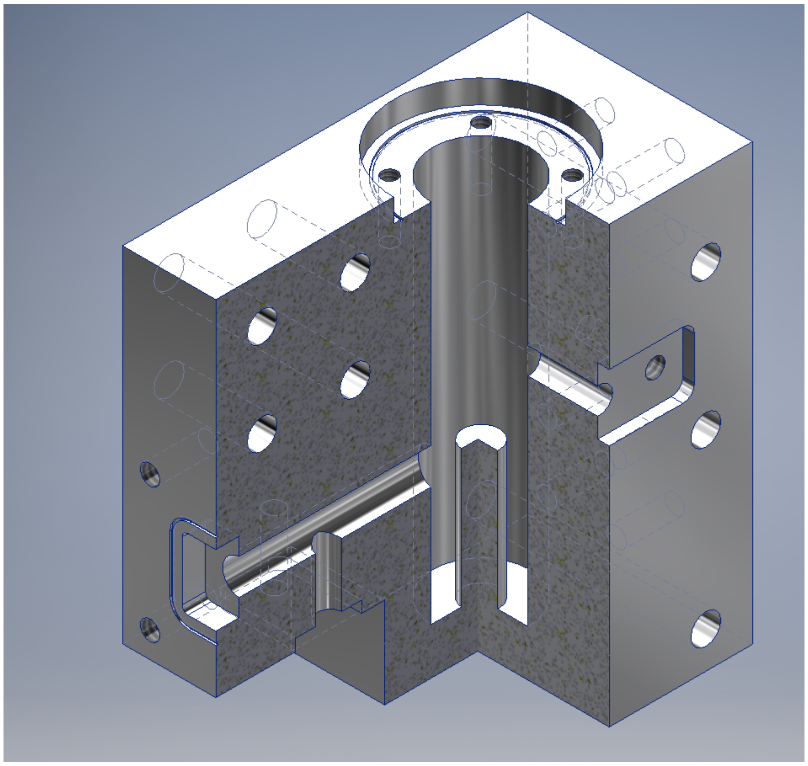


Figure 5.3: $^3\!/_4$-**section view of machined cavity component**.

aluminum is 1.2K. Second, we must remove all blackbody radiation at the frequency of interest of approximately 5 GHz, corresponding to a temperature of $\frac{\hbar\omega}{k_B} \approx 240\ \mathrm{mK}$.

Bringing the base plate of the refrigerator, to which our sample can be firmly attached, to a temperature well below these limits (about 20 mK) is a necessary first step to achieving good thermalization. It would be sufficient too if not for the fact that we need to introduce cables which attach at one end to our sample, and at another end to our control electronics living at room temperature. These cables act as a window, and would carry very hot radiation from the outside world into our sample, preventing it from cooling down. We can overcome this problem by adding attenuators to our cables. These attenuators remove all but an insignificant fraction of the thermal photons from the outside. They remove all but a fraction of the signal (driving) photons as well, but this fraction can be significant if the input drive strength is sufficiently high. The cost of adding attenuators is that they contribute thermal photons of their own, as a result of Johnson-Nyquist noise, which itself is a result of the fluctuation-dissipation theorem. This added component can be made small if the attenuator itself is well thermalized and cold.

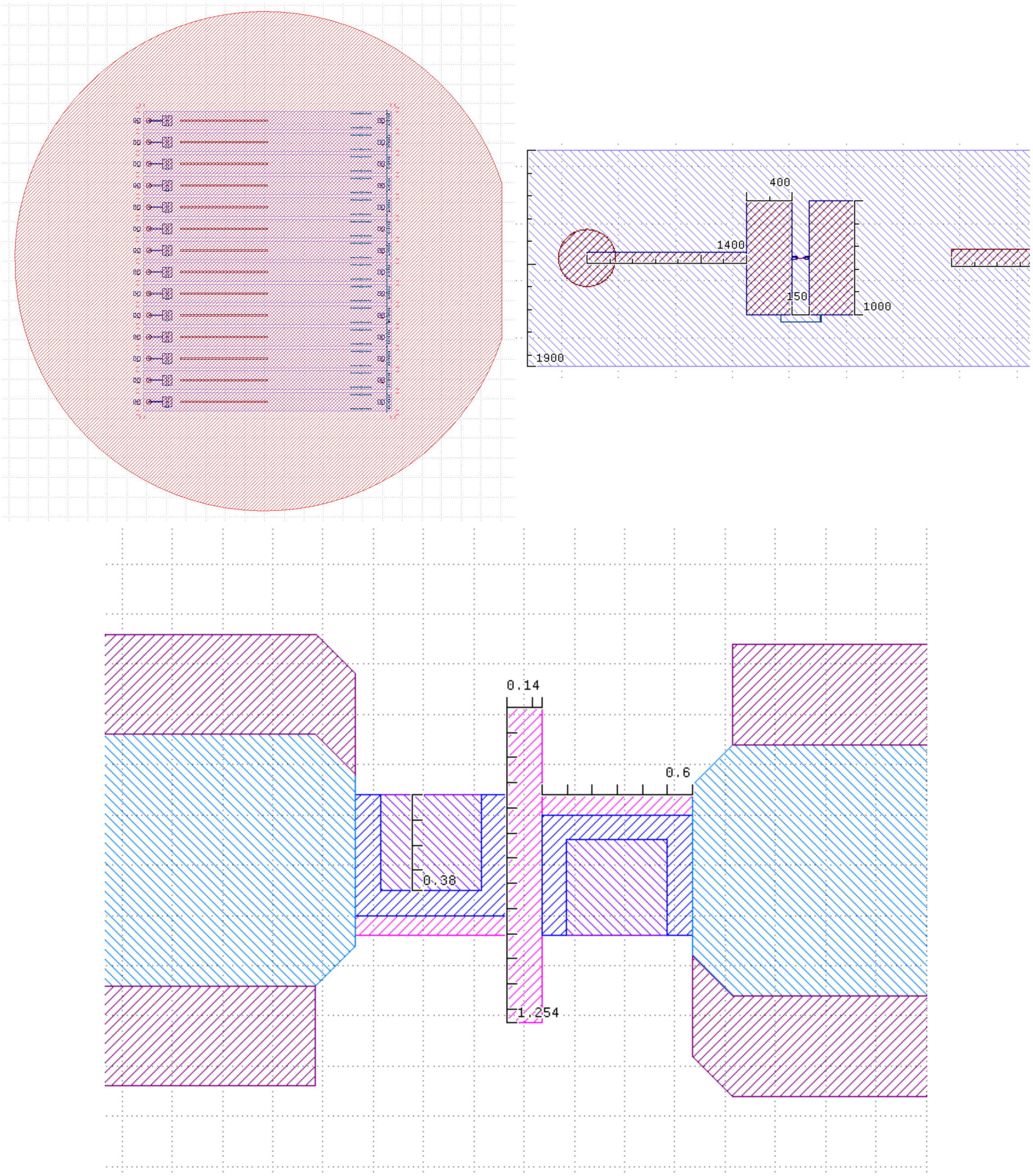


Figure 5.4: **Transmon and readout chip schematic.** (top left) The layout of 14 chips on a 2" wafer. Each chip (top right) contains a transmon and a readout mode, as well as a set of test junctions. The junction (bottom) has several geometry components with varying levels of e-beam dosage applied. The pink leads below and above the U-shaped blue regions should only allow metal to reach the substrate on one orientation of the multi-angle deposition. All indicated numbers are measurements in units of microns

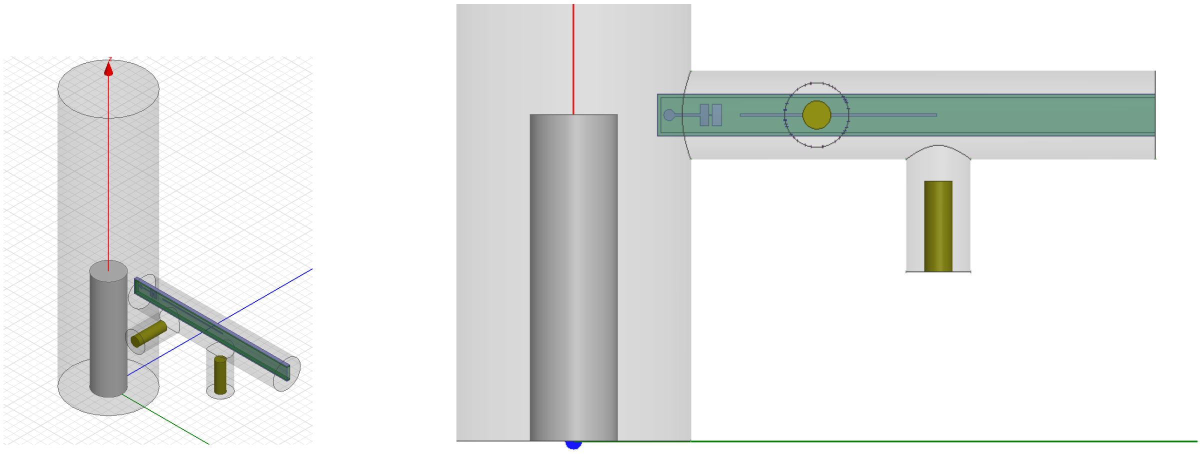


Figure 5.5: **Assembled device schematic**

For this reason, we add attenuators on the input lines and anchor them to the cold plate of the fridge. The isolation factor G required to get the thermal photon population at frequency ω down to a target level $\bar{n}$ can be found via

$$\bar{n} = G\bar{n}_{\mathsf{BE}}(293\ \mathrm{K}, \omega) + (1 - G)\bar{n}_{\mathsf{BE}}(20\ \mathrm{mK}, \omega), \tag{5.8}$$

where we use the Bose-Einstein distribution

$$\bar{n}_{\mathsf{BE}}(T, \omega) = \frac{1}{e^{\frac{\hbar\omega}{k_B T}} - 1}. \tag{5.9}$$

For a target population of $\bar{n} \approx 10^{-3}$ at 5 GHz requires $G \approx -60\ \mathrm{dB}$. About 10 dB of attenuation is present from just the lines themselves, so we need to add at least 50 dB of explicit additional attenuation. If we were to put this amount of attenuation at base, however, for a given amount of power delivered to the sample, we would have to dissipate 10^5 times that power. We do not have the cooling power to maintain 20 mK temperatures with this heat load, and therefore this amount of power would heat up the fridge. We can obtain nearly the same performance without affecting the base temperature by splitting the attenuation in two parts, the first of which is attached to a higher temperature stage (here, -20 dB at 4 K), such that the majority of the heat load is dissipated at this stage, and the remaining attenuation occurs at 20 mK. The logic is that in this scheme, we only need as much attenuation at base as is required to attenuate the 4 K thermal photons, rather than what is needed to attenuate the 290 K thermal photons. The equations for this two stage thermalization are then:

$$\bar{n}_{4\ \mathrm{K}} = G_{4\ \mathrm{K}}\bar{n}_{\mathsf{BE}}(293\ \mathrm{K}, \omega) + (1 - G_{4\ \mathrm{K}})\bar{n}_{\mathsf{BE}}(4\ \mathrm{K}, \omega) \tag{5.10}$$

$$\bar{n}_{\mathsf{base}} = G_{\mathsf{base}}\bar{n}_{4\ \mathrm{K}} + (1 - G_{\mathsf{base}})\bar{n}_{\mathsf{BE}}(20\ \mathrm{mK}, \omega). \tag{5.11}$$

If we say $G_{4\ \mathrm{K}} \approx -20\ \mathrm{dB}$ and $G_{\mathsf{base}} \approx -40\ \mathrm{dB}$, the final population is about 3×10^{-3}, a relatively small increase in thermal population compared with the single-stage approach. We can actually avoid the dissipation of any significant amount of heat at the base stage by using the "direct-tenuator" approach. Instead of using a 20 dB attenuator at base, one can instead use a directional coupler, with a -20 dB coupled port. The majority of the power goes straight through the directional coupler, and is carried on a line which goes back up to the 4 K stage, where it can be dissipated more easily. The downside of this approach is that it sacrifices an additional line between the base and 4 K stages, which is not always available. In addition,

We must use a similar scheme on the output lines as well, but instead of attenuators, we use isolators, which are non-reciprocal devices which act as attenuators for signals travelling in one direction, but have low loss in the other direction, allowing the outgoing signal integrity to be preserved while simultaneously protecting the sample from incoming radiation.

We care not only about photons at the frequency of our modes, but also at higher frequencies, as these can excite quasiparticles and cause all sorts of other nasty effects. If we had perfect, infinite bandwidth attenuators, our presently described setup would be sufficient. However, commercially available attenuators are typically only characterized up to about 20 GHz or so. At higher frequencies these devices can effectively act as passthroughs, allowing harmful infrared radiation to pass through. For this purpose we introduce additional filters, formed by replacing the dielectric in a length of coax with a special epoxy (eccosorb) which is designed to be as absorptive as possible in the infrared. Finally, we can use reflective (lossless) low-pass filters designed to reject frequencies above 10 GHz in order to cover the range between 20 GHz and the infrared band covered by the eccosorb.

We can summarize the attenuation and filters used in our actual experiments via the diagram 5.6.

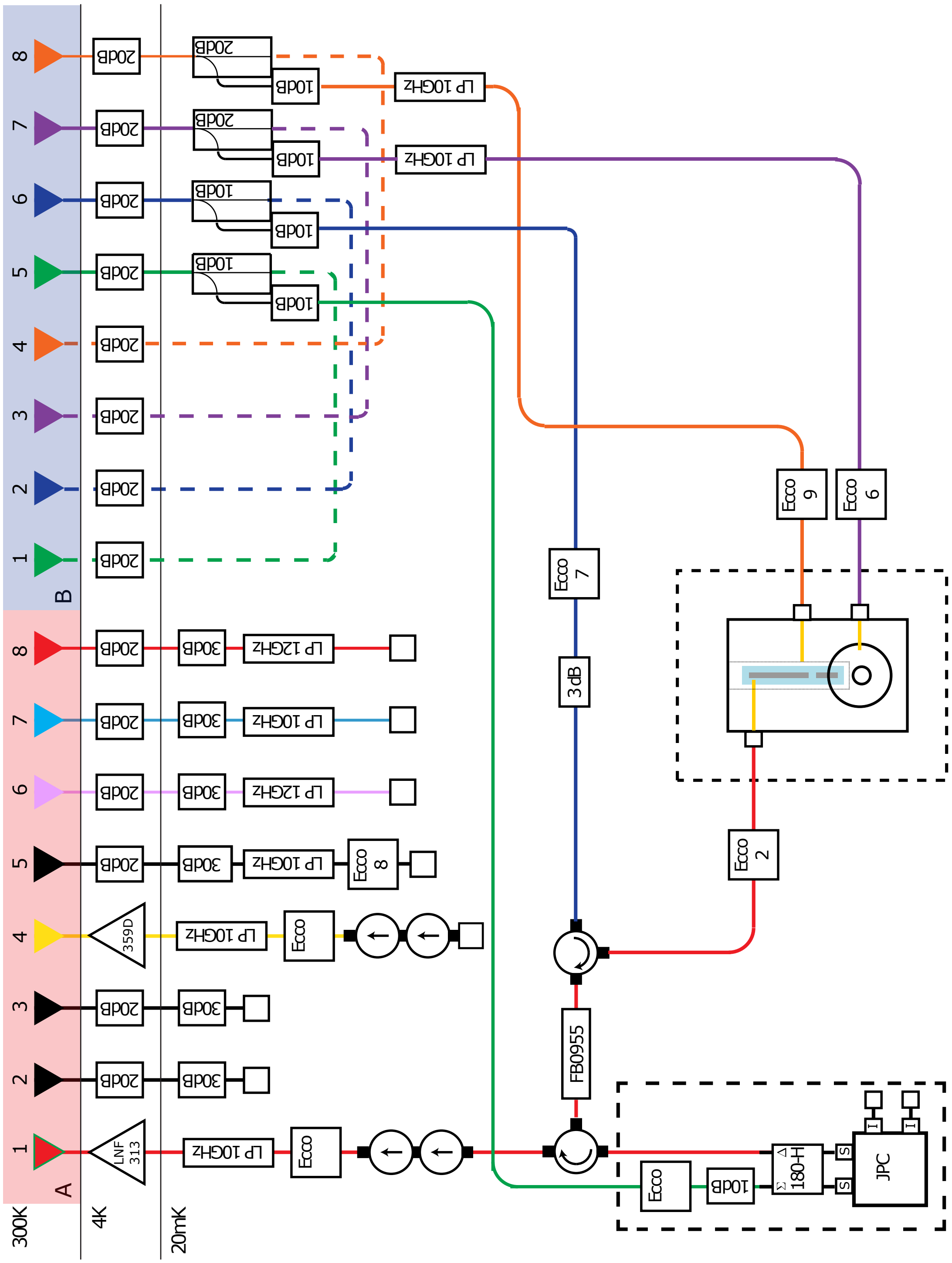


Figure 5.6: **Fridge wiring diagram**

It turns out that the ordering of the components is quite critical. Specifically, the eccosorb should be the last component, and as close to the sample as possible, in order to minimize the temperature of the transmon mode. It is not known precisely why this is the case, but one hypothesis is that the cables and connectors of the microwave components are "leaky," allowing high-frequency photons in from the environment. The ambient photon temperature can be much hotter than the base temperature, especially since most fridges do not have cans which isolate the base plate from the 1 K stage. Additionally, it has been found to be crucial to thermally anchor all of the components at base as directly as possible. It is not desirable to rely upon the coaxial microwave cables themselves to conduct heat. Instead, one should use conductive copper braid, tightly clamped, when direct affixing to the base plate is not possible.

5.5.3 Readout amplification

We are forced to attenuate the signals going to the sample in order to keep the signal to noise ratio high. For the same reason, we must perform amplification of the outgoing signal, as the signal sizes coming out of the system will be dwarfed by the thermal noise in the room temperature acquisition electronics. To do so, we must use a series of low-noise amplifiers, positioned at different temperature stages.

The most important amplifier is the first, which will largely determine the signal-to-noise ratio. For this purpose, we use a Josephson Parametric Converter (JPC), which is a quantum limited amplifier built out of a ring of four Josephson junctions (Bergeal et al., 2010). The system effectively implements the Hamiltonian

$$\boldsymbol{H}_{\mathsf{JPC}} = \omega_a \boldsymbol{a}^\dagger \boldsymbol{a} + \omega_b \boldsymbol{b}^\dagger \boldsymbol{b} + \omega_c \boldsymbol{c}^\dagger \boldsymbol{c} + g(\boldsymbol{a} + \boldsymbol{a}^\dagger)(\boldsymbol{b} + \boldsymbol{b}^\dagger)(\boldsymbol{c} + \boldsymbol{c}^\dagger), \tag{5.12}$$

Where the three modes are called the signal, idler, and pump modes. The system is flux tuned until $\omega_c \approx \omega_a + \omega_b$, within the linewidth of the pump mode. By applying a large displacement ξ_c to the pump mode we can activate the term $\boldsymbol{ab} + \boldsymbol{a}^\dagger \boldsymbol{b}^\dagger$ with strength $g\xi_c$ (see chapter 7 for more discussion of how to analyze "activating" Hamiltonian terms with pumps). The effect of this two-mode squeezing Hamiltonian[7] is to transform the ladder operators over the interaction time t (set by the linewidth) resulting in,

$$\boldsymbol{a} \mapsto \boldsymbol{a} \cosh g\xi_c t - \boldsymbol{b}^\dagger \sinh g\xi_c t \tag{5.13}$$

This can be interpreted as adding amplification to the field of the signal mode (on the order of $\cosh g\xi_c t$), while mixing in some component of the field of the idler mode. This idler mode contribution can be minimized by keeping the idler input to be cold vacuum, in which case the idler mode contribution is to add a “half-photon” of noise. Amplifiers are a deep and subtle topic, to which this hand-wavy description does not do justice. See Sliwa (2016) for a rigorous discussion of how to model the JPC and other types of quantum amplifiers.

JPC amplifiers have a limited dynamic range, meaning that the output begins to saturate at some input power, which we approach when operating the JPC with 20 dB of gain on typical readout signals. Therefore a different type of device is required to perform additional amplification. For this purpose we employ a high electron mobility transistor (HEMT) amplifier. These devices operate at 4K, and provide about 30 dB of gain, with a very low noise figure (≈ 0.1 dB). A final room temperature amplifier, provides the final ≈ 20 dB of gain required to cancel the effect of the attenuators, bringing the total apparent S_{21} of the input-output lines close to unity.

5.5.4 Electronics

The role of the electronics is to firstly, generate the signals which we send into the lines to manipulate the system, and secondly to acquire, interpret, and potentially even act upon the returning signals which carry the information about the system’s behavior.

We can start with the signal generation. The most straightforward approach to this problem is to use a so-called “direct-synthesis” approach using a high-speed, high-bandwidth (~ 50 GS/s, ~ 20 GHz) AWG to create the desired signals *ex nihilo*, as was shown in Raftery et al. (2017). However, this approach requires very expensive hardware, and the capabilities

[7] See Walls and Milburn (1995) for a concise description of the action of both single-mode and multi-mode squeezing operators

and limitations of this approach still need to be explored and characterized. For this reason, we use a more traditional approach involving lower bandwidth (~ 1 GHz) AWGs together with microwave signal generators, which can produce high-quality pure tones up to 20 GHz. With this approach we can produce any signal we want in the same frequency range, with the constraint that the instantaneous bandwidth is within the AWG bandwidth. We can see an example setup in figure 5.7. The main performance metric for the signal generation electronics is *phase noise*, i.e. how well defined is the phase of a signal, relative to a delayed version of itself. The phase noise is the power present in the output, at a given offset from the carrier frequency, relative to the carrier output power. The power spectral density (PSD) can be run through a "filter-function" analysis to produce the effect of phase noise on the Ramsey decay curve (Bylander et al., 2011; Green et al., 2013; Soare et al., 2014). Other components we see in figure 5.7 are amplifiers, which allow us to reach the desired input power, and switches, filters, and isolators, to ensure that the signal is as clean as possible. The switches are a critical component which must be placed after any room temperature amplifier, as these amplifiers increase the apparent noise temperature going into the system. The logic behind the switch is that this noise is acceptable for the short periods of time when the drive is active (say for a fast π pulse) but would be very detrimental over longer periods of idling time. DC blocks are placed around all active components, such as switches, amplifiers, and mixers, to prevent ground loops whenever possible. Finally, several diagnostic ports are created. We tap off the AWG outputs before the mixer, in order to allow us to see the pulse sequence playing in the time domain on an oscilloscope. A small portion of the RF power is diverted to a spectrum analyzer before going to the fridge, which allows us to perform IQ mixer calibrations without disrupting the setup.

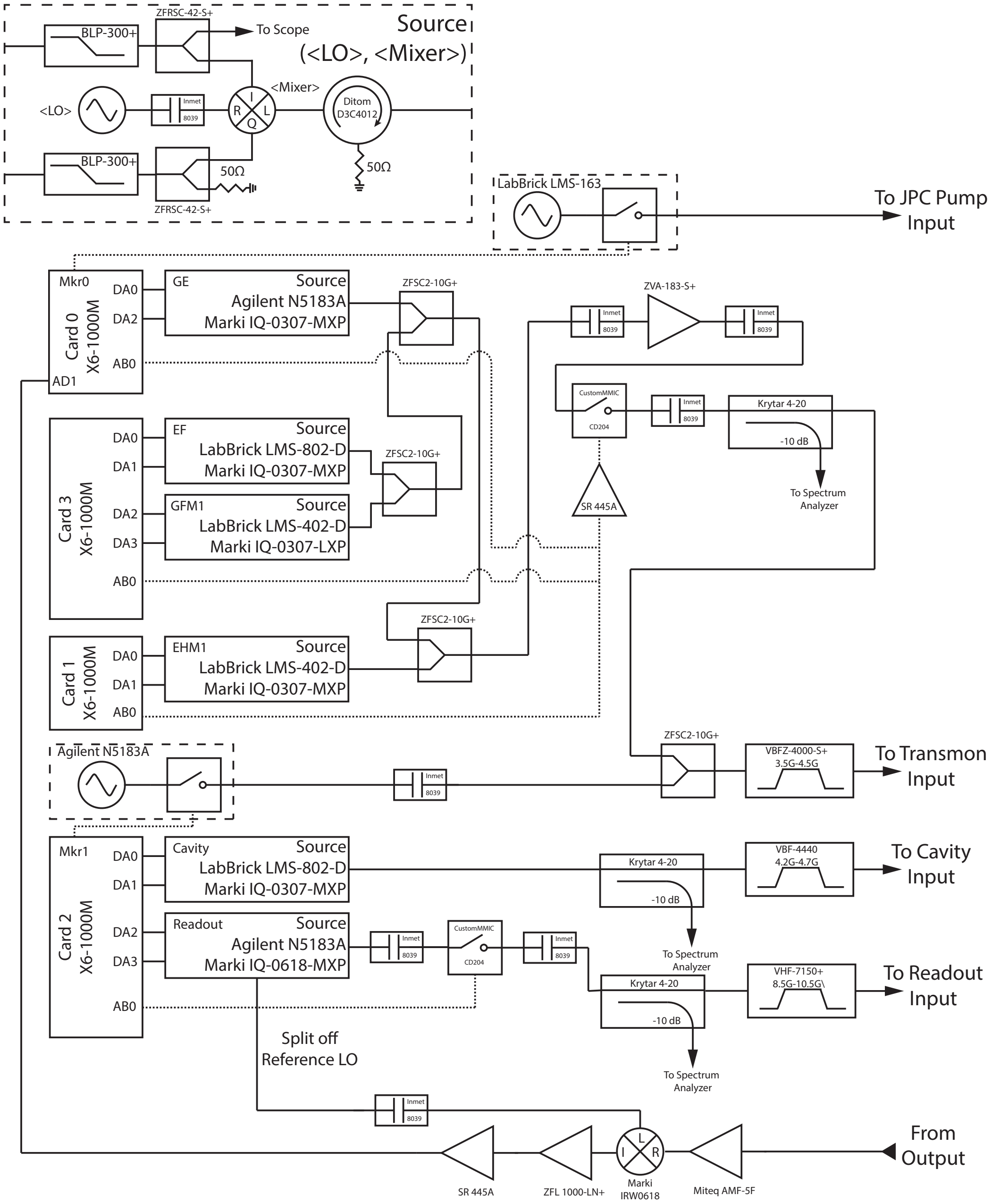


Figure 5.7: **Signal generation and acquisition electronics** This is the final setup used for the FT SNAP experiment (chapter 9). Similar setups are used for the previous two experiments. The "source" block at the top is repeated several times, and is parameterized by the LO hardware, and the mixer hardware, with other components remaining the same. The GFM1 and EHM1 sectors produce the sideband drives corresponding to the $|g, n\rangle \leftrightarrow |f, n-1\rangle$ and $|e, n\rangle \leftrightarrow |h, n-1\rangle$. These were used for photon preparation (section 7.3) and χ cancellation (section 8.2), respectively.

Chapter 6

Universal control of a cat-code qubit

Quantum error correction (QEC) aims at the creation of logical qubits whose information storage and processing capabilities exceed those of its constituent parts. Any logical qubit will inevitably face difficulties in control, by its very nature. The information is encoded in such a way that the environment cannot, without great difficulty or coincidence, learn about, measure, or manipulate it. This means that the designers of operations on a logical qubit must overcome the control barriers they themselves erected in the first place. This is one more facet of the fundamental tension of quantum computing, between isolation and control.

In order to perform gates on a logical qubit one must perform operations on the entire system in such a way that it results in the desired transformation within the two-dimensional logical subspace. In any encoding scheme the system dynamics are not naturally confined within the logical subspace. Therefore, implementing operations requires carefully tailored controls which address each component of the system and manage their mutual interactions.

In encodings based on many entangled two-level systems, which are designed to detect and correct individual errors on any of the subsystems, such as the 7-qubit Steane code, control comes in the form of single-qubit drives on each of components, as well as entangling drives between connected pairs. While this control scheme may seem simple in theory, in practice one will have to reckon with the presence of many types of deviations from the model, including cross-talk between the lines and always-on unwanted interactions between adjacent qubits. These complications will make this task much more difficult in practice.

We have shown in section 2.3 how a dispersively coupled qubit gives universal control to a cavity, and consequently, in principle, of encoded states within the cavity. While the control scheme is much less intuitive than that of multiple qubits, it is actually much simpler to characterize the model deviations, such as Kerr non-linearity. We will begin this chapter by showing how we perform the necessary system characterization, in section 6.1. With this characterization in hand, we can proceed in section 6.2 to employ numerical optimal control techniques detailed in chapter 4, allowing us to demonstrate a contrived, but highly nontrivial action on the cavity. In 6.3 we move the setting back to the cat code detailed in section 3.6 by developing pulses which transfer quantum information between the traditional transmon qubit and the cavity logical qubit. Finally, in 6.4, we can create and test a set of operations on the logical qubit.

6.1 First things first: Characterizing the system

The model we will use to describe the system is a dispersive model, by which we mean the static components are entirely diagonal in the photon number basis. We can partition the Hamiltonian into sectors:

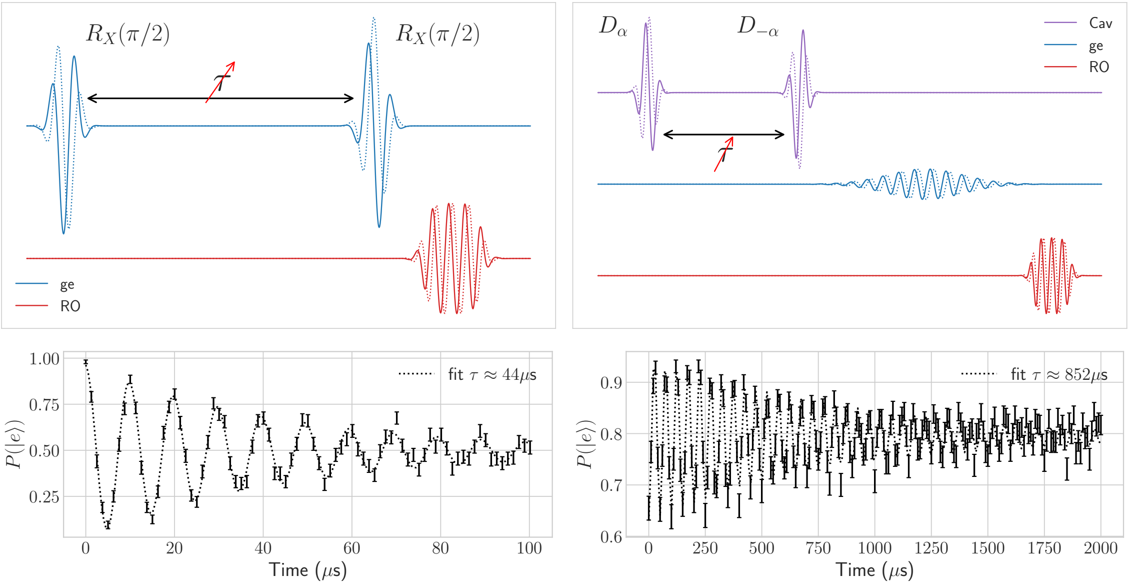
$$\boldsymbol{H}(t) = \boldsymbol{H}_{\text{oscillator}} + \boldsymbol{H}_{\text{transmon}} + \boldsymbol{H}_{\text{interaction}} + \boldsymbol{H}_{\text{drive}}(t) \tag{6.1}$$

We can begin by characterizing the transmon and oscillator sectors

$$\boldsymbol{H}_{\text{transmon}} = \omega_T \boldsymbol{b}^\dagger \boldsymbol{b} + \frac{\alpha_T}{2}(\boldsymbol{b}^\dagger)^2 \boldsymbol{b}^2 \tag{6.2}$$

$$\boldsymbol{H}_{\text{oscillator}} = \omega_C \boldsymbol{a}^\dagger \boldsymbol{a} + \frac{K}{2}(\boldsymbol{a}^\dagger)^2 \boldsymbol{a}^2 \tag{6.3}$$

It is absolutely crucial to know the resonant frequencies ω_C and ω_T, as accurately as possible. The most precise method for determining this is via Ramsey interferometry. The method and fit is slightly different for the cavity and transmon (see figure 6.1) but both operate on the same principle of comparing the phase of the subsystem with that of a local oscillator for varying times.



(a) Transmon Ramsey sequence

(b) Cavity Ramsey sequence

Figure 6.1: **Transmon and cavity Ramsey sequences** Ramsey sequences are a set of time-parameterized experiments which directly yield accurate measurements of the system's resonant frequency. On the left is a transmon Ramsey sequence, which plays two $\pi/2$ qubit rotations separated by a variable time. On the right, a pair of small cavity displacements is separated by a variable time. The photon number can be mapped onto the transmon by a selective transmon rotation. In order to measure the coherence decay constant accurately, it is important to use a detuned local oscillator (either in hardware, or virtually, in software).

The anharmonicities K and α_T differ by many orders of magnitude, and therefore require very different approaches in order to characterize. One might ask why we would need to characterize the anharmonicity, if we are not intending on employing the $|e\rangle \leftrightarrow |f\rangle$ transition in our operation,[1] but this term can, especially at large transmon drive amplitudes induce an effective Stark shift. Therefore including it is essential to achieve the highest level of performance, but the required precision is lower. To measure this, we can perform simple spectroscopy (figure 6.2). Accounting for the cavity anharmonicity, also known as the (self-)Kerr non-linearity K, is no less important. While it may be negligibly small when acting on states within the cat code manifold, the n^2 scaling with respect to photon number means that any intermediate occupation of higher photon number states will induce sensitivity to the value

[1] In principle using this transition is possible, and would likely provide a large benefit for doing faster operations, but would require additional characterization. Extending optimal control to using this resource would be a useful further direction to explore.

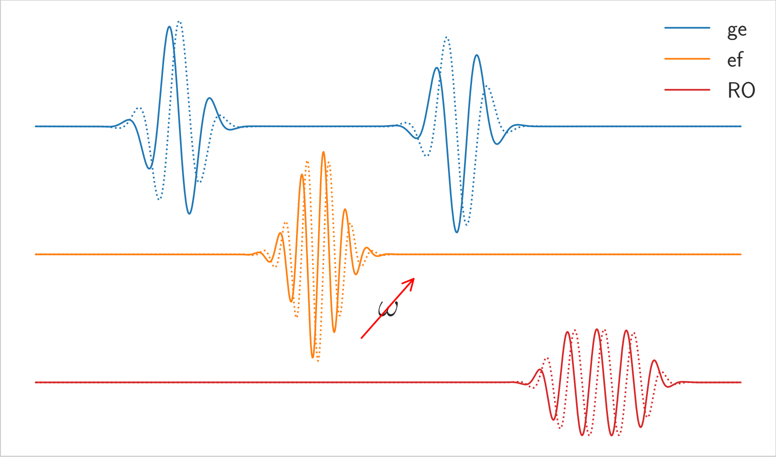


Figure 6.2: **Transmon EF Spectroscopy** The transmon anharmonicity can be found by varying the frequency of the second pulse in this sequence around the expected location of ω_{ef}.

of K. The relevant information for calculating K is contained within an extension of the cavity Ramsey sequence (figure 6.1b) which, in addition to scanning the delay time τ also scans the cavity displacement α. After acquiring this data, we can fit the data using the following model (figure 6.3):

$$f(\omega_C, K, t, \alpha) = |\,\langle\alpha|\, e^{i\omega_C t \boldsymbol{a}^\dagger \boldsymbol{a} + \frac{iKt}{2}(\boldsymbol{a}^\dagger)^2 \boldsymbol{a}^2}\, |\alpha\rangle\,|^2. \tag{6.4}$$

The next piece of characterization that needs to be carried out is to identify the interaction. The most important parts of this Hamiltonian are the linear (χ) and quadratic (χ') photon number dependent dispersive shifts:

$$\boldsymbol{H}_{\text{interaction}} = \chi \boldsymbol{a}^\dagger \boldsymbol{a} \boldsymbol{b}^\dagger \boldsymbol{b} + \frac{\chi'}{2}(\boldsymbol{a}^\dagger)^2 \boldsymbol{a}^2 \boldsymbol{b}^\dagger \boldsymbol{b} \tag{6.5}$$

We can determine these parameters quite accurately by measuring the transmon frequency at different photon numbers. While, at this stage of characterization, we might not be able to prepare individual photon number states deterministically,[2] we can prepare large photon number states via simply driving large displacements in the cavity. Performing spectroscopy of the transmon reveals a frequency comb (see figure 6.4). By scanning the value of α and fitting

[2] see sections 6.2 and 7.3 for how this can be done with optimal control or sideband drives, respectively

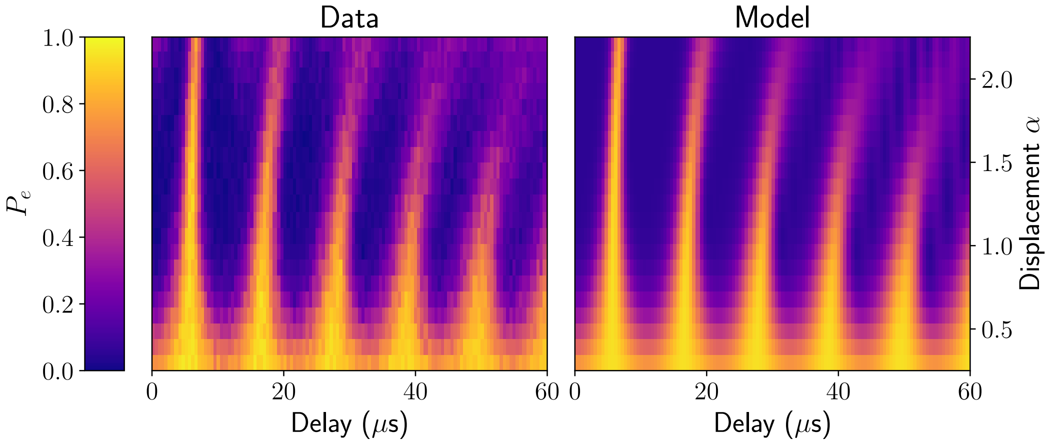


Figure 6.3: **Determining the cavity Kerr non-linearity.** This protocol is the same as the cavity Ramsey sequence seen in figure 6.1b, with the added complication of varying the displacement amplitude. The Kerr non-linearity can be determined from the "bending" of the curves seen here. The quantitative fit can be found using equation 6.4

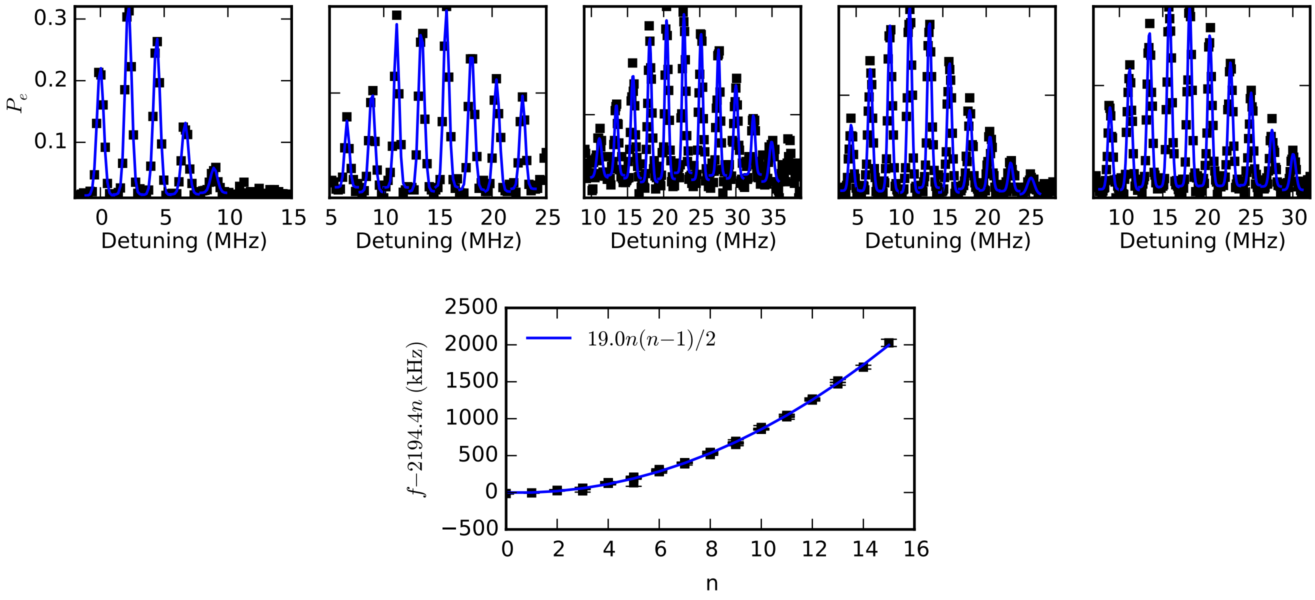


Figure 6.4: **Transmon spectroscopy for determining χ and χ'.**The dispersive shift χ and its second order correction term χ' are determined from transmon spectroscopy experiments with several different displacements (top). Each peak is fit to a Gaussian and the resulting center frequencies are fit using a quadratic model. We see in the bottom plot the deviation of the peak locations from a linear fit, and that a quadratic adjustment is sufficient correction.

the comb using a sum of Gaussians, we can extract the transmon frequency for every photon number, up to the number of photons we are going to allow in our optimal control pulses. We can then fit this set of numbers to a low order polynomial. In this experiment second order was sufficient.

Finally we need to characterize the driven sector of the Hamiltonian:

$$\boldsymbol{H}_{\text{drive}}(t) = \epsilon_C(t)\boldsymbol{a} + \epsilon_T(t)\boldsymbol{b} + \text{h.c.} \tag{6.6}$$

In particular, we need to relate the rates ϵ_C and ϵ_T (in units of Hz) to the AWG amplitude (in either DAC units or voltage). Given that we have a characterized π pulse or cavity displacement.

For an ideal two-level system, we can calculate the drive rate ($\epsilon_{T,\text{max}}$) corresponding to the peak of a Gaussian pulse of with σ required to perform a pi pulse:

$$\begin{aligned} e^{i\theta\boldsymbol{\sigma}_x} &= \cos(\theta) + i\sin(\theta)\boldsymbol{\sigma}_x & (6.7)\\ \frac{\pi}{2} &= \theta & (6.8)\\ &= \epsilon_{T,\text{max}} \int \mathrm{d}t\, e^{-\frac{t^2}{2\sigma^2}} & (6.9)\\ &= \epsilon_{T,\text{max}} \sqrt{2\pi\sigma^2} & (6.10)\\ \epsilon_{T,\text{max}} &= \sqrt{\frac{\pi}{8\sigma^2}} & (6.11) \end{aligned}$$

Assuming a linear relationship between the DAC setting and the drive rate then gives us a map from arbitrary DAC settings to drive rates. Similarly, from a unit displacement, we can calibrate ϵ_C.

$$
\begin{aligned}
e^{i\alpha(\boldsymbol{a}+\boldsymbol{a}^\dagger)} &= \boldsymbol{D}_{i\alpha} && (6.12)\\
1 &= \alpha && (6.13)\\
&= \epsilon_{C,\mathsf{max}} \int \mathrm{d}t\, e^{-\frac{t^2}{2\sigma^2}} && (6.14)\\
&= \epsilon_{C,\mathsf{max}} \sqrt{2\pi\sigma^2} && (6.15)\\
\epsilon_{C,\mathsf{max}} &= \frac{1}{\sqrt{2\pi\sigma^2}} && (6.16)
\end{aligned}
$$

6.2 Showing off a bit: Creating distant Fock states from scratch

After performing the necessary system characterization, resulting in an accurate model of the cavity-transmon Hamiltonian (see Table 5.1 for the results of this characterization), the next step is to tune up the optimal control. We do this largely by following the procedure in figure 4.2. To fill out the details of this protocol, we can specify that the "evaluation pulses" we targeted for the purposes of getting the pulse constraints correct was a set of operations designed to create different Fock states in the cavity, and took the form

$$|0\rangle \otimes |g\rangle \longrightarrow |n\rangle \otimes |g\rangle \qquad (6.17)$$

An example of one of these pulses (specifically for $n = 6$), and its simulated trajectory in phase space, is visualized in figure 6.5. Under the action of this $500\ \mathrm{ns}$ pulse, we see that the cavity and transmon quickly become entangled, with the cavity state having little recognizable or meaningful structure apart from the beginning and end states. We can compare this with the shortest equivalent sequence constructed via SNAP and displacements, we would require at least 2 SNAP operations and 3 displacements, which in this experiment, would take at least 3 μs and would be susceptible to control errors via Hamiltonian terms other than χ.

The real test of this pulse is performing it in practice. This is effectively asking the question: is our model Hamiltonian accurate enough to reliably predict the system evolution within the

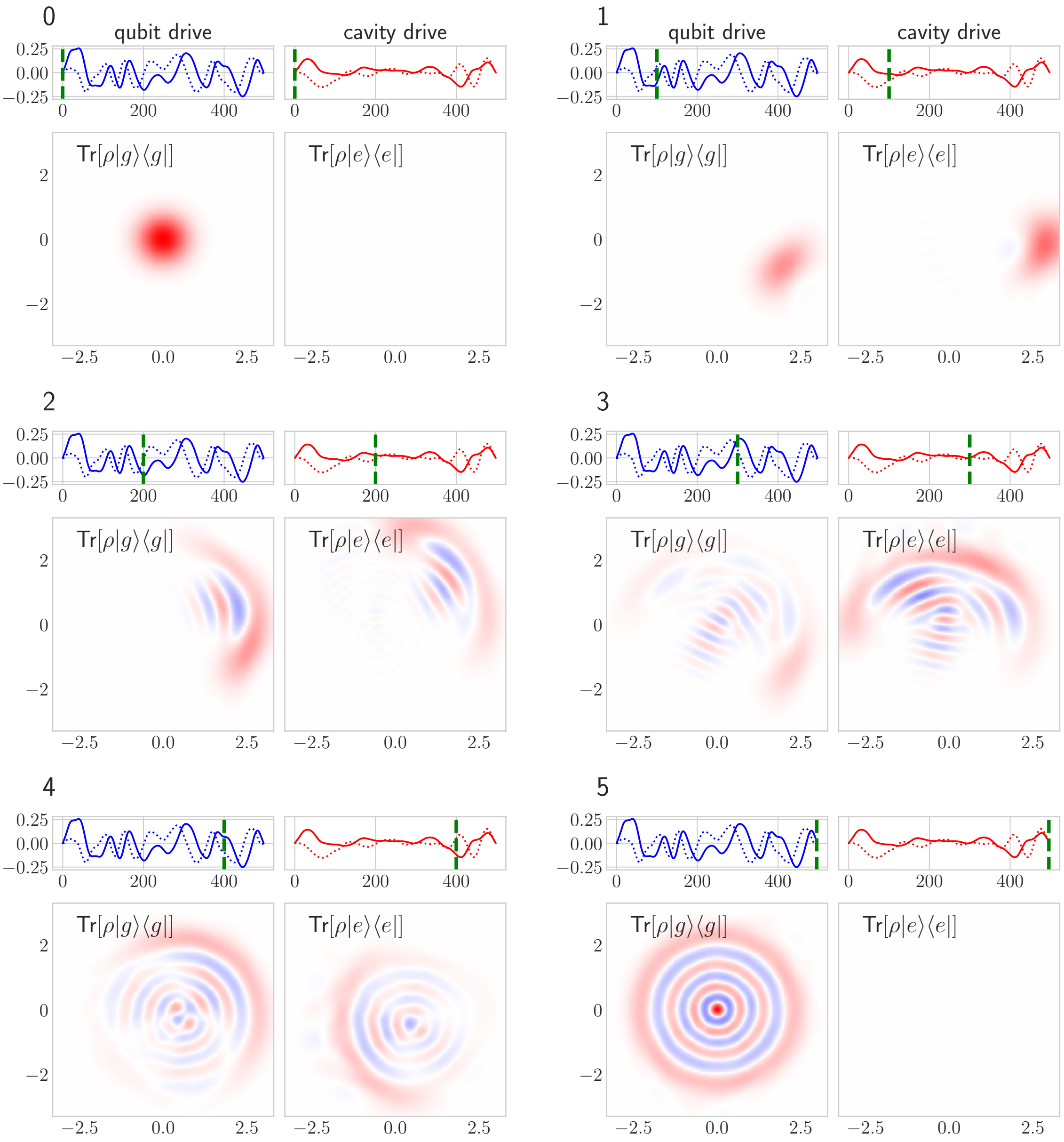


Figure 6.5: **Phase space trajectory of pulse creating 6 photons.** Sub-plots 1-6 are individual time slices of the cavity-transmon trajectory. For each time slice we characterize the cavity state conditional on both $|g\rangle$ and $|e\rangle$ with a Wigner function, which is normalized to the probability of occupying the respective transmon state. Note that this is not quite tomographically complete, which would require four Wigner-like functions per time slice (Vlastakis, 2015, §6.3.1). The trajectory shows how the Fock state is built up in phase space, gradually approaching the correct form seen in the final panel.

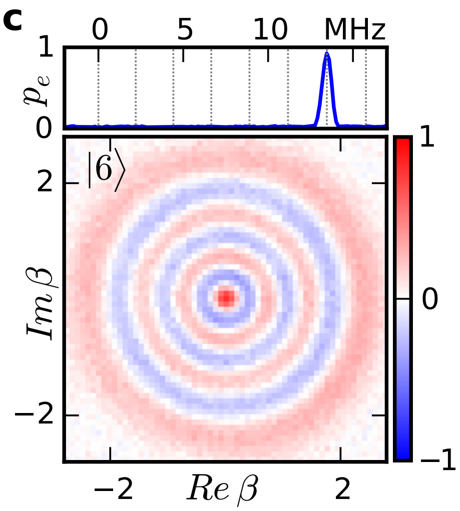


Figure 6.6: **Wigner function Fock state $|6\rangle$ as created by optimal control** Characterization of the oscillator state using Wigner tomography (bottom) and transmon spectroscopy (top), where grey dashed lines indicate the transition frequency associated with the first 7 Fock states. The single peak in the spectroscopy data directly reveals the oscillator's population due to the dispersive interaction giving a frequency shift of $6\chi/2\pi \approx 13$ MHz.

bandwidth, amplitude and photon number occupied during the pulse evolution? We can see in figures 6.7 and 6.6 that it is. In particular, the photon number distribution predicted by the simulation matches the observed trajectory quite well. While the dissipation-free simulation for this pulse has a transfer fidelity $> 99.9\%$, accounting for the known sources of decoherence, mostly transmon dephasing in this sample, reduces the prediction to 98.5%. It is difficult to confirm the fidelity of this Fock state experimentally to this level of accuracy, due to the issues of separating state and measurement fidelity in state tomography, but the best estimates from the transmon spectroscopy indicate a fidelity of 98%, arrived at by comparing the contrast of the vacuum $|0\rangle$ to the prepared state $|6\rangle$.

6.3 Alternating Hilbert spaces: Encode and decode

With the optimal control system brought online (see appendix F for the final code which produced our pulses), we can move on to our more concrete goal of manipulating cat-encoded quantum bits. Our target encoding is the orthogonal cat code specified in equations 3.56 and 3.57, with $\alpha = \sqrt{3}$. In order work with this code, we will need to be able to prepare and measure

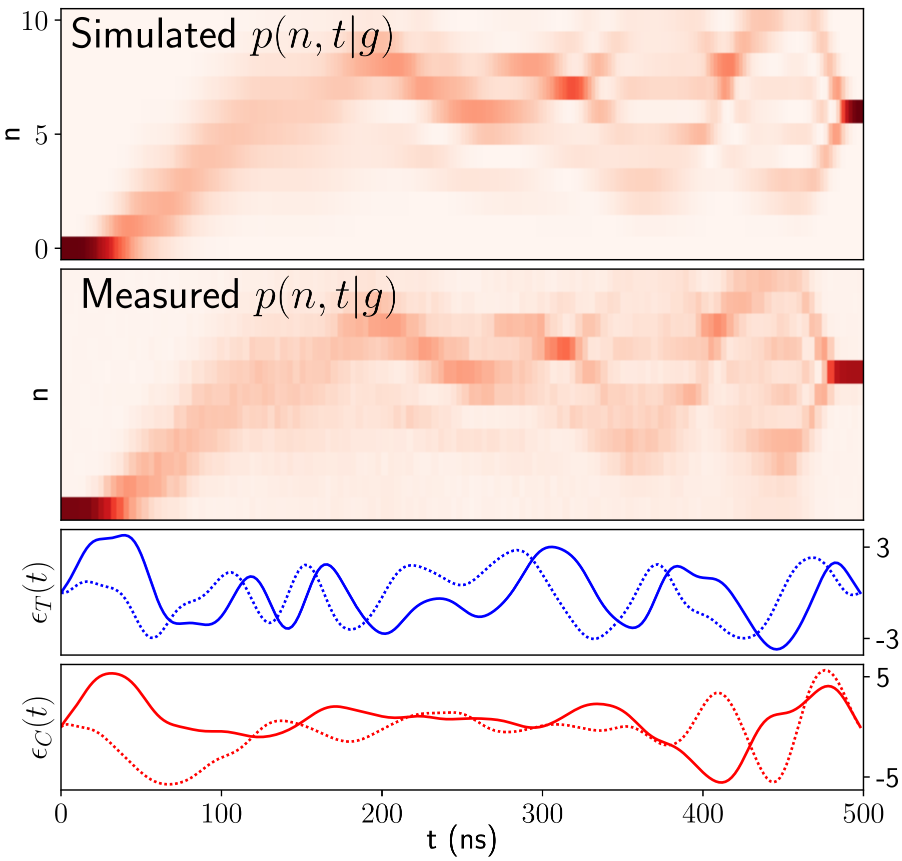


Figure 6.7: **Photon number trajectory of pulse creating 6 photons, compared with experiment.** Lower panels: optimized transmon and oscillator control waveforms of length approximately $2\pi/\chi$ to take the oscillator from vacuum to the 6-photon Fock state. Solid (dotted) lines represent the in-phase (quadrature) field component. Upper panels: oscillator photon-number population trajectory versus time conditioned on transmon in $|g\rangle$, both in simulation (top) and experiment (bottom). A complex trajectory occupying a wide range of photon numbers is taken to perform the intended operation.

its states. We note that we have a very good ability to prepare and characterize states in the transmon. We can extend this capability to the cavity if we can map quantum information from one system to the other. For this reason we create the *encode* ($\boldsymbol{U}_{\text{enc}}$) and *decode* ($\boldsymbol{U}_{\text{dec}}$) pulses (see figure 6.8a)

$$(\alpha\,|g\rangle + \beta\,|e\rangle) \otimes |0\rangle \xrightleftharpoons[\text{decode}]{\text{encode}} |g\rangle \otimes (\alpha\,|0_L\rangle + \beta\,|1_L\rangle) \tag{6.18}$$

These operations will coherently move quantum information between our easily addressed transmon and the cavity. These pulses are each $1100\ \text{ns} \approx 2.4 \times 2\pi/\chi$ in length, and are

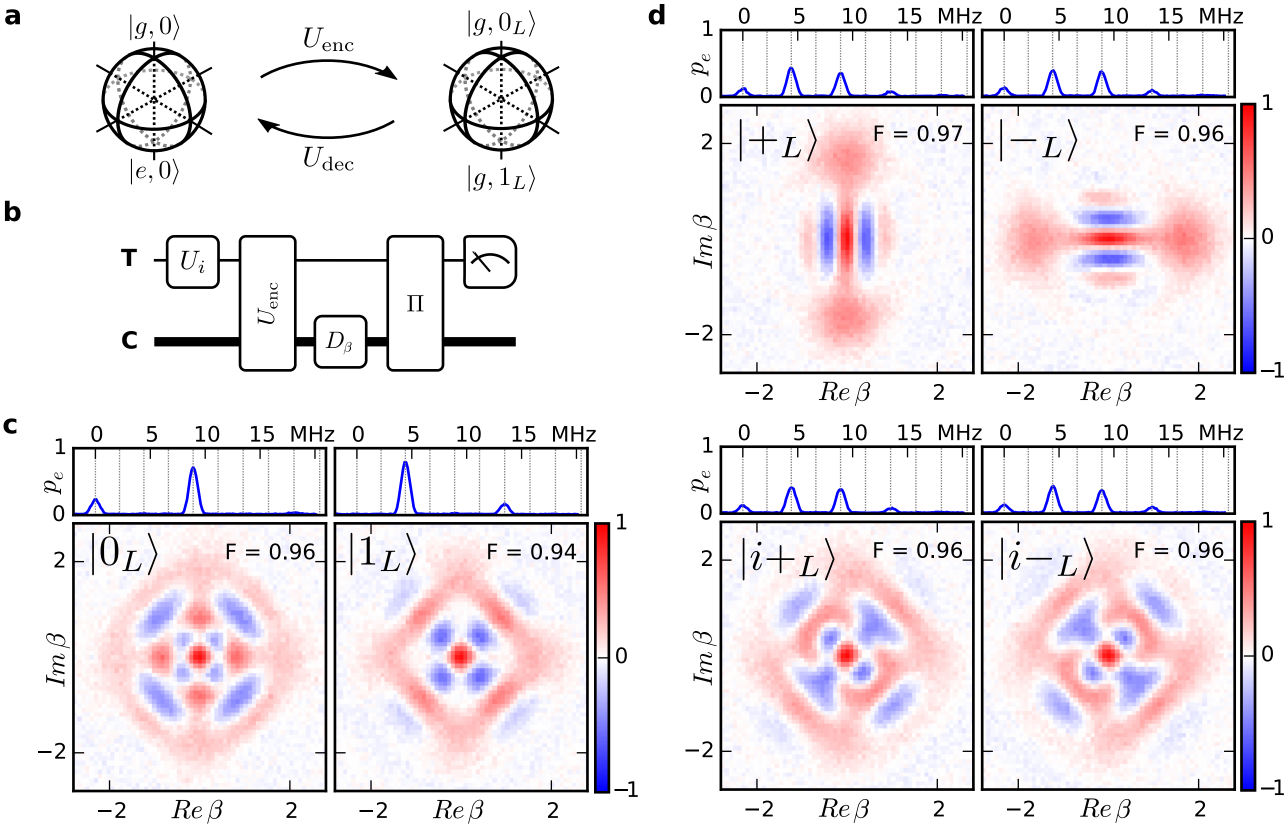


Figure 6.8: **Characterization of encoded states.** **a,** $\boldsymbol{U}_{\text{enc}}$ and $\boldsymbol{U}_{\text{dec}}$ are operations which coherently map between two distinct two-dimensional subspaces, represented by Bloch spheres. The first subspace consists of the transmon $|g\rangle$ and $|e\rangle$ levels, with the oscillator in the vacuum. The second is given by the oscillator-encoded states $|0_L\rangle$ and $|1_L\rangle$ (equations 3.56 and 3.57), with the transmon in the ground state. **b,** Wigner tomography sequence which characterizes the encoded states. A transmon state is prepared by applying an initial rotation $\boldsymbol{U}_i$ and is mapped to the oscillator using $\boldsymbol{U}_{\text{enc}}$. An oscillator displacement $\boldsymbol{D}_\beta$ followed by a parity mapping operation $\boldsymbol{\Pi}$ (implemented using an optimal control pulse) allows one to measure the oscillator Wigner function $W(\beta)$. The transmon can be re-used to measure the oscillator's parity because the encoding pulse leaves the transmon in the ground state with high probability ($p > 98\%$). **c,** Applying $\boldsymbol{U}_{\text{enc}}$ to the transmon states $|g\rangle$ and $|e\rangle$ produces states whose Wigner functions are consistent with the intended encoded basis states . A transmon spectroscopy experiment (top panel) illustrates that only photon number states with $n = 0 \bmod 4$ and $n = 2 \bmod 4$ are present for logical state $|0_L\rangle$ and $|1_L\rangle$ respectively. **d,** Applying $\boldsymbol{U}_{\text{enc}}$ to superpositions of the transmon basis states demonstrates that the relative phase is preserved and that $\boldsymbol{U}_{\text{enc}}$ is a faithful map between the transmon and logical qubit Bloch spheres. These states, on the equator of the Bloch sphere, are equally weighted superpositions of $|0_L\rangle$ and $|1_L\rangle$ and therefore contain all even photon numbers present in the basis states.

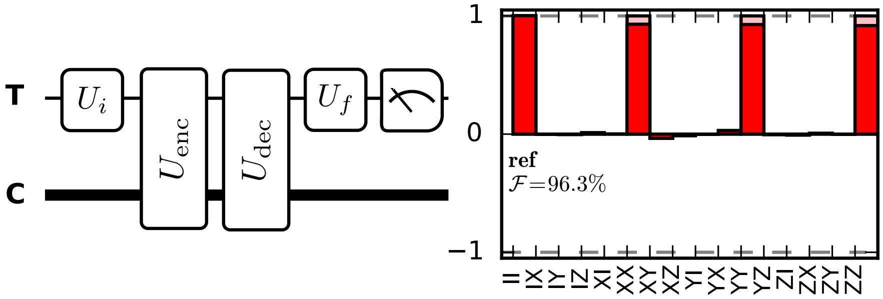


Figure 6.9: **Process tomography of encode-decode.** We perform *transmon* tomography on the process $\boldsymbol{U}_\text{dec}\boldsymbol{U}_\text{enc}$, which is ideally the identity. The result of the tomography is shown in the Pauli transfer representation (appendix C.2.1).

specified with a 2 ns time resolution. About 99% of the spectral content lies within a bandwidth of 33 MHz (27 MHz) for the transmon (oscillator) drive. The pulses which implement these operations can be seen in figure 6.10. In order to validate the encoding, we can perform Wigner tomography (see section 2.3.2, figure 6.8b), and ensure that the resulting state matches expectations. Depending on whether transmon state is $|g\rangle$ or $|e\rangle$ before applying the encode operation, the resulting cavity state could be a four-legged cat state with even or odd superparity. Because of the linearity of the encoding transformation, superposition states in the transmon are transformed into the corresponding superposition states in the cavity, producing either horizontal or vertical two-legged cats. In figure 6.8c-d we see the encoded states corresponding to the 6 cardinal points of the Bloch sphere (c.f. figure 3.2). The encoding pulse is not limited to these points, and works to produce arbitrary encoded states, given the corresponding transmon state. Maximum likelihood reconstruction of the density matrix associated with the measured Wigner functions indicates an average fidelity of 0.96. This metric underestimates the fidelity of $\boldsymbol{U}_\text{enc}$ because it is affected by several sources of error not intrinsic to the encoding operation itself, including error in the parity mapping and measurement infidelity.

We can test the decode operation by performing encode followed by decode. If these operations are ideal, the net transformation should be the identity operation. We can test this property by preparing different states in the transmon, and measuring what transmon state we end up in (figure 6.9). With this method, we can perform process tomography on the transmon (see appendix C.2), and calculate a fidelity to the identity of $\mathcal{F} \approx 96.4\%$. Some of this error comes from inaccuracy in the transmon state preparation and measurement (SPAM) errors, as we will discuss in the next section.

6.4 Testing encoded operations

This brings us to a discussion of the operations manipulating the cat code. Using the same set of conditions as those which generated the encode and decode, we create a universal set of gates on our logical qubit, which includes $\boldsymbol{\sigma}_x$ and $\boldsymbol{\sigma}_y$ rotations by π and $\pi/2$, as well as Hadamard and T gates (see appendix F for the final code which produced our pulses). We include as well an identity operation, which is as long as the other pulses, which intends to cancel the spurious evolution of the cavity under its Kerr non-linearity. Looking at these pulses in 6.10 is not too enlightening. We can note, however, that the T and identity operations, which do not change photon number, but only impart phases to the cavity, can be implemented with only transmon drive, as would be done if these operations were implemented in SNAP. This was a result of the optimization, which in the face of drive amplitude constraints, chose to make the cavity drive negligibly small, as it was not needed.[3]

In order to verify these operations, to begin with, we can prepare encoded states in the cavity, apply the operation, and then characterize the cavity state (figure 6.11). While this is sufficient to establish qualitative agreement, a more precise evaluation can be determined from process tomography (figure 6.12). Process tomography provides a full characterization of a quantum operation, but depends on pre-existing trusted operations and measurements which are not available for our encoded subspace. However, an indirect characterization of a gate $\boldsymbol{U}_X$ on our logical qubit can be performed using the operation $\boldsymbol{U}_{\text{dec}}\boldsymbol{U}_X\boldsymbol{U}_{\text{enc}}$, which maps the transmon subspace onto itself. This allows one to use the trusted state preparations and measurements on the transmon to perform tomography on the composite process. The reconstructed process matrices show qualitative agreement with the intended encoded qubit gates. We can break the calculated infidelity down into 3 parts: transmon preparation and measurement error, encode-decode error and gate error. Using the experimentally determined process fidelities

[3] If we had attempted to make these pulses as fast as possible, it is likely that a cavity drive would still be needed, as the transmon-only approach has a strong speed limit.

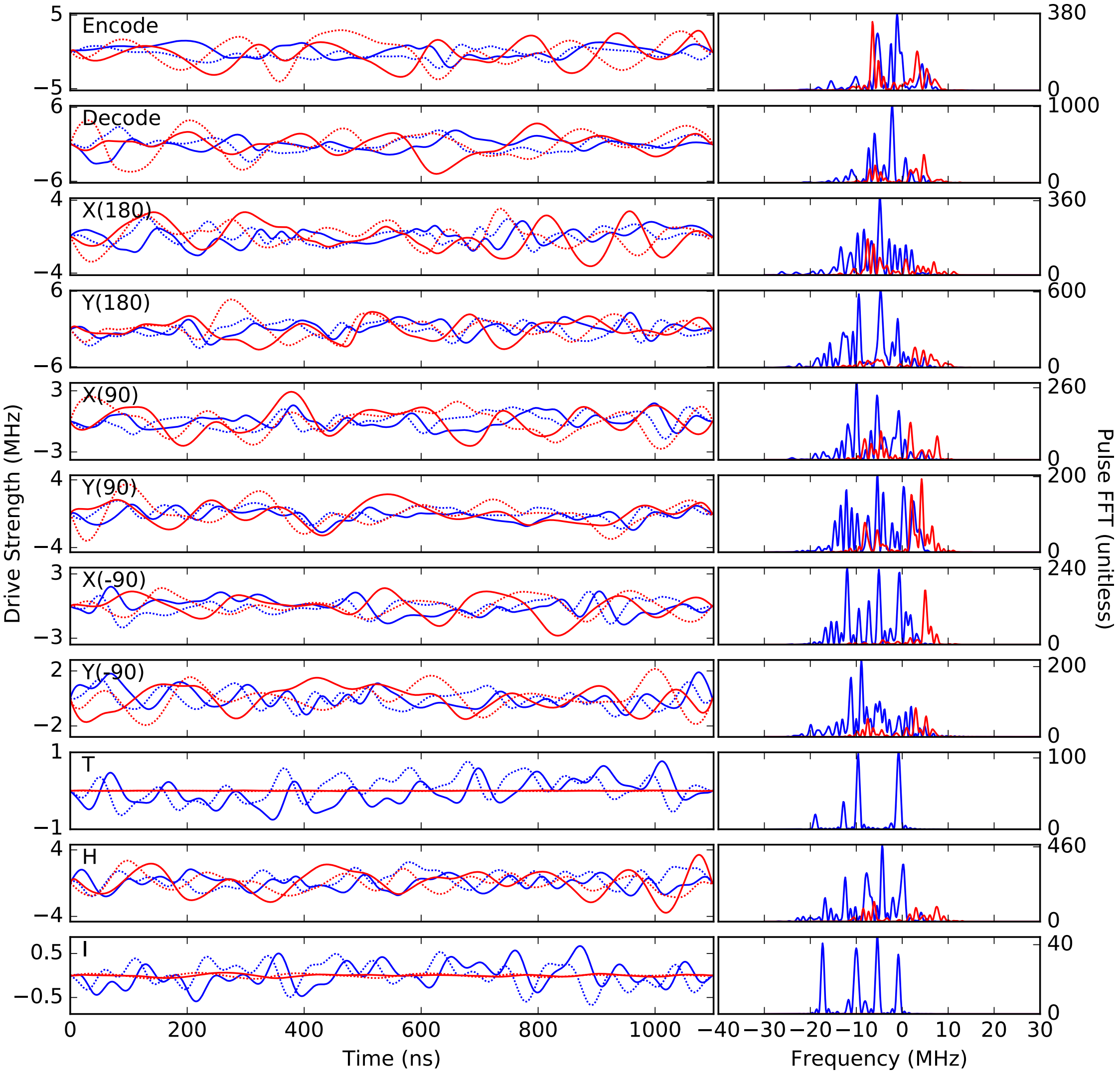


Figure 6.10: **Optimized pulse waveforms.** In the first column, we plot the complex waveforms $\epsilon_T(t)$ and $\epsilon_C(t)$. In the second column, we show the Fourier spectrum $|\tilde{\epsilon}(\omega)|^2$. Blue (red) lines correspond to drives on the transmon (oscillator). Solid (dotted) lines correspond to the in-phase (quadrature) component of the drive. Note that the I and the T gate do not have to change the photon number distribution, but only have to apply different phases to each Fock state component. This can be done by manipulating the transmon (Heeres et al., 2015) only; GRAPE finds a solution with a very small oscillator drive amplitude as well.

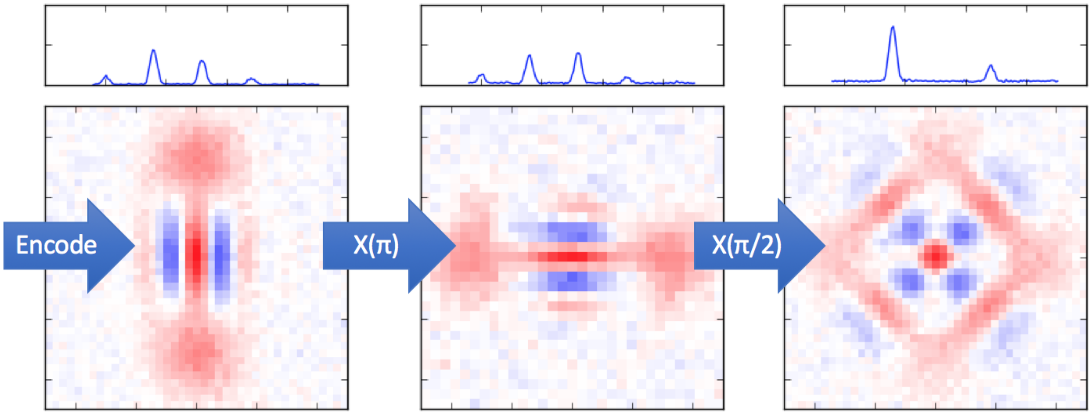


Figure 6.11: **Wigner functions demonstrating the action of encoded operations.** We see the in progressive steps the application of $\boldsymbol{U}_{\text{enc}}$, $\boldsymbol{U}_X\boldsymbol{U}_{\text{enc}}$, and $\boldsymbol{U}_{X2}\boldsymbol{U}_X\boldsymbol{U}_{\text{enc}}$, resulting in the states $|+_L\rangle \rightarrow |-_L\rangle \rightarrow |1_L\rangle$

both without any operation $\mathcal{F}_{\text{PT}}(\text{No Op.}) = 0.982$, as well as with the encode and decode pulses $\mathcal{F}_{\text{PT}}(\boldsymbol{U}_{\text{dec}}\boldsymbol{U}_{\text{enc}}) = 0.964$, we estimate an infidelity contribution of approximately 1.8% for each of the first two components. To account for these factors to first order, the infidelity of operations on the encoded qubit are reported relative to $\mathcal{F}_{\text{PT}}(\boldsymbol{U}_{\text{dec}}\boldsymbol{U}_{\text{enc}})$. We find an average infidelity of 0.75% over our set of 9 gates (see table 6.1).

In order to establish the fidelity of this set of operations more accurately, we perform randomized benchmarking (Magesan et al., 2011) (RB) on our encoded qubit (see appendix C.4). In this protocol, the encode operation is applied, followed by a random selection of N operations from our gate set. Then the operation corresponding to the inverse of the product of the preceding N operations is applied, followed by the decode operation and transmon readout. We do not repeat the same sequence, but instead draw a new random sequence for each single-shot measurement. From the resulting data (figure 6.13) we infer an average gate fidelity of 0.991.

The infidelity of each of the individual gates is isolated using interleaved randomized benchmarking (Magesan et al., 2012) (iRB), which alternates between a single fixed and a random gate. Comparing the fitted decay constants of the RB and iRB results allows us to extract the fidelity of the fixed gate. The results are summarized in table 6.1, together with the gate fidelities based on process tomography (figure 6.12) and Lindblad master equation simulations accounting for finite T_1 and T_2 of the transmon and oscillator (see table 5.1). We note that all

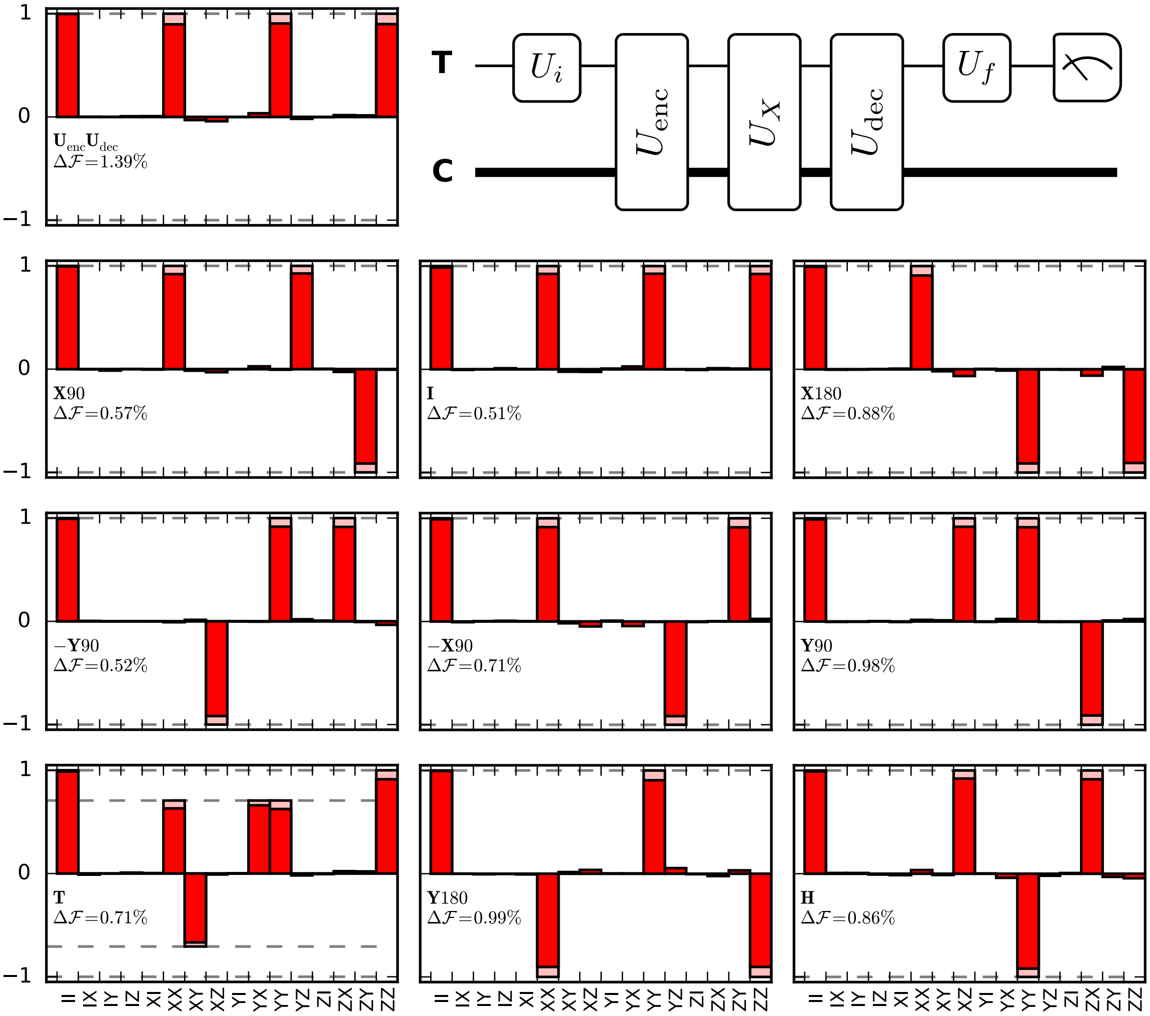


Figure 6.12: **Process tomography of operations on encoded qubit.** In order to characterize a gate U_X on the encoded qubit, transmon process tomography is performed on the operation $\boldsymbol{U}_{\text{dec}}\boldsymbol{U}_X\boldsymbol{U}_{\text{enc}}$. Process tomography is implemented by performing an initial transmon rotation $\boldsymbol{U}_i$ right after state preparation, as well as a final transmon rotation $\boldsymbol{U}_f$, right before measurement of the transmon. We show the process tomography results for selected operations. The process tomography yields an estimated quantum channel G. We represent this channel in the Pauli transfer representation. The bar labeled with operators AB represents $\text{Tr}\left(AG(B)\right)/2$. Red and pink bars indicate the experimental and ideal values, respectively. The infidelity $\Delta\mathcal{F}_{\text{PT}}$ of operation $\boldsymbol{U}_X$ is estimated as the difference between $\mathcal{F}_{\text{PT}}(\boldsymbol{U}_{\text{dec}}\boldsymbol{U}_X\boldsymbol{U}_{\text{enc}})$ and $\mathcal{F}_{\text{PT}}(\boldsymbol{U}_{\text{dec}}\boldsymbol{U}_{\text{enc}}) = 0.964$. The selected set of operations, $\{X180, X90, Y90, T\}$, allows universal control of the logical qubit.

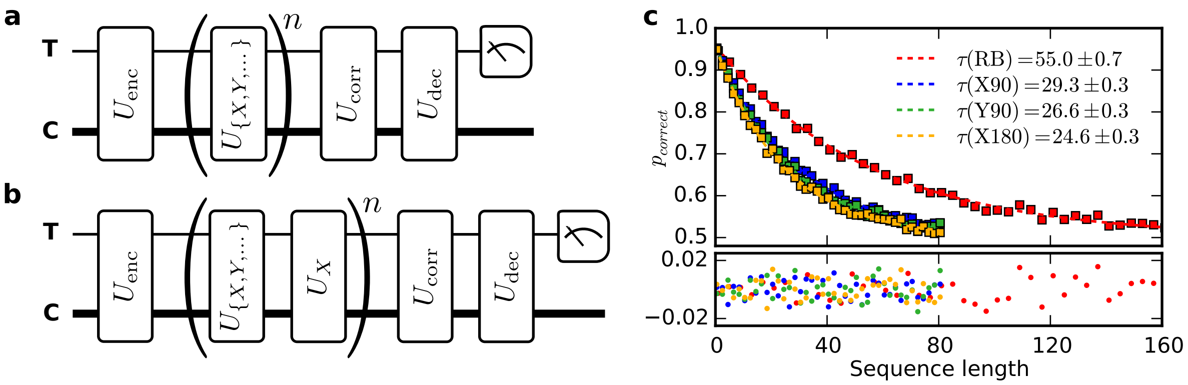


Figure 6.13: **Randomized benchmarking of operations on encoded qubit. a,** Randomized benchmarking (RB) sequence. In RB a sequence of Clifford operations of length n is chosen at random ($\boldsymbol{U}_{\{X,Y,\ldots\}}$), followed by the operation which inverts the effect of the sequence ($\boldsymbol{U}_{\text{corr}}$). In order to apply this technique to the operations on the encoded qubit, we begin the experiment by encoding, and decode before measurement. Our implementation of RB creates a new random gate sequence for every measurement, and is thus not biased by the distribution of sequences which are measured. **b,** Interleaved randomized benchmarking (iRB) sequence: In order to establish the fidelity of a single operation (here, $\boldsymbol{U}_X$), the operation is interleaved with random operations, and the benchmarking result is compared with the non-interleaved case. **c,** The probability of measuring the correct result versus sequence length n is fit to a two parameter model $p_{\text{correct}} = 0.5 + Ae^{-n/\tau}$. The lower panel shows the fit residuals. Each data point is the result of 2000 averages, with a new sequence realization every shot. The error averaged over all gates is computed as $r = (1 - e^{-1/\tau(\text{RB})})/2$ (Magesan et al., 2011). The average error for a single gate X is computed as $r(X) = (1 - e^{1/\tau(X) - 1/\tau(\text{RB})})/2$ (Magesan et al., 2012).

gates are implemented with an approximately equal infidelity of 1% and that process tomography and iRB yield consistent results. While several sources of decoherence are accounted for in the master equation simulations, the dominant source of infidelity in the model is transmon dephasing ($T_2 \approx 43\ \mu$s). The strong agreement between simulations and experiment indicates that the infidelity is primarily caused by decoherence and that additional contributions associated with imperfections in the model Hamiltonian and the applied pulses are a significantly smaller effect.

6.5 Empirical tuning

In order to get the maximum amount of performance out of the pulses, it was necessary to perform compensatory pre-distortions, in order to cancel the effect of a non-ideal transfer

n Gate	$1-\mathcal{F}_{\mathsf{RB}}$ (%)	$\Delta\mathcal{F}_{\mathsf{PT}}$ (%)	$1-\mathcal{F}_{\mathsf{sim}}$ (%)
I	0.78 ± 0.06	0.51	0.31
X90	1.34 ± 0.09	0.57	0.78
-X90	1.54 ± 0.10	0.71	0.83
X180	1.89 ± 0.12	0.88	1.09
Y90	1.63 ± 0.11	0.98	0.76
-Y90	1.38 ± 0.09	0.52	0.75
Y180	2.18 ± 0.14	0.99	1.67
H	1.58 ± 0.11	0.86	1.00
average	1.54 ± 0.10	0.75	0.90
$\boldsymbol{U}_{\mathsf{enc}}\boldsymbol{U}_{\mathsf{dec}}$	2.89 ± 0.03	1.39	1.76
T	-	0.71	0.40

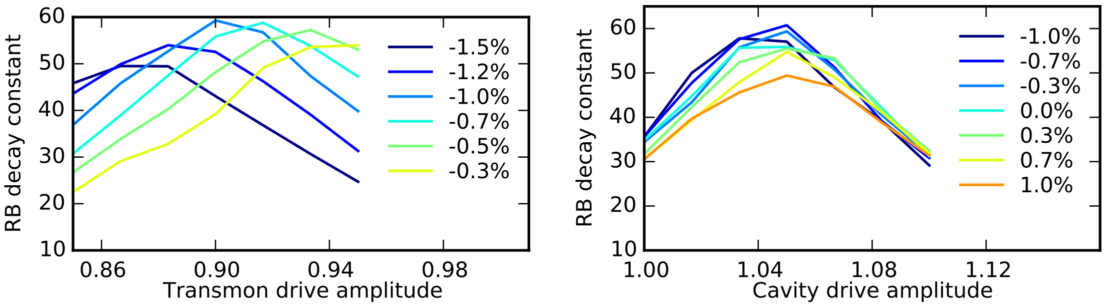
Table 6.1: **Operation fidelities.** Measured and simulated gate infidelities. All fidelities reported are average gate fidelities $\mathcal{F}(\mathcal{E}_1, \mathcal{E}_2) \equiv \int \mathrm{d}\psi F(\mathcal{E}_1(\psi), \mathcal{E}_2(\psi))$, where F is the usual quantum state fidelity $F(\rho_1, \rho_2) = \mathsf{Tr}(\sqrt{\rho_1}\rho_2\sqrt{\rho_1})$. $\mathcal{F}_{\mathsf{RB}}$, $\Delta\mathcal{F}_{\mathsf{PT}}$ and $\mathcal{F}_{\mathsf{sim}}$ are the values extracted from interleaved randomized benchmarking, process tomography (see figure 6.12) and simulations using the Lindblad master equation respectively. The row labeled "average" gives the fidelities averaged over the first 8 gates, which is the set used in the standard randomized benchmarking experiment. The T gate is separated since, as a non-Clifford operation, it is incompatible with standard randomized benchmarking.

function corresponding to the physical lines and components interposing the AWG and device. We perform the distortions described mathematically in section 4.8, specifically, using a first order polynomial to model the transfer function, resulting in the five parameter model:

$$\tilde{\epsilon}_T = A_T \mathscr{F}^{-1}\left[(1 + B_T\omega)\mathscr{F}[\epsilon_T]\right] \tag{6.19}$$

$$\tilde{\epsilon}_C = A_C \mathscr{F}^{-1}\left[(1 + B_C\omega)e^{i\omega\tau}\mathscr{F}[\epsilon_C]\right] \tag{6.20}$$

The overall pulse amplitude correction is captured by A_T and A_C. The "dispersion" (really, frequency dependent loss) of the lines is captured by the parameters B_T and B_C. Finally, the delay between the two channels is captured by τ. We use the randomized benchmarking protocol to perform empirical tuning of these five parameters (Egger and Wilhelm, 2014; Kelly et al., 2014). We optimize the amplitude and dispersion parameters per channel simultaneously, but otherwise tune the parameters independently, resulting in the three parameter scans seen in figures 6.14 and 6.15.



(a) Transmon line dispersion tune-up

(b) Transmon line dispersion tune-up

Figure 6.14: **Dispersion and amplitude optimization** The randomized benchmarking decay constant versus transmon drive amplitude for several different dispersion compensation values (in % per MHz). The amplitude must be changed in parallel when the dispersion value is adjusted, necessitating the 2D sweep.

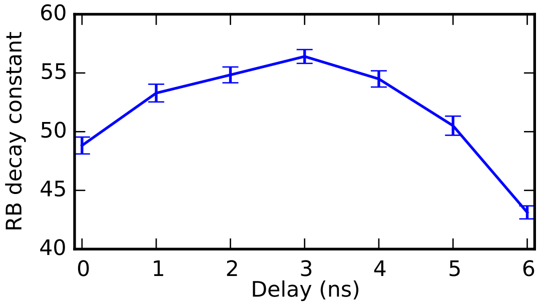


Figure 6.15: **Delay optimization** The delay between the transmon and cavity pulses is necessary to compensate for the varying cables and components which conduct the signals from the AWG to the sample. The delay is implemented via a linear phase shift in the Fourier domain, allowing us to use effective delays smaller than the AWG time discretization of 2 ns.

Chapter 7

Venturing forth in frequency space: sideband drives

Up to this point we have employed a model of a coupled transmon cavity system in the fully "dispersive" regime. Under the assumption that the various modes of the system are sufficiently detuned from one another, we can incorporate bilinear coupling terms (i.e. those which are only second order in the ladder operators) to form a set of normal modes, and model the interactions and nonlinearities as being diagonal in the photon number basis of these normal modes. This derivation relied upon a critical assumption however: that the frequencies of the driving fields would be constrained to a small window in the vicinity of the modes' resonant frequencies. In the experiments discussed in chapters 2 and 6, this was a valid assumption, by design. Even in the general optimal control case, we considered only pulses which were produced by mixing a narrow bandwidth (< 100 MHz) signal with a local oscillator centered on the cavity or transmon resonant frequencies. This allowed us to assume that each pulse acted only via the $\boldsymbol{a}$, $\boldsymbol{a}^\dagger$, $\boldsymbol{\sigma}_-$ and $\boldsymbol{\sigma}_+$ operators, in the rotating frame of the local oscillator.

While this is a useful simplifying condition, and as we have seen, allows for in principle universal control of the system, there are powerful control techniques which are only accessible via driving outside of these frequency ranges.

7.1 Four-wave mixing: A cornucopia of terms

We begin by considering a single mode Hamiltonian (the multi-mode generalization is straightforward), involving three terms. The first gives the resonant frequency of the mode ω_a. The second is a linear drive, with amplitude ϵ, applied to that mode, rotating at some other frequency ω_d. The last is "the rest" of the Hamiltonian, some general function of the mode's ladder operators.

$$\boldsymbol{H}_1 = \omega_a \boldsymbol{a}^\dagger \boldsymbol{a} + \epsilon \left(e^{i\omega_d t} \boldsymbol{a} + \text{h.c.} \right) + f \left(\boldsymbol{a}, \boldsymbol{a}^\dagger \right) \tag{7.1}$$

The goal of the procedure is to eliminate the first two terms, and see how the final term is transformed. We can begin by going into a frame rotating at frequency ω_d to eliminate the time dependence of the second term (see appendix A.3):

$$\boldsymbol{H}_2 = \Delta \boldsymbol{a}^\dagger \boldsymbol{a} + \epsilon \left(\boldsymbol{a} + \boldsymbol{a}^\dagger \right) + f \left(e^{-i\omega_d t} \boldsymbol{a}, e^{i\omega_d t} \boldsymbol{a}^\dagger \right), \tag{7.2}$$

where $\Delta = \omega_a - \omega_d$. Next we can go into a displaced frame with displacement $-\frac{\epsilon}{\Delta}$ (see appendix A.4) to eliminate the second term entirely.

$$\boldsymbol{H}_3 = \Delta \boldsymbol{a}^\dagger \boldsymbol{a} + f \left(e^{-i\omega_d t} \left(\boldsymbol{a} - \frac{\epsilon}{\Delta} \right), e^{i\omega_d t} \left(\boldsymbol{a}^\dagger - \frac{\epsilon}{\Delta} \right) \right) \tag{7.3}$$

Finally we can go into a second rotating frame to eliminate the first term.

$$\boldsymbol{H}_4 = f \left(\boldsymbol{a} e^{-i\omega_a t} - \frac{\epsilon}{\Delta} e^{-i\omega_d t}, \boldsymbol{a}^\dagger e^{i\omega_a t} - \frac{\epsilon}{\Delta} e^{i\omega_d t} \right) \tag{7.4}$$

This is our final form. We can summarize this result by saying that, in the interaction picture, the ladder operators rotate at their resonant frequency, and the drive appears as a scalar term added to the ladder operator each place it appears. This scalar term has magnitude inversely proportional to the detuning from resonance, and itself rotates at the drive frequency.

Now we can apply this knowledge to the specific case of the Josephson junction Hamiltonian to see how 4-wave mixing terms emerge. We take a model of several electromagnetic modes each of which has some coupling to a Josephson junction:

$$\boldsymbol{H} = \boldsymbol{H}_0 + \boldsymbol{H}_{\text{JJ}} \tag{7.5}$$

$$\boldsymbol{H}_0 = \sum_k \omega_k^{(0)} \left(\boldsymbol{a}_k^{(0)}\right)^\dagger \boldsymbol{a}_k^{(0)} \tag{7.6}$$

The superscript (0) indicates that these are the "bare" mode operators and resonant frequencies, independent of the junction. The Josephson junction Hamiltonian is given by

$$\boldsymbol{H}_{\text{JJ}} = \omega_J \cos(\boldsymbol{\Phi}), \tag{7.7}$$

Where Φ is the total flux across the junction, in units of the reduced flux quantum $\phi_0 = \frac{\hbar}{2e}$. This flux is constituted by a linear sum of the fluxes of each of the participating modes,

$$\boldsymbol{\Phi} = \sum_k \phi_k^{(0)} \boldsymbol{a}_k^{(0)} + \text{h.c.} \tag{7.8}$$

where the ϕ_k are participation factors corresponding to the flux induced across the junction from the zero-point fluctuations of the corresponding mode k. Assuming the participations are small, and ignoring the scalar part, the Josephson junction Hamiltonian can be expanded in powers:

$$\boldsymbol{H}_{\text{JJ}} = \omega_J \left(-\frac{\boldsymbol{\Phi}^2}{2!} + \frac{\boldsymbol{\Phi}^4}{4!} - \frac{\boldsymbol{\Phi}^6}{6!} + \cdots \right) \tag{7.9}$$

We note that the terms which are second order in the ladder operators, which come from both $\boldsymbol{H}_0$ and $\boldsymbol{H}_{\text{JJ}}$, can be grouped together and written as

$$\boldsymbol{H}_0 - \omega_J \frac{\boldsymbol{\Phi}^2}{2} = (\vec{\boldsymbol{a}}^{(0)})^T X \vec{\boldsymbol{a}}^{(0)}. \tag{7.10}$$

for some matrix[1] of coefficients X, and for a vector of ladder operators, defined as

$$\vec{\boldsymbol{a}}^{(0)} = \begin{pmatrix} \boldsymbol{a}_1^{(0)} & \cdots & \boldsymbol{a}_N^{(0)} & \left(\boldsymbol{a}_1^{(0)}\right)^\dagger & \cdots & \left(\boldsymbol{a}_N^{(0)}\right)^\dagger \end{pmatrix}. \tag{7.11}$$

[1] While X is a matrix, we write X in standard font rather than boldface to emphasize that it is not an operator on Hilbert space

The matrix of coefficients can be diagonalized, the effect of which is to produce a new set of eigenmodes with ladder operators $\boldsymbol{a}_k$ (note no superscript) which are linear combinations of the $\{\boldsymbol{a}_j^{(0)}\}$ and $\{(\boldsymbol{a}^\dagger)^{(0)}\}$. The resulting Hamiltonian can be written

$$\boldsymbol{H} = \sum_k \omega_k \boldsymbol{a}_k^\dagger \boldsymbol{a}_k + \omega_J \left(\frac{\boldsymbol{\Phi}^4}{4!} - \frac{\boldsymbol{\Phi}^6}{6!} + \cdots \right) \tag{7.12}$$

$\boldsymbol{\Phi}$ can be represented in the new basis as well as $\boldsymbol{\Phi} = \sum_k \phi_k \boldsymbol{a}_k + \text{h.c.}$ where usually, but not necessarily, we have $\phi_k \approx \phi_k^{(0)}$.

If we add drives to this system, with amplitudes $\{\Omega_k\}$ and frequencies $\{\omega_{d,k}\}$, we can go through the process described before of going to a displaced rotating frame, eliminating the drives and the detuning part $\sum_k \omega_k \boldsymbol{a}_k^\dagger \boldsymbol{a}_k$, and using the prescription from equation 7.4 to transform the remaining part.[2]

$$\boldsymbol{H} = \frac{\omega_J}{4!} \left(\sum_k \phi_k (\boldsymbol{a}_k e^{i\omega_k t} + \xi_k e^{i\omega_{d,k} t}) + \text{h.c.} \right)^4 + \cdots \tag{7.13}$$

Here we have defined the unitless mode displacement amplitude $\xi_k = \Omega_k/(\omega_k - \omega_{d,k})$. For n modes there are $\binom{4n+3}{4}$ unique terms in the expansion of the fourth-order term, one corresponding to every choice of 4 elements from the set $\{\boldsymbol{a}_k, \boldsymbol{a}_k^\dagger \xi_k, \xi_k^*\}$, with replacement. However, the vast majority of these terms will be negligible, since they will rotate at a fast frequency. Barring unintended collisions, only two types of terms will remain. The first is those which are "diagonal" in that every $\boldsymbol{a}_k$ or ξ_k is matched with a corresponding $\boldsymbol{a}^\dagger$ or ξ_k^* in the term. These produce anharmonicities ($(\boldsymbol{a}^\dagger)^2 \boldsymbol{a}^2$), dispersive shifts ($\boldsymbol{a}^\dagger \boldsymbol{a} \boldsymbol{b}^\dagger \boldsymbol{b}$), or Stark shifts ($|\xi|^2 \boldsymbol{a}^\dagger \boldsymbol{a}$). The second type is terms which we *intentionally* make resonant by the choice of our drive frequency.

Therefore, we have essentially the ability to pick Hamiltonian terms involving two or three system excitations (with two or one pump excitations making up the energy difference and bringing the total up to four) and bring them forth. These observations have been used extensively in the field of quantum limited amplifiers (Vijay et al., 2009; Bergeal et al., 2010). More recently these processes have been used to create operations in high-Q cQED experiments.

[2] Here we assume there is one drive per mode, although it is possible to do the same treatment with multiple drives per mode, although the notation and indexing becomes more cumbersome

In the following sections, we will look at several examples of such processes that this method enables.

7.2 Q-Switching for faster system reset

Long-lived storage cavities are obviously a boon for the fidelity of storage and manipulation of quantum information. But at the same time it presents an annoyance to the experimenter. Before we can perform an experiment, we must prepare the system in a known, low-entropy state. While there are shortcuts which can be achieved through measurement, or by designing protocols which allow multiple initial states, in general this is achieved by waiting a period of time much longer than the system's lifetime, allowing the system to thermalize with its environment. As the lifetime becomes longer, this method becomes slower.

What we would like to do is be able to switch the decay rate of the system (and therefore the quality factor Q, hence Q-switch) from as slow as possible, during the experiment, to as fast as reasonably possible, during the reset after the experiment. We can achieve this by exploiting the fast decay rate of the *readout* mode. We engineer a swap interaction which allows excitations to jump between the readout and the storage cavity, allowing the cavity to effectively inherit some of the loss of the readout.

To do this, we introduce two pump tones, one at $\omega_c+\Delta$ and the other at $\omega_{\text{RO}}+\Delta$, for some detuning value Δ (figure 7.1). This allows us to make the following Hamiltonian effectively resonant:

$$\boldsymbol{H} = g\left(\boldsymbol{a}^\dagger\boldsymbol{r} + \boldsymbol{a}\boldsymbol{r}^\dagger\right) \tag{7.14}$$

We can determine g from the prefactors in equation 7.13

$$g = \frac{\omega_J}{4!}\phi_c^2\phi_r^2\frac{\Omega_c\Omega_r}{\Delta^2}. \tag{7.15}$$

Given a loss rate on the readout κ_{RO}, The effective loss rate of the storage cavity depends on the relationship between κ_{RO} and g. When $g \gg \kappa_{\text{RO}}$ the excitation swaps rapidly back and forth between the cavity and the readout mode, spending about half of its time, and therefore

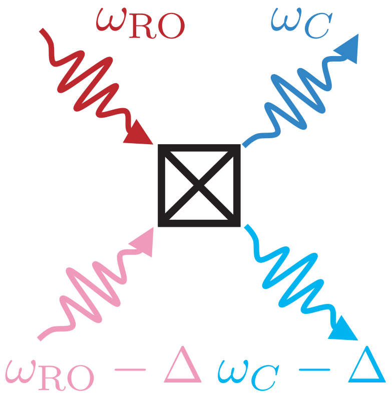


Figure 7.1: **Four wave mixing protocol for Q-Switching** The use of two pump photons allows the conversion of readout photons in storage photons and vice-versa.

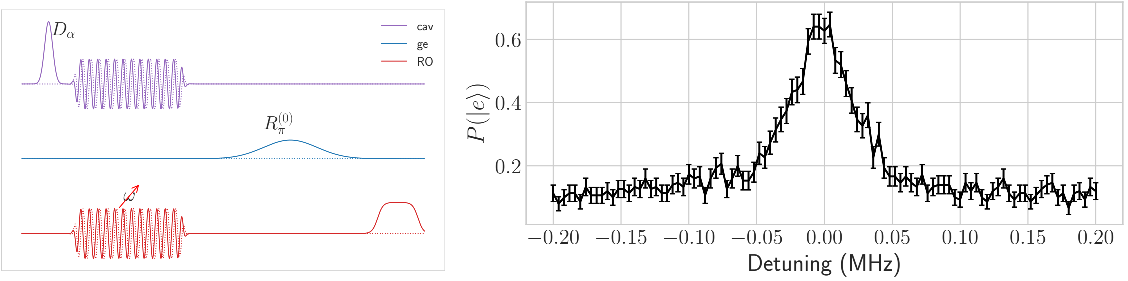


Figure 7.2: **Sequence for tuning up a Q-switch pump.** After preparing some significant number of photons in the cavity, we apply the Q-switch by driving both the cavity and readout modes detuned from resonance. We then probe the probability of having zero photons with a transmon selective pulse and transmon readout. Nominally the detuning of the two drives should be the same, but because of Stark shifts, one should scan the frequency of one of the drives in a small window to identify the ideal driving point.

experiencing about half of the effective decay rate. $\kappa_{c,\text{eff}} \approx \kappa_{\text{RO}}/2$. In the opposite limit, $g \ll \kappa_{\text{RO}}$ the decay rate is counter-intuitively inversely proportional to κ_{RO} as $\kappa_{\text{c,eff}} \approx 4g^2/\kappa_{\text{RO}}$ (Axline, 2018, §6.1.3). The optimum (fastest) decay rate[3] is achieved with $2g = \kappa_{\text{RO}}$.

7.3 Creating photons one at a time

While we showed in section 6.2 how it was possible to create Fock states in the storage mode via optimal control pulses and the dispersive interaction, a constructive approach is possible

[3] Decay rate is a bit of a misnomer, as the occupation of the readout mode is not exactly an exponential decay. It has the form of a critical damping curve $e^{-t/\tau}(a + tb)$ for some constants a and b.

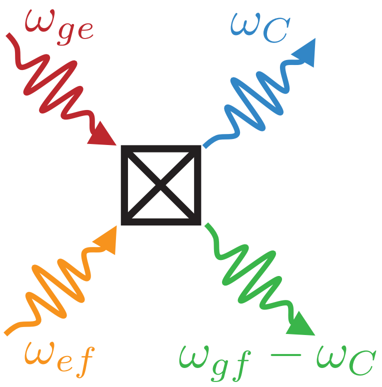


Figure 7.3: **Four wave mixing protocol for photon preparation** The use of a single pump photons allows the conversion of two transmon photons into a single storage photons and vice-versa.

using sidebands. In the same way that Q-switching allowed us to import the energy decay properties of the readout to the storage cavity, a different sideband will allow us to import the single-photon creation ability of the transmon to the storage cavity.

To do so, we will employ drives at frequencies $\omega_{ge}+\Delta$ and $\omega_c+\Delta$ to effect the Hamiltonian $g\boldsymbol{a}^\dagger |g\rangle\langle e| +$ h.c.. We note that this drive, acting for a time $\frac{\pi}{2g}$ will take $|e,0\rangle$ to exactly $|g,1\rangle$. Therefore, by performing a π pulse, followed by a transmon-exchange pulse we can inject a photon into the cavity. We can inject a second photon by performing a subsequent transmon π pulse and exchange operation. However, the second exchange operation must be of time $\frac{\pi}{2\sqrt{2}g}$, as the exchange interaction has a Bosonic enhancement factor which speeds it up as more photons are injected. To do $|e,n\rangle$ to $|g,n+1\rangle$ takes time $\frac{\pi}{2\sqrt{n}g}$.

We can also do this operation in a faster and simpler way by using the $|f\rangle$ state. Using a single drive of frequency $\omega_d = \omega_{gf} - \omega_c$, we can activate the following Hamiltonian (figure 7.3)

$$\boldsymbol{H} = g\left(\boldsymbol{a}^\dagger |g\rangle\langle f| + \boldsymbol{a} |f\rangle\langle g|\right) \tag{7.16}$$

Since the rate g is proportional to the junction participation factors ϕ_k involved, using more transmon excitations rather than cavity mode excitations can result in a faster operation for a given mode displacement[4]. This method uses one additional transmon excitation (requiring both ge and ef π pulses as preparation steps), but uses only a single pump tone.

[4] although generating that displacement may be harder if the needed frequency is very far from resonance

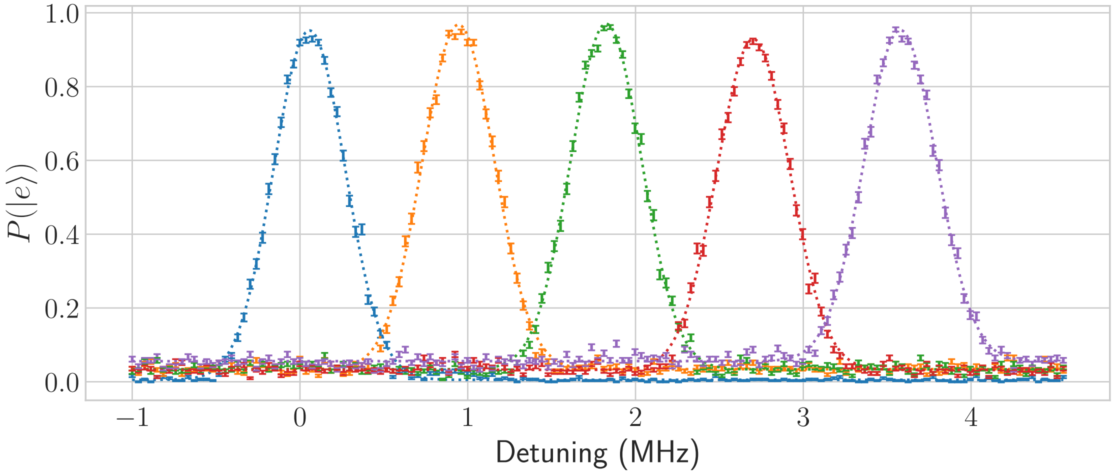


Figure 7.4: **Demonstration of deterministic photon preparation via sideband driving.** Here we see the results of preparing photon number states $|0\rangle$ through $|4\rangle$ via single-pump sideband driving as described in equation 7.16. This qubit spectroscopy vs preparation number shows that the prepared state is high fidelity. The main failure mechanism is the transmon ending in the incorrect state, resulting in the slightly higher background for the higher photon number states. No postselection was performed in this particular experiment.

7.4 Engineered dissipation

We saw in section 7.2 how sidebands could enable the modification of dissipative properties by turning on conversion processes to modes with large native dissipation. In section 7.3 we saw how sidebands could be used to address inherently multi-photon processes, such as creating or removing two transmon excitations at a time. These two processes can be combined to allow the creation of *multi-photon dissipation*, an extremely interesting concept with surprising application. Most interestingly, we can drive a pump at frequency $2\omega_c - \omega_{\text{RO}}$ to convert two photons in the storage cavity into a single photon in the readout cavity. The reverse process is supressed in the limit that the forward rate g_{2ph} is much smaller than the readout loss rate κ_{RO}. This was demonstrated in an experiment by Leghtas et al. (2015) as part of the autonomous cat code stabilization protocol, discussed previously in section 3.6.2. Multiple multi-photon dissipation processes can be concatenated together to create even higher-order effective processes, as shown in the 4-photon dissipation experiment by Mundhada et al. (2018).

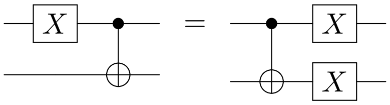
In addition to multi-photon processes, it is possible to create engineered transmon dissipation processes. For instance, one can drive a process at $\omega_{ef} + \Delta$ and $\omega_{\mathsf{RO}} - \Delta$[5] will enable a process $|f\rangle\langle e| \boldsymbol{r}^\dagger +$ h.c.. This effectively creates a dissipation *from* $|e\rangle$ *to* $|f\rangle$. This can create a type of single error channel qubit in the $\{|g\rangle, |f\rangle\}$ subspace, as to first order in the ratio of the pumping rate to the transmon lifetime, decay is converted to dephasing.

[5] note the change in sign, which changes it from conversion to two-mode squeezing

Chapter 8

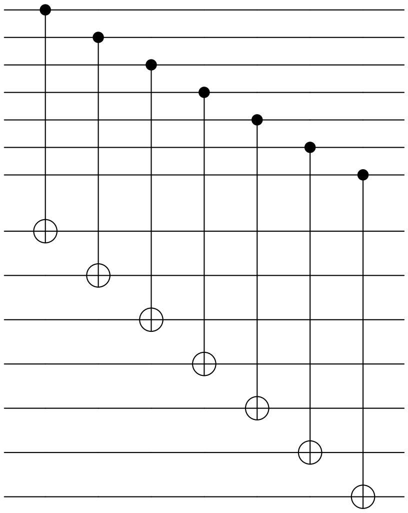
A Fault-Tolerant Parity Measurement

One of the crowning achievements of quantum information theory is the *theory of fault-tolerance* (DiVincenzo and Shor, 1996; Preskill, 1997). While the error correcting codes put forward by Shor (1995) and others showed the feasibility of long-term *storage* of quantum information, it was the theory of fault-tolerance which made possible the prospect of actually doing *computation*. In going from storage to computation one must begin to think of how errors propagate through the computing circuits. While an error correcting code might be designed to deal with a certain class of errors (e.g. single qubit errors), the action of doing computation can transform the errors to a new, larger, potentially uncorrectable set. The prototypical example of this is the transformation of one bit flip into two by the action of a CNOT gate:



(8.1)

If we are dealing with a code which corrects only single qubit errors, then the sudden appearance of two errors will be uncorrectable, and this problem generalizes to higher-order codes. The key to avoiding this problem is to design gates which control the spread of errors so that errors remain correctable. One way of achieving this task is to perform gates "transversally" (Bacon, 2006). This means that you do the gate without performing entangling gates within encoded qubits. For instance, in the Steane code, one can perform a logical CNOT on a pair of encoded qubits as follows



In this scheme, one bit flip error in the target logical does become two errors, but each of these resides in different logical qubits. It can thus be corrected by applying the error correction procedure to both logical qubits. A fully fault-tolerant system requires all of the components responsible for operating an error corrected quantum computer to be made fault tolerant (FT). This includes a universal logical gate set, as well as state preparations, logical measurements, error syndrome measurements, and error correction. It was shown by Aharonov and Ben-Or (1997) that if the error rate per component at the physical level is lower than a threshold value, then the effective error at the logical level is lower, and thus code concatenation provides a path toward qubits with arbitrarily low error rates. While in the first protocols considered this threshold was as low as 10^{-6}, theoretical advances have produced codes with thresholds as high as a few percent (Fowler et al., 2012; Campbell et al., 2017).

In this chapter we will discuss a novel approach to developing fault tolerance which is realizable with orders of magnitude fewer components, building up to the results shown in Rosenblum et al. (2018b). We will start in section 8.1 by discussing an approach to creating fault-tolerance at the Hamiltonian level, by engineering symmetries into our interaction with knowledge of the structure of the dominant error mechanisms. Next, in section 8.2 we will discuss a particular mechanism for engineering interactions, using sideband drives which were discussed in chapter 7. This mechanism will allow us fine grained control of the dispersive interaction Hamiltonian. Next, in section 8.4 we can show how this technique can reduce the propagation of errors in an idling system by temporarily decoupling the transmon and cavity, and showing that the driven sideband method is compatible with high-coherence operation. We will then go beyond idling, and construct an actually useful operation, the parity measurement, in a protected, fault-tolerant way. In section 8.5 we perform experiments which analyze the cavity behavior as a function of transmon state, which allows us to separate out the different error mechanisms and their respective effects on the cavity. In 8.6 we summarize these results to calculate the net performance enhancement derived from the fault-tolerance modifications, and conclude by analyzing the dominant residual sources of error.

8.1 Error Transparency: A paradigm for hardware-efficient fault-tolerance

FT protocols can have an enormous overhead to implement, and for this reason, we should seek out shortcuts wherever we can. Much like the cat code, which took advantage of the *structure* of the errors to produce a parsimonious encoding, one can look for approaches which reach the goal of preventing the propagation of errors and the development of uncorrected errors using the fewest additional complications.

For this purpose, we turn to the concept of *error transparency* of Kapit (2017). In this approach, the physical Hamiltonian implementing a gate is carefully tailored such that it commutes with the correctable errors when acting on the logical state manifold, at all times during the gate operation[1]. In a mathematical language, if we can have some errors $\{\boldsymbol{A}_k\}$ that we anticipate occurring and can correct for, we seek to find an interaction Hamiltonian $\boldsymbol{H}_{\text{int}}$ which implements the gate such that

$$\forall k : [\boldsymbol{H}_{\text{int}}, \boldsymbol{A}_k] = 0 \tag{8.2}$$

This condition implies that we avoid the type of error propagation exhibited in equation 8. It in essence enforces a *symmetry* property of the interaction, which renders it invariant under the action of the errors. Reaching this type of condition, however, requires that the systems

[1] Error transparency (ET) can be thought of as the "computing" equivalent of the concept of decoherence free subspaces (DFS) (Lidar et al., 1998). In some sense the analogy is (FT:QEC)::(ET:DFS)

have many internal degrees of freedom, and that the errors have structure, so that not all transitions are allowed. We need this complexity in order to maintain the properties of the interaction which give us the desired operation, while simultaneously zeroing the components of the interaction which would not commute with the errors. This type of interaction is unlikely to present itself as naturally occurring, and may require us to perform some type of interaction engineering.

How does this apply to our chosen system of cat-encoded cavity qubits coupled to transmon ancillae? As we have developed in section 2.3, and seen applied in chapter 6, our workhorse interaction is the dispersive interaction, $\boldsymbol{H}_{\text{int}} = \chi \boldsymbol{a}^\dagger \boldsymbol{a} \boldsymbol{b}^\dagger \boldsymbol{b}$. There are two primary types of errors that we need consider when analyzing error propagation through this interaction: energy decay ($\boldsymbol{a}$ or $\boldsymbol{b}$) and pure dephasing ($\boldsymbol{a}^\dagger \boldsymbol{a}$ or $\boldsymbol{b}^\dagger \boldsymbol{b}$). The latter errors are diagonal, and therefore commute with the interaction. We should be careful to say now that this does not in itself mean that dephasing errors are harmless, but rather that the application of this interaction Hamiltonian does not exacerbate the harm. Decay events are another story.

$$\left[\boldsymbol{a}^\dagger \boldsymbol{a} \boldsymbol{b}^\dagger \boldsymbol{b}, \boldsymbol{a}\right] = \boldsymbol{a}(\boldsymbol{b}^\dagger \boldsymbol{b}) \tag{8.3}$$

$$\left[\boldsymbol{a}^\dagger \boldsymbol{a} \boldsymbol{b}^\dagger \boldsymbol{b}, \boldsymbol{b}\right] = \boldsymbol{b}(\boldsymbol{a}^\dagger \boldsymbol{a}) \tag{8.4}$$

We can interpret these equations as telling us that the action of the interaction Hamiltonian is to transform decay of one mode to decay of that mode *plus* dephasing of the other mode. This can be seen in a more intuitive way as well. Consider a transmon in the excited state, interacting with a cavity via this interaction for a time Δt which is long compared with $1/\chi$. The cavity rotates at rate χ (in the interaction picture) so long as the transmon is in the excited state. If at some point, say time t_{decay}, during this interaction, the transmon decays to the ground state, the cavity evolution will stop. The cavity will therefore acquire a phase χt_{decay}. The uncertainty we have in the jump time is proportional to Δt. Therefore the uncertainty in the acquired phase is $\chi \Delta t \gg 1$. A large uncertain acquired phase is exactly a pure dephasing event.

From this analysis we see that the dispersive interaction is transparent with respect to pure dephasing, but not with respect to energy decay of the ancillary transmon. To reach full (or at least fully first order) error transparency, we will have to engineer our interaction. The next two sections will develop and demonstrate our preferred method of engineering the dispersive shift. We will return to error transparency in section 8.4.

8.2 Cancelling χ: The simplest useful symmetry

Our tools for modifying our χ interaction spring from the sideband drives which were discussed in chapter 7. These drives allow us to implement terms outside of the native dispersive shift. However, its not immediately clear how we can realize a *driven* dispersive shift with these tools. We can find a clue by considering the origin of the native dispersive shift in the Jaynes-Cummings model (equation 2.11). In this model, the combination of an excitation exchange term $g\left(\boldsymbol{a}^\dagger\boldsymbol{\sigma}_- + \text{h.c.}\right)$ and a detuning term $\Delta\boldsymbol{a}^\dagger\boldsymbol{a}$, in the regime of $\Delta \gg g$, results in a dispersive shift.

We can create a similar effect using sideband drives. First, note that we can create exactly the exchange term, $g\left(\boldsymbol{a}^\dagger\boldsymbol{\sigma}_- + \text{h.c.}\right)$, this time in the rotating frame, using a pair of drives each equally detuned from either the cavity and transmon resonance frequency, say at $\omega_{ge} - \Delta$ and $\omega_c - \Delta$. Such a pair of drives enables the four-wave mixing process which converts photons of one frequency to photons of another frequency. Because this process occurs in the rotating frame, a photon is actually exchanged from one mode to the other as a function of the time the exchange term is applied. However by applying a detuning to one of the drives, we end up with exactly the detuned Jaynes-Cummings Hamiltonian which gave the dispersive interaction in the first place. We can look at this process slightly more rigorously as follows: In the rotating frame, after discarding quickly oscillating terms, the Hamiltonian can be written as

$$\boldsymbol{H} = \boldsymbol{a}^\dagger\boldsymbol{a}\chi_e^{(0)}\left|e\middle\rangle\middle\langle e\right| + g\left(\boldsymbol{a}\boldsymbol{\sigma}_+ e^{i\Delta t} + \text{h.c.}\right) \tag{8.5}$$

$$\equiv \boldsymbol{H}_0 + \boldsymbol{H}_1 e^{i\Delta t} + \boldsymbol{H}_{-1} e^{-i\Delta t} \tag{8.6}$$

We can use a Floquet theory analysis (see appendix A.8) which treats general periodic Hamiltonians of form 8.6 and produces effective time-independent Hamiltonians, under the assumption

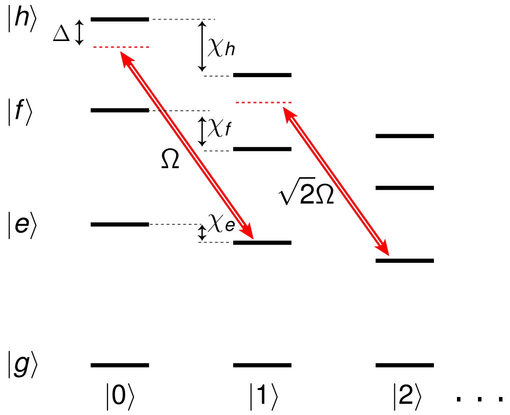


Figure 8.1: **Sideband drive indicated on cavity-ancilla level diagram.** An applied microwave tone (double red arrows) drives the $|e, n\rangle \leftrightarrow |h, n-1\rangle$ transition frequency with Rabi rate $\sqrt{n}\Omega$ and detuning Δ. The resulting Stark shift changes the effective χ_e by an amount $\Omega^2/4\Delta$.

that $|\boldsymbol{H}_0|, |\boldsymbol{H}_{\pm 1}| \ll \Delta$. The generic form is then, to first order in $\frac{g}{\Delta}$[2],

$$\begin{aligned} \boldsymbol{H}_{\text{eff}} &= \boldsymbol{H}_0 + \frac{1}{2\Delta}[\boldsymbol{H}_1, \boldsymbol{H}_{-1}] && (8.7) \\ &= \boldsymbol{H}_0 + \frac{g^2}{\Delta}\left[\boldsymbol{a}^\dagger \boldsymbol{\sigma}_-, \boldsymbol{a}\boldsymbol{\sigma}_+\right] && (8.8) \\ &= \boldsymbol{H}_0 + \frac{g^2}{\Delta}\boldsymbol{a}^\dagger \boldsymbol{a}\boldsymbol{\sigma}_z && (8.9) \end{aligned}$$

This justifies our intuition from before. We see that, (at least in the case that Δ is much larger than χ), we can simply add the driven dispersive interaction to the native one to obtain a net interaction.

We can simplify and strengthen this approach by using a different, but related, sideband process. Instead of converting one cavity photon into one transmon photon using two pump photons, we can convert one cavity photon into two transmon photons using a single pump photon (figure 8.1). For reasons that will become apparent later, we choose to address $|e\rangle \leftrightarrow |h\rangle$ instead of $|g\rangle \leftrightarrow |f\rangle$ as our two-excitation transition. This level of transition selectivity is possible and necessary because the anharmonicity α which separates the sideband transitions is much larger than the sideband exchange rate g. The derived exchange term from this new sideband drive replaces $\boldsymbol{\sigma}_- = |g\rangle\langle e|$ with $|e\rangle\langle h|$. Now if we apply a detuning to this new drive,

[2] Since it is not important we have omitted a Stark shift term which also falls out of the commutator.

the derived interaction replaces $\boldsymbol{\sigma}_z = |e\rangle\langle e| - |g\rangle\langle g|$ with $|h\rangle\langle h| - |e\rangle\langle e|$. This results in a new driven interaction Hamiltonian

$$\boldsymbol{H}_{\mathsf{driven}} = \frac{g^2}{\Delta}\left(|h\rangle\langle h| - |e\rangle\langle e|\right) \tag{8.10}$$

which can be added to the native interaction to form the total,

$$\begin{aligned}\boldsymbol{H}_{\mathsf{int}} &= \boldsymbol{H}_{\mathsf{int}}^{(0)} + \boldsymbol{H}_{\mathsf{driven}} &(8.11)\\ &= \boldsymbol{a}^\dagger\boldsymbol{a}\left(\left(\chi_e^{(0)} - \frac{g^2}{\Delta}\right)|e\rangle\langle e| + \chi_f^{(0)}\,|f\rangle\langle f| + \left(\chi_h^{(0)} + \frac{g^2}{\Delta}\right)|h\rangle\langle h|\right). &(8.12)\\ &\equiv \boldsymbol{a}^\dagger\boldsymbol{a}\left(\chi_e\,|e\rangle\langle e| + \chi_f\,|f\rangle\langle f| + \chi_h\,|h\rangle\langle h|\right) &(8.13)\end{aligned}$$

From this formula for the interaction, there are two interesting points we will explore. The first is the case of

$$\chi_e^{(0)} + \frac{g^2}{\Delta} = 0, \tag{8.14}$$

where the cavity does not distinguish between $|g\rangle$ and $|e\rangle$. With this tool we will be able to decouple the cavity from transmon thermal population, and improve its idling coherence time (section 8.3). In the next case, we can choose

$$\chi_e^{(0)} + \frac{g^2}{\Delta} = \chi_f^{(0)}. \tag{8.15}$$

Here the cavity does not distinguish between $|e\rangle$ and $|f\rangle$, and will be protected against decay from $|f\rangle$. We call these cases χ_{eg}-cancelling and χ_{fe}-cancelling, respectively, where we introduce the notation $\chi_{ab} \equiv \chi_a - \chi_b$. Note that both χ-cancelling points are possible (albeit with different drives) because the sign of the induced dispersive shift can change with the sign of the detuning from sideband resonance. We can see an implementation of both of these cases in figure 8.2. We can measure this change in χ by applying the drive for varying sideband detunings, varying the number of photons in the cavity, and characterizing the transmon frequency, via standard transmon spectroscopy methods. By carefully inspecting the photon number dependence of the transmon frequency, we find we can completely eliminate the linear

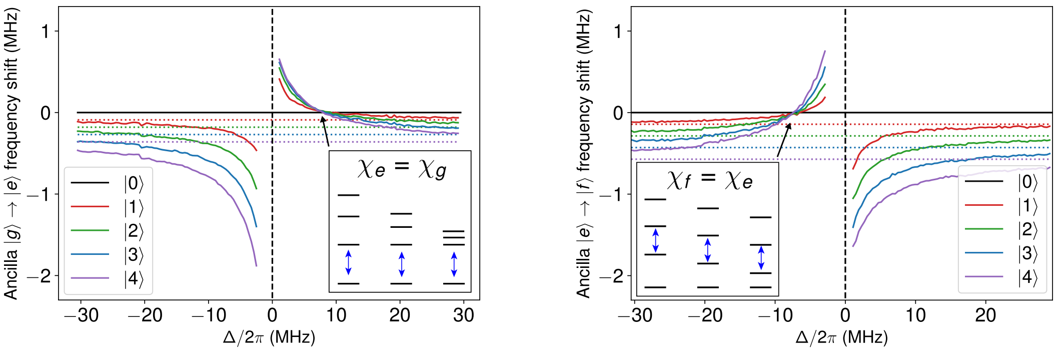


Figure 8.2: **Cancelling the dispersive interaction with a sideband drive** Spectroscopy of the $|g\rangle$ to $|e\rangle$ (left) and $|e\rangle$ to $|f\rangle$ (right) transitions performed with a varying number of photons in the cavity. χ_{eg} (χ_{fe}), as well as higher order nonlinear dispersive shifts can be extracted from the spread in transition frequencies with respect to photon number. The indicated crossing points show where χ_{eg} (χ_{fe}) is approximately zero, as emphasized by the blue arrows in the insets depicting the effective driven level diagram. The dotted lines refer to the transition frequencies when no sideband drive is applied.

dispersive shift, while leaving a residual $5\ \mathrm{kHz}$ nonlinear dispersive shift (figure 8.3).

8.2.1 Choosing drive parameters

We note that there are two parameters available to us in order to satisfy either equation 8.14 or 8.15. Assuming the validity of the model, for any value of pump strength (g) we can find a detuning (Δ) which yields the appropriate χ. How can we break this degeneracy in practice? The first obvious concern is the validity of the dispersive approximation, which relies on $\frac{g}{\Delta}$ being small. The next lowest order term is a driven χ' of order $\frac{g^4}{\Delta^3}$, smaller by a factor of $\frac{g^2}{\Delta^2} = \frac{\chi_e^{(0)}}{\Delta}$. Therefore, to minimize this concern, we choose Δ large, and therefore large g. There are additional reasons to prefer this direction. Firstly, we see decreased hybridization (which is again of order $\chi_e^{(0)}/\Delta$) between the $|e\rangle$ and $|h\rangle$ states. This hybridization allows for the possibility of undesired transitions, say from $|f\rangle$ to $|h\rangle$ via T_1^{fe} processes. Additionally, in order to avoid spurious transitions from $|e\rangle$ to $|h\rangle$ or vice versa, the pump must be turned on adiabatically, a process that becomes slower as the detuning is reduced.

A different set of considerations keeps us from putting the drive power (and therefore detuning) at whatever maximum is supported by the control electronics. Firstly, we have to balance the gain obtained from the χ cancellation process with the cost of adding a heat load

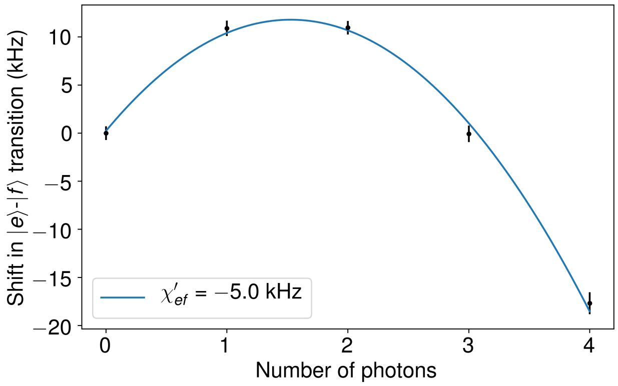


Figure 8.3: **Measuring the nonlinear component of the driven dispersive shift.** We prepare photon number states $|0\rangle$ through $|4\rangle$, turn on the pump to the detuning used for χ_{ef} cancellation, and perform transmon ef spectroscopy as in figure 8.2. From each trace we extract the detuning from bare resonance. We fit these frequencies to a quadratic model, from which we can infer the non-linear frequency shift, $\chi'_{ef} \approx 5\,\mathrm{kHz}$. This is largely a driven effect, resulting from the higher order terms in the perturbative Floquet analysis derived in appendix A.8.

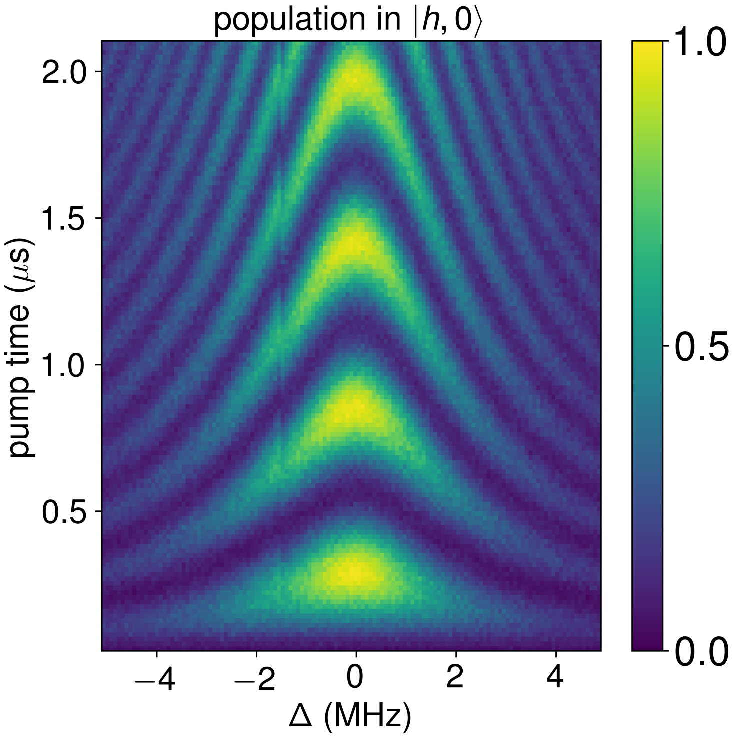


Figure 8.4: **Characterizing χ-cancelling pump.** Chevron pattern observed in the population of $|h, 0\rangle$ when preparing $|e, 1\rangle$ and switching on the sideband drive at a detuning Δ from resonance for a varying amount of time.

to system in the form of a strong drive. If we heat up components in the transmission lines, or even the base temperature of the fridge, we will adversely affect the performance of the system. Even with no added line noise or heat load, however, there may still be issues with high drive levels. For instance, we can see how a large displacement can combine with native transmon dephasing to produce heating:

$$\begin{aligned} \gamma_\phi D[\boldsymbol{b}^\dagger \boldsymbol{b}] &\to \gamma_\phi D[(\boldsymbol{b}^\dagger + \xi)(\boldsymbol{b} + \xi^*)] && (8.16) \\ &\approx \xi \gamma_\phi D[\boldsymbol{b}] + \xi \gamma_\phi D[\boldsymbol{b}^\dagger] + D[\boldsymbol{b}^\dagger \boldsymbol{b}] && (8.17) \end{aligned}$$

In addition, higher drive powers enables higher order transitions between levels. The number of accessible transitions becomes denser and denser in frequency space as drive power increases, until it is no longer an unlikely coincidence to drive spurious transitions, but rather an inevitability. For a more rigorous exploration of this and other issues, see Sank et al. (2016); Zhang et al. (2019).

8.3 Extending the cavity lifetime by protecting it from the transmon

Dispersive shifts, when combined with thermal population leads to dephasing. The typical setting for this statement is that of a two level qubit and a dispersively coupled readout cavity (Gambetta et al., 2006). The exact formula for how much dephasing to expect from a coupled mode, with dispersive shift χ, decay rate Γ, and equilibrium population $\bar{n}_{\text{th}}$, was calculated in Rigetti et al. (2012).

$$\begin{aligned} \frac{1}{T_\phi} &= \frac{\Gamma}{2} \text{Re} \left(\sqrt{\left(\frac{1 + i\chi}{\Gamma} \right)^2 + \frac{4i\chi\bar{n}_{\text{th}}}{\Gamma}} \right) && (8.18) \\ &\approx \frac{\bar{n}_{\text{th}} \Gamma \chi^2}{\Gamma^2 + \chi^2} && (8.19) \\ &\approx \begin{cases} \bar{n}_{\text{th}} \Gamma & \chi \gg \Gamma \\ \frac{\bar{n}_{\text{th}} \chi^2}{\Gamma} & \chi \ll \Gamma \end{cases} && (8.20) \end{aligned}$$

These formulae apply equally to the case of a readout mode inducing dephasing in a transmon as to a transmon inducing dephasing in a storage cavity. In this latter case, we are almost always operating in the photon number resolved regime, allowing us to take the $\chi \gg \Gamma$ limit of equation 8.20. This quantity $\bar{n}_{\mathsf{th}}\Gamma$ is (for $\bar{n}_{\mathsf{th}} \ll 1$) precisely the rate of jumps from the ground state to the excited state. This has a simple interpretation: In the photon number resolved limit, any jump from the ground state to the excited state results in dephasing of the coupled cavity. Removing the transmon from our picture, and viewing only the cavity, it appears that we have a new error channel $\bar{n}_{\mathsf{th}}\Gamma D[\boldsymbol{a}^\dagger \boldsymbol{a}]$. This is an error which is not corrected by the cat code, and therefore can be a limiting factor if the rate is at all comparable to the rate of cavity single-photon loss. Indeed, this was one of the issues which limited performance in the cat code demonstration by Ofek et al. (2016).[3]

We could avoid this problem if we could decouple the cavity from the transmon, for instance by setting $\chi = 0$. We need non-zero dispersive shift at certain times, when we are deliberately interacting the two systems to achieve a goal such as performing a parity measurement, but during other times, say idling between parity measurements, decoupling would be advantageous. We can apply the result shown in figure 8.2 which allows us to do just that. What remains to show is that this *actually helps*. In order to do this, we zoom in on the region of frequency space around the χ_{eg}-cancelling point. At each point here, in addition to measuring χ_{eg}, we measure the cavity coherence time T_2^c, i.e. the characteristic time over which a superposition of $|0\rangle$ and $|1\rangle$ Fock state loses coherence. In figure 8.5 we see the improvement in the cavity coherence time in the region where $|\chi_{eg}| < \Gamma = 1/T_1^{ef}$. We note that the coherence time breaks down into two contributions, energy decay and pure dephasing as $1/T_2 = 1/(2T_1^c) + 1/T_\phi^c$. Removing the dephasing component via χ_{eg}-cancelling brings us close to the decay imposed limit of $2T_1^c$. In fact at the optimal point, the pure dephasing time was increased from $T_1^{eg}/\bar{n}_{\mathsf{th}} \approx 1.1\ \mathrm{ms}$ to $14\ \mathrm{ms}$. This demonstration proves two things. Firstly the thermal occupation of the transmon was indeed was limiting the cavity coherence and this limit can be lifted by turning off the

[3] One prospective method for addressing the problem of dephasing in the cat code is to employ the driven-dissipative autonomous stabilization protocols discussed in section 3.6.2. This method protects against pure, white noise dephasing quite well, but against thermal excitation induced dephasing, we require that the stabilization rate Γ_{2ph} is large not only compared with $1/T_\phi$, but with χ as well, which can be much larger. This effect limited the effectiveness of the coherent state stabilization by Leghtas et al. (2015).

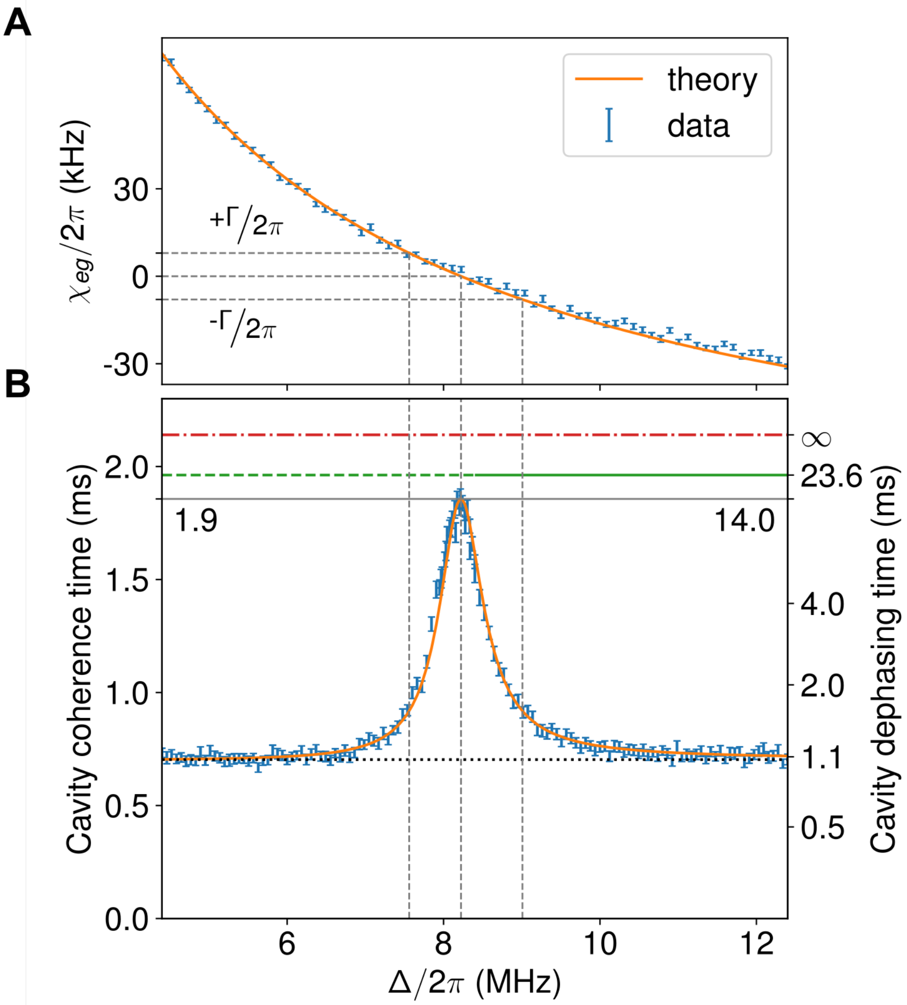


Figure 8.5: **Improving the cavity coherence time by decoupling the cavity from thermal ancilla excitations.** While a bare cavity is nearly completely limited by single photon loss, a cavity dispersively coupled to an ancilla experiences dephasing because of spontaneous ancilla excitations. **(A)** The measured dispersive interaction (blue markers) varies as a function of sideband drive detuning from resonance Δ as $\chi_{eg} = \chi_{eg}^0 + \Omega^2/4\Delta$ (solid orange line). **(B)** Cavity coherence times as a function of the sideband drive frequency obtained from cavity Ramsey experiments. In the absence of quantum error correction, the cavity coherence time is limited to $2T_1^c \approx 2.2$ ms (red dot-dashed line). Without sideband drive, thermal ancilla excitations limit the cavity coherence to about 700 μs (dotted black line). Protection against these excitations starts occurring when $|\chi_{eg}| < \Gamma/2\pi$ (dashed grey lines), where $\Gamma = 1/T_1^{eg}$ is the ancilla $|e\rangle$ to $|g\rangle$ decay rate. This dephasing source is almost entirely removed for $\chi_{eg} = 0$, resulting in a coherence time $T_2^c(\chi_{eg} = 0) = 1.9$ ms (solid grey line), close to the 2 ms limit set by second-order thermal excitation from $|e\rangle$ to $|f\rangle$ (dashed green line). The analytical behavior of the cavity coherence (orange line, equation 8.18) closely matches the observed values.

dispersive interaction. Second, and more critical, is that the negative effects of applying a strong driving tone continuously do not overwhelm the benefits. Other coherence properties of the system seem to be largely preserved in the presence of the χ-cancelling drive. One major reason for this is the fact that the drive is only "active" (in the sense of being close to resonance with any accessible transition) when the transmon is in the excited state $|e\rangle$. Since this is only the case $\bar{n}_{\text{th}} \approx 3\%$ of the time, some of the ill effects of the drive, (e.g. hybridization with $|h\rangle$) are "second order" errors, in the sense of occurring with a rate which is the product of two small numbers.

8.4 Parity measurement using $|f\rangle$

We now return to the task of implementing fault-tolerance via error transparency. We note that, when dealing with the states in the transmon, we have a separation of scales between the rates associated with various decoherence processes, with decay ($|g\rangle\langle e|$, $|e\rangle\langle f|$, etc.[4]) and dephasing ($|g\rangle\langle g|$, $|e\rangle\langle e|$, etc.) occurring at a rate of tens to hundreds of microseconds, and excitation rates ($|e\rangle\langle g|$, $|f\rangle\langle e|$, etc) being orders of magnitude slower. Other transition rates, say those involving a direct two excitation loss such as $|g\rangle\langle f|$ appear to be zero within the error of measurement, a fact which can be justified by symmetry arguments leading to selection rules. This does not mean that two-excitation transitions do not happen, but rather that such transitions are mediated by the single-excitation processes, e.g. $|e\rangle\langle f|$ followed by $|g\rangle\langle e|$, and are thus suppressed over timescales short compared to the single-excitation transition rate.

With these facts in mind, we can consider, over short time periods, what are the dominant transitions which occur starting from the within the subspace spanned by $|g\rangle$ and $|f\rangle$. These are decay $|e\rangle\langle f|$ and dephasing $|f\rangle\langle f|$.[5] If these are our relevant errors, how can we satisfy the error transparency condition (equation 8.2)? Using the form of the interaction given by

[4] Here we associate processes with the operator found in the corresponding Lindblad dissipator

[5] Note either a $|g\rangle\langle g|$ or a $|f\rangle\langle f|$ dissipator produce equivalent results when constrained to the two dimensional subspace.

equation 8.13:

$$[\boldsymbol{H}_{\mathsf{int}}, |f\rangle\langle f|] = 0 \tag{8.21}$$

$$[\boldsymbol{H}_{\mathsf{int}}, |e\rangle\langle f|] = (\chi_e - \chi_f)\, \boldsymbol{a}^\dagger \boldsymbol{a}\, |e\rangle\langle f| \tag{8.22}$$

We have seen in figure 8.2 that we can indeed make $\chi_e = \chi_f$ under the appropriate drive frequency (a detuning with the opposite sign of that used in section 8.3). Moreover, we can do this without giving up our productive resource for entangling with the cavity, which within the $\{|g\rangle, |f\rangle\}$ subspace is simply χ_f, the difference between the cavity frequency in the $|g\rangle$ and $|f\rangle$ states. This is in a sense a fault tolerant dispersive interaction.

We can employ this dispersive interaction towards nearly all of the uses of this interaction explored in chapter 2. Specifically, here, we will address the creation of a fault-tolerant version of the parity measurement discussed in section 2.3.1. The required protocol is a straightforward modification of the traditional parity measurement. Instead of simply preparing a superposition $|g\rangle + |e\rangle$, we additionally perform a π pulse on the $|e\rangle \leftrightarrow |f\rangle$ transition, producing the superposition $|g\rangle + |f\rangle$. We allow this state to evolve for a time $t = \pi/\chi_f$ such that the parity is mapped to the sign of the superposition. The preparation sequence is then reversed, which maps even parity to $|g\rangle$ and odd parity to the $|e\rangle$.

In order to make this fault tolerant, in the sense of preventing cavity dephasing arising from decay from $|f\rangle$ to $|e\rangle$, we apply a drive at the χ_{fe}-cancelling point during this wait time. In figure 8.6 we see a diagram indicating how one can visualize the fact that the interaction commutes with the error, and therefore produces no backaction on the cavity.

8.5 Postselecting on errors: Identifying transmon-induced cavity dephasing

In this protocol the outcome of the parity measurement depends on what, if any, error event occurs in the transmon. If no errors occur, the transmon will be in $|g\rangle$ for even parity, and $|e\rangle$ for odd parity. If a dephasing event occurs, these outcomes will be reversed, with $|e\rangle$ for

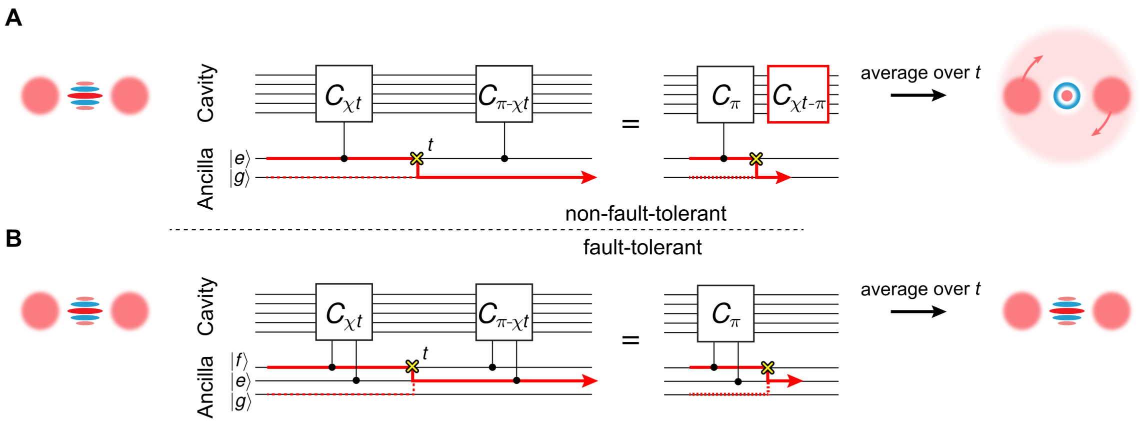


Figure 8.6: **Schematic circuit diagram of a FT parity measurement**. Circuit schematic showing the effect of ancilla energy relaxation on a Schrödinger cat state (depicted by its Wigner tomogram, left) during a parity map in both the traditional (A) and FT (B) schemes. In these circuit diagrams the lines within a bundle represent the individual states of the associated mode. $C_\theta = e^{i\theta \boldsymbol{a}^\dagger \boldsymbol{a}}$ represents a cavity phase shift of angle θ conditional on the state of the ancilla. **(A)** In the non-FT implementation, an error occurring at time $t \in (0, \pi/\chi)$ results in a cavity phase shift of χt. This completely dephases the cavity state when averaged over t. **(B)** In the FT implementation, an error occurring at time t is equivalent to the same error occurring at the end of the parity map, since the error commutes with the interaction.

even parity and $|g\rangle$ for odd parity. If a decay event occurs, the final state will be $|f\rangle$. If we can ensure that the cavity is to a high degree of certainty in an even parity state, then the states $|g\rangle$, $|e\rangle$, $|f\rangle$ cleanly map to the trajectories, of no-error, dephasing error, and decay error, respectively. This is in contrast with the original protocol for the parity measurement where the two outcomes cannot separate the three possible error types. In this case the decay error is folded into both the $|g\rangle$ and $|e\rangle$ outcomes.

We employ three different parity measurement protocols in order to demonstrate the enhancement in cavity coherence obtained from χ_{fe}-cancelling:

- $\boldsymbol{\Pi}_{ge}$: The traditional parity measurement from section 2.3.1
- $\boldsymbol{\Pi}_{gf}$: The parity measurement using the $|g\rangle$ and $|f\rangle$ states
- $\boldsymbol{\Pi}_{\text{FT}}$: The fault-tolerant parity measurement, employing the χ_{fe}-cancelling drive during the wait time.

Circuit schematics corresponding to each of these types of protocols can be found on the left half of figure 8.8. There are two important metrics when it comes to characterizing these

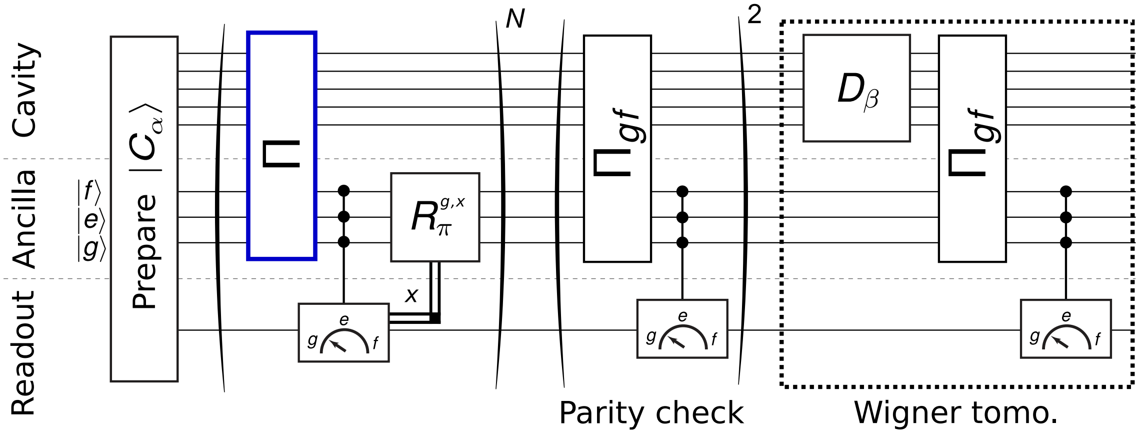


Figure 8.7: **Circuit protocol for characterizing the parity syndrome measurement** We start by initializing the cavity in the cat state $|C_\alpha^+\rangle$. After every parity map Π (indicated in blue), we perform a three-outcome ancilla readout, and reset the ancilla using π-pulses (R_π). We can repeat the parity map and reset procedures N times before moving on to do cavity tomography via Wigner measurement. For the results in figure 8.8, we have $N = 1$, while in figure 8.9, we have N varying along the x axis. We do two additional parity measurements before the final Wigner measurement in order to ensure that the cavity parity is even, using the $\mathbf{\Pi}_{gf}$ protocol.

protocols. First, we are concerned with the *assignment fidelity*, which is the probability of correctly learning the cavity parity. Second, we are concerned with non-fault-tolerance, which here manifests as a *probability of cavity dephasing*. In order to compare these different parity measurement protocols, and extract both the assignment fidelity and the probability of dephasing, we employ the sequence depicted in figure 8.7. We perform Wigner tomography (section 2.3.2, appendix C.1.2) which is postselected on the outcome of the measurement. The results can be seen in figure 8.8. To begin with, we can determine the parity measurement fidelity from the outcome probabilities. In the case of $\mathbf{\Pi}_{ge}$, the determination is quite simply the probability of measuring $|g\rangle$, which is approximately 84%, as this is the correct outcome for the even parity cat state. In the case of three outcomes, however, we must treat the case of measuring $|f\rangle$ not as a complete failure. This outcome is rather a *heralded* failure. Therefore instead of abandoning hope of learning the parity correctly, we can try again. The probability of success in this repeated protocol is then $p_g + p_e p_g + p_e^2 p_g + \cdots = \frac{p_g}{1-p_e}$. For $\mathbf{\Pi}_{gf}$ this is about 89%, and for $\mathbf{\Pi}_{\mathsf{FT}}$ it is about 86%.

While we see a modest degradation of the transmon coherence properties under the influence

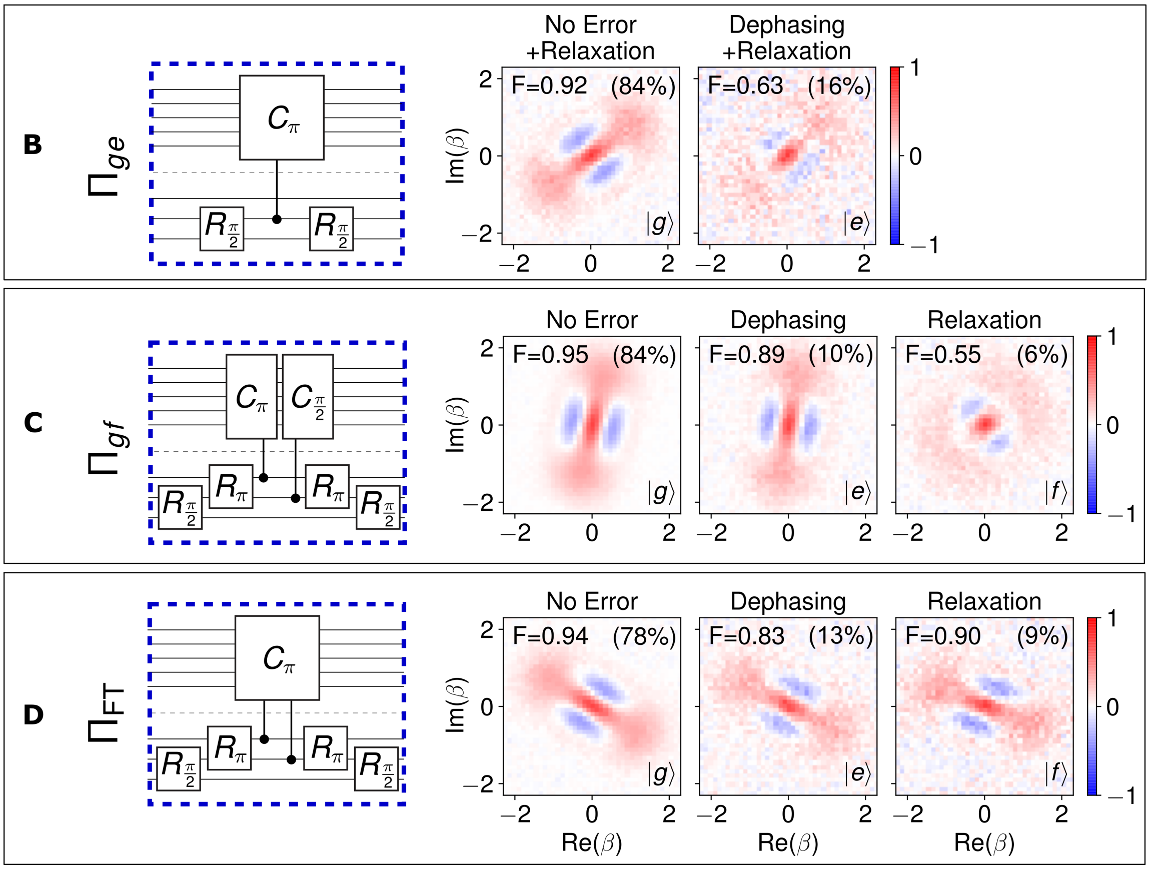


Figure 8.8: **Cavity Wigner functions, postselected on parity measurement outcomes.** The outcome (shown in the bottom right of each Wigner plot) informs us about ancilla behavior during the parity mapping (top). The prevalence of this outcome is indicated in the top right. For each Wigner tomogram, a state fidelity F (shown in the top left) is given, each with statistical error smaller than 0.01. The fidelity of the initial cat state is 0.95 due to imperfections in state preparation and tomography. For $\mathbf{\Pi}_{ge}$ (B) and $\mathbf{\Pi}_{gf}$ (C), ancilla relaxation results in a dephased cavity state, whereas for $\mathbf{\Pi}_{\mathrm{FT}}$ (D) the logical qubit is preserved.

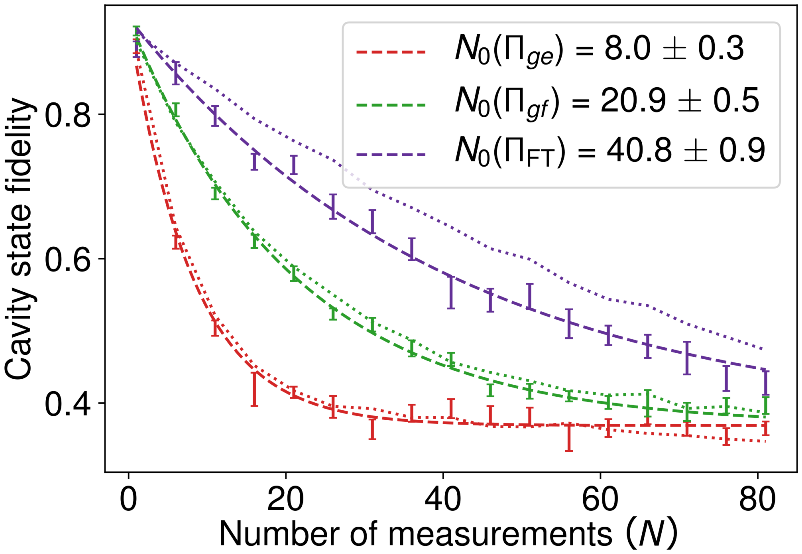


Figure 8.9: **Comparing protocols with repeated parity measurements.** Fidelity vs. number of measurements (N) for the three types of parity measurement. The dotted lines are simulated fidelities extracted from Monte-Carlo trajectories (section8.6), and the dashed lines are exponential fits to the data $F(N) = Ae^{-N/N_0} + c$ with $A \approx 0.56$ and $c \approx 0.37$ for all curves.

of the χ_{fe}-cancelling drive leading to a slight reduction in assignment fidelity, we see a large improvement in terms of the decay-induced cavity dephasing. Looking at the Wigner functions in figure 8.8, we can visually confirm the improvement in cavity coherence. While the component corresponding to either no error or transmon dephasing is coherent regardless (recall that the dephasing event commutes with the interaction), the coherence of the decay component is severely degraded unless the χ_{fe}-cancelling pump is applied.

In order to quantitatively extract the average coherence reduction per parity measurement, we repeatedly measure the parity, and estimate the state fidelity as a function of the number of measurements applied (figure 8.9). The fidelity exponentially approaches the fidelity of a completely dephased cat state[6], with different number constants for each of the protocols which were considered.

With an exponential fit, we can assign a characteristic number of measurements (N_0) in which the cavity fidelity decays. At this point, we can quantify the improvement offered by the FT protocol. We see that $N_0(\mathbf{\Pi}_{gf})/N_0(\mathbf{\Pi}_{ge}) = 2.6 \pm 0.2$, showing that even without sideband drive, the $\mathbf{\Pi}_{gf}$ protocol offers some advantages compared to $\mathbf{\Pi}_{ge}$. The first reason

[6] The fidelity would deviate from this in the long time limit as the average photon number drops to zero.

is that the probability of relaxation is lower for $\mathbf{\Pi}_{gf}$, since the relaxation time of $|f\rangle$ (24 μs) is nearly equal to that of $|e\rangle$ (26 μs), while the parity measurement time of $\mathbf{\Pi}_{gf}$ (as well as $\mathbf{\Pi}_{\mathsf{FT}}$) is less than half that of $\mathbf{\Pi}_{ge}$. The second reason is that the cavity is less dephased given that an ancilla relaxation event occurred, since the cavity angle is distributed between 0 and $\pi\chi^0_{fe}/\chi^0_{fg} = 0.6\pi$ As evident from the residual coherence after a relaxation event in figure 8.8C). The FT implementation improves on $\mathbf{\Pi}_{gf}$ by a factor of 2.0 ± 0.1, resulting in a total fault-tolerance gain of $N_0(\mathbf{\Pi}_{\mathsf{FT}})/N_0(\mathbf{\Pi}_{ge}) = 5.1 \pm 0.3$.

We can compare the observed cavity dephasing rates with predictions for residual uncorrected errors, the largest of which are thermal excitation during the parity map and decay during readout. Monte-Carlo simulations (see following section) of how the cavity phase distribution is affected by these factors produce fidelity decay curves which are in good agreement with the observed results. The agreement is best in the case of the non-FT measurements, where cavity dephasing is dominated by a single well-understood mechanism, namely, ancilla decay during the parity map. The simulation underestimates the decay in the FT case, indicating that there are additional mechanisms for dephasing which are not captured in our model. Some of these mechanisms may be explained by ancilla decoherence induced by the strong sideband drive.

8.6 Performance analysis

How should we interpret these results? To start, we can compare the measured performance to our predictions based on the model provided by the decoherence parameters, and other known sources of uncorrected error. As we can see in table 8.1, there are three main categories of failure: unprotected transitions during the parity map, such as heating or double decay, unprotected transitions during the transmon readout, and readout assignment errors. We also see that, in this parameter regime, at least for $\mathbf{\Pi}_{\mathsf{FT}}$ none of the error sources are truly dominant over the others. Heating from $|f\rangle$ to $|h\rangle$ is the largest predicted issue, but double decay, and decay during the readout, are all equivalently important. However, these issues do add up to a substantial error rate. However, this predicted aggregate dephasing probability (1.36%) is smaller than the corrected component of decay during the parity map (2.84%). If we compare

	Failure mode		**Probability of occurrence**	$\delta\chi$	$[t_0, t_1]$	**Dephasing per occurrence**	**Probability of dephasing**
Parity Map	$\|f\rangle \to \|e\rangle$	$\mathbf{\Pi}_{gf}$	$\frac{t_{\mathsf{map}}}{2T_1^{fe}} = 4.77\%$	χ_{ef}	$[0, t_{\mathsf{map}}]$	62%	2.84%
		$\mathbf{\Pi}_{\mathsf{FT}}$		0		0%	0%
	$\|f\rangle \to \|e\rangle \to \|g\rangle$		$\frac{t_{\mathsf{map}}^2}{4T_1^{fe}T_1^{eg}} = 0.20\%$	χ_{gf}	$[\frac{t_{\mathsf{map}}}{3}, t_{\mathsf{map}}]$	100%	0.19%
	$\|f\rangle \to \|h\rangle$		$\frac{3}{2}\frac{t_{\mathsf{map}}\bar{n}_{\mathsf{th}}^e}{T_1^{eg}} = 0.38\%$	χ_{fh}	$[0, t_{\mathsf{map}}]$	100%	0.38%
	$\|g\rangle \to \|e\rangle$		$\frac{1}{2}\frac{t_{\mathsf{map}}\bar{n}_{\mathsf{th}}^e}{T_1^{eg}} = 0.13\%$	χ_{ge}	$[0, t_{\mathsf{map}}]$	83%	0.11%
Readout	$\|g\rangle \to \|e\rangle$		$\frac{p_g\bar{n}_{\mathsf{th}}^e t_{\mathsf{RO}}}{T_1^{eg}} = 0.12\%$	χ_{ge}	$[0, t_{\mathsf{RO}}]$	42%	0.05%
	$\|e\rangle \to \|g\rangle$		$\frac{p_e t_{\mathsf{RO}}}{T_1^{eg}} = 0.58\%$	χ_{ge}	$[0, t_{\mathsf{RO}}]$	42%	0.25%
	$\|f\rangle \to \|e\rangle$		$\frac{p_f t_{\mathsf{RO}}}{T_1^{fe}} = 0.42\%$	χ_{ge}	$[0, t_{\mathsf{RO}}]$	72%	0.3%
Assignment	Assign $\|g\rangle$ as $\|e\rangle$		0.04%	χ_{ge}	$[t_{\mathsf{RO}}, t_{\mathsf{RO}}]$	100%	0.04%
	Assign $\|e\rangle$ as $\|g\rangle$		0.01%	χ_{ge}	$[t_{\mathsf{RO}}, t_{\mathsf{RO}}]$	100%	0.01%
	Assign $\|e\rangle$ as $\|f\rangle$		0.02%	χ_{ef}	$[t_{\mathsf{RO}}, t_{\mathsf{RO}}]$	100%	0.02%
	Assign $\|f\rangle$ as $\|e\rangle$		0.01%	χ_{ef}	$[t_{\mathsf{RO}}, t_{\mathsf{RO}}]$	100%	0.01%
						Total error probability ($\mathbf{\Pi}_{\mathsf{FT}}$)	1.36%
						Total error probability ($\mathbf{\Pi}_{gf}$)	4.20%

Table 8.1: **Error budget for FT parity measurement.** In these formulae $t_{\mathsf{map}} = \pi/\chi_{fg} = 2.1\,\mu\mathsf{s}$ is the time required to perform the parity mapping, $t_{\mathsf{RO}} = 1.2\,\mu\mathsf{s}$ is the time required to perform the readout of the ancilla. $\{p_g, p_e, p_f\} \approx \{0.8, 0.12, 0.08\}$ are the probabilities of ending the protocol in g, e, f, respectively. The probability of ancilla assignment error is estimated from the overlap of the Gaussian distributions in the histograms of the readout outcomes, as well as the prior probability of measuring a given state. Dephasing per occurrence is calculated from $\tilde{f}(\delta\chi, t_0, t_1)$ as defined in 8.26

the dephasing probabilities to $1/N_0(\mathbf{\Pi}_{gf}) \approx 4.78\%$ and $1/N_0(\mathbf{\Pi}_{\mathsf{FT}}) \approx 2.45\%$, it seems that there is some unaccounted for contribution to the error of the $\mathbf{\Pi}_{\mathsf{FT}}$ protocol.

In order to simulate the fidelity of the cavity state after a sequence of parity measurements more accurately, we can employ a Monte Carlo approach to sample from the ultimate distribution of cavity phases. This produces the dotted lines in figure 8.9. We construct a Monte Carlo model that takes into account the errors listed in table 8.1. Each of the k errors has a probability of occurrence $(p_1, \ldots, p_k)$, a change in cavity frequency $(\delta\chi^{(1)}, \ldots, \delta\chi^{(k)})$, and a range of times for which this shift is active $([t_0^{(1)}, t_1^{(1)}], \ldots, [t_0^{(k)}, t_1^{(k)}])$. We simulate the cavity's trajectory over N parity measurements by sampling the number of each event from the multinomial distribution $n_1, \ldots, n_k \sim \text{Multinomial}(N, p_1, \ldots, p_k)$. Then, for each event, we sample the change in cavity phase from a uniform distribution associated with that event $\theta_{i,j} \sim \text{Unif}(\delta\chi^{(i)}t_0^{(i)}, \delta\chi^{(i)}t_1^{(i)})$. Finally, we sum these phases $\theta = \sum_{i=1}^{k}\sum_{j=1}^{n_i}\theta_{i,j}$, and compute the final fidelity $\left|\left\langle C_\alpha^+ \middle| C_{\alpha e^{i\theta}}^+ \right\rangle\right|^2$. We repeat this procedure 10,000 times and compute an average fidelity as a function of N. While the Monte-Carlo simulation should be the most accurate method of predicting the fidelity decay curve, we would also like to have a mechanism for assessing the relative importance of each error channel in determining the final dephasing rate. To do so, we first calculate an effective "dephasing per occurrence" for each event, which is a number between 0 and 1 indicating the degree of dephasing induced by the error. The dephasing probability is then the product of the probability of occurrence and dephasing per occurrence (see table 8.1). We compute the infidelity f per occurrence as follows:

$$f(\delta\chi, t_0, t_1) = 1 - \frac{1}{t_1 - t_0}\int_{t_0}^{t_1} \mathrm{d}t \left|\left\langle C_\alpha^+ \middle| C_{\alpha e^{i\delta\chi t}}^+ \right\rangle\right|^2 \tag{8.23}$$

$$= 1 - \frac{4}{\mathcal{N}^2(t_1 - t_0)}\int_{t_0}^{t_1} \mathrm{d}t \left|\left\langle \alpha \middle| \alpha e^{i\delta\chi t} \right\rangle + \left\langle \alpha \middle| -\alpha e^{i\delta\chi t} \right\rangle\right|^2 \tag{8.24}$$

$$= 1 - \frac{4}{\mathcal{N}^2(t_1 - t_0)}\int_{t_0}^{t_1} \mathrm{d}t \left| e^{-\alpha^2(1+\exp(-i\delta\chi t))} + e^{-\alpha^2(1-\exp(-i\delta\chi t))} \right|^2 \tag{8.25}$$

Where $\mathcal{N} = 2\left(1 + e^{-2|\alpha|^2}\right)$ is a normalization factor. The effective dephasing per occurrence can be found by comparing the fidelity to the fidelity of the completely dephased state:

$$\tilde{f}(\delta\chi, t_0, t_1) = \text{min}\ \left\{1, \frac{f(\delta\chi, t_0, t_1)}{f(\delta\chi, 0, 2\pi/\delta\chi)}\right\} \tag{8.26}$$

The sampling results produce very good agreement for the $\mathbf{\Pi}_{ge}$ and $\mathbf{\Pi}_{gf}$ cases, while also similarly modestly underestimating the error in the $\mathbf{\Pi}_{\text{FT}}$ case.

In summary, we have demonstrated the fault-tolerance of the parity measurement, in the sense of reducing the impact of transmon energy decay on the cavity phase coherence. We should now return to our original goal of evaluating how this will effect the performance of cat code error correction, in the style of the experiments shown by Ofek et al. (2016). Unfortunately, with the sample used in this demonstration, we do not have the capability of applying the parity measurement to improve the cat code lifetime substantially. This is a result of the very low $\chi \times T_2$ product which sets the assignment fidelity of the parity measurement. In the complete absence of any sort of non-fault-tolerance, even very low assignment fidelities are acceptable, because one can simply repeat the measurement until a sufficient confidence is established. However, in this case, we would require about 5 parity measurements to have 99% assignment certainty, at which point the five-fold reduction in non-fault-tolerance from the improved protocol is cancelled.

In order to successfully apply this technique in the context of error correction we have several improvements to seek out. First we should use a sample with state-of-the art coherence times and thermal populations, as there is nothing intrinsic about our setup which makes this particularly difficult. Second, we should increase the dispersive shift χ so the parity measurement can be performed faster. Implementing this requires being able to *cancel* larger values of χ, which could be aided by ongoing efforts to reduce the negative effects of strong pumps. Finally, we should make the readout faster, and more ideal. The cavity is completely exposed to qubit transitions during the readout, and if we cannot find a way to decouple the storage and qubit during this time, the best we can do is to make the readout as fast as possible. Order of magnitude improvements in this area are possible (Heinsoo et al., 2018) and could significantly reduce the residual non-fault-tolerance of the protocol.

Chapter 9

Fault-tolerant SNAP

In chapter 8, we saw how we could construct an implementation of the photon number parity measurement, an essential component of cavity-encoded error correction, in a manner which was fault-tolerant with respect to transmon decoherence. While this is one essential component, in order to make progress towards a fully fault-tolerant system, we must implement *every* component in a protected way. In this chapter, we will see how our previous approach can be extended to a different task: manipulating the cavity state. While we saw how cavity gates could be implemented in chapter 6, these implementations left a lot to be desired. While relatively high fidelity, these operations were limited by both transmon decay and dephasing processes. Additionally, these operations would not preserve the structure of cavity errors, i.e. cavity photon loss during the pulse would not be correctable via parity measurement, as it would be if such an error had occurred during an idling period.

If we compare the SNAP operation and the parity measurement, there are many similarities. Both rely on the dispersive shift χ acting over a period of time, and both involve direct driving of the transmon around its resonance frequency with a bandwidth set by $\bar{n}\chi$. The major difference is the fact that the transmon drive in the SNAP pulse is more or less continuously engaged. This distinction complicates the simple picture of error transparency (section 8.1) which gave us fault tolerance before. However, a more subtle set of algebraic criteria can show how we can still get fault-tolerance in SNAP (Ma and Jiang, in preparation). In practice we can largely proceed by analogy with the FT parity protocol: we will replace all usages of the excited state $|e\rangle$ with the excited state $|f\rangle$, and during the χ evolution we will drive the system so that $\chi_e = \chi_f$. We will measure the transmon to determine whether any decoherence events occurred, and to determine if the operation succeeded. There are two large questions to be addressed, one theoretical, and one practical. Firstly, we must be much more careful about how transmon dephasing acts on our system, since it will occur during periods where both χ and a transmon drive are present. Secondly, the SNAP protocol relies on a drive which continuously takes $|g\rangle$ to $|e\rangle$; following our prescription, how can we implement a drive which takes $|g\rangle$ directly to $|f\rangle$?

9.1 An interaction picture for SNAP

We begin by recalling the SNAP operation as discussed in section 2.4. This operation consists of several simultaneous drives applied to the transmon at frequencies which are detuned from vacuum resonance by an integer multiple of the dispersive shift χ. Each of these drives has the same amplitude, but differs in the phase. We can write this as follows:

$$\boldsymbol{H} = \frac{\chi}{2}\boldsymbol{a}^\dagger\boldsymbol{a}\boldsymbol{\sigma}_z + \Omega\sum_k e^{i(k\chi t+\theta_k)}\boldsymbol{\sigma}_- + \text{h.c.}, \tag{9.1}$$

Where Ω is the drive rate and θ_k are the phases associated with each drive. We can modify this Hamiltonian, following our prescription, adding awareness of the third transmon level $|f\rangle$, and driving transitions directly between $|g\rangle$ and $|f\rangle$ (discussion of how to do this in section 9.3).

$$\boldsymbol{H}_{\text{int}} = (\chi_e\,|e\rangle\langle e| + \chi_f\,|f\rangle\langle f|)\,\boldsymbol{a}^\dagger\boldsymbol{a} + \Omega\sum_k e^{i(\chi_f kt+\theta_k)}\,|g\rangle\langle f| + \text{h.c.} \tag{9.2}$$

In order to find a time-independent picture for this operation, we can perform the canonical transformation (appendix A.1) using the time dependent unitary

$$\boldsymbol{U} = \exp\Big\{it\,(\chi_e\,|e\rangle\langle e| + \chi_f\,|f\rangle\langle f|)\,\boldsymbol{a}^\dagger\boldsymbol{a}\Big\}. \tag{9.3}$$

Under this transformation, the ladder operators are transformed as follows:

$$\boldsymbol{a} \mapsto e^{i(\chi_e |e\rangle\langle e| + \chi_f |f\rangle\langle f|)} \boldsymbol{a} \tag{9.4}$$

$$|g\rangle\langle f| \mapsto e^{i\chi_f t \boldsymbol{a}^\dagger \boldsymbol{a}} |g\rangle\langle f| \tag{9.5}$$

$$|g\rangle\langle e| \mapsto e^{i\chi_e t \boldsymbol{a}^\dagger \boldsymbol{a}} |g\rangle\langle e| \tag{9.6}$$

$$|e\rangle\langle f| \mapsto e^{i(\chi_f - \chi_e) t \boldsymbol{a}^\dagger \boldsymbol{a}} |e\rangle\langle f| \tag{9.7}$$

It will be important to remember that the *jump operators* are also transformed in precisely this way. The resulting interaction Hamiltonian can be written as follows:

$$\boldsymbol{H}_{\text{int}} = \Omega \sum_k e^{i(\chi_f k t - \chi_f \boldsymbol{a}^\dagger \boldsymbol{a} t + \theta_k)} |f\rangle\langle g| + \text{h.c.} \tag{9.8}$$

$$\approx \Omega \sum_k e^{i\theta_k} |f, k\rangle\langle f, k| + \text{h.c.} \tag{9.9}$$

where the rotating wave approximation that we have made in the second line is valid in the limit where $\chi_f \gg \Omega$. We can simplify this one more layer by recalling the definition of the SNAP operation

$$\boldsymbol{S}(\vec{\theta}) = \sum_k e^{i\theta_k} |k\rangle\langle k| \tag{9.10}$$

We can therefore rewrite 9.9 as:

$$\boldsymbol{H}_{\text{int}} = \Omega \left(\boldsymbol{S}(\vec{\theta}) |f\rangle\langle g| + \boldsymbol{S}(-\vec{\theta}) |g\rangle\langle f| \right) \tag{9.11}$$

The evolution under this Hamiltonian is trivial to solve due to the simple fact that the Hamiltonian, constrained to the $\{|g\rangle, |f\rangle\}$ subspace, is 'Pauli-like' in that it squares to the identity within the subspace:

$$\boldsymbol{H}_{\text{int}}^2 = \Omega^2 (|g\rangle\langle g| + |f\rangle\langle f|), \tag{9.12}$$

and therefore we have

$$e^{i\boldsymbol{H}_{\text{int}} t} = |e\rangle\langle e| + \cos(\Omega t) (|g\rangle\langle g| + |f\rangle\langle f|) + i \sin(\Omega t) \left(\boldsymbol{S}(\vec{\theta}) |f\rangle\langle g| + \boldsymbol{S}(-\vec{\theta}) |g\rangle\langle f| \right). \tag{9.13}$$

In practice, Ω will not be constant, but rather have some Gaussian envelope profile. In this case, we can replace Ωt with the integrated area under the envelope.

9.2 Analyzing fault propagation in SNAP

The simplest way to see the behavior of the SNAP operation under the action of decoherence is to consider what happens if a discrete jump happens at some time t in the middle of the operation lasting time T. Let's start by considering a decay event $|e\rangle\langle f|$. Recalling equation 9.7, in this picture we must write $e^{i(\chi_f-\chi_e)t\boldsymbol{a}^\dagger\boldsymbol{a}}\,|e\rangle\langle f|$. We assume we start with the transmon in the ground state, and some state in the cavity $|\psi_{\mathsf{cav}}\rangle$. We analyze the evolution in three steps, an initial Hamiltonian evolution, the application of a jump operator, and the remaining Hamiltonian evolution

$$|\psi_{\mathsf{final}}\rangle \propto e^{i\boldsymbol{H}_{\mathsf{int}}(T-t)}\left(e^{i(\chi_f-\chi_e)t\boldsymbol{a}^\dagger\boldsymbol{a}}\,|e\rangle\langle f|\right)e^{i\boldsymbol{H}_{\mathsf{int}}t}\left(|\psi_{\mathsf{cav}}\rangle\otimes|g\rangle\right) \tag{9.14}$$

$$\propto e^{i(\chi_f-\chi_e)t\boldsymbol{a}^\dagger\boldsymbol{a}}\left(\boldsymbol{S}(\vec{\theta})\,|\psi_{\mathsf{cav}}\rangle\right)\otimes|e\rangle \tag{9.15}$$

So we see that the operation still effectively takes place![1] The intuition here is that, in order to have a decay event in the first place, we must have made our transit from $|g\rangle$ to $|f\rangle$. The only trouble is the unwanted rotation $e^{i(\chi_f-\chi_e)t\boldsymbol{a}^\dagger\boldsymbol{a}}$, which is not deterministic since it depends on the random jump time t. We will remove this as before, by the second part of our prescription, pumping to ensure that $\chi_e=\chi_f$. Of course the transmon ends up in the incorrect state, but this is no matter if we can measure and reset the transmon as part of our protocol.

What of transmon dephasing? In this case the operator we wish to consider is $|f\rangle\langle f|$. In

[1] This is only the case when we start in the ground state $|g\rangle$. We can implement the SNAP operation starting in $|f\rangle$, with very similar results, except that in the case of decay ($|e\rangle\langle f|$), the effective cavity operation is the identity, as in the case of dephasing.

this case the rotating frame has no effect on the jump operator: [2]

$$
\begin{aligned}
\left|\psi_{\text{final}}\right\rangle &\propto e^{i\boldsymbol{H}_{\text{int}}(T-t)}\left|f\right\rangle\!\left\langle f\right| e^{i\boldsymbol{H}_{\text{int}}t}\left(\left|\psi_{\text{cav}}\right\rangle \otimes \left|g\right\rangle\right) && (9.16)\\
&\propto e^{i\boldsymbol{H}_{\text{int}}(T-t)}\left(\boldsymbol{S}(\vec{\theta})\left|\psi_{\text{cav}}\right\rangle\right)\otimes\left|f\right\rangle && (9.17)\\
&\propto \cos\left((\Omega(T-t)\right)\left(\boldsymbol{S}(\vec{\theta})\left|\psi_{\text{cav}}\right\rangle\right)\otimes\left|f\right\rangle + i\sin\left(\Omega(T-t)\right)\left(\left|\psi_{\text{cav}}\right\rangle\otimes\left|g\right\rangle\right) && (9.18)
\end{aligned}
$$

This appears complicated, but as before, the act of measuring the transmon at the end of the protocol makes everything simple. We either measure $\left|f\right\rangle$ and obtain $\boldsymbol{S}(\vec{\theta})\left|\psi_{\text{cav}}\right\rangle$, our desired final state, or we measure $\left|g\right\rangle$ and obtain $\left|\psi_{\text{cav}}\right\rangle$, i.e. our original state with no operation performed.

The final piece to consider is cavity decay, $\boldsymbol{a}$. Recalling equation 9.4, this term has an $e^{i(\chi_e|e\rangle\langle e|+\chi_f|f\rangle\langle f|)t}$ time dependency. This effectively induces a transmon dephasing event, meaning that as before, we will have some final probability of measuring either $\left|g\right\rangle$ or $\left|f\right\rangle$.

$$
\begin{aligned}
\left|\psi_{\text{final}}\right\rangle \propto\, & e^{i\boldsymbol{H}_{\text{int}}(T-t)}\left(e^{i(\chi_e|e\rangle\langle e|+\chi_f|f\rangle\langle f|)t}\boldsymbol{a}\right)e^{i\boldsymbol{H}_{\text{int}}t}\left(\left|\psi_{\text{cav}}\right\rangle\otimes\left|g\right\rangle\right) && (9.19)\\
\propto\, & c_1\left(\boldsymbol{a}\left|\psi_{\text{cav}}\right\rangle\otimes\left|g\right\rangle\right) + c_2\left(\boldsymbol{a}\boldsymbol{S}(\vec{\theta})\left|\psi_{\text{cav}}\right\rangle\otimes\left|f\right\rangle\right)+\\
& c_3\left(\boldsymbol{S}(\vec{\theta})\boldsymbol{a}\left|\psi_{\text{cav}}\right\rangle\otimes\left|f\right\rangle\right) + c_4\left(\boldsymbol{S}(-\vec{\theta})\boldsymbol{a}\boldsymbol{S}(\vec{\theta})\left|\psi_{\text{cav}}\right\rangle\otimes\left|g\right\rangle\right), && (9.20)
\end{aligned}
$$

where the unimportant prefactors $c_1, \ldots, c_4$ can be computed easily, and will in general depend on the time of the jump t. We see that, in order to be obviously harmless, it would be sufficient to have $\left[\boldsymbol{a}, \boldsymbol{S}(\vec{\theta})\right] = 0$. However, this is impossible so long as $\theta \neq 0$. We can, however, make this work if the less restrictive $\boldsymbol{S}(\vec{\theta})\boldsymbol{a}\left|\psi_{\text{cav}}\right\rangle = \boldsymbol{a}\boldsymbol{S}(\vec{\theta})\left|\psi_{\text{cav}}\right\rangle$ condition is met. For this purpose, we will consider that we are in something like a cat code state, where $\left|\psi_{\text{cav}}\right\rangle$ only occupies even photon number parity. We can write our condition in operator form as

$$\boldsymbol{S}(\vec{\theta})\boldsymbol{a}\boldsymbol{P}_{\text{even}} = \boldsymbol{a}\boldsymbol{S}(\vec{\theta})\boldsymbol{P}_{\text{even}}, \qquad (9.21)$$

[2] A similar analysis can be performed for different models of dephasing, say $|g\rangle\langle g|$ or $|f\rangle\langle f| - |g\rangle\langle g|$, and yield equivalent results

where $\boldsymbol{P}_{\text{even}}$ is the projector on the even parity subspace. We can expand the definitions above:

$$\boldsymbol{S}(\vec{\theta})\boldsymbol{a}\boldsymbol{P}_{\text{even}} = \left(\sum_k e^{i\theta_k} |k\rangle\langle k|\right)\left(\sum_n \sqrt{n+1}\,|n\rangle\langle n+1|\right)\left(\sum_{m\,\text{even}} |m\rangle\langle m|\right) \tag{9.22}$$

$$= \left(\sum_k e^{i\theta_k} |k\rangle\langle k|\right)\left(\sum_{n\,\text{odd}} \sqrt{n+1}\,|n\rangle\langle n+1|\right) \tag{9.23}$$

$$= \sum_{k\,\text{odd}} \sqrt{k+1}e^{i\theta_k} |k\rangle\langle k+1| \tag{9.24}$$

$$\boldsymbol{a}\boldsymbol{S}(\vec{\theta})\boldsymbol{P}_{\text{even}} = \left(\sum_n \sqrt{n+1}\,|n\rangle\langle n+1|\right)\left(\sum_k e^{i\theta_k} |k\rangle\langle k|\right)\left(\sum_{m\,\text{even}} |m\rangle\langle m|\right) \tag{9.25}$$

$$= \left(\sum_n \sqrt{n+1}\,|n\rangle\langle n+1|\right)\left(\sum_{k\,\text{even}} e^{i\theta_k} |k\rangle\langle k|\right) \tag{9.26}$$

$$= \sum_{k\,\text{odd}} \sqrt{k+1}e^{i\theta_{k+1}} |k\rangle\langle k+1| \tag{9.27}$$

Comparing 9.24 with 9.27 we see that this condition can be satisfied by setting $\theta_k = \theta_{k+1}$ for k odd. We can return to equation 9.20 and simplify (using $|\psi_{\text{cav}}\rangle = \boldsymbol{P}_{\text{even}} |\psi_{\text{cav}}\rangle$)

$$|\psi_{\text{final}}\rangle \propto (c_1 + c_4)\left(\boldsymbol{a}\,|\psi_{\text{cav}}\rangle \otimes |g\rangle\right) + (c_2 + c_3)\left(\boldsymbol{a}\boldsymbol{S}(\vec{\theta})\,|\psi_{\text{cav}}\rangle \otimes |f\rangle\right). \tag{9.28}$$

This is quite remarkable: we have a transformation that we can effect on a cavity encoded qubit which is completely compatible with error correction! We simply need to dial the cavity phases such that the phases are equivalent in the even and odd parity subspaces. There is of course some probability of failing to implement the operation (as in the case of transmon dephasing), but this property is determined by a measurement of the transmon. The intuition for this fact is the observation that we can make the transmon trajectory identical in the even and odd subspaces, (assuming we know the starting photon number parity), along with the knowledge that transmon dephasing (an inevitable result of χ combined with cavity decay) is not a catastrophic error.

9.3 Raman SNAP

Let us turn now to the problem of how to implement this drive. The missing piece is the drive component of equation 9.2, $f(t)\,|g\rangle\langle f|+\text{h.c.}$. A single drive, applied at frequency $\omega_{gf}=\omega_{ge}+\omega_{ef}$ cannot implement this as a result of symmetry: The drive only couples states of differing parity. This means that we will need another approach to driving this transition. We turn to the method of *stimulated Raman transitions* (Linskens et al., 1996; Bateman et al., 2010). In this method, we apply drives to both the $|g\rangle \leftrightarrow |e\rangle$ and $|e\rangle \leftrightarrow |f\rangle$ transitions. We detune these drives by an equal and opposite amount: resulting in frequencies $\omega_{ge}-\Delta$ and $\omega_{ef}+\Delta$. If Δ is sufficiently large compared to the drive amplitude, then the effect of this scheme is to drive transitions between $|g\rangle$ and $|f\rangle$ without any intermediate occupation of $|e\rangle$. Given individual drive amplitudes Ω, the effective gf Rabi rate is $\frac{\Omega^2}{\Delta}$.

We wish to combine stimulated Raman driving with the simultaneous number-selective driving of SNAP. In order to do this, we must engineer a situation where, for each n up to the maximum number of addressed photons, there exists a pair of drives with frequencies which satisfy $\omega_1+\omega_2=\omega_{gf}+n\chi_f$. In addition, these frequencies must avoid ω_{ge} and ω_{gf} individually, and we must be careful not to drive spurious transitions by other pairs of drives not considered.

One elegant way of satisfying all of these constraints is to adopt the approach shown in figure 9.1. One strong drive is placed at a given detuning Δ from ω_{ge}. This drive is shared among all of the photon-number selective transitions. The matching pairs for each of these transitions is then given by a weaker tone at $\omega_{ef}+\Delta+n\chi_f$. In comparison with schemes where every Raman transition has a distinct pair of drive tones, this scheme is much simpler, and avoids the problem of accidentally driving unintended transitions with other tone pairings.

9.4 Some assembly required: The FT SNAP protocol

Implementing a fault-tolerant gate requires more than careful pulse shaping. There are several components which must be characterized and assembled. Putting these all together results in the protocol seen in figure 9.2. Here we see the combination transmon drives as well as the χ-cancelling pump and readout pulses.

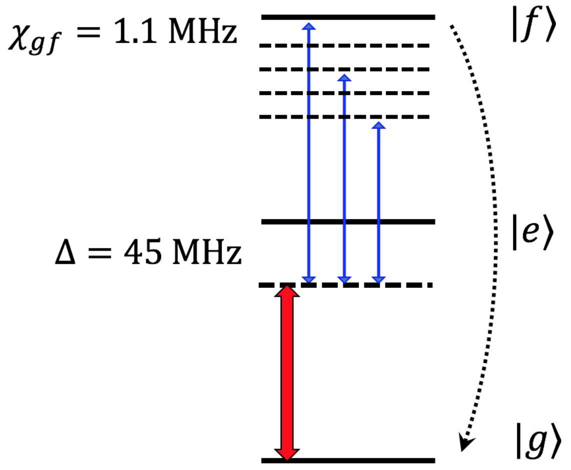


Figure 9.1: **Energy level diagram for Raman SNAP scheme.** The SNAP protocol is implemented with one strong drive, detuned from ω_{ge} by $\Delta = 45$ MHz, and 3 weaker drives, allowing the simultaneous driving of $|n, g\rangle \leftrightarrow |n, f\rangle$ for $n = 0, 2, 4$, as required for the kitten code (equation 3.66). The SNAP evolution is completed by a fast pulse taking $|f\rangle$ to $|g\rangle$ unconditionally.

We anticipate that one of the major sources of failure in this operation is transmon decay during the transmon readout. Since we are unable to effectively cancel χ during the readout operation[3] transitions during the readout will lead to cavity dephasing. In order to minimize this error, we can exploit the fact that the transmon states have different probabilities of transitioning, specifically, $|f\rangle$ is shorter lived than $|e\rangle$, which is shorter lived than $|g\rangle$. We can perform a permutation of the states before measuring (specifically, swapping $|f\rangle$ and $|g\rangle$) so that the most likely state ends up in the state least likely to decay.

The transmon readout is a major source of concern, and requires some optimizing in order to balance the many factors which it influences. There are several desiderata:

- We wish to shorten the length of time, such that the probability of transmon decay is minimized.
- We wish to decrease the number of photons used, so that the readout does not accidentally learn the number of photons in the storage cavity via cross-Kerr interaction ($\chi_{rs}\boldsymbol{a}^\dagger\boldsymbol{a}\boldsymbol{r}^\dagger\boldsymbol{r}$).

[3] The reason for this is that, while there are many photons in the readout cavity, the transmon frequency becomes undefined on the scale of $\bar{n}\chi_{\text{RO}}$ by the shot-noise in the readout population. Since this quantity is comparable to the detuning (Δ, in equation 8.15), the chi-cancelling point is also ill-defined.

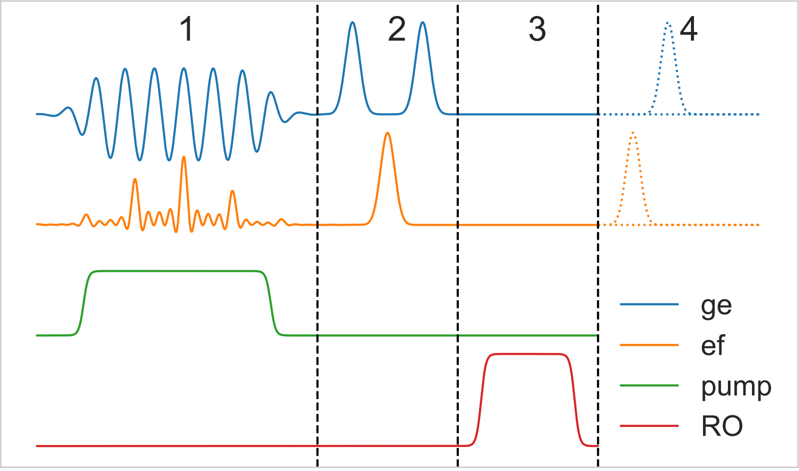


Figure 9.2: **Pulses comprising the FT SNAP protocol.** There are five stages to the protocol (times and amplitudes are not to scale). **(1)** Apply the (half-)SNAP operation with three drives. Two are Raman drives, detuned from the ω_{ge} and ω_{ef} transitions, respectively. The third is the χ_{ef}-cancelling pump. **(2)** Swap the $|g\rangle$ and $|f\rangle$ states, so that no error is mapped to $|g\rangle$, decay is mapped to $|e\rangle$, and dephasing is mapped to $|f\rangle$, in accordance with the outcome probabilities to minimize the probability of decay during the transmon readout. **(3)** Readout the transmon. **(4)** On the basis of the readout result we reset the transmon to the ground state. **(5)** (not shown) In the case of measuring $|f\rangle$, which corresponds primarily to transmon dephasing, the logical operation was the identity, so we try again, restarting the procedure at step 1.

- We wish to readout with sufficient length and photon number to completely discriminate $|g\rangle$, $|e\rangle$, $|f\rangle$, and even potentially $|h\rangle$.

These factors cannot be completely eliminated with any choice of readout parameters, and therefore must be balanced in the final implementation.

Finally we must use the readout result to inform our subsequent actions. There are three main components to how we use the result. Firstly, we must return the transmon to the ground state by performing the appropriate set of π pulses. Secondly, we must compensate for the deterministic rotation of the storage cavity during the readout pulse by an amount $\chi_x t_{\text{RO}}$ when we measure the state $|x\rangle$. We can perform this rotation "in software" by simply rotating all future cavity drives by the appropriate angle. In the characterization experiments we run here, this includes cavity drives which are either part of optimal control cavity manipulations, or displacements which are part of Wigner tomography. The ability to adjust these phases in real time on the basis of recorded measurements is a critical capability for our implementation. Finally, we must determine whether the gate was successful. In our chosen implementation, starting in the ground state, performing the SNAP, and returning quickly to the ground state, the cavity gate is implemented in the case of either 'no error' ($|g\rangle$) or decay ($|e\rangle$, and is not implemented in the case of dephasing $|f\rangle$. Therefore, in order to make our operation deterministic, we must be ready to repeat the entire protocol in the case that we observe $|f\rangle$.

9.5 Tuneup procedure

Given the number of components in the protocol, the procedure for arriving at a fully calibrated operation involves many steps.

1. Perform standard tuneup to identify ω_{ge}, ω_{ef}, as well as the corresponding π pulses. Measure χ_e and χ_f.

2. Tune up a transmon readout protocol that can distinguish $|g\rangle$, $|e\rangle$ and $|f\rangle$ in a single-shot.

3. Tune up a method of preparing photon number states in the cavity, either directly or by postselection.

4. Find the rough $\chi_e = \chi_f$ pump driving point as done in figure 8.2.

5. Tune up a direct $|f\rangle\langle g|$ operation using a Raman pair, with the pump on for the duration. Optimize the detuning parameter Δ. Choose the amplitude so that π pulse time is significantly smaller than χ_f.

6. Replace one of the Raman drives with a SNAP comb with frequency separation χ_f. Adjust the amplitude and frequency of the SNAP components such that an optimal π pulse is performed starting from any relevant photon number.

7. Tune the phase offsets by performing Wigner tomography on $\boldsymbol{S}(\vec{0})$, $\boldsymbol{S}(\vec{0})^2$, $\boldsymbol{S}(\vec{0})^4$, etc.

8. Scan the pump detuning in a small range and measure the cavity coherence after a SNAP, postselecting on decay ($|e\rangle$). This is needed because the χ cancelling point during idle

operation is not necessarily the same as during SNAP, as a result of various Stark shifts. Choose the pump frequency and re-perform steps 6 and 7.

9. Measure the cavity phase adjustments which must be performed following a $|e\rangle$ or $|f\rangle$ result.

There are several "free" parameters which we might vary in an attempt to improve the final operational fidelity of the protocol. Changing each of these parameters can require different re-calibration.

- Changing the χ-cancelling pump amplitude or detuning requires performing the calibration beginning at step 4
- Changing the Raman detuning Δ requires re-performing all of the calibration steps beginning at step 5
- Changing the readout pulse parameters, such as readout pulse length or amplitude requires re-performing steps 7 and 9, since the cavity phases coming out of the readout can be affected.

Some of these steps merit a bit of explanation. In step 6 we prescribe a method for calibrating the SNAP pulse. In this step, for a pulse targeting n photon levels, we wish to identify $2n$ parameters, specifically the amplitude and frequency for each of the components. To first order, the frequencies are known to be $n\chi_f$ and the amplitudes are simply the amplitude required for a single selective pulse, as determined in step 5. However in practice these pulses are not completely independent from each other, especially as we would like to push the pulse time down, to minimize the error rate, away from the limit where $t_{\text{pulse}} \gg 1/\chi_f$. Our main goal is to have an operation which effects a π rotation to the transmon regardless of the number of photons in the cavity. To do this, we prepare a given photon number n, play the current pulse, scanning the amplitude and detuning, to determine a correction which enhances the probability of ending up in $|f\rangle$. We apply this correction to the n-th component of the SNAP pulse, and then proceed to the next photon number $n+1$. Because the components are not necessarily independent, we may need to repeat this entire procedure a few times in order to fully converge.

In step 7 we seek to determine the relative phases for each of the n components of the pulse. Ideally, if we set all of the pulses to the same phase, the result should be the identity operation, up to an overall cavity rotation of $t_{\text{pulse}}\chi_f$. In practice this is not the case for two reasons: first the pulses are not acting independently, second the cavity evolves under its own internal non-linearity $\frac{K}{2}(\boldsymbol{a}^\dagger)^2\boldsymbol{a}^2$. In order to compensate for both of these effects (and the overall cavity rotation if desired, although this is not strictly necessary), we can adjust the phases of each of the components.

There are many possible methods for tuning the cavity phases. Our approach involves characterizing a series of states via Wigner tomography. For each state we construct a maximum likelihood density matrix, the largest eigenvalue eigenvector of which should resemble the desired state. The argument of each of the amplitudes comprising this eigenvector give us a set of phases to assign to the state. We begin by preparing a state of interest which is a superposition of all of the relevant photon number states, in our case the “kitten code” state $(|0\rangle + \sqrt{2}\,|2\rangle + |4\rangle)/2$. We characterize the phases of this state using the above method to establish a baseline. We then prepare the same state and apply the SNAP operation, with zero nominal phases, such that the target operation is the identity, and characterize the resulting state. The difference between the phases inferred from the second state and the first state are used to correct the phases of the SNAP drive. To get better accuracy we can go on to repeated SNAP applications: prepare our baseline state and apply the nominal identity operation 2, 4, or 8 times, and characterize the phases of the resulting state. This allows us to exacerbate small phase offsets and get better calibration accuracy.

Finally, we should discuss how to measure the cavity coherence, as is required in step 8. The most general method to achieve this result is to perform a Wigner tomography. However, when we are scanning over many points, and wish to establish the coherence at each point, the overhead of full state tomography is cumbersome. Instead of this, we construct using optimal control, an operation which implements the following transformation:

$$|g, +_L\rangle \mapsto |g, \psi_1\rangle \tag{9.29}$$

$$|g, -_L\rangle \mapsto |e, \psi_2\rangle \tag{9.30}$$

$$|g, +_E\rangle \mapsto |g, \psi_1\rangle \tag{9.31}$$

$$|g, -_E\rangle \mapsto |e, \psi_2\rangle \tag{9.32}$$

where the cavity logical states are the binomial code words (equation 3.66), and the cavity states $|\psi_1\rangle, \ldots, |\psi_4\rangle$ are arbitrary states which are fixed by the gauge choice in the optimal control, following the methods presented in section 4.3.3. This allows us to map the preservation or loss of the relative photon number phases, regardless of cavity decay, onto the transmon state. in a way which maximizes contrast.

9.6 Characterizing the FT SNAP

We proceed now to the actual experimental results, as performed on the sample described in table 5.1. We can begin by demonstrating the effectiveness of the SNAP pulse in rotating the transmon state between $|g\rangle$ and $|f\rangle$ without occupying $|e\rangle$. We see the pulse and transmon population trajectory in figure 9.3. The observed trajectory matches well against the predicted trajectory, and there is essentially no excited state population except for the slow accumulation resulting from decay from $|f\rangle$. We note that the evolution is not smoothly continuous, but rather goes in steps. As the number of photons addressed increases, and consequently the number of frequency components, the pulse times become more and more defined. This does not drastically change the analysis, although there can be a concern regarding the divergence of different photon number trajectories. The degree to which the trajectories differ is the degree to which the time of the jump conveys information about which photon number state we are in.[4] This knowledge of photon number translates directly to cavity dephasing, and therefore should be avoided by using a long enough pulse.

[4] It would be an interesting, as of yet unperformed, exercise to bound the difference in trajectories as a function of the length of the SNAP operation.

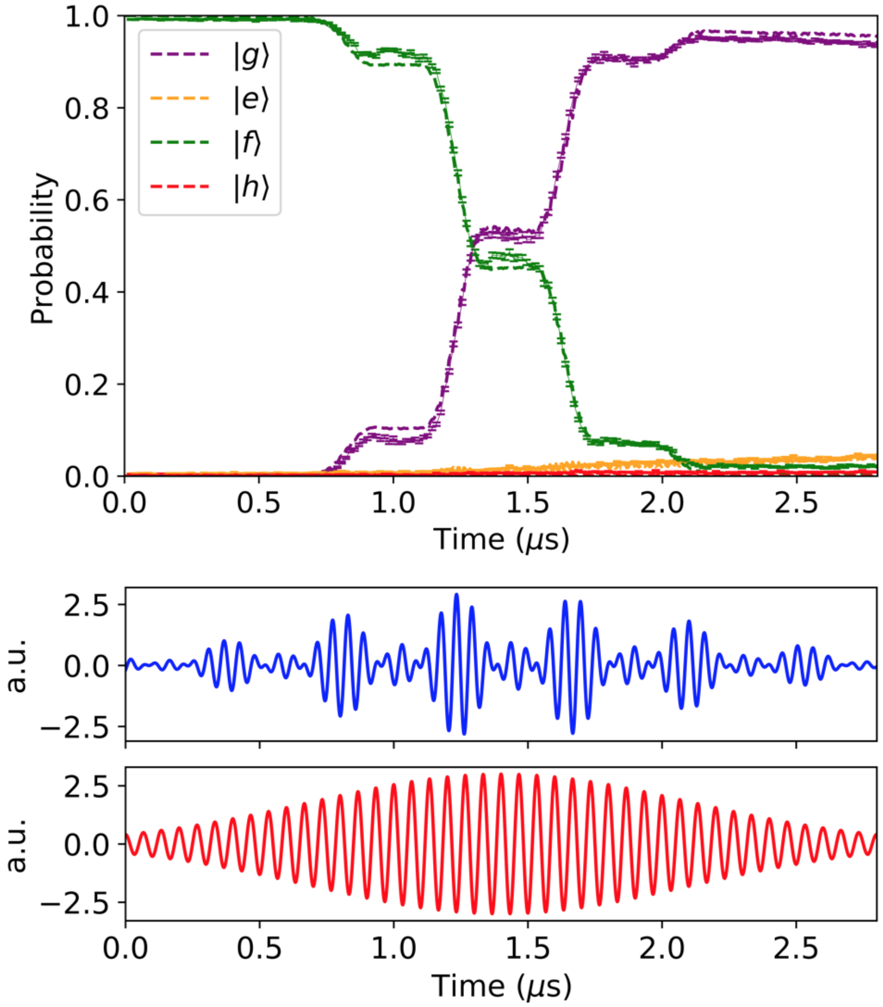


Figure 9.3: **Measured trajectory of transmon state throughout Raman SNAP.** We can measure the trajectory by stopping the pulse suddenly and measuring the transmon state using a readout which discriminates between $|g\rangle$, $|e\rangle$, $|f\rangle$ and $|h\rangle$. The total pulse has peaks separated by $\frac{\pi}{\chi_f} \approx 416$ ns as a result of the drives components separated by $2\chi_f$ in frequency. We see good agreement with the simulated trajectory (dashed lines).

We can next demonstrate the maintenance of cavity coherence regardless of the transmon measurement outcome. To do so we will start by preparing an encoded cavity state, in this case, a binomial state $|{+}_L\rangle$ (section 3.7.1). We will then apply a logical Z rotation by angle $\pi/2$, which should ideally result in the state $|i{+}_L\rangle$. We implement this operation using the SNAP protocol, acting on Fock states $|0\rangle$, $|2\rangle$ and $|4\rangle$, with phases 0, $\pi/2$ and 0, respectively. In figure 9.4 we can see the cavity state postselected on the outcome of the measurement used in the FT SNAP protocol. The three outcomes correspond to the three dominant trajectories: no error, transmon decay, and transmon dephasing. In the no error case, the operation completes as desired, resulting in the correct final state. In the decay trajectory, we see (and this only holds when the χ-cancelling pump is applied) that the cavity coherence is preserved and additionally

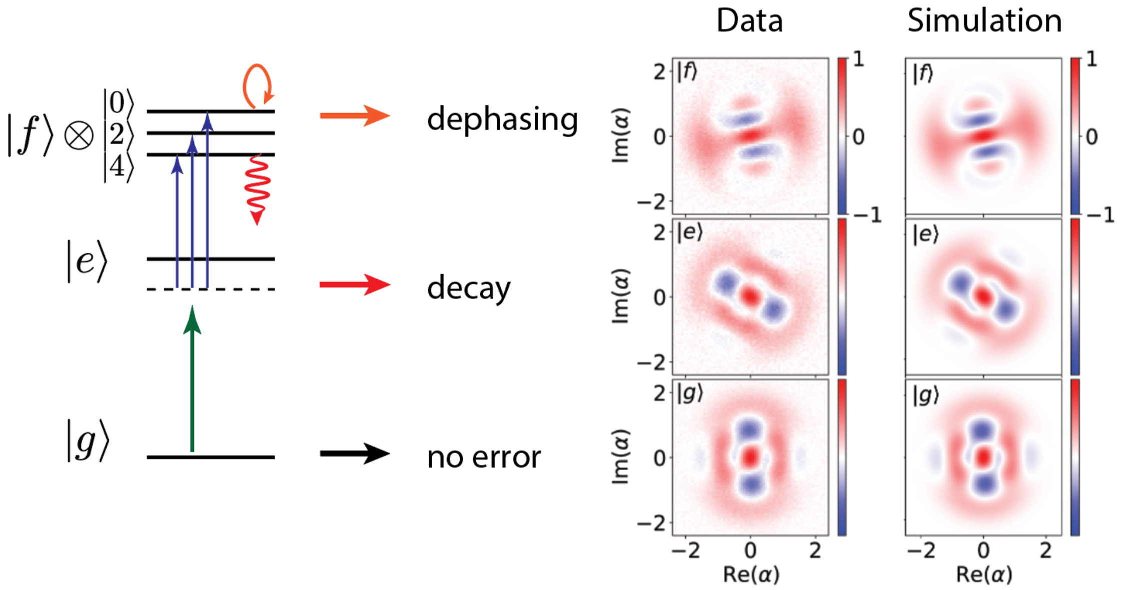


Figure 9.4: **Wigner functions showing cavity state from FT SNAP conditional on transmon.** The three transmon states correspond to the three dominant transmon trajectories, no error ($\approx 94\%$), decay ($\approx 3\%$) and dephasing ($\approx 1\%$). We start the cavity off in the binomial code word $|{+}_L\rangle$, resembling a horizontal cat state. The effect of the operation is a $\pi/2$ rotation about the logical $\boldsymbol{\sigma}_z$, ideally producing the state $|i{+}_L\rangle$. We see this state in the $|g\rangle$ and $|e\rangle$ Wigner functions. In $|e\rangle$ there is an additional cavity phase space rotation which can be dealt with in software by updating the phase of the cavity drives. In the dephasing case, the Wigner function resembles the input state, with the exception of some Kerr evolution producing a slight distortion. These states agree well with numerical simulation.

that the operation completes. There is an overall cavity rotation, which can be compensated for in software as discussed. Finally the dephasing trajectory maintains coherence as well, and is *close* to the original state. The slight modification is due to the Kerr evolution. In principle this additional evolution could be cancelled in the subsequent SNAP pulse by modifying the SNAP phases, however because of the limited pulse memory and lack of ability to dynamically construct the SNAP pulse from its components, this was not performed here.

We can measure the SNAP performance quantitatively by performing randomized benchmarking. In order to do so, we prepare a set of optimal control pulses implementing the Clifford operations on the encoded subspace, as was done in chapter 6. We then can string these operations together (as done in figure 6.13) in sequences of varying length, in order to establish a baseline fidelity curve. Then, by interleaving the SNAP operation between the optimal control pulses, and comparing with the baseline, we can establish the operation fidelity (figure 9.5). We compare the fidelity of the original SNAP protocol (omitting the χ cancelling drive as well as the transmon measurement and feedback) with that of the full fault-tolerant protocol. The exponential decay is fit with a number constant of approximately 14 and 20 for the non-FT and FT protocols respectively. After subtracting the baseline constant of roughly 40, we are left with error rates of 4.6% and 2.4% for the respective protocols. This is approximately a factor of two performance gain as a result of the work done to recapture the cavity coherence.

This result indicates that the FT protocol effectively addresses half of the error in the system. Where is the rest of the error coming from? We can look at the diagram in figure 9.6 to analyze the ways in which error arises and propagates. Several types of errors are accounted for here, including first and second order transmon transitions during the SNAP, transitions during the readout, cavity transitions, and readout induced cavity dephasing. The total predicted 2.1% error per operation is in moderately good agreement with the measured value of 2.4%. In this diagram, we can identify which error channels are dominant, and which are negligible. To start with second order transitions, such as double decay from $|f\rangle$ to $|g\rangle$, are of small enough probability to not contribute. Cavity decay however is quite significant, both during the SNAP (0.4%) and the readout (0.5%). In principle this component can be addressed by introducing the cavity error correction, since the operation can be made to preserve the

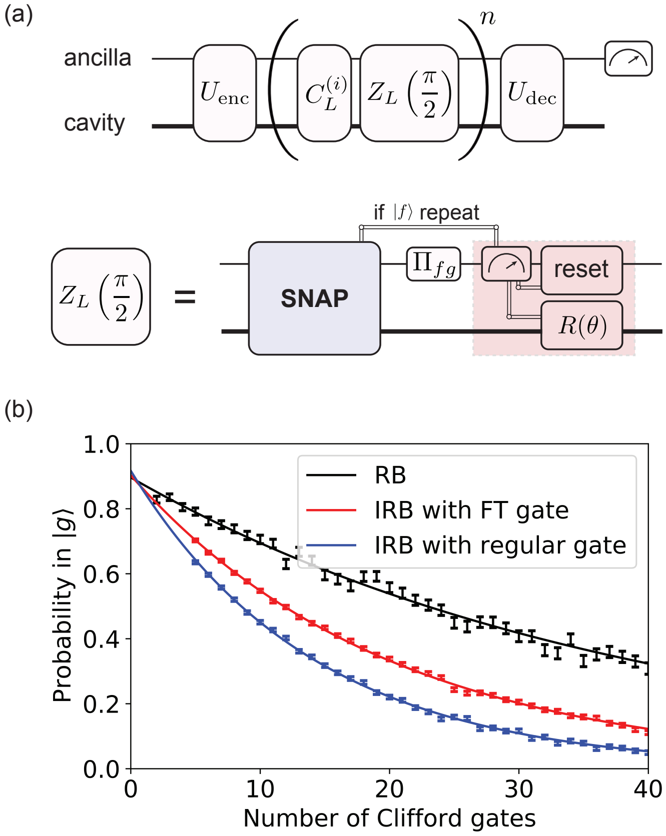


Figure 9.5: **Randomized benchmarking the FT SNAP operation.** The interleaved randomized benchmarking (iRB) protocol (Magesan et al., 2012) is similar to the one performed in the optimal control characterization experiment in figure 6.13. The general logical Clifford operations $(C_L^(i))$ are again performed by optimal control pulses. The interleaved part is the SNAP operation, which in the FT variant contains a readout, feedforward reset of transmon, feedforward update of cavity phase, and potential feedback restarting of the sequence when measuring $|f\rangle$. By comparing the decay number constants of 14 and 20 for the non-FT and FT iRB to the decay constant of ≈ 40 for the RB, we can extract error rates of 4.6% and 2.4% for the respective protocols.

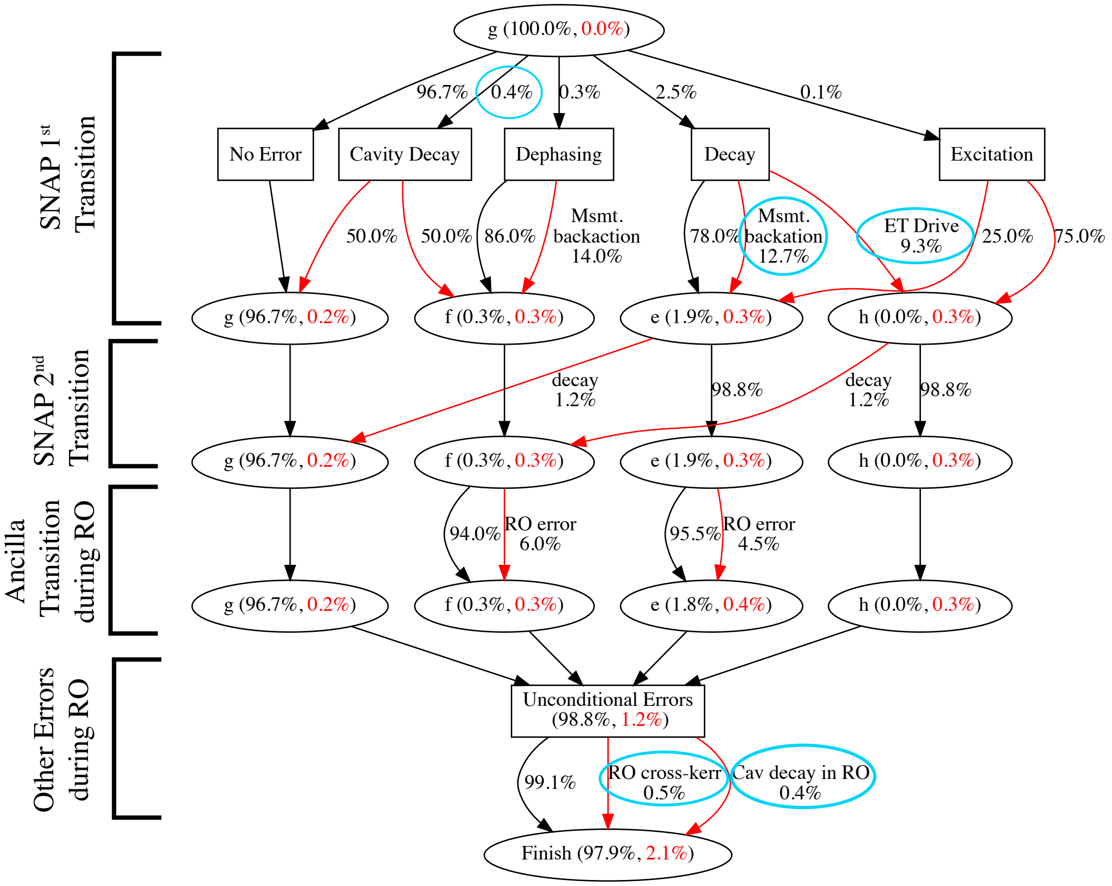


Figure 9.6: **Graph of possible error trajectories in FT SNAP protocol** At each node on the tree, the pair of numbers are probabilities. The first of these is the probability of being in the labelled state and having the logical qubit retain its coherence. The second is the probability of being in the labelled state and having the logical qubit dephased. The sum of all the numbers in a given row of nodes should be 100%. Paths between nodes are colored red if the logical qubit is effectively dephased. In the non error-corrected protocol (S_{NC}), all paths except the "No Error" case would lead to loss of logical qubit coherence. The first layer indicates single-errors occurring the SNAP pulse. The second layer accounts for second-order double ancilla transition events. The third layer accounts for ancilla decay during the readout. The final step is accounting for readout errors which do not depend on the ancilla state. Circled in blue are the primary contributions to the final total, in order, cavity loss ($\sim 0.8\%$), readout cross-Kerr ($\sim 0.5\%$), Measurement back-action from path-independence violation ($\sim 0.3\%$), decay to the third excited state $|h\rangle$ from hybridization induced by the error-transparency drive ($\sim 0.3\%$) and decay from $|e\rangle$ to $|g\rangle$ during readout ($\sim 0.2\%$).

structure of cavity errors as discussed at the end of section 9.2.[5] Therefore the dominant and concerning unaddressed error components are the readout cross-Kerr (0.5%), transmon decay during the readout (especially from $|e\rangle$, 0.2%), and χ cancelling drive induced hybridization resulting in population of $|h\rangle$ (0.4%).

While we show an improvement over the standard protocol by about a factor of 2, the presence of the previously discussed errors, against which we are unprotected, limits the gain without further experimental improvements. In order to demonstrate the degree to which we are eliminating the error channel associated with transmon dephasing and decay during the SNAP pulse, we can deliberately increase the rates associated with these processes, and measure the ratio of the fidelities of the standard and fault-tolerant protocols. Of course both fidelities will decrease as errors are added, but the rate of fidelity loss should be much faster for the standard SNAP protocol.

In order to increase the transmon dephasing rate, without affecting its decay rate, we can add a weak resonant drive to the readout mode, increasing its steady state population, and thus the dephasing rate (following an analogue to equation 8.18). In order to increase the decay rate independently of the dephasing rate, we can add a noisy input, with a narrow bandwidth (~ 10 MHz) centered on ω_{ef}. For a timescale long compared to the bandwidth ($\delta t \gg 100$ ns) the effect of this drive appears to be an incoherent transition probability from $|f\rangle$ to $|e\rangle$ or vice versa. The fact that we increase the excitation probability from $|e\rangle$ to $|f\rangle$ is actually acceptable, since we are tolerant to this transition as well, although now we should interpret the added rate as a "bit-flip" rate rather than a decay rate. We can see in the top half of figure 9.7 how the measured final transmon state populations change as a function of injected noise strength. The fact that only the $|e\rangle$ ($|f\rangle$) population remains flat for increased dephasing (bit-flip) rates indicates that our modifications to the decoherence properties are precise. Additionally, these probabilities agree with a simulation (dotted lines) accounting for the (independently measured) dephasing and bit flip rates.

We can go on to re-measure the gate infidelity, as measured by interleaved RB, and show this as a function of injected error rate, which we compared with the predicted infidelity, which

[5] In this work, however, we only performed the drive on the even photon number peaks. Maintaining the structure of the errors would require driving on the odd peaks as well.

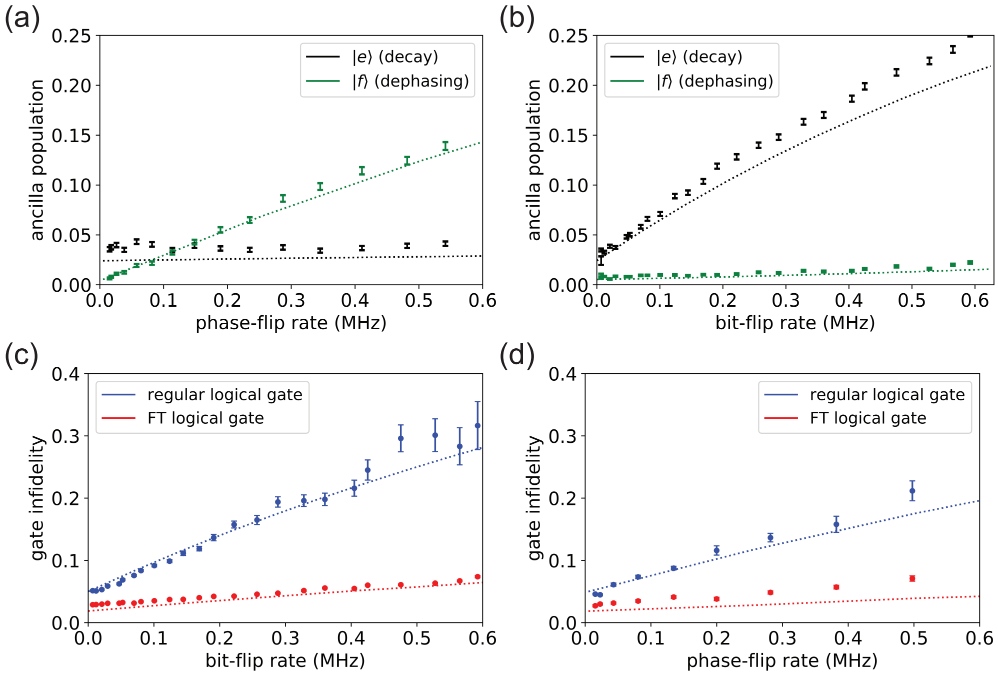


Figure 9.7: **Characterizing the effect of injected errors on the FT SNAP operation** Errors are injected into the system in one of two ways. Either the steady state readout mode occupation is increased by a weak resonant drive at ω_{RO}, inducing an additional phase-flip rate, or a noise source, with 5 MHz bandwidth, is placed on the ω_{ef} mixing chain, producing an additional bit-flip rate (the noisy drive produces both $|f\rangle \rightarrow |e\rangle$ as well as $|e\rangle \rightarrow |f\rangle$). We see in the top two plots that increasing the phase flip rate increases the probability of measuring $|f\rangle$ and that increasing the bit flip rate increases the probability of measuring $|e\rangle$. While both the FT and non-FT protocols' fidelities suffer as a result of these added errors, the FT protocol suffers much less, increasing the FT/non-FT performance ratio from 2 (with native errors) to a factor of 4.9 (with maximal added bit-flip errors).

is calculated as in figure 9.6. In our model, we assume we are completely protected against, the action of the decay or dephasing event itself, but we still predict increasing infidelity as a result of both decay during the readout and pump-induced hybridization. The predictions match the measured infidelities, for the most part, although we moderately underestimate the infidelity associated with increased transmon dephasing. At maximum the infidelity ratio between the standard and FT protocols reaches 3.3 for injected phase flips, and 4.9 for injected bit flips.

Chapter 10

Conclusion and Prospects

I would like to conclude by discussing some of the options I see for future projects. Some of these ideas are further developments of the ideas presented in this thesis, while others are simply areas I think are promising directions that build on the work done here at Yale.

Optimal control beyond the dispersive regime We have shown how control can be made faster and higher fidelity by moving from time-disjoint transmon and cavity drives using Gaussian pulses to a more general class of operations. However there are several "arbitrary" restrictions which remain. Most notably is our use of drives which are centered on the transmon and cavity resonant frequencies, with a relatively narrow bandwidth. This restriction prevents us from using higher states of the transmon, or many of the types of sideband transitions considered in chapter 7. There are two approaches one could take. The more conservative approach is to add new frequency ranges one at a time, adding new effective Hamiltonians for drives in these ranges. More radical would be to move to a "full-bandwidth" Hamiltonian, for instance of the form 7.5, trying to allow drives which wring the maximal amount of performance from the system allowed. This would be interesting to try in conjunction with a full-bandwidth AWG, as was used in Raftery et al. (2017).

Using χ cancellation to improve transmon coherence A very straightforward modification of the experiment shown in section 8.3 is to apply a χ cancelling drive between the *readout mode* and the transmon. In this case, the readout is the "fast" mode and transmon the "slow" mode. By decoupling them we can protect the transmon from the finite-temperature induced transitions in the readout. The value of this proposition would depend on the readout temperature as well as the strength of the readout-transmon interaction χ_{RO}. However, the prospect for generally applicable performance enhancement is tantalizing. A more desirable situation would be to go the other way, introducing a drive to *turn on* the interaction rather than turn off, although I know of no way to do this currently.

Engineering χ via transmon tunability It is possible to achieve the effect $\chi_e = \chi_f$ without introducing any drive terms necessarily if one is willing to finely tune the cavity-transmon detuning.[1] One can work out from a second order perturbation theory argument that, in the limit of the large detuning ($\Delta \gg g$), that we have

$$\chi_{ef} \approx \frac{g^2}{\Delta}\frac{\alpha_T(\alpha_T - \Delta)}{(\alpha_T + \Delta)(2\alpha_T + \Delta)}. \tag{10.1}$$

This is can be made zero by choosing $\Delta \equiv \omega_{ge} - \omega_c = \alpha_T$. Note that this is a detuning which puts the cavity above the transmon, as α_T is typically negative. This can be seen in a numerical diagonalization as well (figure 10.1). While it might be possible to reach this point by careful fabrication, a more robust method would be to use a flux-tunable transmon. Attempting this would require the further development of methods of introducing flux bias lines which do not spoil the high quality factors of the seamless aluminum cavities.

Extending error-transparent fault-tolerance to GKP protocols The success of cat code error correction has reignited interest in the oldest of oscillator based encodings, the GKP code (Gottesman et al., 2001). As with the cat code, the fundamental task in GKP error correction is performing error syndrome measurements. Unlike the cat code, the outcome of this syndrome measurement is not a single bit (i.e. parity) but rather a continuous quantity, namely the position or momentum modulo some constant. There have been proposals (e.g. (Terhal and Weigand, 2016)) on how this can be implemented in cQED. Much like the parity measurement, the possibility of decay during the syndrome measurement could limit the performance of the error

[1] This was pointed out to us by Mostafa Khezri in a private communication

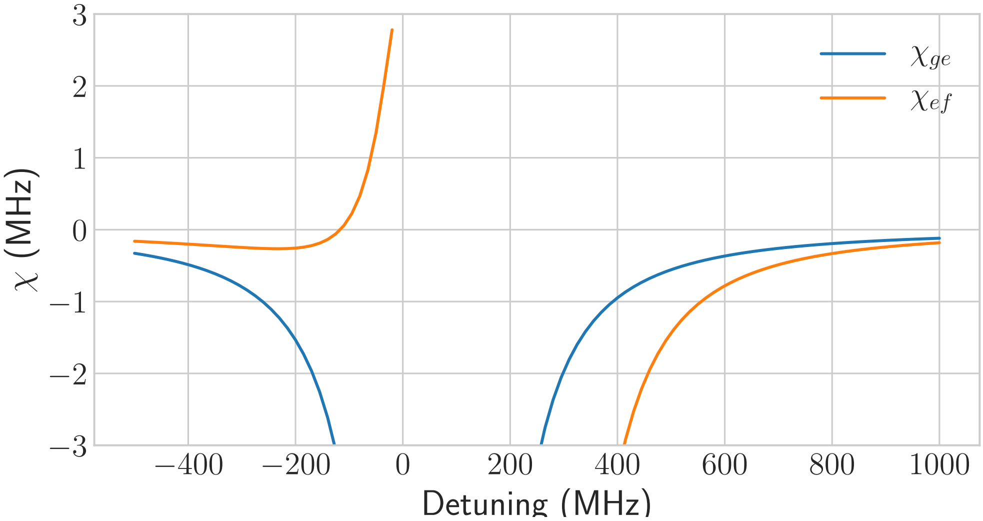


Figure 10.1: "Natural" χ matching by setting the detuning. Here a numerical diagonalization of an anharmonic Jaynes-Cummings Hamiltonian ($\alpha_T = 130$ MHz, $g = 20$ MHz) has a zero-crossing in the value of χ_{fe} at around α_T.

correction. The central component is the conditional displacement, $e^{(\alpha \boldsymbol{a} - \text{h.c.})|e\rangle\langle e|}$. Making this component, which is the subject of ongoing work at Yale, fault-tolerant is a simple modification of the protocols shown in chapter 8 and 9.

Optimal control for fault-tolerance An ultimate synthesis of the methods presented in this thesis would be to find ways of using numerical optimal control which also maintains some form of fault-tolerance. It would be desirable to both prevent propagation of transmon errors, and to maintain the structure of cavity errors, so that cavity error correction could deal with the errors occuring during the operation. This is a difficult task. The strength of optimal control comes from its ability to navigate the continuously connected space of unitary transformations. As soon as we impose fault-tolerance, our landscape could become much more disjointed. We may ask for a local variation which minimizes error susceptibility, but baking it in from the beginning would require a different approach to just adding penalty terms.

Improving transmon readout with non-linear filtering Error correction and fault-tolerance inherently requires performing measurements and acting on that information. These measurements must have high state-discrimination fidelity, but moreover, should produce the measured state as its output. Improving the readout in both of these regards is a crucial ingredient in realizing error correction protocols which are actually useful in practice. While the theory of how to best turn a measured signal into a state assignment is well worked out in the ideal case (Hatridge et al., 2013), the analysis breaks down in the presence of either transmon state transitions, or of non-linearity of the readout mode. In order to extract the maximum performance of the readout, it would be advantageous to employ a more sophisticated method of state assignment, using more recently developed machine learning tools as the assignment algorithm.

Hardware efficient quantum von Neumann architecture The paradigm of "hardware-efficiency" has led us to the cat code, where a whole array of two level systems is replaced by a single box, allowing an error corrected logical qubit to be constructed and controlled via a single Josephson junction. Can we push hardware efficiency further? One way of doing so follows the approach developed by Naik et al. (2017), where a single multimode resonator can host many Bosonic modes, each of which could encode a qubit. The advantage of this approach is that all of these degrees of freedom can be controlled by a single ancillary qubit (in their case, a parametrically modulated SQUID junction). This architecture mirrors the traditional "von Neumann architecture" for classical computers, which separates the processing unit from the memory. From my perspective, this is a natural extension of the types of simplifications which have been pursued at Yale. If such a device could be made with high-Q cavity resonators, then it would be in a great position to integrate with cavity encoded logical qubits. As I see it, the main problem with this approach is the set of always-on dispersive interactions between the modes. This problem could be alleviated by using the newly designed SNAIL element (Sivak et al., 2019) which allows one to implement bilinear SWAP operations between modes without introducing the four-wave mixing which gives rise to the dispersive shift.

Bibliography

S. Aaronson and D. Gottesman. Improved simulation of stabilizer circuits. *Phys. Rev. A*, 70(5):052328 (2004). doi:10.1103/PhysRevA.70.052328. (Cited on page 209.)

D. Aharonov and M. Ben-Or. Fault-tolerant quantum computation with constant error. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 176–188. ACM, New York, NY, USA (1997). ISBN 0-89791-888-6. doi:10.1145/258533.258579. (Cited on page 126.)

Y. Aharonov and J. Anandan. Phase change during a cyclic quantum evolution. *Phys. Rev. Lett.*, 58(16):1593–1596 (1987). doi:10.1103/PhysRevLett.58.1593. (Cited on page 23.)

V. V. Albert, K. Noh, K. Duivenvoorden, D. J. Young, R. T. Brierley, P. Reinhold, C. Vuillot, L. Li, C. Shen, S. M. Girvin, B. M. Terhal, and L. Jiang. Performance and structure of single-mode bosonic codes. *Phys. Rev. A*, 97(3):032346 (2018). doi:10.1103/PhysRevA.97.032346. (Cited on pages 51 and 52.)

V. Ambegaokar and A. Baratoff. Tunneling Between Superconductors. *Phys. Rev. Lett.*, 10(11):486–489 (1963). doi:10.1103/PhysRevLett.10.486. (Cited on page 79.)

B. E. Anderson, H. Sosa-Martinez, C. A. Riofrío, I. H. Deutsch, and P. S. Jessen. Accurate and Robust Unitary Transformations of a High-Dimensional Quantum System. *Phys. Rev. Lett.*, 114(24):240401 (2015). doi:10.1103/PhysRevLett.114.240401. (Cited on page 54.)

C. Axline. *Building Blocks for Modular Circuit QED Quantum Computing*. Ph.D. thesis, Yale University (2018). (Cited on pages 8, 83, and 121.)

C. Axline, M. Reagor, R. Heeres, P. Reinhold, C. Wang, K. Shain, W. Pfaff, Y. Chu, L. Frunzio, and R. J. Schoelkopf. An architecture for integrating planar and 3D cQED devices. *Appl. Phys. Lett.*, 109(4):042601 (2016). doi:10.1063/1.4959241. (Cited on page 81.)

D. Bacon. Fault-tolerant quantum computation and the threshold theorem. `https://courses.cs.washington.edu/courses/cse599d/06wi/lecturenotes19.pdf` (2006). (Cited on page 125.)

J. Bateman, A. Xuereb, and T. Freegarde. Stimulated Raman transitions via multiple atomic levels. *Phys. Rev. A*, 81(4):043808 (2010). doi:10.1103/PhysRevA.81.043808. (Cited on page 153.)

N. Bergeal, F. Schackert, M. Metcalfe, R. Vijay, V. E. Manucharyan, L. Frunzio, D. E. Prober, R. J. Schoelkopf, S. M. Girvin, and M. H. Devoret. Phase-preserving amplification near the quantum limit with a Josephson ring modulator. *Nature*, 465(7294):64–68 (2010). doi:10.1038/nature09035. (Cited on pages 93 and 119.)

M. V. Berry. Quantal Phase Factors Accompanying Adiabatic Changes. *Proc. R. Soc. A*, 392(1802):45–57 (1984). doi:10.1098/rspa.1984.0023. (Cited on page 23.)

L. Bishop. *Circuit Quantum Electrodynamics*. Ph.D. thesis, Yale University (2010). (Cited on page 8.)

R. Blume-Kohout, J. K. Gamble, E. Nielsen, J. Mizrahi, J. D. Sterk, and P. Maunz. Robust, self-consistent, closed-form tomography of quantum logic gates on a trapped ion qubit. *arXiv* (2013). (Cited on page 207.)

J. Blumoff. *Multiqubit experiments in 3D circuit quantum electrodynamics*. Ph.D. thesis, Yale University (2017). (Cited on pages 7, 8, 81, and 83.)

H. Bombín. Gauge color codes: optimal transversal gates and gauge fixing in topological stabilizer codes. *New J. Phys.*, 17(8):083002 (2015). doi:10.1088/1367-2630/17/8/083002. (Cited on page 33.)

T. Brecht. *Micromachined Quantum Circuits*. Ph.D. thesis, Yale University (2017). (Cited on pages 8 and 77.)

J. Bylander, S. Gustavsson, F. Yan, F. Yoshihara, K. Harrabi, G. Fitch, D. G. Cory, Y. Nakamura, J.-S. Tsai, and W. D. Oliver. Noise spectroscopy through dynamical decoupling with a superconducting flux qubit. *Nature Physics*, 7(7):565–570 (2011). doi:10.1038/nphys1994. (Cited on page 95.)

R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A Limited Memory Algorithm for Bound Constrained Optimization. *SIAM J. Sci. Comput.*, 16(5):1190–1208 (1995). doi:10.1137/0916069. (Cited on page 58.)

K. Cahill and R. Glauber. Density Operators and Quasiprobability Distributions. *Phys. Rev.*, 177(5):1882–1902 (1969). doi:10.1103/PhysRev.177.1882. (Cited on pages 20 and 203.)

A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane. Quantum Error Correction and Orthogonal Geometry. *Phys. Rev. Lett.*, 78(3):405–408 (1997). doi:10.1103/PhysRevLett.78.405. (Cited on page 33.)

E. T. Campbell, H. Anwar, and D. E. Browne. Magic-state distillation in all prime dimensions using quantum reed-muller codes. *Phys. Rev. X* (2012). (Cited on page 33.)

E. T. Campbell, B. M. Terhal, and C. Vuillot. Roads towards fault-tolerant universal quantum computation. *Nature*, 549(7671):172–179 (2017). doi:10.1038/nature23460. (Cited on page 126.)

H. Carmichael. *An Open Systems Approach to Quantum Optics*. Springer (1993). ISBN 3540566341. (Cited on page 7.)

J. Chiaverini, D. Leibfried, T. Schaetz, M. D. Barrett, R. B. Blakestad, J. Britton, W. M. Itano, J. D. Jost, E. Knill, C. Langer, R. Ozeri, and D. J. Wineland. Realization of quantum error correction. *Nature*, 432(7017):602–605 (2004). doi:10.1038/nature03074. (Cited on page 32.)

K. Chou. *Teleported operations between logical qubits in circuit quantum electrodynamics*. Ph.D. thesis, Yale University (2018). (Cited on pages 7 and 8.)

J. Chow. *Quantum Information Processing with Superconducting Qubits*. Ph.D. thesis, Yale University (2010). (Cited on pages 7 and 8.)

A. D. Córcoles, E. Magesan, S. J. Srinivasan, A. W. Cross, M. Steffen, J. M. Gambetta, and J. M. Chow. Demonstration of a quantum error detection code using a square lattice of four superconducting qubits. *Nature Communications*, 6:6979 (2015). doi:10.1038/ncomms7979. (Cited on page 32.)

D. G. Cory, M. D. Price, W. Maas, E. Knill, R. Laflamme, W. H. Zurek, T. F. Havel, and S. S. Somaroo. Experimental quantum error correction. *Phys. Rev. Lett.*, 81(10):2152–2155 (1998). (Cited on page 4.)

J. Cramer, N. Kalb, M. A. Rol, B. Hensen, M. S. Blok, M. Markham, D. J. Twitchen, R. Hanson, and T. H. Taminiau. Repeated quantum error correction on a continuously encoded qubit by real-time feedback. *Nature Communications*, 7:11526 (2016). doi:10.1038/ncomms11526. (Cited on page 32.)

D. L. Creedon, M. Goryachev, N. Kostylev, T. B. Sercombe, and M. E. Tobar. A 3D printed superconducting aluminium microwave cavity. *Appl. Phys. Lett.*, 109(3):032601 (2016). doi:10.1063/1.4958684. (Cited on page 75.)

A. W. Cross, E. Magesan, L. S. Bishop, J. A. Smolin, and J. M. Gambetta. Scalable randomised benchmarking of non-Clifford gates. *npj Quantum Inf.*, 2:16012 (2016). doi:10.1038/npjqi.2016.12. (Cited on page 209.)

J. Dalibard, K. Mølmer, and Y. Castin. Monte Carlo wave-function method in quantum optics. *J. Opt. Soc. Am. B, JOSAB*, 10(3):524–538 (1993). doi:10.1364/JOSAB.10.000524. (Cited on page 45.)

P. de Fouquieres. Implementing Quantum Gates by Optimal Control with Doubly Exponential Convergence. *Phys. Rev. Lett.*, 108(11):110504 (2012). doi:10.1103/PhysRevLett.108.110504. (Cited on page 59.)

P. de Fouquieres, S. G. Schirmer, S. J. Glaser, and I. Kuprov. Second order gradient ascent pulse engineering. *arXiv*, (2):412–417 (2011). (Cited on page 58.)

M. H. Devoret. Quantum Fluctuations In Electrical Circuits (1997). (Cited on page 4.)

D. P. DiVincenzo and P. W. Shor. Fault-Tolerant Error Correction with Efficient Quantum Codes. *Phys. Rev. Lett.*, 77(15):3260–3263 (1996). doi:10.1103/PhysRevLett.77.3260. (Cited on pages 3 and 125.)

F. Dolde, V. Bergholm, Y. Wang, I. Jakobi, B. Naydenov, S. Pezzagna, J. Meijer, F. Jelezko, P. Neumann, T. Schulte-Herbrüggen, J. Biamonte, and J. Wrachtrup. High-fidelity spin entanglement using optimal control. *Nature Communications*, 5 (2014). doi:10.1038/ncomms4371. (Cited on page 54.)

D. J. Egger and F. K. Wilhelm. Adaptive Hybrid Optimal Quantum Control for Imprecisely Characterized Systems. *Phys. Rev. Lett.*, 112(24):240503 (2014). doi:10.1103/PhysRevLett.112.240503. (Cited on page 114.)

A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Phys. Rev. A*, 86(3) (2012). doi:10.1103/PhysRevA.86.032324. (Cited on pages 33 and 126.)

J. M. Gambetta, A. Blais, D. I. Schuster, A. Wallraff, L. Frunzio, J. Majer, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf. Qubit-photon interactions in a cavity: Measurement-induced dephasing and number splitting. *Phys. Rev. A*, 74(4):042318 (2006). doi:10.1103/PhysRevA.74.042318. (Cited on page 134.)

D. Gottesman. *Stabilizer Codes and Quantum Error Correction*. Ph.D. thesis, California Institute of Technology (1997). (Cited on page 32.)

D. Gottesman. The Heisenberg Representation of Quantum Computers (1998). (Cited on page 209.)

D. Gottesman, A. Kitaev, and J. Preskill. Encoding a qubit in an oscillator. *Phys. Rev. A*, 64(1):012310 (2001). doi:10.1103/PhysRevA.64.012310. (Cited on pages 51 and 168.)

T. J. Green, J. Sastrawan, H. Uys, and M. J. Biercuk. Arbitrary quantum control of qubits in the presence of universal noise. *New J. Phys.*, 15(9) (2013). doi:10.1088/1367-2630/15/9/095004. (Cited on page 95.)

D. Greenbaum. Introduction to Quantum Gate Set Tomography (2015). (Cited on page 207.)

D. J. Griffiths. *Introduction to Quantum Mechanics (2nd Edition)*. Pearson Prentice Hall (2004). ISBN 0131118927. (Cited on page 7.)

E. L. Hahn. Spin Echoes. *Phys. Rev.*, 80(4):580–594 (1950). doi:10.1103/PhysRev.80.580. (Cited on page 61.)

S. Haroche and J.-M. Raimond. *Exploring the Quantum* (2006). ISBN 9780198509141. (Cited on page 8.)

M. Hatridge, S. Shankar, M. Mirrahimi, F. Schackert, K. Geerlings, T. Brecht, K. M. Sliwa, B. Abdo, L. Frunzio, S. M. Girvin, R. J. Schoelkopf, and M. H. Devoret. Quantum Back-Action of an Individual Variable-Strength Measurement. *Science*, 339(6116):178–181 (2013). doi:10.1126/science.1226897. (Cited on page 170.)

R. W. Heeres, P. Reinhold, N. Ofek, L. Frunzio, L. Jiang, M. H. Devoret, and R. J. Schoelkopf. Implementing a universal gate set on a logical qubit encoded in an oscillator. *Nature Communications*, 8(1):94 (2017). doi:10.1038/s41467-017-00045-1. (Cited on pages 6 and 85.)

R. W. Heeres, B. Vlastakis, E. Holland, S. Krastanov, V. V. Albert, L. Frunzio, L. Jiang, and R. J. Schoelkopf. Cavity State Manipulation Using Photon-Number Selective Phase Gates. *Phys. Rev. Lett.*, 115(13):137002 (2015). doi:10.1103/PhysRevLett.115.137002. (Cited on pages 27 and 110.)

J. Heinsoo, C. K. Andersen, A. Remm, S. Krinner, T. Walter, Y. Salathé, S. Gasparinetti, J.-C. Besse, A. Potočnik, A. Wallraff, and C. Eichler. Rapid High-fidelity Multiplexed Readout of Superconducting Qubits. *Phys. Rev. Applied*, 10(3):034040 (2018). doi:10.1103/PhysRevApplied.10.034040. (Cited on page 146.)

M. Hofheinz, H. Wang, M. Ansmann, R. C. Bialczak, E. Lucero, M. Neeley, A. D. O'Connell, D. Sank, J. Wenner, J. M. Martinis, and A. N. Cleland. Synthesizing arbitrary quantum states in a superconducting resonator. *Nature*, 459(7246):546–549 (2009). doi:10.1038/nature08005. (Cited on pages 14 and 15.)

E. T. Jaynes and F. W. Cummings. Comparison of quantum and semiclassical radiation theories with application to the beam maser. *Proc. IEEE*, 51(1):89–109 (1962). doi:10.1109/PROC.1963.1664. (Cited on page 13.)

E. Kapit. Error-transparent quantum gates for small logical qubit architectures (2017). (Cited on page 127.)

J. Kelly, R. Barends, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, A. G. Fowler, I. C. Hoi, E. Jeffrey, A. Megrant, J. Mutus, C. Neill, P. J. J. O'Malley, C. Quintana, P. Roushan, D. Sank, A. Vainsencher, J. Wenner, T. C. White, A. N. Cleland, and J. M. Martinis. Optimal Quantum Control Using Randomized Benchmarking. *Phys. Rev. Lett.*, 112(24):240504 (2014). doi:10.1103/PhysRevLett.112.240504. (Cited on page 114.)

J. Kelly, R. Barends, A. G. Fowler, A. Megrant, E. Jeffrey, T. C. White, D. Sank, J. Y. Mutus, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, I. C. Hoi, C. Neill, P. J. J. O'Malley, C. Quintana, P. Roushan, A. Vainsencher, J. Wenner, A. N. Cleland, and J. M. Martinis. State preservation by repetitive error detection in a superconducting quantum circuit. *Nature*, 519(7541):66–69 (2015). doi:10.1038/nature14270. (Cited on page 32.)

N. Khaneja, T. Reiss, C. Kehlet, T. Schulte-Herbrüggen, and S. J. Glaser. Optimal control of coupled spin dynamics: design of NMR pulse sequences by gradient ascent algorithms. *Journal of Magnetic Resonance*, 172(2):296–305 (2005). doi:10.1016/j.jmr.2004.11.004. (Cited on pages 54 and 57.)

E. Knill and R. Laflamme. Theory of quantum error-correcting codes. *Phys. Rev. A*, 55(2):900–911 (1997). doi:10.1103/PhysRevA.55.900. (Cited on page 30.)

E. Knill, R. Laflamme, and G. J. Milburn. A scheme for efficient quantum computation with linear optics. *Nature*, 409(6816):46–52 (2001). doi:10.1038/35051009. (Cited on page 4.)

E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland. Randomized benchmarking of quantum gates. *Phys. Rev. A*, 77(1):012307 (2008). doi:10.1103/PhysRevA.77.012307. (Cited on page 208.)

J. Koch, T. Yu, J. M. Gambetta, A. A. Houck, D. I. Schuster, J. Majer, A. Blais, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf. Charge-insensitive qubit design derived from the Cooper pair box. *Phys. Rev. A*, 76(4):042319 (2007). doi:10.1103/PhysRevA.76.042319. (Cited on page 79.)

S. Krastanov, V. V. Albert, C. Shen, C.-L. Zou, R. W. Heeres, B. Vlastakis, R. J. Schoelkopf, and L. Jiang. Universal control of an oscillator with dispersive coupling to a qubit. *Phys. Rev. A*, 92(4):040303 (2015). doi:10.1103/PhysRevA.92.040303. (Cited on pages 23 and 26.)

J. Krause, M. O. Scully, T. Walther, and H. Walther. Preparation of a pure number state and measurement of the photon statistics in a high- Qmicromaser. *Phys. Rev. A*, 39(4):1915–1921 (1989). doi:10.1103/PhysRevA.39.1915. (Cited on page 14.)

R. Laflamme, C. Miquel, J. P. Paz, and W. H. Zurek. Perfect Quantum Error Correcting Code. *Phys. Rev. Lett.*, 77(1):198–201 (1996). doi:10.1103/PhysRevLett.77.198. (Cited on page 32.)

C. K. Law and J. H. Eberly. Arbitrary Control of a Quantum Electromagnetic Field. *Phys. Rev. Lett.*, 76(7):1055–1058 (1996). doi:10.1103/PhysRevLett.76.1055. (Cited on pages 14 and 15.)

F. Lecocq, I. M. Pop, Z. Peng, I. Matei, T. Crozes, T. Fournier, C. Naud, W. Guichard, and O. Buisson. Junction fabrication by shadow evaporation without a suspended bridge. *Nanotechnology*, 22(31):315302 (2011). doi:10.1088/0957-4484/22/31/315302. (Cited on page 86.)

Z. Leghtas, G. Kirchmair, B. Vlastakis, M. H. Devoret, R. J. Schoelkopf, and M. Mirrahimi. Deterministic protocol for mapping a qubit to coherent state superpositions in a cavity. *Phys. Rev. A*, 87(4):042315 (2013a). doi:10.1103/PhysRevA.87.042315. (Cited on page 22.)

Z. Leghtas, G. Kirchmair, B. Vlastakis, R. J. Schoelkopf, M. H. Devoret, and M. Mirrahimi. Hardware-Efficient Autonomous Quantum Memory Protection. *Phys. Rev. Lett.*, 111(12):120501 (2013b). doi:10.1103/PhysRevLett.111.120501. (Cited on page 38.)

Z. Leghtas, S. Touzard, I. M. Pop, A. Kou, B. Vlastakis, A. Petrenko, K. M. Sliwa, A. Narla, S. Shankar, M. J. Hatridge, M. Reagor, L. Frunzio, R. J. Schoelkopf, M. Mirrahimi, and M. H. Devoret. Confining the state of light to a quantum manifold by engineered two-photon loss. *Science*, 347(6224):853–857 (2015). doi:10.1126/science.aaa2085. (Cited on pages 46, 123, and 135.)

M. Leskes, P. K. Madhu, and S. Vega. Floquet theory in solid-state nuclear magnetic resonance. *Progress in Nuclear Magnetic Resonance Spectroscopy*, 57(4):345–380 (2010). doi:10.1016/j.pnmrs.2010.06.002. (Cited on pages 194 and 195.)

N. Leung, M. Abdelhafez, J. Koch, and D. Schuster. Speedup for quantum optimal control from automatic differentiation based on graphics processing units. *Phys. Rev. A*, 95(4):042318 (2017). doi:10.1103/PhysRevA.95.042318. (Cited on page 66.)

L. Li, C.-L. Zou, V. V. Albert, S. Muralidharan, S. M. Girvin, and L. Jiang. Cat Codes with Optimal Decoherence Suppression for a Lossy Bosonic Channel. *Phys. Rev. Lett.*, 119(3):030502 (2017). doi:10.1103/PhysRevLett.119.030502. (Cited on pages 44 and 48.)

D. A. Lidar, I. L. Chuang, and K. B. Whaley. Decoherence-Free Subspaces for Quantum Computation. *Phys. Rev. Lett.*, 81(12):2594–2597 (1998). doi:10.1103/PhysRevLett.81.2594. (Cited on page 127.)

A. F. Linskens, I. Holleman, N. Dam, and J. Reuss. Two-photon Rabi oscillations. *Phys. Rev. A*, 54(6):4854–4862 (1996). doi:10.1103/PhysRevA.54.4854. (Cited on page 153.)

S. Lloyd and S. L. Braunstein. Quantum computation over continuous variables. *Phys. Rev. Lett.*, 82(8):1784–1787 (1999). doi:10.1103/PhysRevLett.82.1784. (Cited on page 12.)

D. Loss and D. P. DiVincenzo. Quantum computation with quantum dots. *Phys. Rev. A*, 57(1):120–126 (1998). doi:10.1103/PhysRevA.57.120. (Cited on page 4.)

W. Ma and L. Jiang. General fault-tolerant quantum gates for markovian ancilla noise (in preparation). (Cited on page 147.)

E. Magesan, J. M. Gambetta, and J. Emerson. Scalable and Robust Randomized Benchmarking of Quantum Processes. *Phys. Rev. Lett.*, 106(18):180504 (2011). doi:10.1103/PhysRevLett.106.180504. (Cited on pages 111, 113, 208, and 209.)

E. Magesan, J. M. Gambetta, B. R. Johnson, C. A. Ryan, J. M. Chow, S. T. Merkel, M. P. da Silva, G. A. Keefe, M. B. Rothwell, T. A. Ohki, M. B. Ketchen, and M. Steffen. Efficient Measurement of Quantum Gate Error by Interleaved Randomized Benchmarking. *Phys. Rev. Lett.*, 109(8):080505 (2012). doi:10.1103/PhysRevLett.109.080505. (Cited on pages 111, 113, 163, and 209.)

W. Magnus. On the exponential solution of differential equations for a linear operator. *Communications on Pure and Applied Mathematics*, 7(4):649–673 (1954). doi:10.1002/cpa.3160070404. (Cited on page 11.)

E. S. Mananga and T. Charpentier. Introduction of the Floquet-Magnus expansion in solid-state nuclear magnetic resonance spectroscopy. *J. Chem. Phys.*, 135(4):044109 (2011). doi:10.1063/1.3610943. (Cited on page 194.)

D. C. McKay, C. J. Wood, S. Sheldon, J. M. Chow, and J. M. Gambetta. Efficient Z gates for quantum computing. *Phys. Rev. A*, 96(2):022330 (2017). doi:10.1103/PhysRevA.96.022330. (Cited on page 62.)

P. Meystre, J. J. Slosser, and M. Wilkens. Cat in a cold niobium box. *Optics Communications*, 79(5):300–304 (1990). doi:10.1016/0030-4018(90)90073-3. (Cited on page 14.)

M. H. Michael, M. Silveri, R. T. Brierley, V. V. Albert, J. Salmilehto, L. Jiang, and S. M. Girvin. New Class of Quantum Error-Correcting Codes for a Bosonic Mode. *Phys. Rev. X*, 6(3):031006 (2016). doi:10.1103/PhysRevX.6.031006. (Cited on pages 35, 46, and 50.)

M. Mirrahimi, Z. Leghtas, V. V. Albert, S. Touzard, R. J. Schoelkopf, L. Jiang, and M. H. Devoret. Dynamically protected cat-qubits: a new paradigm for universal quantum computation. *New J. Phys.*, 16(4) (2014). doi:10.1088/1367-2630/16/4/045014. (Cited on pages 38 and 46.)

M. Mirrahimi and P. Rouchon. Dynamics and control of open quantum systems. `https://who.rocq.inria.fr/Mazyar.Mirrahimi/QuantSys2015.pdf` (2015). (Cited on page 194.)

F. Motzoi, J. M. Gambetta, S. T. Merkel, and F. K. Wilhelm. Optimal control methods for rapidly time-varying Hamiltonians. *Phys. Rev. A*, 84(2):022307 (2011). doi:10.1103/PhysRevA.84.022307. (Cited on page 65.)

S. O. Mundhada, A. Grimm, J. Venkatraman, Z. K. Minev, S. Touzard, N. E. Frattini, V. V. Sivak, K. Sliwa, P. Reinhold, S. Shankar, M. Mirrahimi, and M. H. Devoret. Experimental implementation of a Raman-assisted six-quanta process (2018). (Cited on pages 46 and 123.)

R. K. Naik, N. Leung, S. Chakram, P. Groszkowski, Y. Lu, N. Earnest, D. C. McKay, J. Koch, and D. I. Schuster. Random access quantum information processors using multimode circuit quantum electrodynamics. *Nature Communications*, 8(1):1904 (2017). doi:10.1038/s41467-017-02046-6. (Cited on pages 76 and 170.)

I. Najfeld and T. F. Havel. Derivatives of the Matrix Exponential and Their Computation. *Advances in Applied Mathematics*, 16(3):321–375 (1995). doi:10.1006/aama.1995.1017. (Cited on pages 57 and 193.)

Y. Nakamura, Y. A. Pashkin, and J. S. Tsai. Coherent control of macroscopic quantum states in a single-Cooper-pair box. *Nature*, 398(6730):786–788 (1999). doi:10.1038/19718. (Cited on page 78.)

M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press (2011). ISBN 9781107002173. (Cited on pages 7, 22, and 30.)

D. Nigg, M. Müller, E. A. Martinez, P. Schindler, M. Hennrich, T. Monz, M. A. Martin-Delgado, and R. Blatt. Quantum computations on a topologically encoded qubit. *Science*, 345(6194):302–305 (2014). doi:10.1126/science.1253742. (Cited on page 32.)

S. E. Nigg, H. Paik, B. Vlastakis, G. Kirchmair, and S. Shankar. Black-box superconducting circuit quantization. *Phys. Rev. Lett.* (2012). doi:10.1103/PhysRevLett.108.240502. (Cited on page 81.)

J. Nocedal and S. J. Wright. *Numerical Optimization (Springer Series in Operations Research and Financial Engineering)*. Springer (2000). ISBN 0387987932. (Cited on page 58.)

K. Noh, V. V. Albert, and L. Jiang. Quantum Capacity Bounds of Gaussian Thermal Loss Channels and Achievable Rates With Gottesman-Kitaev-Preskill Codes. *IEEE Trans. Inform. Theory*, 65(4):2563–2582 (2019). doi:10.1109/TIT.2018.2873764. (Cited on page 51.)

N. Ofek, A. Petrenko, R. Heeres, P. Reinhold, Z. Leghtas, B. Vlastakis, Y. Liu, L. Frunzio, S. M. Girvin, L. Jiang, M. Mirrahimi, M. H. Devoret, and R. J. Schoelkopf. Extending the lifetime of a quantum bit with error correction in superconducting circuits. *Nature*, 536(7617):441–445 (2016). doi:10.1038/nature18949. (Cited on pages 42, 135, and 145.)

H. Paik, D. I. Schuster, L. S. Bishop, G. Kirchmair, G. Catelani, A. P. Sears, B. R. Johnson, M. J. Reagor, L. Frunzio, L. I. Glazman, S. M. Girvin, M. H. Devoret, and R. J. Schoelkopf. Observation of High Coherence in Josephson Junction Qubits Measured in a Three-Dimensional Circuit QED Architecture. *Phys. Rev. Lett.*, 107(24):240501 (2011). doi:10.1103/PhysRevLett.107.240501. (Cited on page 80.)

D. M. Pozar. *Microwave Engineering*. Wiley (2011). ISBN 0470631554. (Cited on pages 76 and 83.)

J. Preskill. Fault-tolerant quantum computation. *arXiv* (1997). (Cited on pages 3 and 125.)

J. Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79 (2018). doi:10.22331/q-2018-08-06-79. (Cited on page 4.)

T. Proctor, K. Rudinger, K. Young, M. Sarovar, and R. Blume-Kohout. What Randomized Benchmarking Actually Measures. *Phys. Rev. Lett.*, 119(13):130502 (2017). doi:10.1103/PhysRevLett.119.130502. (Cited on page 209.)

H. Rabitz, M. Hsieh, and C. Rosenthal. Landscape for optimal control of quantum-mechanical unitary transformations. *Phys. Rev. A*, 72(5):052337 (2005). doi:10.1103/PhysRevA.72.052337. (Cited on page 54.)

J. Raftery, A. Vrajitoarea, G. Zhang, Z. Leng, S. J. Srinivasan, and A. A. Houck. Direct digital synthesis of microwave waveforms for quantum computing (2017). (Cited on pages 94 and 167.)

M. Reagor. *Superconducting Cavities for Circuit Quantum Electrodynamics*. Ph.D. thesis, Yale University (2015). (Cited on pages 75 and 86.)

M. Reagor, W. Pfaff, C. Axline, R. W. Heeres, N. Ofek, K. M. Sliwa, E. Holland, C. Wang, J. Blumoff, K. Chou, M. J. Hatridge, L. Frunzio, M. H. Devoret, L. Jiang, and R. J. Schoelkopf. Quantum memory with millisecond coherence in circuit QED. *Phys. Rev. B*, 94(1):014506 (2016). doi:10.1103/PhysRevB.94.014506. (Cited on pages 76 and 81.)

M. Reed. *Entanglement and Quantum Error Correction with Superconducting Qubits*. Ph.D. thesis, Yale University (2013). (Cited on pages 7 and 8.)

M. D. Reed, L. DiCarlo, S. E. Nigg, L. Sun, L. Frunzio, S. M. Girvin, and R. J. Schoelkopf. Realization of three-qubit quantum error correction with superconducting circuits. *Nature*, 482(7385):382–385 (2012). doi:10.1038/nature10786. (Cited on page 32.)

M. D. Reed, B. R. Johnson, A. A. Houck, L. DiCarlo, J. M. Chow, D. I. Schuster, L. Frunzio, and R. J. Schoelkopf. Fast reset and suppressing spontaneous emission of a superconducting qubit. *Appl. Phys. Lett.*, 96(20):203110 (2010). doi:10.1063/1.3435463. (Cited on page 83.)

P. Reinhold, S. Rosenblum, wenlong Ma, L. Frunzio, L. Jiang, and R. Schoelkopf. Error-corrected gates on an encoded qubit (in preparation). (Cited on page 7.)

C. Rigetti, J. M. Gambetta, S. Poletto, B. L. T. Plourde, J. M. Chow, A. D. Córcoles, J. A. Smolin, S. T. Merkel, J. R. Rozen, G. A. Keefe, M. B. Rothwell, M. B. Ketchen, and M. Steffen. Superconducting qubit in a waveguide cavity with a coherence time approaching 0.1 ms. *Phys. Rev. B*, 86(10):100506 (2012). (Cited on page 134.)

D. Ristè, S. Poletto, M. Z. Huang, A. Bruno, V. Vesterinen, O. P. Saira, and L. DiCarlo. Detecting bit-flip errors in a logical qubit using stabilizer measurements. *Nature Communications*, 6:6983 (2015). doi:10.1038/ncomms7983. (Cited on page 32.)

G. Riviello, K. M. Tibbetts, C. Brif, R. Long, R.-B. Wu, T.-S. Ho, and H. Rabitz. Searching for quantum optimal controls under severe constraints. *Phys. Rev. A*, 91(4):043401 (2015). doi:10.1103/PhysRevA.91.043401. (Cited on page 63.)

S. Rosenblum, Y. Y. Gao, P. Reinhold, C. Wang, C. J. Axline, L. Frunzio, S. M. Girvin, L. Jiang, M. Mirrahimi, M. H. Devoret, and R. J. Schoelkopf. A CNOT gate between multiphoton qubits encoded in two cavities. *Nature Communications*, 9(1):652 (2018a). doi:10.1038/s41467-018-03059-5. (Cited on pages 48 and 85.)

S. Rosenblum, P. Reinhold, M. Mirrahimi, L. Jiang, L. Frunzio, and R. J. Schoelkopf. Fault-tolerant detection of a quantum error. *Science*, 361(6399):266–270 (2018b). doi:10.1126/science.aat3996. (Cited on pages 7 and 126.)

D. Sank, Z. Chen, M. Khezri, J. Kelly, R. Barends, B. Campbell, Y. Chen, B. Chiaro, A. Dunsworth, A. Fowler, E. Jeffrey, E. Lucero, A. Megrant, J. Mutus, M. Neeley, C. Neill, P. J. J. O'Malley, C. Quintana, P. Roushan, A. Vainsencher, T. White, J. Wenner, A. N. Korotkov, and J. M. Martinis. Measurement-Induced State Transitions in a Superconducting Qubit: Beyond the Rotating Wave Approximation. *Phys. Rev. Lett.*, 117(19):190503 (2016). doi:10.1103/PhysRevLett.117.190503. (Cited on page 134.)

I. Scholz, J. D. van Beek, and M. Ernst. Operator-based Floquet theory in solid-state NMR. *Solid State Nuclear Magnetic Resonance*, 37(3-4):39–59 (2010). doi:10.1016/j.ssnmr.2010.04.003. (Cited on page 194.)

D. Schuster. *Circuit Quantum Electrodynamics*. Ph.D. thesis, Yale University (2007). (Cited on pages 8 and 78.)

M. O. Scully and M. S. Zubairy. *Quantum Optics*. Cambridge University Press (1997). ISBN 0521434580. (Cited on page 7.)

R. Shankar. *Principles of Quantum Mechanics, 2nd Edition*. Plenum Press (2011). ISBN 9780306447907. (Cited on page 7.)

C. Shen, R. W. Heeres, P. Reinhold, L. Jiang, Y.-K. Liu, R. J. Schoelkopf, and L. Jiang. Optimized tomography of continuous variable systems using excitation counting. *Phys. Rev. A*, 94(5):052327 (2016). doi:10.1103/PhysRevA.94.052327. (Cited on page 202.)

P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *SFCS-94*, pages 124–134. IEEE Comput. Soc. Press (1994). ISBN 0-8186-6580-7. doi:10.1109/SFCS.1994.365700. (Cited on page 2.)

P. W. Shor. Scheme for reducing decoherence in quantum computer memory. *Phys. Rev. A*, 52(4):R2493–R2496 (1995). doi:10.1103/PhysRevA.52.R2493. (Cited on pages 3, 29, 32, and 125.)

V. V. Sivak, N. E. Frattini, V. R. Joshi, A. Lingenfelter, S. Shankar, and M. H. Devoret. Kerr-free three-wave mixing in superconducting quantum circuits (2019). (Cited on page 170.)

K. Sliwa. *Improving the Quality of Heisenberg Back-Action of Qubit Measurements made with Parametric Amplifiers*. Ph.D. thesis, Yale University (2016). (Cited on page 94.)

J. A. Smolin, J. M. Gambetta, and G. Smith. Efficient Method for Computing the Maximum-Likelihood Quantum State from Measurements with Additive Gaussian Noise. *Phys. Rev. Lett.*, 108(7):070502 (2012). doi:10.1103/PhysRevLett.108.070502. (Cited on pages 202 and 206.)

A. Soare, H. Ball, D. Hayes, J. Sastrawan, M. C. Jarratt, J. J. McLoughlin, X. Zhen, T. J. Green, and M. J. Biercuk. Experimental noise filtering by quantum control. *Nature Physics*, 10(11):825–829 (2014). doi:10.1038/NPHYS3115. (Cited on page 95.)

R. Spekkens. Negativity and Contextuality are Equivalent Notions of Nonclassicality. *Phys. Rev. Lett.*, 101(2):020401 (2008). doi:10.1103/PhysRevLett.101.020401. (Cited on page 20.)

A. Steane. The ion trap quantum information processor. *Appl. Phys. B*, 64(6):623–643 (1997). doi:10.1007/s003400050225. (Cited on page 4.)

A. M. Steane. Error Correcting Codes in Quantum Theory. *Phys. Rev. Lett.*, 77(5):793–797 (1996). doi:10.1103/PhysRevLett.77.793. (Cited on pages 32 and 33.)

A. M. Steane. Space, Time, Parallelism and Noise Requirements for Reliable Quantum Computing. *Fortschritte der Physik*, 46(4-5):443–457 (1998). doi:10.1002/(SICI)1521-3978(199806)46:4/5¡443::AID-PROP443¿3.0.CO;2-8. (Cited on page 3.)

D. Steck. Quantum and atom optics. `http://atomoptics-nas.uoregon.edu/~dsteck/teaching/quantum-optics/quantum-optics-notes.pdf` (2007). (Cited on pages 7 and 9.)

B. M. Terhal and D. Weigand. Encoding a qubit into a cavity mode in circuit QED using phase estimation. *Phys. Rev. A*, 93(1):012315 (2016). doi:10.1103/PhysRevA.93.012315. (Cited on pages 51 and 168.)

W. G. Unruh. Maintaining coherence in quantum computers. *Phys. Rev. A*, 51(2):992–997 (1995). doi:10.1103/PhysRevA.51.992. (Cited on page 3.)

R. Vijay, M. H. Devoret, and I. Siddiqi. Invited Review Article: The Josephson bifurcation amplifier. *Review of Scientific Instruments*, 80(11):111101 (2009). doi:10.1063/1.3224703. (Cited on page 119.)

L. Viola, E. Knill, and S. Lloyd. Dynamical Decoupling of Open Quantum Systems. *Phys. Rev. Lett.*, 82(12):2417–2421 (1999). (Cited on page 61.)

B. Vlastakis. *Controlling coherent state superpositions with superconducting circuits*. Ph.D. thesis, Yale University (2015). (Cited on pages 7, 16, and 104.)

B. Vlastakis, G. Kirchmair, Z. Leghtas, S. E. Nigg, L. Frunzio, S. M. Girvin, M. Mirrahimi, M. H. Devoret, and R. J. Schoelkopf. Deterministically Encoding Quantum Information Using 100-Photon Schrödinger Cat States. *Science*, 342(6158):607–610 (2013). doi:10.1126/science.1243289. (Cited on pages 22 and 38.)

K. Vogel, V. M. Akulin, and W. P. Schleich. Quantum state engineering of the radiation field. *Phys. Rev. Lett.*, 71(12):1816–1819 (1993). doi:10.1103/PhysRevLett.71.1816. (Cited on page 14.)

D. F. Walls and G. J. Milburn. *Quantum optics* (1995). ISBN 9783540588313. doi:10.1007/978-3-642-79504-6. (Cited on pages 7 and 94.)

M. M. Wolf. *Quantum channels & operations: Guided tour* (2012). (Cited on page 204.)

S. Yasin, D. G. Hasko, and H. Ahmed. Comparison of MIBK/IPA and water/IPA as PMMA developers for electron beam nanolithography. *Microelectronic Engineering*, 61-62:745–753 (2002). doi:10.1016/S0167-9317(02)00468-9. (Cited on page 86.)

Y. Zhang, B. J. Lester, Y. Y. Gao, L. Jiang, R. J. Schoelkopf, and S. M. Girvin. Engineering bilinear mode coupling in circuit QED: Theory and experiment. *Phys. Rev. A*, 99(1):012314 (2019). doi:10.1103/PhysRevA.99.012314. (Cited on page 134.)

Appendix A

Identities and derivations

A.1 Changing frames

A frame change is a way of re-organizing a quantum system by transforming the states and operators all in such a way that the observable quantities (expectation values) remain unchanged for all times. The purpose of doing so is to write down a form where the desired effects are obvious, or at least familiar. A frame change is specified first by identifying how states will transform, using some unitary operator $\boldsymbol{U}$.

$$|\widetilde{\psi}\rangle \equiv \boldsymbol{U}\,|\psi\rangle \tag{A.1}$$

Density matrices and observable operators transform as well, so as to preserve expectation values $\langle\psi|\,\boldsymbol{M}\,|\psi\rangle$ or $\mathrm{Tr}\{\boldsymbol{\rho}\boldsymbol{M}\}$

$$\widetilde{\boldsymbol{\rho}} \equiv \boldsymbol{U}\boldsymbol{\rho}\boldsymbol{U}^\dagger \tag{A.2}$$

$$\widetilde{\boldsymbol{M}} \equiv \boldsymbol{U}\boldsymbol{M}\boldsymbol{U}^\dagger \tag{A.3}$$

The Hamiltonian changes in the same way if $\boldsymbol{U}$ is time-independent. If the transform is time dependent, then it is slightly different. We can derive this by looking at the time evolution of

$|\widetilde{\psi}\rangle$

$$\partial_t |\widetilde{\psi}\rangle = \boldsymbol{U} \left(\partial_t |\psi\rangle\right) + \left(\partial_t \boldsymbol{U}\right) |\psi\rangle \tag{A.4}$$

$$= i\boldsymbol{U}\boldsymbol{H} |\psi\rangle + \left(\partial_t \boldsymbol{U}\right) |\psi\rangle \tag{A.5}$$

$$= i \left(\boldsymbol{U}\boldsymbol{H}\boldsymbol{U}^\dagger - i \left(\partial_t \boldsymbol{U}\right) \boldsymbol{U}^\dagger\right) |\widetilde{\psi}\rangle \tag{A.6}$$

$$\equiv i\widetilde{\boldsymbol{H}} |\widetilde{\psi}\rangle \tag{A.7}$$

Therefore the final form of the transformed Hamiltonian is

$$\widetilde{\boldsymbol{H}} = \boldsymbol{U}\boldsymbol{H}\boldsymbol{U}^\dagger - i\dot{\boldsymbol{U}}\boldsymbol{U}^\dagger \tag{A.8}$$

We will often have Hamiltonians $\boldsymbol{H}$ which are built out of products of a smaller "alphabet" of operators, most prominently the ladder operators $\boldsymbol{a}$ and $\boldsymbol{a}^\dagger$. In order to calculate the transformed form $\boldsymbol{U}\boldsymbol{H}\boldsymbol{U}^\dagger$, it is sufficient to calculate the effect on each member of the "alphabet." For instance, if $\boldsymbol{H} = \boldsymbol{X}_1\boldsymbol{X}_{\boldsymbol{2}}\boldsymbol{X}_{\boldsymbol{3}}$ then

$$\boldsymbol{U}\boldsymbol{H}\boldsymbol{U}^\dagger = \left(\boldsymbol{U}\boldsymbol{X}_1\boldsymbol{U}^\dagger\right)\left(\boldsymbol{U}\boldsymbol{X}_2\boldsymbol{U}^\dagger\right)\left(\boldsymbol{U}\boldsymbol{X}_3\boldsymbol{U}^\dagger\right) \tag{A.9}$$

More generally, for $\boldsymbol{H} = f(\boldsymbol{a}, \boldsymbol{a}^\dagger)$

$$\boldsymbol{U}\boldsymbol{H}\boldsymbol{U}^\dagger = f(\boldsymbol{U}\boldsymbol{a}\boldsymbol{U}^\dagger, \boldsymbol{U}\boldsymbol{a}^\dagger\boldsymbol{U}^\dagger) \tag{A.10}$$

A.2 Commutator relations

For ladder operators the most important equation is the base commutation relation

$$\left[\boldsymbol{a}, \boldsymbol{a}^\dagger\right] = 1 \tag{A.11}$$

It is helpful to be able to compute higher order commutators as well. In particular we can compute:

$$\Big[\boldsymbol{a}, (\boldsymbol{a}^\dagger)^n\Big] = \boldsymbol{a}(\boldsymbol{a}^\dagger)^n - (\boldsymbol{a}^\dagger)^n\boldsymbol{a} \tag{A.12}$$

$$= (\boldsymbol{a}^\dagger\boldsymbol{a} + 1)(\boldsymbol{a}^\dagger)^{n-1} - (\boldsymbol{a}^\dagger)^n\boldsymbol{a} \tag{A.13}$$

$$= \boldsymbol{a}^\dagger\left(\boldsymbol{a}(\boldsymbol{a}^\dagger)^{n-1} - (\boldsymbol{a}^\dagger)^{n-1}\boldsymbol{a}\right) + (\boldsymbol{a}^\dagger)^{n-1} \tag{A.14}$$

$$= \boldsymbol{a}^\dagger\Big[\boldsymbol{a}, (\boldsymbol{a}^\dagger)^{n-1}\Big] + (\boldsymbol{a}^\dagger)^{n-1} \tag{A.15}$$

Here we have reduced the computation of $\Big[\boldsymbol{a}, (\boldsymbol{a}^\dagger)^n\Big]$ to that of $\Big[\boldsymbol{a}, (\boldsymbol{a}^\dagger)^{n-1}\Big]$, a simple recurrence relation. This can be solved as

$$\Big[\boldsymbol{a}, (\boldsymbol{a}^\dagger)^n\Big] = \boldsymbol{a}^\dagger\left((n-1)(\boldsymbol{a}^\dagger)^{(n-2)}\right) + (\boldsymbol{a}^\dagger)^{n-1} \tag{A.16}$$

$$= n(\boldsymbol{a}^\dagger)^{n-1} \tag{A.17}$$

$$\tag{A.18}$$

We note that this is exactly the form of the "derivative" of $(\boldsymbol{a}^\dagger)^n$ with respect to $\boldsymbol{a}^\dagger$. Because of the linearity of the commutator, this extends to any differentiable function f:

$$\Big[\boldsymbol{a}, f(\boldsymbol{a}^\dagger)\Big] = f'(\boldsymbol{a}^\dagger) \tag{A.19}$$

Similarly, we can compute the action of taking the commutator with $\boldsymbol{a}^\dagger$ and find:

$$\Big[f(\boldsymbol{a}), \boldsymbol{a}^\dagger\Big] = f'(\boldsymbol{a}) \tag{A.20}$$

A.3 Rotating frame

One common frame change is the so-called *rotating frame*, which tries to account for a detuning term $\omega\boldsymbol{a}^\dagger\boldsymbol{a}$ in the Hamiltonian using the following transformation:

$$\boldsymbol{U} = e^{-i\omega\boldsymbol{a}^\dagger\boldsymbol{a}t}. \tag{A.21}$$

If we start with a Hamiltonian

$$\boldsymbol{H} = \omega \boldsymbol{a}^\dagger \boldsymbol{a} + f(\boldsymbol{a}, \boldsymbol{a}^\dagger), \tag{A.22}$$

we can compute the new Hamiltonian (equation A.8),

$$\widetilde{\boldsymbol{H}} = \boldsymbol{U}\boldsymbol{H}\boldsymbol{U}^\dagger - i\dot{\boldsymbol{U}}\boldsymbol{U}^\dagger \tag{A.23}$$

$$= \boldsymbol{U}\boldsymbol{H}\boldsymbol{U}^\dagger - \omega \boldsymbol{a}^\dagger \boldsymbol{a} \tag{A.24}$$

In order to understand the $\boldsymbol{U}\boldsymbol{H}\boldsymbol{U}^\dagger$ component, let us first see how the ladder operators transform (equation A.10). Using the Baker-Campbell-Hausdorff (BCH) expansion we can see

$$\boldsymbol{U}\boldsymbol{a}\boldsymbol{U}^\dagger = \boldsymbol{a} + i\omega t\Big[\boldsymbol{a}^\dagger\boldsymbol{a}, \boldsymbol{a}\Big] - \frac{(\omega t)^2}{2}\Big[\boldsymbol{a}^\dagger\boldsymbol{a}, \Big[\boldsymbol{a}^\dagger\boldsymbol{a}, \boldsymbol{a}\Big]\Big] + \cdots \tag{A.25}$$

$$= \sum_{k=0} \frac{(i\omega t)^n}{n!}\Big[\boldsymbol{a}^\dagger\boldsymbol{a}, \cdot\Big]^n(\boldsymbol{a}) \tag{A.26}$$

We not that, because $\boldsymbol{a}$ seems to be an "eigenoperator" of the commutator with $\boldsymbol{a}^\dagger\boldsymbol{a}$, the terms in this expansion simplify:

$$\Big[\boldsymbol{a}^\dagger\boldsymbol{a}, \boldsymbol{a}\Big] = \boldsymbol{a} \Longrightarrow \Big[\boldsymbol{a}^\dagger\boldsymbol{a}, \cdot\Big]^n(\boldsymbol{a}) = \boldsymbol{a} \tag{A.27}$$

Therefore we can write,

$$\boldsymbol{U}\boldsymbol{a}\boldsymbol{U}^\dagger = \sum_{k=0} \frac{(i\omega t)^n}{n!}\boldsymbol{a} \tag{A.28}$$

$$= e^{i\omega t}\boldsymbol{a} \tag{A.29}$$

and more generally

$$\boldsymbol{U} f(\boldsymbol{a}, \boldsymbol{a}^\dagger)\boldsymbol{U}^\dagger = f(\boldsymbol{a}e^{i\omega t}, \boldsymbol{a}^\dagger e^{-i\omega t}). \tag{A.30}$$

This result gives us the final form for the Hamiltonian in the new frame (from A.24)

$$\widetilde{\boldsymbol{H}} = f(\boldsymbol{a}e^{i\omega t}, \boldsymbol{a}^\dagger e^{-i\omega t}). \tag{A.31}$$

This is to say, we have eliminated the $\omega \boldsymbol{a}^\dagger \boldsymbol{a}$ term, at the cost of adding time dependence to any $\boldsymbol{a}$ and $\boldsymbol{a}^\dagger$ operators that remain. Diagonal terms (with equal numbers of $\boldsymbol{a}$ and $\boldsymbol{a}^\dagger$) will remain unchanged, as the time dependence terms cancel.

A.4 Displaced frame

The next most common transformation is to invoke a displacement operator, with the intent of simplifying the treatment of a constantly applied drive term.

$$\boldsymbol{U} = \boldsymbol{D}_\alpha = e^{\alpha \boldsymbol{a}^\dagger - \alpha^* \boldsymbol{a}} \tag{A.32}$$

We again invoke BCH to see how the ladder operators transform

$$\boldsymbol{U}\boldsymbol{a}\boldsymbol{U}^\dagger = \sum_{k=0} \frac{1}{n!} \Big[\alpha \boldsymbol{a}^\dagger - \alpha^* \boldsymbol{a}, \cdot\Big]^n (\boldsymbol{a}) \tag{A.33}$$

Once again, a trick will allow us to simplify this. we note that the commutator of the first term produces a scalar value. Therefore all subsequent terms will be zero, since the commutator with a scalar is zero.

$$\Big[\alpha \boldsymbol{a}^\dagger - \alpha^* \boldsymbol{a}, \boldsymbol{a}\Big] = \alpha \Longrightarrow \Big[\alpha \boldsymbol{a}^\dagger - \alpha^* \boldsymbol{a}, \cdot\Big]^{n>1} (\boldsymbol{a}) = 0 \tag{A.34}$$

This means the BCH expansion stops after the first term, producing

$$\boldsymbol{U}\boldsymbol{a}\boldsymbol{U}^\dagger = \boldsymbol{a} + \alpha. \tag{A.35}$$

This means that general operators transform as

$$\boldsymbol{U} f(\boldsymbol{a}, \boldsymbol{a}^\dagger) \boldsymbol{U}^\dagger = f(\boldsymbol{a} + \alpha, \boldsymbol{a}^\dagger + \alpha^*) \tag{A.36}$$

A.5 Dispersive Hamiltonian for a multi-level ancilla

In order to derive the dispersive (diagonal) Hamiltonian for a resonator coupled to an ancilla, we will carry out some perturbation theory calculations. Our Hamiltonian is split into a diagonal part $\boldsymbol{H}_0$ and an off-diagonal perturbation $\boldsymbol{H}_{\text{int}}$:

$$\boldsymbol{H} = \boldsymbol{H}_0 + \boldsymbol{H}_{\text{int}} \tag{A.37}$$

We say the resonator is linear, and the ancilla spectrum is arbitrary:

$$\boldsymbol{H}_0 = \omega_a \boldsymbol{a}^\dagger \boldsymbol{a} + \sum_j E_j \left|j\right\rangle\!\left\langle j\right| \tag{A.38}$$

The structure of the interaction is to couple some ancilla operator $\boldsymbol{B}$ to one quadrature of the resonator $(\boldsymbol{a} + \boldsymbol{a}^\dagger)$:

$$\boldsymbol{H}_{\text{int}} = \boldsymbol{B}\left(\boldsymbol{a} + \boldsymbol{a}^\dagger\right). \tag{A.39}$$

Before perturbation, the energy associated with state with n photons and the j-th ancilla state $(\left|n, j\right\rangle)$ is:

$$E^{(0)}_{n,j} = n\omega_a + E_j. \tag{A.40}$$

The first order perturbation theory contribution is zero because the interaction is purely off-diagonal:

$$E^{(1)}_{n,j} = \left\langle n, j\right| \boldsymbol{H}_{\text{int}} \left|n, j\right\rangle = 0. \tag{A.41}$$

So the first contribution comes from the second order term:

$$E^{(2)}_{n,j} = \sum_{(m,k)\neq(n,j)} \frac{\left|\left\langle n, j\right| \boldsymbol{H}_{\text{int}} \left|m, k\right\rangle\right|^2}{E^{(0)}_{n,j} - E^{(0)}_{m,k}}. \tag{A.42}$$

We can separately calculate the matrix element in the numerator

$$\begin{aligned} \langle n, j| \boldsymbol{H}_{\mathsf{int}} |m, k\rangle &= \langle n, j| \boldsymbol{B} \left(\boldsymbol{a} + \boldsymbol{a}^\dagger\right) |m, k\rangle && \text{(A.43)} \\ &= \boldsymbol{B}_{jk} \left(\sqrt{m}\delta_{n,m-1} + \sqrt{m+1}\delta_{n,m+1}\right), && \text{(A.44)} \end{aligned}$$

as well as the energy difference in the denominator (defining $E_{jk} \equiv E_j - E_k$),

$$\begin{aligned} E^{(0)}_{n,j} - E^{(0)}_{m,k} &= (n\omega_a + E_j) - (m\omega_a + E_k) && \text{(A.45)} \\ &\equiv (n-m)\omega_a + E_{jk}, && \text{(A.46)} \end{aligned}$$

to produce the final result

$$\begin{aligned} E^{(2)}_{n,j} &= \sum_{k\neq j} \frac{|\boldsymbol{B}_{jk}|^2(n+1)}{E^{(0)}_{n,j} - E^{(0)}_{n+1,k}} + \frac{|\boldsymbol{B}_{jk}|^2 n}{E^{(0)}_{n,j} - E^{(0)}_{n+1,k}} && \text{(A.47)} \\ &= \sum_{k\neq j} \frac{|\boldsymbol{B}_{jk}|^2(n+1)}{E_{jk} + \omega_a} + \frac{|\boldsymbol{B}_{jk}|^2 n}{E_{jk} - \omega_a} && \text{(A.48)} \\ &= \sum_{k\neq j} |\boldsymbol{B}_{jk}|^2 \frac{(2n+1)E_{jk} - \omega_a}{E^2_{jk} - \omega^2_a} && \text{(A.49)} \end{aligned}$$

The *dispersive shift* to the cavity frequency χ_j is the energy cost of adding a photon when the ancilla is in the j-th state:

$$\begin{aligned} \chi_j &= (E_{n+1,j} - E_{n,j}) - \omega_a && \text{(A.50)} \\ &= \sum_{k\neq j} \frac{2E_{jk}|\boldsymbol{B}_{jk}|^2}{E^2_{jk} - \omega^2_a}. && \text{(A.51)} \end{aligned}$$

This is a well defined linear frequency shift since the result does not depend on the number of photons n. If our ancilla is a two level system (with levels $|g\rangle$ and $|e\rangle$) and the coupling operator is $\boldsymbol{B} = \boldsymbol{\sigma}_x$, then we can calculate the change in cavity frequency associated with each state:

$$\chi_g = \frac{-2\omega_{ge} g^2}{\omega_{ge}^2 - \omega_a^2} \tag{A.52}$$

$$\chi_e = \frac{2\omega_{ge} g^2}{\omega_{ge}^2 - \omega_a^2} \tag{A.53}$$

Typically we will absorb χ_g into the final value of ω_a and therefore we can focus on the change in cavity frequency between $|g\rangle$ and $|e\rangle$:

$$\begin{aligned}
\chi_{eg} &\equiv \chi_e - \chi_g & \text{(A.54)}\\
&= \frac{4\omega_{ge} g^2}{\omega_{ge}^2 - \omega_a^2} & \text{(A.55)}\\
&= \frac{4\omega_{ge} g^2}{\Delta(2\omega_{ge} - \Delta)} & \text{(A.56)}\\
&\approx \frac{2g^2}{\Delta} & \text{(A.57)}
\end{aligned}$$

Here the final approximation assumes that the detuning is small compared with the frequency of either mode $\Delta \ll \omega_{ge}, \omega_a$. This is the exact form which is obtained if the Jaynes-Cummings form is assumed from the beginning ($\boldsymbol{H}_{\text{int}} = g(\boldsymbol{a}^\dagger \boldsymbol{\sigma}_- + \boldsymbol{a}\boldsymbol{\sigma}_+)$). In the case of a three-level ancilla (third state $|f\rangle$), we represent the ancilla anharmonicity with $E_f = 2\omega_{ge} + \alpha$, and assume a coupling operator

$$\begin{aligned}
\boldsymbol{B} &= (\boldsymbol{b} + \boldsymbol{b}^\dagger) & \text{(A.58)}\\
&= \Big(|g\rangle\!\langle e| + \sqrt{2}\,|e\rangle\!\langle f| + \text{h.c.}\Big), & \text{(A.59)}
\end{aligned}$$

where the final line assumes that we limit our consideration to the first three levels. In this case, χ_g remains unchanged from the two-level case ($|g\rangle$ does not directly couple to $|f\rangle$) but we must modify χ_e as follows:

$$\chi_e = \frac{2\omega_{ge} g^2}{\omega_{ge}^2 - \omega_a^2} - \frac{4\omega_{ef} g^2}{\omega_{ef}^2 - \omega_a^2}. \tag{A.60}$$

Note the factor of 4 in the numerator of the second term, which arises from the "bosonic enhancement" of the coupling matrix element. The calculation of the final dispersive shift then is

$$\chi_{eg} = \frac{4\omega_{ge}g^2}{\Delta(2\omega_{ge} - \Delta)} - \frac{4(\omega_{ge} + \alpha)g^2}{(\Delta + \alpha)(2\omega_{ge} + \alpha - \Delta)} \tag{A.61}$$

$$\approx 2\frac{g^2}{\Delta}\left(\frac{1}{\Delta} - \frac{1}{\Delta + \alpha}\right) \tag{A.62}$$

$$= \frac{2g^2\alpha}{\alpha(\Delta + \alpha)} \tag{A.63}$$

Again we have invoked the approximation that we have a hierarchy of frequency scales $\Delta, \alpha \ll \omega_{ge}, \omega_a$.

A.6 Damped harmonic oscillator master equation

There are many approaches one can take to deriving the master equation. I'd like to take an intuitive approach, which starts from the belief that photons will be lost from the system at rate κ. From this baseline, we can find a continuous equation which models this loss, while respecting the constraints needed to keep the solutions physical.

We will work in the operator-sum picture, where the evolution of a quantum state $\boldsymbol{\rho}$ is given by

$$\boldsymbol{\rho} \mapsto \sum_k \boldsymbol{M}_k \boldsymbol{\rho} \boldsymbol{M}_k^\dagger, \tag{A.64}$$

for some set of linear operators $\{\boldsymbol{M}_k\}$The statement that probability is conserved, and that the density matrix retains its trace-one property, is equivalent to the following restriction

$$\sum_k \boldsymbol{M}_k^\dagger \boldsymbol{M}_k = \boldsymbol{I}. \tag{A.65}$$

For pure unitary evolution, we have the sole operator $\boldsymbol{M}_0 = \boldsymbol{U}$, where $\boldsymbol{U}$ is a solution to the Schrödinger equation. For sufficiently short time, we can model $\boldsymbol{M}_0 \approx \boldsymbol{I} + i\boldsymbol{H}\delta t$, which satisfies equation A.65 to first order in δt.

We wish to add a term capturing photon loss. Under a sufficiently short amount of time

$\delta t \ll 1/\kappa$, we can neglect the possibility of two or more photons leaking out. The natural way to model single photon loss is via an operator $\boldsymbol{M}_1 \propto \boldsymbol{a}$. If we wish to have the term $\kappa \delta t \boldsymbol{a} \boldsymbol{\rho} \boldsymbol{a}^\dagger$ in our version of equation A.64, then we need $\boldsymbol{M_1} = \sqrt{\kappa \delta t} \boldsymbol{a}$. However, at this stage, with $\boldsymbol{M}_0$ as before and $\boldsymbol{M}_1$, we no longer satisfy the probability conservation constraint A.65, as $\boldsymbol{M}_0 \boldsymbol{M}_0^\dagger + \boldsymbol{M}_1 \boldsymbol{M}_1^\dagger = \boldsymbol{I} + \kappa \delta t \boldsymbol{a}^\dagger \boldsymbol{a} + O((\kappa \delta t)^2)$.

This first model failed because we did not account for the back-action inherent in *learning* that no photon was lost. An extreme example is elucidating. Consider that we start with an equal superposition of $|0\rangle$ and $|100\rangle$ in the cavity, and that there is no Hamiltonian evolution ($\boldsymbol{H} = 0$). We wait some amount of time, and observe that no photon leaks out. Are we still in the state $|0\rangle + |100\rangle$? Well, to answer that, we can ask: do we give equal credence to the possibility that there is zero photons in the cavity as the possibility that there is 100? To this latter question, the answer is of course not. If there were 100 photons, we would be exceedingly likely to see one leak out. Therefore the observation that none leaked out increases the likelihood of $|0\rangle$. This is a simple application of Bayes rule. The act of observing photons leaking or not leaking is a form of partial measurement, which has back-action regardless of the outcome.

In order to account for this, we must modify our no-jump operator in such a way to restore conservation of probability.

$$\boldsymbol{M}_0 = \boldsymbol{I} + \delta t \left(i\boldsymbol{H} - \frac{\kappa}{2} \boldsymbol{a}^\dagger \boldsymbol{a} \right) \tag{A.66}$$

It is easy to check that this now satisfies equation A.65 to first order in δt. We can convert this into a differential equation

$$\begin{aligned}
\boldsymbol{\rho}(t+\delta t) =& \boldsymbol{M}_0 \boldsymbol{\rho}(t) \boldsymbol{M}_0 + \boldsymbol{M}_1 \boldsymbol{\rho}(t) \boldsymbol{M}_1 && \text{(A.67)}\\
=& \boldsymbol{\rho}(t) + \delta t \left(i\boldsymbol{H} - \frac{\kappa}{2} \boldsymbol{a}^\dagger \boldsymbol{a} \right) \boldsymbol{\rho}(t) + \delta t \boldsymbol{\rho}(t) \left(-i\boldsymbol{H} - \frac{\kappa}{2} \boldsymbol{a}^\dagger \boldsymbol{a} \right) \\
& + \kappa \delta t \boldsymbol{a} \boldsymbol{\rho}(t) \boldsymbol{a}^\dagger + O((\delta t)^2) && \text{(A.68)}\\
\partial_t \boldsymbol{\rho} =& i[\boldsymbol{H}, \boldsymbol{\rho}] + \kappa \boldsymbol{a} \boldsymbol{\rho} \boldsymbol{a}^\dagger - \frac{1}{2} \left(\boldsymbol{a}^\dagger \boldsymbol{a} \boldsymbol{\rho} + \boldsymbol{\rho} \boldsymbol{a}^\dagger \boldsymbol{a} \right) && \text{(A.69)}
\end{aligned}$$

A.7 Fréchet Derivative of the matrix exponential

In order to compute the maximally accurate derivative of the matrix exponential $\boldsymbol{U} = e^{\boldsymbol{A}}$ with respect to some parameter x upon which $\boldsymbol{A}$ depends ($\partial_x \boldsymbol{A} \neq 0$), which is necessary for the correct implementation of GRAPE (chapter 4) we can take the following derivation (Najfeld and Havel, 1995). Eventually, we will take $\boldsymbol{A} = \frac{i\delta t}{\hbar}\boldsymbol{H}$. We begin by employing the following definition of an exponential function:

$$e^{\boldsymbol{A}} = \lim_{N\to\infty}\left(1 + \frac{\boldsymbol{A}}{N}\right)^{N} \tag{A.70}$$

This form allows us to use the product rule for differentiation to come up with an integral form for the derivative.

$$\begin{aligned}
\partial_x \boldsymbol{U} &= \partial_x e^{\boldsymbol{A}} && \text{(A.71)}\\
&= \lim_{N\to\infty}\sum_{k=0}^{N}\left(1+\frac{\boldsymbol{A}}{N}\right)^{k}\frac{i\delta t}{\hbar N}\left(\partial_x \boldsymbol{A}\right)\left(1+\frac{\boldsymbol{A}}{N}\right)^{N-k-1} && \text{(A.72)}\\
&= \int_0^1 \mathrm{d}s\, e^{s\boldsymbol{A}}\left(\partial_x \boldsymbol{A}\right) e^{(1-s)\boldsymbol{A}} && \text{(A.73)}\\
&= \left(\int_0^1 \mathrm{d}s\, e^{s\boldsymbol{A}}\left(\partial_x \boldsymbol{A}\right) e^{-s\boldsymbol{A}}\right)\boldsymbol{U} && \text{(A.74)}\\
& && \text{(A.75)}
\end{aligned}$$

This form can be simplified if we assume that $\boldsymbol{A}$ is normal, and thus diagonalized by some unitary transformation $\boldsymbol{V}$.

$$\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{\dagger} \to e^{s\boldsymbol{A}} = \boldsymbol{V}e^{s\boldsymbol{\Lambda}}\boldsymbol{V}^{\dagger} \tag{A.76}$$

where $\boldsymbol{\Lambda}_{ij} = \lambda_i \delta_{ij}$. We can put this in our above expression, and identify several sub-expressions which will help us analyze this.

$$\partial_x \boldsymbol{U} = \left(\int_0^1 \mathrm{d}s\, \boldsymbol{V} e^{\Lambda s} \boldsymbol{V}^\dagger \left(\partial_x \boldsymbol{A} \right) \boldsymbol{V} e^{-\Lambda s} \boldsymbol{V}^\dagger \right) \boldsymbol{U} \tag{A.77}$$

$$= \boldsymbol{V} \underbrace{\left(\int_0^1 \mathrm{d}s\, e^{\Lambda s} \overbrace{\boldsymbol{V}^\dagger \left(\partial_x \boldsymbol{A} \right) \boldsymbol{V}}^{\boldsymbol{X}} e^{-\Lambda s} \right)}_{\boldsymbol{Z}} \boldsymbol{V}^\dagger \boldsymbol{U} \tag{A.78}$$

We can see that $\boldsymbol{Z}$ factorizes into an entrywise product between $\boldsymbol{X}$ and a simple integral expression dependent only on the eigenvalues of $\boldsymbol{A}$:

$$\boldsymbol{Z}_{ij} = \underbrace{\int_0^1 \mathrm{d}s\, e^{s(\lambda_i - \lambda_j)}}_{\boldsymbol{\Gamma}_{ij}} \boldsymbol{X}_{ij} \tag{A.79}$$

$$\boldsymbol{\Gamma}_{ij} = \begin{cases} \frac{e^{(\lambda_i - \lambda_j)} - 1}{(\lambda_i - \lambda_j)} & \lambda_i \neq \lambda_j \\ 1 & \lambda_i = \lambda_j \end{cases} \tag{A.80}$$

$$\boldsymbol{Z} = \boldsymbol{\Gamma} \circ \boldsymbol{X} \tag{A.81}$$

The final notation $\circ$ represents the Hadamard (entrywise) product $(\boldsymbol{A} \circ \boldsymbol{B})_{ij} = \boldsymbol{A}_{ij} \boldsymbol{B}_{ij}$. We can put this all together for our final expression.

$$\partial_x \boldsymbol{U} = \boldsymbol{V} \left(\Gamma \circ \left(\boldsymbol{V}^\dagger \left(\partial_x \boldsymbol{A} \right) \boldsymbol{V} \right) \right) \boldsymbol{V}^\dagger \boldsymbol{U} \tag{A.82}$$

A.8 Approximate time-independent Hamiltonians

There are many equivalent ways of deriving approximate time-independent Hamiltonians. There is the Floquet-Magnus expansion (Mananga and Charpentier, 2011), which modifies the derivation of the Magnus series. There are approaches which construct a full Floquet Hamiltonian and proceed via block diagonalization procedures (Leskes et al., 2010; Scholz et al., 2010)[1]. Finally, this procedure is also described as a higher-order rotating wave approximation (Mirrahimi and

[1] These diagnolization procedures are called the van Vleck transformation in the NMR literature, but is closely related to the Schrieffer-Wolff transformation in practice

Rouchon, 2015). In my opinion the clearest treatment can be found in Leskes et al. (2010). The high-level procedure involves three steps. One takes the time-dependent Hamiltonian, and translates it into a time-independent Hamiltonian via Floquet theory. This procedure involves expanding the Hilbert space to include a time-sector. The time-dependence manifests as off-diagonal elements in the time basis. One then wishes to perform block-diagonalization of this Floquet Hamiltonian. Typically this will be only an approximate form. One then projects this form back to the non-Floquet space, yielding the effective time-independent form.

A.8.1 Floquet formalism

In the "two-time" version of the Floquet formalism, we introduce a formal time variable t' which we will associate with the periodic behavior. We can recover the original behavior by setting $t' = t$. If the period is $2\pi/\omega$ we can write the Hamiltonian in a Fourier series:

$$\boldsymbol{H}(t') = \sum_n \boldsymbol{H}_n e^{in\omega t'}. \tag{A.83}$$

We write our states as having two time variables

$$|\psi(t,t')\rangle = \sum_n |\psi_n(t)\rangle\, e^{in\omega t'}. \tag{A.84}$$

Finally the Schrödinger equation becomes

$$i(\partial_t + \partial_{t'})\,|\psi(t,t')\rangle = \boldsymbol{H}(t')\,|\psi(t,t')\rangle\,. \tag{A.85}$$

We define the "Floquet Hamiltonian" as $\boldsymbol{H}_F = \boldsymbol{H}(t') - i\partial_{t'}$, such that

$$\partial_t\,|\psi(t,t')\rangle = -i\boldsymbol{H}_F\,|\psi(t,t')\rangle\,. \tag{A.86}$$

The next step is to think of $\boldsymbol{H}_F$ as an operator on a bigger Hilbert space $\mathcal{H}\otimes\mathcal{H}_F$, where the new sector of Hilbert space is spanned by with states $|n_F\rangle = e^{in\omega t'}$, where n is any integer, positive or negative. We note that the action of multiplying by $e^{im\omega t'}$ is to take $|n_F\rangle$ to $|(n+m)_F\rangle$, and that the action of $-i\partial_{t'}$ is to take $|n_F\rangle$ to $n\,|n_F\rangle$, we can define the operators

$$\boldsymbol{F}_n \equiv \sum_m |(m+n)_F\rangle\!\langle n_F| \tag{A.87}$$

$$\boldsymbol{N} \equiv \sum_m m\,|m_F\rangle\!\langle m_F| \tag{A.88}$$

allowing us to write down our Floquet-space formulation of the periodically driven system:

$$|\psi(t,t')\rangle = \sum_n |\psi_n(t)\rangle \otimes |n_F\rangle \tag{A.89}$$

$$\boldsymbol{H} = \sum_n \boldsymbol{H}_n \boldsymbol{F}_n + \omega \boldsymbol{N} \tag{A.90}$$

A.8.2 Block Diagonalization

With the Floquet Hamiltonian in hand, we now have a time-independent representation. This is convenient because it allows us to analyze the system using time-independent perturbation theory. Our goal will be to perform a transformation (of the form $\boldsymbol{U} = e^{i\boldsymbol{S}}$) putting the Floquet Hamiltonian in block-diagonal form:

$$e^{i\boldsymbol{S}}\boldsymbol{H}_F e^{-i\boldsymbol{S}} = \boldsymbol{H}_{\text{eff}}\boldsymbol{F}_0 + \omega\boldsymbol{N} \tag{A.91}$$

We can put the left side of this equation into BCH form. We are going to look only at terms which are second order or lower in any of the $\boldsymbol{H}_j$, meaning we can truncate at second order for now.

$$e^{i\boldsymbol{S}}\boldsymbol{H}_F e^{-i\boldsymbol{S}} = \boldsymbol{H}_F + i[\boldsymbol{S},\boldsymbol{H}] - \frac{1}{2}[\boldsymbol{S},[\boldsymbol{S},\boldsymbol{H}]] + O(|\boldsymbol{H}_j|^3) \tag{A.92}$$

If we take $\boldsymbol{S} = \sum_{m\neq 0} \frac{i\boldsymbol{H}_m}{n\omega}\boldsymbol{F}_n$, we have

$$[\boldsymbol{S},\boldsymbol{H}] = \sum_{m\neq 0} \frac{1}{m\omega}\left(\sum_n [\boldsymbol{H}_m\boldsymbol{F}_m, \boldsymbol{H}_n\boldsymbol{F}_n] + \omega\boldsymbol{H}_m[\boldsymbol{F}_m,\boldsymbol{N}]\right) \tag{A.93}$$

We can use the easily verified facts $[\boldsymbol{F}_n, \boldsymbol{F}_m] = 0$ and $[\boldsymbol{F}_n, \boldsymbol{N}] = -n\boldsymbol{F}_n$ to then show

$$[\boldsymbol{S}, \boldsymbol{H}] = i \sum_{m\neq 0} \left(\sum_n \frac{[\boldsymbol{H}_m, \boldsymbol{H}_n]}{m\omega} \boldsymbol{F}_{n+m} - \boldsymbol{H}_m \boldsymbol{F}_m \right) \tag{A.94}$$

We can use this result to find the relevant part of the second order BCH term.

$$[\boldsymbol{S}, [\boldsymbol{S}, \boldsymbol{H}]] = \sum_{k\neq 0} \sum_{m\neq 0} \frac{[\boldsymbol{H}_k, \boldsymbol{H}_m]}{k\omega} \boldsymbol{F}_{k+m} + O(|\boldsymbol{H}_j|^3) \tag{A.95}$$

Putting this all back into equation A.92 yields

$$e^{i\boldsymbol{S}} \boldsymbol{H}_F e^{-i\boldsymbol{S}} = \boldsymbol{H}_0 \boldsymbol{F}_0 + \omega \boldsymbol{N} + \frac{1}{2} \sum m, n \neq 0 \frac{[\boldsymbol{H}_m, \boldsymbol{H}_n]}{m\omega} \boldsymbol{F}_{m+n} + \sum_{m\neq 0} \frac{[\boldsymbol{H}_m, \boldsymbol{H}_0]}{m\omega} \boldsymbol{F}_m + O(|\boldsymbol{H}_j|^3) \tag{A.96}$$

Now this form is not exactly the block-diagonal form we were looking for, but it might be closer. The size of the off-diagonal elements (i.e. those involving $F_{m\neq 0}$) have gone from $O(|\boldsymbol{H}_j|)$ to $O(|\boldsymbol{H}_j|^2/\omega)$, which is smaller in the limit $|\boldsymbol{H}| \ll \omega$. In this case, we can take our first-order answer to $\boldsymbol{H}_{\text{eff}}$ (from equation A.91) to be the part proportional to $\boldsymbol{F}_0$, namely:

$$\boldsymbol{H}_{\text{eff}} \approx \boldsymbol{H}_0 + \frac{1}{2} \sum_{n\neq 0} \frac{[\boldsymbol{H}_n, \boldsymbol{H}_{-n}]}{n\omega} \tag{A.97}$$

By carrying this procedure out to one further order we can get the further correction

$$\boldsymbol{H}^{(2)}_{\text{eff}} = \frac{1}{3} \sum_{n\neq 0, n'\neq 0, n\neq n'} \frac{[[\boldsymbol{H}_{n-n'}, \boldsymbol{H}_{-n'}], \boldsymbol{H}_{-n}]}{nn'\omega^2} - \frac{1}{2} \sum_{n\neq 0} \frac{[[\boldsymbol{H}_0, \boldsymbol{H}_n], \boldsymbol{H}_{-n}]}{n^2\omega^2} \tag{A.98}$$

Appendix B

Construction of unitary operations from a universal control set

A d-dimensional quantum control system is a Hamiltonian of the form

$$H(t) = \sum_k \epsilon_k(t) H_k, \tag{B.1}$$

where the control Hamiltonians are elements of the (d^2-dimensional) Lie algebra $\mathfrak{su}(d)$[1]. A set of Lie algebraic operators (G) generates a sub-algebra (span $S(G)$), which is found by taking the closure of the set under the Lie bracket (commutator).

$$\begin{aligned} S_0(G) &= G \\ S_{i+1}(G) &= S_i(G) \cup \{[a,b] \mid a, b \in S_i(G)\} \\ S(G) &= S_{d^2}(G) \end{aligned}$$

We can say that G is *universal* if

$$\text{span } S(G) = \mathfrak{su}(d). \tag{B.2}$$

[1] More accurately iH is a member of the Lie algebra, as traditionally defined. Please forgive my abuse of terminology

This is because if $G = \{H_k\}$ is universal, then for every unitary operation U there is some set of controls $\{\epsilon_k(t)\}$ such that the evolution produces that unitary, i.e.

$$\mathcal{T}\exp\left(\int \mathrm{d}\tau\, H(\tau)\right) = U$$

While constructing such controls $\{\epsilon_k(t)\}$ is difficult exactly, we can easily find controls which realize an operation approximately, up to a desired level of approximation, using "bang-bang" controls, i.e. controls which are either on or off at any given time.

To do so, we begin by processing the input to our problem, the unitary operation U, by taking the matrix logarithm.

$$A = -i\log(U) \rightarrow U = e^{iA} \tag{B.3}$$

A is a member of the Lie algebra, and as such, can be represented as a weighted sum of components from $S(G)$.

$$A = \sum_k c_k B_k,\ B_k \in S(G) \tag{B.4}$$

We will begin with the Suzuki-Trotter decomposition, or more colloquially, *trotterization*:

$$e^{X+Y} \approx (e^{X/N} e^{Y/N})^N. \tag{B.5}$$

We can therefore break up the evolution into terms, each of which relies on only a single member of the set $S(G)$:

$$U \approx \left(\prod_k e^{i\frac{c_k B_k}{N}}\right)^N \tag{B.6}$$

What remains is to show how to realize each of the elements $e^{ic_k B_k}$. These elements (B_k) fall into one of two cases, either they are in G, in which case they are directly realizable, with piecewise-constant controls, or they are formed by the commutator $(B_k = [X, Y])$, in which case they can be approximated using the group commutator:

$$e^{[X,Y]} \approx \left(e^{\frac{X}{N}} e^{\frac{Y}{N}} e^{\frac{-X}{N}} e^{\frac{-Y}{N}}\right)^N. \tag{B.7}$$

Appendix C

Tomographies, large and small

In the context of quantum information the word *tomography* refers to the act of characterization, i.e. taking a set of measurements, and coming to some conclusion about the numbers representing the underlying phenomenon. There are, broadly speaking, three types of objects that we are most often interested in characterizing in quantum information: states, measurements, and processes.

C.1 State tomography

The fundamental equation for state tomography is for computing the expected outcome of a measurement x, via

$$x = \mathrm{Tr}\{\boldsymbol{M}\boldsymbol{\rho}\} = \sum_{ij} \boldsymbol{M}_{ij}\boldsymbol{\rho}_{ji}. \tag{C.1}$$

Here $\boldsymbol{\rho}$ represents the underlying state, and $\boldsymbol{M}$ represents the measurement protocol. We can use the process of vectorization (equation 4.22) to make the structure more apparent.

$$x = \langle\!\langle \boldsymbol{M} | \boldsymbol{\rho} \rangle\!\rangle \tag{C.2}$$

In order to characterize a quantum state, we can perform a set of measurements $\{\boldsymbol{M}_i\}$, to the state. The resulting set of experimental outcomes $\vec{x} \equiv \sum_i x_i |i)$[1] can be related as follows

$$\vec{x} = \left(\sum_i^{N_{\text{exp}}} |i)\langle\!\langle \boldsymbol{M}_i| \right) |\boldsymbol{\rho}\rangle\!\rangle \tag{C.3}$$

$$\equiv \mathcal{M}|\boldsymbol{\rho}\rangle\!\rangle \tag{C.4}$$

If the number of experiments N_{exp} is exactly the number of components of $\boldsymbol{\rho}$ (i.e $N_{\text{exp}} = (\dim \mathcal{H})^2 \equiv d^2$), then $\mathcal{M}$ may be directly invertible and the solution is simply $|\boldsymbol{\rho}\rangle\!\rangle = \mathcal{M}^{-1}\vec{x}$. If more experiments are performed then (with probability 1) an exact solution does not exist. The best we can hope to do is minimize the distance between the two[2]:

$$\underset{\boldsymbol{\rho}}{\text{minimize}} \, \| \mathcal{M}|\boldsymbol{\rho}\rangle\!\rangle - \vec{x} \| \tag{C.5}$$

The solution to this is found by the Moore-Penrose pseudo-inverse

$$|\boldsymbol{\rho}_{\text{least-sq}}\rangle\!\rangle = \left(\mathcal{M}\mathcal{M}^T \right)^{-1} \mathcal{M}^T \vec{x} \tag{C.6}$$

The problem with this approach is that the resulting density matrix does not always obey physicality constraints, namely being positive semidefinite (all eigenvalues are non-negative) and probability conservation ($\mathrm{Tr}\{\boldsymbol{\rho}\} = 1$). We can address these concern with two modifications to the protocol. First, we note that the trace constraint is a linear constraint, representable as:

$$\langle\!\langle \boldsymbol{I} | \boldsymbol{\rho} \rangle\!\rangle = 1 \tag{C.7}$$

We note that, if we want to minimize $\left\| A\vec{x} - \vec{b} \right\|$ while satisfying $C\vec{x} = \vec{d}$, we can replace this (by writing the Karush-Kuhn-Tucker conditions and solving) with the expanded, *unconstrained*

[1] Here $|i)$ is an abstract basis vector indicating a contribution toward the i-th experimental result

[2] This is equivalent to maximizing the (log-)likelihood of the data, $\log \mathcal{L}(\vec{x}) \propto \sum_i |\langle\!\langle \boldsymbol{M}_i | \boldsymbol{\rho} \rangle\!\rangle - x_i|^2$ assuming Gaussian uncertainty on the data, and ignoring physicality constraints, which we will address later

minimization problem

$$\begin{bmatrix} A^T A & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} \vec{x} \\ \vec{\lambda} \end{bmatrix} = \begin{bmatrix} A^T \vec{b} \\ \vec{d} \end{bmatrix}, \tag{C.8}$$

where $\vec{\lambda}$ is a set of Lagrangian multipliers. For us then, we set $A = \mathcal{M}$ and $C = (\langle\!\langle \boldsymbol{I}|)$ is a $1 \times (\dim \mathcal{H})^2$ matrix for the single constraint, and $\vec{d} = (1)$.

We can use this method to identify a generally non-positive semidefinite matrix $\widetilde{\boldsymbol{\rho}}$. We wish to find the closest matrix to $\widetilde{\boldsymbol{\rho}}$ which is also positive semidefinite. For this purpose, we can turn to the method of Smolin et al. (2012), who prove that the closest (in the sense of the 2-norm) positive semidefinite matrix with the same trace can be computed by a simple algorithm, in which $\widetilde{\boldsymbol{\rho}}$ is diagonalized. The positive semidefinite output $\boldsymbol{\rho}$ is formed by redistributing the eigenvalues in such a way that the sum of the eigenvalues is preserved but all are made non-negative.

C.1.1 Optimizing tomography

So far, there has been no discussion of how to pick the measurements $\{\boldsymbol{M}_i\}$ to perform. The key assumption we have made is the invertibility of the matrix $\mathcal{M}$ or rather (in the overdetermined case, equation C.6), $\mathcal{M}\mathcal{M}^T$. In order for this to be true, we need the set of measurements to be "complete," i.e.

$$\text{span}\{\boldsymbol{M}_1, \ldots \boldsymbol{M}_{N_{\text{exp}}}\} = \mathcal{H}. \tag{C.9}$$

However, not all complete sets of measurements are made equal. In particular, we can quantify "how invertible" a matrix is by its *condition number*, κ, defined as the ratio of the largest (magnitude) and smallest (magnitude) eigenvalues.

$$\kappa(\mathcal{M}) = \frac{|\lambda_{\text{max}}(\mathcal{M})|}{|\lambda_{\text{min}}(\mathcal{M})|} \tag{C.10}$$

The condition number provides a bound on the amplification of the relative error. For instance if there is a certain amount of relative noise in the measurement results ϵ, then the relative noise in the inferred density matrix is no bigger than $\kappa\epsilon$. Therefore it is highly desirable to minimize the condition number to maximize the tomography efficiency (Shen et al., 2016).

C.1.2 Wigner tomography

Wigner tomography proceeds from the previous analysis. However, it can be a bit tricky to correctly write down the appropriate form for the matrix $\mathcal{M}$. We need to know how every component of the density matrix ($|n\rangle\, m$, for Fock states $|n\rangle$ and $|m\rangle$) affect every experimental outcome (designated by the corresponding displacement α) to produce the entire Wigner function $W_\alpha(\boldsymbol{\rho})$. Typically we actually look at $\frac{2}{\pi}W_\alpha(\boldsymbol{\rho})$ as this is more experimentally sensible, representing the displaced parity. We can follow some math from Cahill and Glauber (1969, eq. 7.15) in order to arrive at the following:

$$\frac{2}{\pi}W_\alpha(|n\rangle\langle m|) = \langle m|\, \boldsymbol{D}_\alpha \boldsymbol{\Pi} \boldsymbol{D}_{-\alpha}\, |n\rangle \tag{C.11}$$

$$= e^{-2|\alpha|^2}\sqrt{\frac{m!}{n!}}(-1)^{m-1}(2\alpha)^{n-m}L_m^{n-m}(4|\alpha|^2), \tag{C.12}$$

where L_m^k represents the generalized Laguerre polynomial.

We can optimize Wigner tomography by minimizing the condition number as discussed in section C.1.1. In order to do so, one must first choose a photon number cutoff N_{ph}, as otherwise, no finite number of measurements would ever be invertible. Next, one must choose a number of displacements $N_{\text{exp}} \geq N_{\text{ph}}^2$. Then, one can numerically minimize the condition number. This process can be sped up dramatically by calculating the *gradient* of the condition number with respect to the displacements, and using a gradient-aware optimization protocol, as was used for pulse optimization in chapter 4. An example of this is shown in appendix G. An example result of this optimization is shown in C.1. The result is characterized by a distribution of points which is denser closer to the center. It would be an interesting task to try to find some non-numerical method of generating optimally spaced Wigner displacements.

C.2 Process tomography

Processes in quantum mechanics are linear transformations on the space of density matrices. In order to be physical, quantum processes must be *trace-preserving*, that is produce trace-one density matrices when given trace-one density matrices on their input. Next the must be

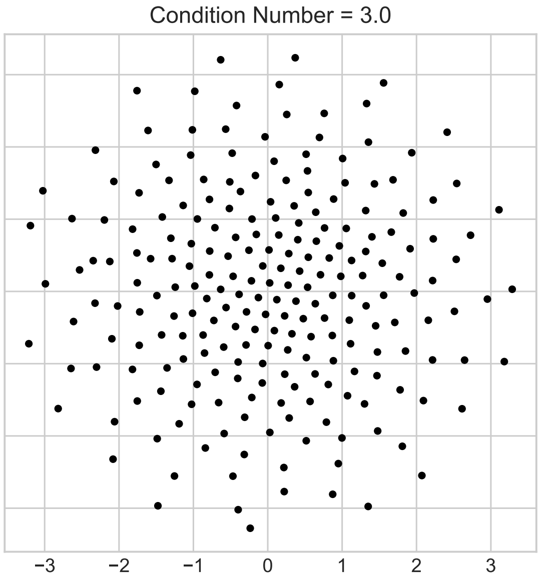


Figure C.1: **Numerically optimized displacements for Wigner tomography**. These displacements were produced by the code in appendix G, and are targeting $N_{\text{ph}} = 14$ and $N_{\text{exp}} = 225$, yielding a condition number $\kappa \approx 3$. For a square lattice with the same number of points, the minimal condition number is approximately 628 ($\alpha_{\text{max}} \approx 2.08$). In order to get a square lattice with the same condition number, one would need $25^2 = 625$ experiments ($\alpha_{\text{max}} \approx 2.88$).

positive, that is produce positive density matrices when given positive density matrices. More subtly, they must be *completely positive*, meaning that they must preserve positivity even when acting within a subspace of a larger Hilbert space. We therefore describe the space of physical processes as completely positive trace-preserving (CPTP) maps.

There are a few equivalent ways of describing superoperators (Wolf, 2012). The first is the Kraus operator-sum representation

$$\boldsymbol{\rho}_{\text{out}} = \sum_{k=1}^{d^2} \boldsymbol{A}_k \boldsymbol{\rho}_{\text{in}} \boldsymbol{A}_k^\dagger \tag{C.13}$$

In this case, complete positivity is ensured by the structure of the representation. The process is trace-preserving when we have

$$\sum_k \boldsymbol{A}_k^\dagger \boldsymbol{A}_k = \boldsymbol{I}. \tag{C.14}$$

We also have the superoperator representation, where the process is given by a matrix ($\mathcal{U}$)

acting on vectorized density matrices.

$$|\boldsymbol{\rho}_{\mathsf{out}}\rangle\!\rangle = \mathcal{U}|\boldsymbol{\rho}_{\mathsf{in}}\rangle\!\rangle \tag{C.15}$$

In this case the trace-preserving property is given by the "unitality of the inverse," i.e.

$$\mathcal{U}^T|\boldsymbol{I}\rangle\!\rangle = |\boldsymbol{I}\rangle\!\rangle \tag{C.16}$$

Before discussing positivity, we must introduce a new representation, which exploits the Choi-Jamiolkowski isomorphism between processes and states. The so-called *Choi matrix*, which I will denote $\boldsymbol{\rho}_{\mathcal{U}}$ is defined as a density matrix on the product Hilbert space $\mathcal{H} \otimes \mathcal{H}$. The Choi matrix is the state which results from applying the process to one half of a completely entangled state $|\Omega\rangle = \frac{1}{\sqrt{d}} \sum_{k=1}^{\dim \mathcal{H}} |k\rangle \otimes |k\rangle$

$$\boldsymbol{\rho}_{\mathcal{U}} = (\mathcal{U} \otimes \mathcal{I})(|\Omega\rangle\!\langle\Omega|) \tag{C.17}$$

Both $\mathcal{U}$ and $\boldsymbol{\rho}_{\mathcal{U}}$ are $d^2 \times d^2$ matrices, whose entries are related by a simple reshuffling of their elements (and rescaling by the dimension d) as follows:

$$\mathcal{U}_{(ij),(kl)} = d \times (\boldsymbol{\rho}_{\mathcal{U}})_{(ik),(jl)} \tag{C.18}$$

We have complete positivity when $\boldsymbol{\rho}_{\mathcal{U}}$ is a positive semidefinite matrix. Finally, we can relate the Choi representation to the Kraus representation. The Choi matrix can be diagonalized in terms of vectorized Kraus operators.

$$\rho_{\mathcal{U}} = \sum_j |\boldsymbol{A}_k\rangle\!\rangle\langle\!\langle \boldsymbol{A}_k| \tag{C.19}$$

In reverse, the Kraus operators can be formed by un-vectorizing the eigenvectors of the Choi matrix and multiplying by the square root of the corresponding eigenvalue.

The way we characterize processes experimentally is by preparing a set of states, applying the process to those states, and performing state tomography on those states. We can describe each experiment as the following

$$x_i = \langle\!\langle \boldsymbol{M}_i|\mathcal{U}|\boldsymbol{\rho}_i\rangle\!\rangle \tag{C.20}$$

This is a linear relationship between the N_{exp}-dimensional experimental result vector $\vec{x}$ and the d^4-dimensional process matrix $\mathcal{U}$. If you will forgive our ridiculous abuse of notation, we can write this as some operator $\mathcal{S}$ relating the experimental result vector and the "process vector" $|\mathcal{U}\rangle\!\rangle$

$$\vec{x} = \left(\sum_i |i)\left(\langle\!\langle M_i|^* \otimes \langle\!\langle \rho_i|\right)\right)|\mathcal{U}\rangle\!\rangle \equiv \mathcal{S}|\mathcal{U}\rangle\!\rangle \tag{C.21}$$

We can now proceed to perform least squares inversion as was done in the case of state tomography (equation C.6). We can maintain the trace-preserving aspect by taking the linear constraints from equation C.16 and using the form from equation C.8. However, if we wish to maintain complete positivity, we cannot simply perform the process from Smolin et al. (2012) on $\boldsymbol{\rho}_{\mathcal{U}}$ as one might like to do, as the process only preserves the trace of the operator, rather than preserving "trace-preserving." If we wish to find the actual least-squares CPTP map, our only recourse is the use of iterative "conic solvers," i.e. optimization tools which can work within the cone of positive semidefinite matrices. This is possible, but cumbersome, and for the results shown in this thesis, we have been content to not force our process matrices to be completely positive.

C.2.1 Pauli transfer representation

For the purpose of visualizing quantum processes, it is convenient to use a representation with real parameters. When we are dealing with a system of n qubits, we can do this with the "Pauli transfer" representation:

$$\mathcal{P}_{ij}(\mathcal{U}) = \frac{1}{2^n}\operatorname{tr}\{\boldsymbol{X}_i\mathcal{U}(\boldsymbol{X}_j\}) = \langle\!\langle \boldsymbol{X}_i|\mathcal{U}|\boldsymbol{X}_j\rangle\!\rangle, \tag{C.22}$$

Where $\boldsymbol{X}_i$ are the elements of the n-qubit Pauli group $\{\boldsymbol{I}, \boldsymbol{\sigma}_x, \boldsymbol{\sigma}_y, \boldsymbol{\sigma}_z\}^{\otimes n}$. Essentially, this is the same as the superoperator representation (equation C.15) except the operator basis is the Paulis, rather than the matrix elements $|i\rangle\langle j|$. As a result of using a Hermitian operator basis, we can get a real representation of the process, which is convenient for visualization.

C.3 Gate set tomography

State and process tomography both rely on crucial assumptions in order to be valid. For state tomography we must trust that we have a reliable description of the measurements $\{\boldsymbol{M}_i\}$ we are performing. In process tomography we must trust both the measurements and the prepared states $\{\boldsymbol{\rho}_i\}$. This is a valid assumption if we are trying to characterize a state or process which has a fidelity much lower than that of our states or measurements. In this case the SPAM errors will not effect the final result too much. But in general this is not the case. What is necessary, is to develop a model of the entire system which is self-consistent. Gate set tomography (Blume-Kohout et al., 2013; Greenbaum, 2015) tries to do this by modeling each of the gates, states, and measurements as full parameterized quantum objects, and using a set of measurements which determine how these objects interact with each other. In the standard procedure, we imagine a system with only one initial state $\boldsymbol{\rho}$ and only one (binary) measurement $\boldsymbol{M}$ (although it can be generalized, this is sufficient for our systems), and an entire set of gates $\{\mathcal{G}_1, \dots \mathcal{G}_K\}$. We have d^2-1 parameters for the density matrix, d^2 parameters for the measurement POVM, and $d^2(d^2-1)$ parameters for each of the K gates, for a total of $Kd^4+(2-K)d^2-1$ free parameters. This can be quite large. However, with enough measurements we can begin to constrain these parameters. The measurements will be determined by the "string" $\vec{i} \in \bigcup_n \{1, \dots, K\}^n$ designating which set of gates is applied to the starting state, and in which order, before measurement:

$$x_{\vec{i}} = \langle\!\langle \boldsymbol{M} | \mathcal{G}_{i_k} \cdots \mathcal{G}_{i_1} | \boldsymbol{\rho} \rangle\!\rangle \tag{C.23}$$

The "fitting" procedure is quite generic. We simply write down the (completely non-linear it should be noted) set of equations relating our parameters and the vector of measurement results, and we maximize the likelihood of observing the data by minimizing the difference between the predicted and observed results. We must do this in an iterative way, lacking any formula. Because of the non-linearity of the cost function, it is possible to end up in a local minima if one is not careful. This chance can be minimized by using a good initial guess for the parameters.

We note that for any invertible transformation $\mathcal{B} \in \mathbb{C}^{d^2 \times d^2}$, the result in equation C.23 will remain invariant under the following "gauge transformation":

$$|\boldsymbol{\rho}\rangle\!\rangle \mapsto \mathcal{B}^{-1} \tag{C.24}$$

$$\langle\!\langle \boldsymbol{M}| \mapsto \langle\!\langle \boldsymbol{M}|\mathcal{B} \tag{C.25}$$

$$\mathcal{G}_k \mapsto \mathcal{B}^{-1}\mathcal{G}_k\mathcal{B} \tag{C.26}$$

This means that, for whatever result we obtain as a result of the likelihood maximization, we can get another equally good result by applying this transformation. There are two reasons we can do this. First, we may wish to make the model "more physical", say by ensuring positivity or complete positivity of the model matrices. Second, we can use this to make the model more reminiscent of our conception of the system. For instance, we are free to perform a permutation gauge transformation which "relabels" the basis vectors so that they match what we expect.

C.4 Randomized benchmarking

Gate set tomography learns the maximum amount of information to come up with the most complete consistent model for a quantum system. On the other end of the spectrum is randomized benchmarking (RB) (Knill et al., 2008; Magesan et al., 2011), which measures a very small set of data to learn one particular aspect of the system, namely the average fidelity of a particular set of gates. Randomized benchmarking is based on a very simple metric: if we use a quantum computer to do a thing, how often do we get the correct answer? Of course, the answer depends on what thing (protocol) we do, but if we wish to get a more generically applicable answer, we can take the average over a large set of protocols. In quantum computing, unlike classical (deterministic) computing, verifying that we get the correct result requires many measurements in general ($O(4^n/\epsilon^2)$ to do full state tomography of n qubits with additive error ϵ). However, if the expected final state is a simultaneous eigenstate of $\{\boldsymbol{\sigma}_z^{(1)}, \ldots, \boldsymbol{\sigma}_z^{(n)}\}$ (i.e. a bit string in the $\boldsymbol{\sigma}_z$ basis) then we can determine if we got the correct answer in a single shot, simply by measuring. In randomized benchmarking, we measure the probability of getting the correct answer over a subset of such protocols, constructed by applying gates from the Clifford group.[3] The Clifford group (the group generated by H, S, and CNOT on n qubits) is a useful setting because the Gottesman-Knill theorem (Gottesman, 1998; Aaronson and Gottesman, 2004) allows one to efficiently classically simulate the effect of Clifford group circuits.

In randomized benchmarking we measure the probability of measuring $|0\rangle^{\otimes n}$, averaged over all Clifford circuits of length n whose ideal final state is $|0\rangle^{\otimes n}$. We then vary the parameter n, and see how the average success probability decays as a function of the sequence length. Under modest assumptions, this decay is exponential. While the success probability for any given n depends on the SPAM errors, the decay constant should not. The decay constant should be related to the "average gate fidelity":

$$\mathcal{F}_{\text{ave}} = \frac{1}{|\text{Clif}|} \sum_{j=1}^{|\text{Clif}|} \int \mathrm{d}\psi \, \mathrm{Tr}\Big\{ \mathcal{U}_j^{\text{ideal}} \left(|\psi\rangle\langle\psi| \right) \mathcal{U}_j^{\text{actual}} \left(|\psi\rangle\langle\psi| \right) \Big\} \tag{C.27}$$

where |Clif| is the number of elements in the Clifford group. If the RB curve is fit to a model $Ap^m + B$, the value p can be used, under certain assumptions, to compute $\mathcal{F}_{\text{ave}} = p + (1-p)/2^n$ (Magesan et al., 2011). However, the practicality of these assumptions has been criticized (Proctor et al., 2017), and the relationship between the RB decay constant and any precise definition of gate fidelity is murky. When we apply RB to logical qubits, such as cat-code qubits (as done in sections 6.4 and 9.6) is one such situation in which the assumption providing that relationship to average state fidelity do not precisely hold. However, the RB constant, by itself, is still a useful metric for evaluating the performance of a quantum computer, which can be meaningfully compared to other RB constants.

One way in which RB decay constant comparison is useful is in the case of *interleaved* randomized benchmarking (iRB) (Magesan et al., 2012). In this setting, we construct a sequence in which we alternate between a given fixed operation and a randomly selected Clifford

[3] There are proposals which use other groups, allowing for benchmarking of a different set of gates (Cross et al., 2016). Using a universal set of gates is possible (see e.g. "cross-entropy benchmarking"), but not scalable to arbitrary numbers, as it requires the ability to classically simulate the entire circuit

operation. By comparing the decay constant of the resulting curve to that of standard RB, the we can determine how the fidelity of the interleaved pulse compares with the average of the entire gate set. In this way one can identify the proportion of the error which is attributable to each gate.

Appendix D

Taming the Black Mamba: Structure and software for the Yngwie FPGA quantum controller

In this section we will discuss the many components of the quantum control stack. In figure D.1 we see an outline of the components involved. We will go through each of these components, including the hardware, the input and output processing chains, the execution model, the instruction sets, higher level sequencing language, experiment organization, and user interface.

D.1 Understanding the hardware

The Yngwie quantum controller is built on top of the Innovative Integration X6-1000M board. The board provides digital-analog converters (DACs) which create the control signals, analog-digital converters (ADCs) which record the readout results, and a field-programmable gate array (FPGA) which synchronizes and orchestrates their operation. On the output side, the X6-1000M board has four 16-bit DAC channels running at 500 MS/s (or 2 DAC channels at 1 GS/s). These outputs have a maximum output voltage of $2V_{pp}$. On the input side, there are two 12-bit ADC channels running at 1 GS/s, which accepts signals smaller than $1V_{pp}$. The FPGA controlling them is a Xilinx Virtex 6 SX475T. The FPGA is loaded with the Yngwie "logic," a hardware specification which specifies how the FPGA is internally configured. This logic is specified using VHDL, and will be discussed at length in the following section. In addition to the analog inputs and outputs there are several 2V digital inputs and outputs as well. These can be used to trigger additional instruments, to gate RF switches, or to communicate measurement results.

Up to four X6-1000M cards can be operated synchronously using the VPXI-ePC (colloquially known as the *Black Mamba*). This card distributes a clock and trigger signal so that each card begins its experimental sequence simultaneously, with a trigger jitter smaller than 4 ps.

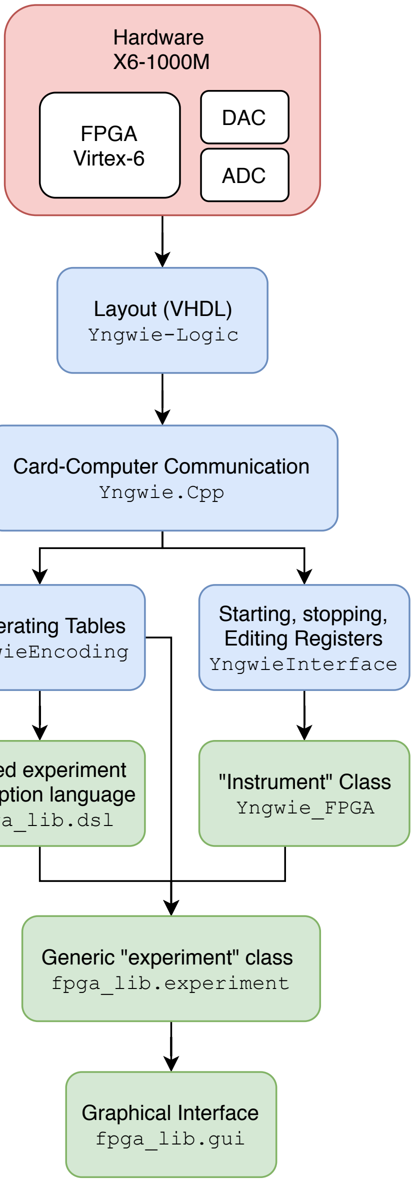


Figure D.1: **Outline of organization of components in FPGA stack.** The hardware (red) was done by Innovative Integration LLC. The logic and low level software (green) was written by Nissim Ofek. The higher-level software (green) including `fpga_lib` components were written by Philip Reinhold, with assistance from Reinier Heeres.

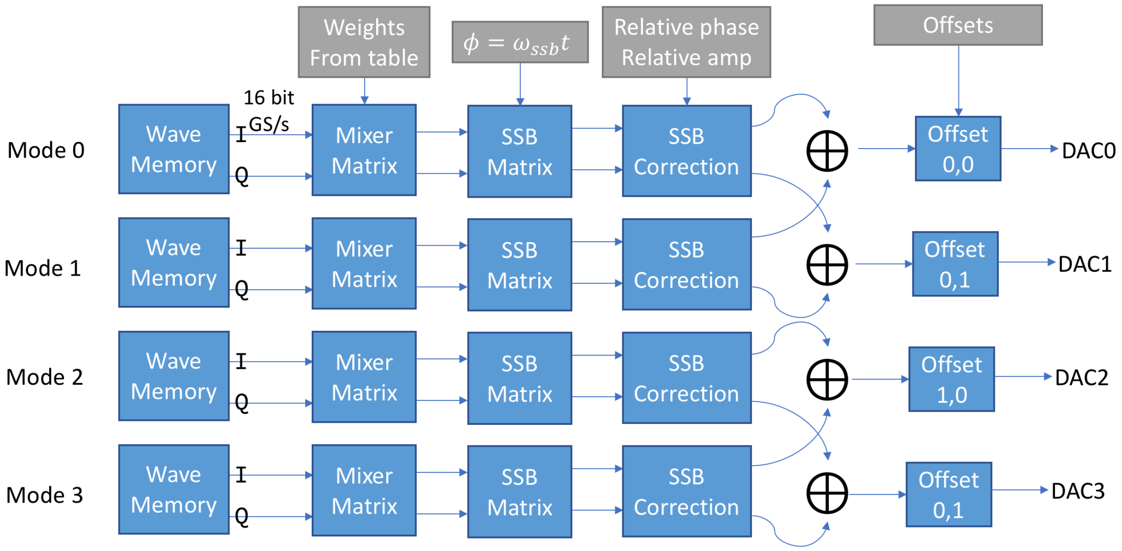


Figure D.2: **Yngwie analog output chain** In this example, the system is configured in four-mode two output pair operation

D.2 The Yngwie VHDL logic

The architecture of the Yngwie controller is quite complicated. There are several concurrently executing "threads"[1] of execution, several of which have completely different instruction sets and capabilities. There are up to four *analog* sequences, whose responsibility it is to schedule pulses to be sent to the DACs. There are two *digital* sequences, which schedule turning on and off digital "marker" pulses. There is a *CPU* sequence, which contains descriptions of real-time computations. Finally, there is the *master* sequence, which manages the acquisition chain and schedules the CPU computations. Each of these 5-8 threads operates in parallel, on every card in the system. For a full crate of cards, this is up to 32 simultaneous threads. These threads have no way of staying synchronized with each other aside from careful design of the programs which control them. In order to simplify this task, the `fpga_lib` sequencing language and compiler can be used, in order to write the entire set of instruction sets in a unified way, and minimizing the chance of unintentional desynchronization.

D.2.1 Analog output chain

The output chain, defined in the logic, connects the waveform memory to the physical DACs on the board. We can see a block diagram in figure D.2. The first thing to note about the output chain is that it operates on pairs. The idea is that a pair of physical outputs can be used with an IQ mixer to achieve single-sideband (SSB) modulation of the carrier tone. There are four physical outputs of the card, and so it supports up to two pairs. The card can be configured to use either a single pair (DA0/DA2) or two pairs (DA0/DA1 and DA2/DA3). When using two pairs, the sample rate is cut from 1 GS/s to 500 MS/s. In addition to configuring the number of outputs, one can also configure the number of *modes*. A mode is an independent processing

[1] I am not invoking the technical definition of thread, as used in an operating systems context

channel which can carry out the tasks of scaling, rotating, modulation, and mixer adjustment. Each mode produces a pair of streams. If there are more modes than outputs, the modes are distributed among the outputs and summed together. Since the number of modes can be 1, 2, or 4 we have five ways of configuring the system

1. 1 mode, 1 output pair
2. 2 modes, 1 output pair. The two modes I and Q outputs are summed before going to the DAC
3. 4 modes, 1 output pair. All four modes I and Q outputs are summed.
4. 2 modes, 2 outputs: Each mode gets an output (mode 0 to DA0/1, mode 1 to DA2/3)
5. 4 modes, 2 outputs: Two modes to each output. (mode 0 + mode 1 to DA0/1, mode 2 + mode 3 to DA2/3)

Each mode gets its own segment of wave memory, but since the total amount of memory is fixed, adding more modes reduces the wave memory available to any given mode, which is why one might prefer to use fewer modes.

Each mode gets assigned its own analog sequencing table, and therefore can be scheduled completely independently. When a pulse is played, it generates a stream of sample pairs. We will denote the samples at the k-th stage as

$$\begin{pmatrix} I^{(k)} \\ Q^{(k)} \end{pmatrix} \tag{D.1}$$

Each mode performs three major processing tasks to the samples requested by the respective tables. The first is apply its own *mixer matrix* a two-by-two matrix (MX)

$$\begin{pmatrix} I^{(1)} \\ Q^{(1)} \end{pmatrix} = \begin{pmatrix} \mathsf{MX}[0,0] & \mathsf{MX}[0,1] \\ \mathsf{MX}[1,0] & \mathsf{MX}[1,1] \end{pmatrix} \begin{pmatrix} I^{(0)} \\ Q^{(0)} \end{pmatrix} \tag{D.2}$$

The components of this matrix are taken either explicitly from the instruction which plays the pulse, or from the dynamic mixer values (§D.2.5). The most common use case is to scale the amplitude of a pulse (for this, a diagonal mixer matrix is used) or to adjust the pulse phase (in which case a rotation matrix is used).

Next is the SSB modulator.

$$\begin{pmatrix} I^{(2)} \\ Q^{(2)} \end{pmatrix} = \begin{pmatrix} \cos \omega_{\mathsf{SSB}}(t-t_0) & \sin \omega_{\mathsf{SSB}}(t-t_0) \\ -\sin \omega_{\mathsf{SSB}}(t-t_0) & \cos \omega_{\mathsf{SSB}}(t-t_0) \end{pmatrix} \begin{pmatrix} I^{(1)} \\ Q^{(1)} \end{pmatrix} \tag{D.3}$$

Here t is the real time the sample is played and t_0 is either the beginning of the experiment, or the last time the sideband frequency was updated. This is "single-sideband" because assuming a constant input, the complex-valued output $I + iQ \propto \cos \omega_{\mathsf{SSB}} t + i \sin \omega_{\mathsf{SSB}} t = e^{i\omega_{\mathsf{SSB}} t}$ has only a single frequency component. The final step is to perform an adjustment for imperfections in the mixer. These values are specified via the registers—accessed by `YngwieInterface` or the

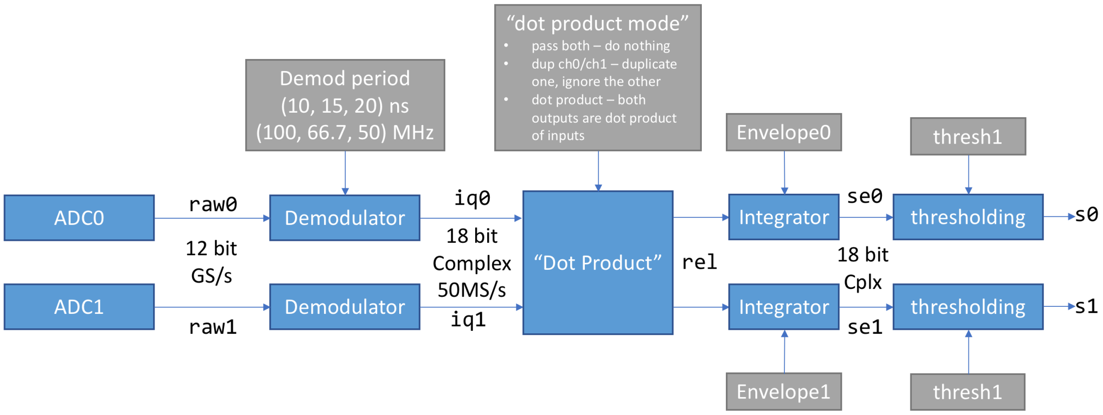


Figure D.3: **Yngwie analog input chain**

`Yngwie_FPGA` instrument (§D.4)—and perform the final transformation:

$$\begin{pmatrix} I^{(3)} \\ Q^{(3)} \end{pmatrix} = \begin{pmatrix} u^{1/2}\cos(\phi) & u^{1/2}\sin(\phi) \\ u^{-1/2}\sin(\phi) & u^{-1/2}\cos(\phi) \end{pmatrix} \begin{pmatrix} I^{(2)} \\ Q^{(2)} \end{pmatrix}, \tag{D.4}$$

where the parameters ϕ and u correspond directly to the `ssbtheta` and `ssbratio` parameters (§D.4.) Note in this formula is not like a rotation matrix (in which the $\sin(\phi)$ components would have opposite sign). The effect of this adjustment is to change the relative phase of the I and Q components, as well as the relative amplitude.

Next the modes are combined in order to match the number of outputs if necessary. Finally, the outputs have offsets applied. These offsets, which are used to cancel the mixer leakage, are on so long as an experiment is running (regardless of whether pulses are being currently played).

D.2.2 Analog input chain

The analog inputs are processed in a series of steps which refine the acquired data all the way from an array of raw ADC samples to a final binary value indicating a quantum measurement result, upon which we can branch the controller's execution. We have the ability to tap of the data at any point in this processing chain. We see the chain in figure D.3. We start with the `raw0/1` data, which comes straight from the ADC0/1 inputs, and is 12 bit data with 1 ns resolution. This is first processed by the *demodulators* to produce the `iq0/1` data, each of which is an 18-bit complex pair.

$$\mathtt{iq0/1} = \sum_{k=0}^{\mathtt{demod\_period}-1} \mathtt{raw0/1}[k] \left(\mathtt{demod\_table\_I\_0/1}[\mathtt{idx}_k] + i\mathtt{demod\_table\_Q\_0/1}[\mathtt{idx}_k]\right) \tag{D.5}$$

$$\mathtt{idx}_k = (k+t) \bmod \mathtt{demod\_period} \tag{D.6}$$

The `demod_period` is set by a register and can be either 10, 15, or 20 ns. Here t is the number of nanoseconds either since the beginning of the experiment (if `demod_reset_mode` is True) or since the beginning of the acquisition (if demod_reset_mode is False). Typically the values in the demodulator table are set as follows:

$$\texttt{demod\_table\_I\_0/1}[k] = \cos(2\pi k/\texttt{demod\_period}) \qquad \text{(D.7)}$$

$$\texttt{demod\_table\_Q\_0/1}[k] = \sin(2\pi k/\texttt{demod\_period}) \qquad \text{(D.8)}$$

This results in a demodulation by 100, 66.7 or 50 MHz depending on the period.

The next step is the "dot product" step, whose behavior depends on the value of dot_product_mode (§D.4) which transforms iq0/1 into rel0/1.[2] The behavior of this module is trivial unless dot product mode is enabled, in which case the output is

$$\texttt{rel0/1}[k] = \texttt{iq0}[k]^* \times \texttt{iq1}[k]. \qquad \text{(D.9)}$$

This is useful if one of the inputs is a reference signal which determines what phase at which to expect the other signal to arrive. If there are fewer values in the iq0 stream than iq1 (i.e. because the channels acquire for different lengths) then the dot product will continue using the last recorded value of iq0 (or vice versa). This is often used if the reference signal is only present for a short time, and because the reference signal is expected to be constant (after demodulation anyway).

Next comes the "state estimation" component, which consists of *integration* and *thresholding*. The integration consumes the rel0/1 streams and for each produces a single 18 bit real number, called se0/1.

$$\texttt{se0/1} = \text{Re}\left\{\sum_k \texttt{rel0/1}[k] \times \texttt{envelope0/1}[k]\right\} \qquad \text{(D.10)}$$

The point is to compare the observed trajectory to some reference trajectory, and yield some similarity metric. If both rel0/1 contain the same contents (which they will unless dot_product_mode is "pass both") then we can compare to two reference trajectories, which is useful for assigning more than two outcomes to a readout. We can access these se values in the CPU (§D.2.5) but most commonly we will instead access the thresholded binary values

$$\texttt{s0/1} = \texttt{se0/1} > \texttt{thresh0/1} \qquad \text{(D.11)}$$

The thresholds thresh0/1 and envelopes envelope0/1 are specified in the "state estimator table." The current address in the state estimator table can be set by the master sequence, allowing the envelope and thresholds to be changed on the fly.

D.2.3 Digital inputs and outputs

There are many digital (2V) signals which are accessible from the labelled BNC outputs. The mapping between labels and function is given in table D.1. There are three types of outputs present on the card, with different functions and control mechanisms:

[2] Only one rel stream is externally accessible at a time. The default is rel0, but can be switched using the rel_stream_selector in YngwieInterface.

Description	In/Out	PCI-E (single card)	Black Mamba
External Function 0	Out	P14,P16,P17,P18,P19	N15,N16,N17,N18,N19
External Function 1	Out	P15,P20,P21,P22,P23	N14,N20,N21,N22,N23
X0	In	N14	P15
X1	In	N15	P14
X00	In	N16	P17
X01	In	N17	P16
X10	In	N18	P19
X11	In	N19	P18
X20	In	N20	P21
X21	In	N21	P20
X30	In	N22	P23
X31	In	N23	P22
Automatic Buffer 0	Out	P8,P9	N8,N9
Automatic Buffer 1	Out	N8,N9	P8,P9
Automatic Buffer 2	Out	P10,P11	N10,N11
Automatic Buffer 3	Out	N10,N11	P10,P11
Digital Marker 0	Out	P12	N13
Digital Marker 1	Out	N12	P13
Digital Marker 2	Out	P13	N12
Digital Marker 3	Out	N13	P12

Table D.1: **Yngwie FPGA digital output mapping**

- External functions: These reflect some Boolean function of the signals (§D.3.1). The exact function can be dynamically set in the master sequence.

- Automatic Buffers: These digital outputs are designed to go high when the pulses are actively being played, allowing for an RF switch which is normally closed to be opened. Exactly which modes activate which buffers can be configured with the `buffer_mask` settings (§D.4).

- Digital Markers: These outputs can be manually controlled and scheduled in a very similar way to the analog outputs (§D.3.4).

In addition to the outputs there are 10 inputs. These are all functionally identical, in that they control the value of a named signal (§D.3.1) which goes high or low depending on the corresponding input voltage.

Setting up intercard communication

In a "Black Mamba" crate of four cards, we may wish for the ability to have each of the cards branch on a signal generated in any of the other cards. For this purpose we use the "external function" digital outputs in conjunction with the $\mathtt{X}nm$ digital inputs. While one can build up the needed physical connections between the cards in a piecemeal way as needed, a more organized and thorough approach can pay off in the long run. To allow for the maximum

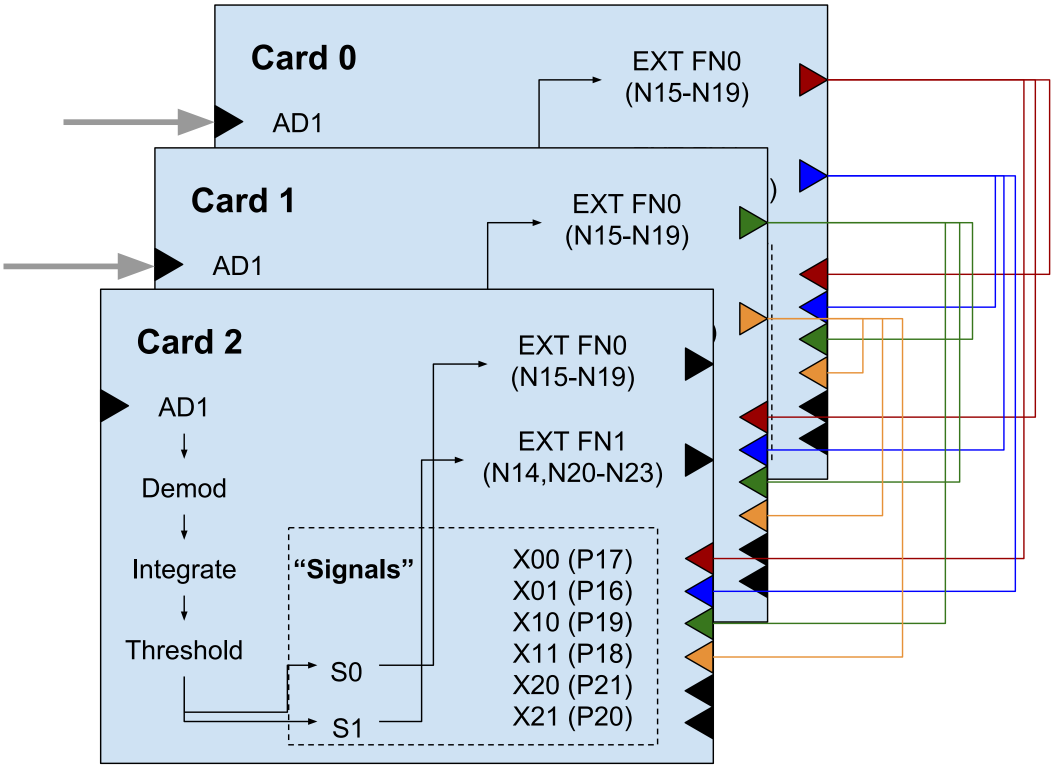


Figure D.4: **Inter-card connections to use for multi-card feedback**

amount of intercard communication, one should hook up the inputs and outputs as shown in figure D.4. In this setup, each card distributes a copy of its external functions to each other card. The naming convention is that input Xnm is connected to a copy of external function m on card n.

With this connectivity in place, in order for each card to branch on a signal (§D.3.1) generated in some card (say card n), it is only necessary to first assign that signal to the external function m on card n, and second, have every card branch on the signal Xnm. In typical use, the only signal which needs to be distributed are those which cannot be generated in parallel on all cards, namely the measurement results. In this case, one typically assigns the external functions to the state estimator signals s0 and s1 to external functions 0 and 1 respectively at the beginning of each sequence, keeping the interpretation of Xnm fixed as "the m-th readout result on card n."

D.2.4 Tables

The behavior of the Yngwie controller is determined in two ways. The first is a set of *registers* which contain discrete settings. These can be accessed via the `YngwieInterface` module (§D.3. This can be integrated with the `instrumentserver` via the `Yngwie_FPGA` instrument class (§D.4). The second way the behavior is determined is by the *tables*. The tables can be created directly using the `YngwieEncoding` module. The overhead of managing the large number of tables can be handled using the `fpga_lib` tools in sections D.5 and D.7.

- *Master*: Schedules acquisition periods and CPU programs
- *Analog*: Schedules the analog outputs
- *Digital*: Schedules the digital marker outputs
- *Program*: Contains descriptions of CPU programs
- *Waves*: Contains the waves which are played by the analog outputs
- *Memory*: Contains the initial contents of the CPU accessible memory
- *Demodulation*: Contains the waveforms used for demodulation (typically $\cos$ and $\sin$)
- *Integration*: Contains the envelopes used for integrating
- *State estimator*: Contains the state estimator parameters.
- *Arbitrary function*: Contains the arbitrary function definitions (§D.2.5).

D.2.5 CPU

The Yngwie controller has a very unique computation model for its so-called "super-CPU"[3]. Architecturally, compared with standard CPU implementations, it resembles most a "transport-triggered architecture."

In this model, the CPU consists of a set of functional units (adders, multipliers, memory accessors, etc.) each of whose inputs are connected to a multiplexer, which is in turn connected to the outputs of many other functional units. On each cycle, the CPU advances to the next instruction (there is no branching in this model), and each CPU instruction specifies the behavior for each multiplexer.

This is a difficult model to program to, because one must keep in mind the connectivity diagram for each of the functional units as well as the processing time. For instance, if one makes a memory request on instruction N, one must access the result on instruction $N+3$, and only then. However this model also allows for a great deal of parallelism, as each functional unit can be manually scheduled, and operates in parallel. In addition, each functional unit is fully pipelined, allowing one to put multiple inputs into a functional unit before the first input has finished processing, and retrieve the corresponding answers in order from the output.

In what follows I will describe each of the functional units (bold face) as well as their input operands (italics). The `IMM0/1` values are not functional units per se but constant values which are stored in the CPU instructions.

- **MEM**k(*action, value, address*) (3 cycles) for $k \in 0, \ldots, 3$

[3] Named because the previous implementation was a "mini-CPU"

  - MEM0/1 and MEM2/3 refer to completely separate banks of memory
  - *action*
    * read: produce on its output the contents of the memory at the given address.
    * write: Take the given input value and store it at the given address
  - *value*: one of `IMM`$< k \bmod 2 >$, ADD1...3, MEM0...3
  - **address**: either mem_addr$< k >$ (explicit in CPU instruction) or ADD1...3

- **ADD1**(*left op, right op, action*) (1 cycle)
  - *action*: The output of the adder depends on the action specified. Here a is the *left op* and b is the *right op*.
    * $a + b$
    * $a - b$
    * $a\&b$ (bitwise AND)
    * $b - a$
    * $a|b$ (bitwise OR)
    * $a \wedge b$ (bitwise XOR)
    * $b >> 1$ (divide by two)
  - *left op*: one of 0, ARB, MUH0, MUL0, ADD1, ADD2, `IMM0`, `IMM1`
  - *right op*: one of 0, ADD1...3, MUH1, MUL1, SENS, TST

- **ADD2**(*left op, right op, action*) (1 cycle)
  - *action* (same as ADD1)
  - *left op* one of 0, ARB, MUH1, MUL1, ADD1, ADD2, `IMM0`, `IMM1`
  - *right op* one of 0, ADD1...3, MUH0, MUL0, SENS, TST

- **ADD3**(*left op, right op, action*)
  - *action* (same as ADD1)
  - *left op* one of 0, ADD1...3, MEM0...3
  - *right op* (same as left op)

- **MUL0/1**(*left op, right op*) (4 cycles).
  - Note that multiplication of two n-bit numbers produces a $2n$ bit number. We need to take an n-bit part of the answer and throw away the rest. There are two conventions which are supported. The inputs can be considered integers, in which case the least significant bits are taken, which is the default behavior. Otherwise, the inputs can be considered as fixed-point numbers, contained in the range $[-2, 2]$. In this case the (almost) highest bits are taken. MUH0/1 designates this part of the answer.
  - *left op* one of `IMM0/1`, ADD1...3, MEM0...3
  - *right op* one of SE$0/1$, ADD1...3, MEM0...3

- **TST**(*op*) (1 cycle)
  - Putting a value x into TST will cause it to update the `r0` $(x = 0)$ and `r1` $(x < 0)$ signals which can be used for branching or sending to the external functions. It presents on its output the value loaded into it, meaning that this is a convenient way of getting the SE0 and SE1 values into the CPU.
  - *op*: one of TST, ADD1. . . 3, MEM0, MEM2, SE0, SE1
- **ARB**(*op, function*) (9 cycles)
  - Evaluates one of the 8 "arbitrary functions" which are defined in the arbitrary function table. Evaluation works by interpolation over 512 points spanning the full range. Most commonly used to evaluate $\sin$ and $\cos$.
  - *op*: one of 0, ADD1. . . ADD3, MEM0. . . MEM3
  - *function*, an integer in 0,. . . ,7
- **CPU Outputs**(*value*)
  - Assigning to these values changes behavior elsewhere in the FPGA
    1. **MX00**,**MX01**,**MX10**,**MX11**: Dynamic mixer values
    2. **SSB**: dynamic SSB value
    3. **DYN1. . . 3**: Dynamic length
    4. **GOF0. . . 3**: Dynamic goto jump
    5. **REC0. . . 3**: Set value in result record
  - *value* one of ADD1. . . 3, MEM0. . . 3 (**MX** only)

D.3 The `Yngwie` python libraries

The primary `Yngwie` code interface is a C++ package which mediates the communication between the cards and the host PC. Knowledge of this layer is largely unnecessary. On top of this is the Python interface, which functionally serves as the documentation of the Yngwie logic. It is separated into two parts. The first is `YngwieEncoding` which contains the tools necessary for building all of the tables (§D.2.4), including python objects corresponding to the various instruction sets. The second is `YngwieInterface`, which contains tools for manipulating the registers—most of which correspond to settings in the `Yngwie_FPGA` instrument (§D.4)—as well as methods for starting and stopping experiments.

D.3.1 Basic sequencing

Instructions in the master, analog and digital tables share a common execution model. Each instruction specifies a `length` as well as a method of deciding what instruction to jump to next. The length is specified in units of cycles (4 ns), and can be up to 2^{19} cycles (about 2 ms). In addition to the static length, a `dynamic_length` can be added from any of the three dynamic length CPU outputs (§D.2.5) (`'DYN1'`,`'DYN2'` or `'DYN3'`).

In addition to having a length, each instruction carries a mechanism for deciding which instruction address to go to next. There are four ways of determining the next instruction to jump to.

- `GOTO` x: explicitly specifying the instruction to jump to, either absolutely or relatively to the current instruction. The jump value x has an explicit component written in the instruction, but to this one can add one of the four dynamically computed "goto offset" values. The most common instruction is `GOTO +1`.
- `IF` x `ELSE` y: Examine the current value of the internal function (§D.3.2. If the value is high, go to x, otherwise y.
- `GOSUB` x `RET. TO` y: Jump to x, pushing the value y onto a stack which can contain up to four levels (don't try to use explicit recursion here)
- `RETURN` Pop an address y off of the stack and go to that point

D.3.2 Master table

The master table is in charge of data acquisition, scheduling the CPU, and setting the internal and external functions. Entries in the master table can be created using the `MasterInstruction()` object. It has the following properties:

1. `trigger_levels`: Indicates which, if any, oft the two input channels should be acquiring during this instruction.
2. `estimation_params_addr`: The address into the state estimator table which should be used for the subsequent acquisitions
3. `demod_reset`: Signal to restart the phase of the demodulator (§D.2.2)
4. `generate_record`: Signal to generate a single result record at this point
5. `program_entry`: Starts a CPU program using this address into the program table
6. `internal_function`: Accepts a Boolean function (see below) which is used for branching `IF THEN ELSE` instruction address resolution (§D.3.1). item `external_function0/1`: Accepts a Boolean function (see below) which is used to set the corresponding digital output (§D.2.3).

Signals

The internal and external functions take values which correspond to an arbitrary Boolean function of the *signals*. There are three types of signals:[4]

- `s0/1`: Is the state estimator result `se0/1` is larger than the corresponding threshold?

[4] Perusing the code you may find references to a fourth type, c0/1 which corresponds to a "counter" feature present in an older version of the logic.

- `r0`: Is the current value in the TST CPU unit (§D.2.5) equal to zero?
- `r1`: Is the current value in the TST CPU unit less than zero?
- `x0/1`, `x`n`0/1` for $n \in 0, \ldots, 3$: Is the voltage associated digital input (§D.2.3) currently high?

These signals can be combined using the standard Boolean functions AND (`&`), OR (|), XOR ($\wedge$), and the unary NOT ( ). The resulting function can be completely arbitrary, but can have no more than four variables (signals).

D.3.3 Analog tables

In addition to the common components listed in §D.3.1, each instruction in the analog tables (created with the `AnalogPairInstruction()`) has the following properties.

- `wave_address`: Indicates the address in wave memory at which to begin playing
- `unique_wave`: If this is true, the wave address increments every cycle. If false, the wave address stays the same, repeating the same 4 ns segment of wave memory. This option allows for a contant pulse with no memory overhead.
- `mixer_amplitudes`: A set of four constants which specify the mixer matrix values (§D.2.1).
- `mixer_mask`: Specifies which of the mixer matrix values are taken from the mode's dynamically determined mixer matrix. Note that these are the values which were in the CPU outputs (§D.2.5) when the `mixer_fetch` option was used, not the current value in the CPU outputs.
- `mixer_fetch`: Specifies that the mode should load the dynamic mixer matrix values from the CPU.
- `ssb_reload`: Specifies that the mode should change its SSB frequency using the value in the CPU output.
- `silent_wave`: If true, specifies that the wave should not trigger the autobuffers (§D.2.3).

It should be noted that in actuality the I and Q outputs actually run independent threads of execution. They operate on the same table, but begin their executions at address 0 and 1 respectively. In practice, by using the `AnalogPairInstruction` objects to build the analog table, we can ignore this implementation detail, and just consider it to be a single thread which directs both outputs, but in principle the possibility to manipulate these separately exists.

D.3.4 Digital tables

The digital tables are much simpler than the master and analog tables. There are two digital sequences, each of which controls two of the digital marker outputs (§8). The `DigitalInstruction()` has ony a single additional field, the `marker_levels` parameters, which contains two bits indicating which of the sequence's two marker outputs is high for the duration of the instruction.

D.3.5 Result records

In addition to the output types described in the analog input processing chain (§D.2.2), the FPGA also has the ability to generate *result records* (called `rec`, when considered to be an output stream). These contain a lot of data about the internal state of the controller at the time that it was generated. Result records are generated in one of two ways: either when we advance to an instruction in the master sequence with the `generate_record` flag enabled (§D.3.2) or if one of the conditions specified in the current value of the `rrec_generate` register is met (§D.4). The result records can be interpreted using the `YngwieDecoding.ResultRecord()` objects, and contain the following fields:

- `s0/1`, `x0/1`, `r0/1`: a Boolean value describing the current value of the indicated signal (§D.3.2)
- `ir`, `ex0/1`: The current (Boolean) value of the internal function or external functions
- `time_stamp`: The time, measured in cycles since the start of the experiment
- `addr_curr` and `addr_next`: The

D.4 The `Yngwie_FPGA` instrument class

- `demod_reset_mode`: Confusingly, true indicates that the phase of the demodulator is never reset, but continues real time. False indicates that the demodulator phase should be reset every time a new acquisition is started.
- `dot_product_mode`: specifies how the “dot product” module in the input chain functions: (§D.2.2)
  - `dot product`: $\texttt{rel0} = \texttt{rel1} = \texttt{iq0} \times \texttt{iq1}^*$
  - `dup ch0`: $\texttt{rel0} = \texttt{rel1} = \texttt{iq0}$
  - `dup ch0`: $\texttt{rel0} = \texttt{rel1} = \texttt{iq1}$
  - `pass both`: $\texttt{rel0} = \texttt{iq0}$ and $\texttt{rel1} = \texttt{iq1}$.
- `dump_path`: The directory into which the next acquired data blocks should be written. Typically automatically managed by the `fpga_lib.experiment` class if used (§D.7).
- `nmodes`: The number of modes to create. Either 1, 2, or 4. Cannot be set dynamically, but rather must be specified at instrument creation.
- `noutputs`: How many physical outputs to enable. 2 indicates one pair, and allows for GS/s operation. 4 indicates two pairs, and allows for 500 MS/s operation.
- `rrec_generate`: Controls when result records are generated. This integer should be considered as a bit stream indicating which conditions cause result records to be generated. One can set which conditions one wants by summing a subset of the following terms.
  - $2^0 = 1$: When the signal `s0` changes

  - $2^1 = 2$: When the signal s1 changes
  - $2^4 = 16$: When the signal x0 changes
  - $2^5 = 32$: When the signal x1 changes
  - $2^6 = 64$: When the signal r0 changes
  - $2^7 = 128$: When the signal r1 changes
  - $2^8 = 256$: When the internal function result changes
  - $2^9 = 512$: When the external function0 result changes
  - $2^{10} = 1024$: When the external function1 result changes
  - $2^{13} = 8192$: When the current (master) instruction address changes
  - $2^{14} = 16384$: When the proposed next (master) instruction address changes
  - $2^{16} = 65536$: When the state estimator result se0 changes
  - $2^{17} = 131072$: When the state estimator result se1 changes
  - $2^{18} = 262144$: When the state estimator result se0 is finalized
  - $2^{19} = 524288$: When the state estimator result se1 is finalized
  - $2^{20} \ldots 2^{23}$: When the CPU output registers REC0...3 change

- run_status: Indicates the presence of some sort of initialization error. If this value is non-zero, then there was a problem. The first bit being high indicates that the attempt at starting a run failed. The second bit being high (2 or 3) indicates that the system is waiting for a trigger. The third bit indicates an invalid trigger. The fourth bit indicates that the trigger is overloaded.

- unlimited: False(/True) means that the FPGA does(/does not) stop running when the requested amount of data is acquired.

- delay_analog: The delay (in units of cycles) applied to all of the analog outputs. For obscure reasons, this has a minimum value of 13 cycles, and a maximum of 63 cycles.

- delay_marker0...3: The delay (in units of cycles) applied to a given digital marker output. Can take any value between 0 and 63.

- ext_fn_enabledEXxy: Each of the external functions can be replicated on up to five outputs. These copies can be enabled or disabled individually for the purpose of reducing the internal noise and crosstalk.

- buffer_mask0...3: Indicates which modes trigger each of the automatic buffers (§D.2.3). In this four bit number (0-15) each bit (lowest bit gives mode 0, highest bit gives mode 3) indicates whether that mode triggers this automatic buffer.

- buffer_combiner0...3: In addition to being triggered by the analog output of a mode being non-zero, the automatic buffers can also be sensitive to the value of the external function. This two bit field describes whether or not the buffer should be activated on external function 0 (bit 0) and/or external function 1 (bit 1).

- buffer_gen_delay: A delay (in cycles) between 0 and 63 applied to all of the autobuffers

- buffer_gen_width: A 32-bit field which is convolved with the "mode non-zero" signal to produce the autobuffer output. Generally used to pad the width of the autobuffer by up to 31 cycles. To pad by k cycles set this value to $2^{k+1} - 1$. Setting this value to zero will disable the autobuffer.
- offset0/1: Each of these is a pair of numbers which controls the DC offsets of the two physical outputs corresponding to each output pair. These are given as integers, where the full DAC range is $[-2^{17}, +2^{17}]$.
- ssb_resolution: When loading the SSB dynamically, the integer value in the SSB CPU output is multiplied by this factor to produce the real SSB frequency needed. The options are 1 kHz, 500 Hz, 250 Hz and 125 Hz.
- ssbfreq0...3: The initial SSB frequency for mode 0. . . 3, in Hz.
- ssbratio0...3: The relative amplitude of I and Q (for modes 0. . . 3) as applied to the mixer correction matrix (§D.2.1).
- ssbtheta0...3: The relative phase correction between I and Q. Zero indicates that they should be exactly (nominally) 90° out of phase.

D.4.1 Running experiments

The procedure for actually running experiments has a few steps.

1. Compile each of the tables listed in §D.2.4. The output files should be stored in a single directory
2. Use the Yngwie_FPGA instrument instance method load_tables() on the directory
3. Calculate the amount of data that the experiment should produce of each type: raw0/1, iq0/1, rel, se0/1, and rec (result records, §D.3.5). Call the instrument instance accept_stream() method for each type. The result can be split into "blocks" which will be written to hard drive as soon as they are ready. Typically one will pick a block size which is small enough to be acquired quickly, and set the number of blocks to be the full amount of requested data.
4. Call the instrument instance start() method. The success of this method can be checked using the check_trigger() method.

After start() has been called, the FPGA will continue running. If the unlimited option (§D.4) is false, it will automatically stop running when all of the data has been acquired. While it is running, it will write blocks to the dump_path directory as they become ready. In addition to the data blocks, one can also find a log file, which can be used to aid debugging a failure to properly launch.

This summarizes the perspective of running experiments at a low level. In practice, one can use the tools in the fpga_lib.experiment.FPGAExperiment class (§D.7) to aid running experiments and monitoring the output.

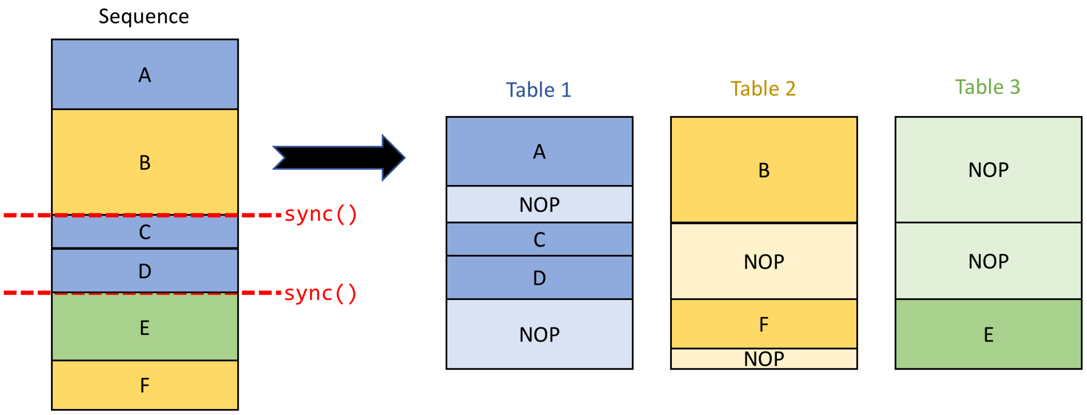


Figure D.5: **fpga_lib basic sequencing model**

D.5 The fpga_lib.dsl sequencing language

In order to alleviate the problem of managing the creation of the many tables required to operate the FPGA (§D.2.4), one can turn to the fpga_lib software package. In this package we define an *domain specific language* (DSL) that is used to specify experiment sequences. For n cards with m analog modes, we have $n(m+3)$ tables to fill out, each of which is scheduled using the format described in §D.3.1. Each of these sequences must be designed to track each other in parallel. This is a cumbersome task to do by hand, and once done, it becomes difficult to modify the sequence while preserving the timing synchronization. The approach taken by fpga_lib is to write a single, linear, sequence containing elements of all of the sequenced tables, which is compiled down into a result accepted by YngwieEncoding.

fpga_lib employs a "singleton" pattern, where each of the commands modifies an underlying "sequence" (contained in the implicit state, accessible by fpga_lib.dsl.state.get_state()) being built, in order to avoid every line containing some version of seq.append(...). After the sequence is built, it is compiled. The basic underlying method of the compilation can be seen in figure D.5. We can think of each of the sequencing tables as "bins." Every element of the unified sequence corresponds to one of these bins. When we process the unified sequence we put each item in the corresponding bin until we reach a sync() instruction. At this point we calculate the duration in each bin (since either the start of the sequence or the last sync()) ℓ_k, i.e. the sum of the length field for each instruction in the bin. We are going to "pad" each bin with blank instructions of total duration $(\max_n \ell_n) - \ell_k$. This way each bin will start the set of instructions following the sync() command simultaneously. There are four ways of padding supported

- sync(alignment='left'): Add the padding to the end of the bins (the default, if alignment is omitted)
- sync(alignment='right'): Add the padding to the beginning of the bins, just following the previous sync().
- sync(alignment='center'): Split the padding in two components, half goes to the

beginning, half to the end.

- sync(alignment='pad_inst'): Don't add a new instruction, but rather increase the duration of the final instruction. Useful for "jump tables" where the total number of instructions is critical. Must be used with caution.

D.5.1 Output

The basic way of controlling the output is to play a pulse. There are two ways of doing so, depending on whether the pulse has constant amplitude or not. For such pulses one can use the constant_pulse() directive, which requires only that one specify the length (in units of ns, which must be some multiple of 4) and amplitude. For all other pulses, one specifies the complete array of values, with 1 ns resolution (if four outputs are used, every other point is simply ignored) using the array_pulse() directive. Both accept a set of arguments (amp and phase) which can specify the value of the mixer matrix as: (§D.2.1)

$$\begin{pmatrix} \mathtt{amp}\cos(\mathtt{phase}) & \mathtt{amp}\sin(\mathtt{phase}) \\ -\mathtt{amp}\sin(\mathtt{phase}) & \mathtt{amp}\cos(\mathtt{phase}) \end{pmatrix} \tag{D.12}$$

By specifying the argument amp="dynamic", one can instead take the mixer matrix to be the last value which was loaded from the CPU outputs (§D.5.5). Additionally, the length parameter can be specified as "DYN1" to, for instance, use the length stored in the **DYN1** CPU output (§D.2.5). This can be used in either constant_pulse or array_pulse, although in the latter case, the pulse will be truncated if the dynamic length is short, and play some other pulse in wave memory if the dynamic length is too long.

There are also convenience arguments, which allow one to modify the pulse as it is stored in wave memory. The static_amp argument allows for the pulse to be scaled in memory, either in amplitude or phase. The detune argument allows for the pulse to be modulated in memory, resulting in the center frequency of the resulting waveform to be shifted by the specified amount.

In addition to the analog outputs, there are also the digital outputs, of which there are four channels. These can be controlled with the marker_pulse() directive, which accepts a channel (which of the 4 markers) and length parameter (which can be dynamic)

D.5.2 Acquisition

The acquisition periods can be scheduled in nearly an identical way to the digital marker pulses using the set_trigger_levels() command, which specifies which of the two analog inputs are acquiring and for how long. By convention we denote the first input (ADC0) as "reference" and the second input (ADC1) as "signal" and can refer to the acquisition using the convenience commands acquire_signal(), acquire_ref(), and acquire_both().[5] In addition to controlling when acquisitions happen, we can also control how they are processed

[5] Note that a single acquisition can be split across multiple instructions, commonly used if the signal acquisition is longer, e.g. acquire_both(reference_length); acquire_signal(signal_length - reference_length).

to a degree using the `estimation_params_addr` keyword argument in any of the previous commands to specify what address in the state estimator table we want to use.

While this manual level of control exists, it is unfortunately a bit cumbersome. For this reason, it is prefereable to use the higher-level `Readout` abstraction (§D.6.2). This automatically performs many of the manipulations needed to add a readout to a sequence including modifying the data-decoding procedure needed to interpret the results (§D.5.6).

D.5.3 Control flow

By default, the sequencer creates instructions which indicate that the execution should jump to the instruction which follows, incrementing the instruction address by 1. There are several ways in which we can deviate from this default behavior.

- `goto('A')`: Unconditionally jump to the location labelled as `A`
- `goto('A', 'GOF0')`: Unconditionally jump to `A` plus the offset stored in **GOF0** (§D.2.5).
- `set_int_func(signal)`: Update the signal (§D.3.2) used for jumping. This is often not needed to be called manually, if using higher-level control flow objects like `if_then_else()` or `Cond`.
- `conditional_jump('A', 'B')`: Jump to `A` if the current value of the internal function is high, otherwise jump to `B`.
- `it_then_else(signal, 'A', 'B')`: Sets the internal function to the given signal, and then performs the `conditional_jump`
- `if_then(signal, 'A')`: Same as `if_then_else` except if the signal is false, we go to the subsequent instruction.
- `Cond(signal)`: When used with a python `with` statement, the indented code is performed only when the signal is high, otherwise the block is skipped.
- CPU comparisons: We can use the CPU to do branching as well. We can use statements of the form

```
# In a with statement
with expr1 <op> expr2:
    ...

# Alternatively in other places which accept a signal
if_then_else(expr1 <op> expr2, 'A', 'B')
```

where `expr1` and `expr2` are arbitrary CPU expressions (§D.5.4) and <op> is any comparison operator, such as <, <=, ==, > or >=. This statement compiles down into three steps. First the CPU computes the quantity `expr2 - expr1`, and stores it in the TST functional unit (§D.2.5). Next it assigns the correct combination of the `r0` and `r1` registers to the internal function, and finally adds the necessary conditional jumping code. as necessary

- `subroutine`: A python decorator which causes the tables generated by the function body to be stored as a subroutine (technically, the tables will be appended to the end of the main experiment sequence). When the function is called the thread of execution will jump to the function body, and return to the subsequent instruction after finishing the function body. The purpose of this is to lessen repetition of instructions which could cause the table lengths to exceed maximum allowed values.

D.5.4 CPU

We discussed the underlying model of the Yngwie CPU in §D.2.5. We saw how complicated but powerful the programming model is. We can give up some amount of performance in order to make this functionality accessible, by using the `fpga_lib.dsl.cpu` compiler, which is an entirely separate compiler from that used for compiling the basic sequences (master, analog, digital). While not necessarily able to extract the maximum amount of performance, in the majority of the cases it is preferable to be able to quickly come up with functional, if suboptimal, programs for achieving basic computational tasks.

The `fpga_lib.dsl.cpu` compiler creates CPU programs corresponding to statements of the form *target* <<= *expression*, meaning store the computed value of the expression in the location specified by the target.[6] The target can be any of the following

- a `Register()` or `FloatRegister()` instance, i.e. `r = Register(); r <<= expr`.
- an indexed `Array()` instance, i.e. `a = Array(); a[idx] = expr`. [7] The index can be either an integer or a register.
- Any of the CPU dynamic outputs
  - `DynamicSSB[0]`
  - `DynamicLength[idx]` for `idx` in $\{0, 1, 2\}$
  - `DynamicJump[idx]` for `idx` in $\{0, 1, 2, 3\}$.
  - `DynamicMixer[idx0][idx1]` for `idx0/1` in $\{0, 1\}$.
  - `OutputRec[idx]` for `idx` in $\{0, 1, 2, 3\}$.

An expression can be any of the following forms (`expr1` and `expr2` can be any expressions themselves, making this definition recursive)

- Integer, in the range $[2^{-17}$, $2^{17} - 1]$
- Float, in the range [-2, 2]
- a `Register()` or `FloatRegister()` instance
- an indexed `Array()` instance, i.e. `a = Array(); target <<= a[expr]`

[6] The <<= operator in python technically means "in-place bitwise left shift.", but in `fpga_lib` we overwrite this meaning to refer to assignment in the CPU.

[7] For technical reasons the syntax for array assignment is slightly different, using = rather than <<=.

- `expr1` *op* `expr2` where *op* can be any of `+, -, *, &, |,` $\wedge$. Note there is no division.
- `expr >> k` for k any small integer.[8]
- `f(expr)` where `f` is an "arbitrary function" defined using the `arbitrary_function` python decorator.
- `GetSE0/1()` which returns the current value of the state estimator integration results `se0/1` (§D.2.2).

In the preceding definitions we see many references to `Register` objects. These are objects which refer to locations in the local memory of the CPU (§D.2.5). There are a total of 2048 available addresses split across the two memory blocks (the first accessible via MEM0/1 and the second via MEM2/3). When a register object is created, it's initial value can be specified as an argument.[9] When we create a register, we can specify its "datatype" as either an integer or float by using either the `Register` or `FloatRegister` commands, respectively. This is not really a property of the underlying object in the CPU memory, but rather one of how we interpret and manipulate this object. Most importantly is how we treat the object in multiplication (see the difference between MUL and MUH in §D.2.5). Note that the product of two integers is an integer, but in all other cases (float/float or float/integer) the result is a float.

The `fpga_lib.dsl.cpu` compiler works essentially via a breadth-first search of the possible methods of constructing a given expression out of the available functional units, scheduling their execution and making sure there are no collisions. This works well for most small cases, and the result is correct if the compilation succeeds, but it is not too hard to come across cases where the compiler fails to find an implementation. In this case, one can assist the compiler by breaking the expression down into sub-components by assigning intermediate values to registers.

D.5.5 Sweeping parameters dynamically

One of the most important tools in building experiments is the notion of "scanning" a parameter and seing how the result varies. This can be implemented on the FPGA in several ways. The most obvious way is to produce a produce a program in which each step is explicitly written in the sequence. This is the behavior produced when a python `for` loop is invoked, as the subsequent sequence-writing code is executed repeatedly. For a loop involving n steps, each of which produces m instructions, then the total sequence length becomes $n \times m$. This method has the most flexibility, but incurs the penalties of requiring longer compilation times, and being limited in size to a number of steps which can fit within the maximum sequence length. This limit can be quickly reached if scans are nested. The tools available within the FPGA allow for scans to be programmed, rather than explicitly written. In this case, instead of using python's `for` statement, we use the python's `with` statement, which does not repeatedly call the sequence-writing code contained, but rather adds statements to the beginning and end of this code which cause the block to loop *on the FPGA*. There are several methods which can

[8] What is implemented at the hardware level is `expr >> 1` (i.e. divide by two). In order to make bigger divisions, we can repeat this many times `expr >> 3 = ((expr >> 1) >> 1) >> 1`.

[9] Note that this is the initial value at the start of a run. If many experiments are performed per run (as is typical). It will need to be re-initialized manually at the start of each experiment.

be used in this way to scan various properties. In addition to constructing the actual loops, these methods provide metadata to the decoder (§D.5.6) which allows for it to annotate the axis created in the resulting data with information about the scanned parameters.

- `Repeat(n)`: Don't sweep anything, but simply repeat the indented block a given number of times.
- `scan_register(min, max, npts) as x`: Create a register named `x` whose value will be scanned between `min` and `max` as the indented block is repeated `npts` times.
- `scan_float_register(min, max, npts) as x`: same as `scan_register`, but the `min` and `max` parameters are treated as floating point values (between -2 and 2). `x` is an instance of `FloatRegister()`.
- `scan_length(min, max, npts) as dynlen`: The result `dynlen` contains the name of one of the three CPU outputs (§D.2.5) in which dynamic lengths can be stored. This output will be scanned as the indented block is repeated.
- `scan_ssb(mode, min, max, npts)`: Repeat the contained block while sweeping the SSB frequency of the indicated mode. The frequency range accessible is $[-2^{17}, 2^{17}] \times$ `ssb_resolution`, where the resolution is set by the register value, accessible via the instrument class (§D.4). Note that, even for the largest resolution, the range is limited to about ± 131 MHz.
- `scan_amplitude(mode, min, max, npts)`: On the indicated channel, sweep the dynamic mixer matrix (§D.2.1). The contents of the mixer matrix are $\begin{bmatrix} a & 0 \\ 0 & a \end{bmatrix}$ for a swept value of a. In order to use the dynamic amplitude on a given pulse, you must specify `amp="dynamic"` while calling it.
- `scan_phase(mode, min, max, npts, rel_amp=1)`: On the indicated channel, sweep the dynamic mixer matrix (§D.2.1). The contents of the mixer matrix are `rel_amp` $\times \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$ for a swept value of θ. As with `scan_amplitude`, you must specify `amp="dynamic"` on the pulse(s) which should have a swept phase.

D.5.6 Data decoding

The FPGA produces unstructured streams of data (output to the file system as blocks of a desired size) which must be interpreted to be useful. In order to structure the data it is necessary to anticipate how much data is produced by a given experiment. When either a `Readout` object (§D.6.2) or `generate_record` is called, with the intent of producing recorded data, it modifies the `fpga_lib.dsl.results.Results` object, stored in the compiler's implicit state (accessible by `fpga_lib.dsl.state.get_state()`) by calling either the `Results.add_measurement()` or `Results.add_rrec()` method, respectively. By combining this information with information from the looping constructs (§D.5.5), the system can build a model of the structure of the resulting data. This model allows it to interpret the data streams to create structured datasets, which can be easily processed and visualized.

The structure of the resulting data can be quite complicated, especially when multiple datasets are created (requiring a more complicated restructuring than a simple array reshape), and when different data streams being accessed, each of which provide different amounts of data per invocation. While the system can handle many variations, there are a few important limitations to keep in mind.

First, data cannot be acquired during feedback. In a feedback loop, the number of measurement results cannot be determined at run-time. While it is in principle possible to build a more sophisticated system which detects feedback and devises measures to structure the data resulting from feedback measurements, the simplest approach is to omit this data from the record. Therefore, if a measurement is performed in a feedback loop, it must be called with the keyword argument `log=False`. This indicates (by making a modified entry in the state-estimator table) that the `se0/1` values generated by this readout should not be output to the data stream.

Secondly, as a result of this design, the `raw0/1`, `iq0/1` and `rel` streams cannot be accessed when measurement-based feedback is used. These streams are not filtered by the presence of the `log` flag. Therefore, if feedback occurs, there is no way to ensure that these listed data streams will have a consistent structured form.

D.6 The `fpga_lib.entities` quantum control abstractions

The objects provided by `fpga_lib.entities` are designed to group together commonly used methods together with the data required to operate them. The canonical example is the π-pulse. One does not wish to specify an array of values to provide to `array_pulse` every time. Specifically, we wish to have certain properties of this pulse be consistent across each application, yet also have these properties (the amplitude, length and detuning for instance) be easily modified without editing every program which uses the pulse. While one could simply write a function for this purpose, the entities provide the additional benefit of storing their parameters (using methods from `fpga_lib.parameters`) in a consistent way. This allows the pulse parameters to be both easily modified from the graphical interface (§D.8) as well as retrieved and stored along with experimental results.

Most of the entities described are instances of `fpga_lib.parameters.Calibratable`, which allows their properties to be both easily modifiable and have the modifications be persistent without requiring code changes. In order for this to work, after all of the instances of `Calibratable` in are created, one should call the method `Calibratable.load_all_parameters()`. In order to keep this behavior consistent, it is preferable to have all calibratable objects defined in the same location. This location is, by convention, the `init_script.py` file. Typically, one imports from this file at the beginning of all experiment files (§D.7), in order to set up the python environment for writing a sequence.

D.6.1 `Mode` objects

`Mode` objects are created to correspond to the "modes" present in the architecture of the analog output system (§D.2.1). Primarily, these objects help by removing the need to recall the translation between physical objects in the system (and the associated LO frequencies) and (card, mode) pairs specifying outputs of the FPGA. Mode objects have as internal methods many of the functions which require output pair specification, such as array_pulse, constant_pulse, scan_amplitude, and scan_ssb, among others.

In addition to the base Mode objects, there are domain specific objects such as Qubit and Cavity objects. These objects have two associated CalibratedPulseObject, one for *selective* pulses, and another for *unselective* pulses, the distinction being the length (and therefore spectral width) of the pulse. These calibrated pulses have associated properties such as sigma for the Gaussian pulse width, unit_amp for the amplitude corresponding to either a π pulse or unit displacement, and detuning. Additionally, theses objects can be used to store information about the associated mode lifetimes, which can be used in determining the appropriate amount of time required to reset the system to thermal equilibrium.

D.6.2 Readout objects

One of the most important abstractions is the Readout() object and its sub-classes. It's use is necessary in order to get the most benefit out of the data decoding system (§D.5.6). The base Readout() object has several properties which govern its behavior:

- acq_len: The amount of time (in ns) to acquire data (by convention from the AD1 input) per readout
- ref_len: The amount of time (in ns) to acquire a reference signal (by convention from the AD0 input) per readout
- thresh0/1: The threshold (in units of se0/1) on which to update the signals s0/1.
- use_envelope: If false, the integration weights (§D.2.2) are a constant function of time ("boxcar" envelope). If true, use the integration from envelope_file.
- envelope_file: The name of a file produced via numpy.savez(), which should contain entries named "envelope0/1" corresponding to the envelopes desired. This file can be created using the set_envelope() method.

When a readout instance is called, by default, the action taken is to record the se value (i.e. $\mathtt{se0} + i\mathtt{se1}$) to a dataset named "default." Different datasets can be created by passing keyword arguments of the form <dataset_name>="<stream_name>" to the readout call. The dataset name can be any string and the possible streams correspond to the sections of the input processing pipeline (§D.2.2). The shape of the resulting dataset is inferred in one of two ways. If there are multiple calls to the readout object (e.g. when called within a for loop) with the same dataset name, an axis of length equal to the number of calls is formed. Additionally, if a readout is called within a scan object (§D.5.5) then an axis is formed with length corresponding to the length of the scan.

The base Readout object performs an acquisition, but without a pulse to stimulate the readout cavity, the data acquired will be meaningless. Therefore several sub-classes exist which augment the base readout to perform a pulse as well when called. The TriggeredReadout

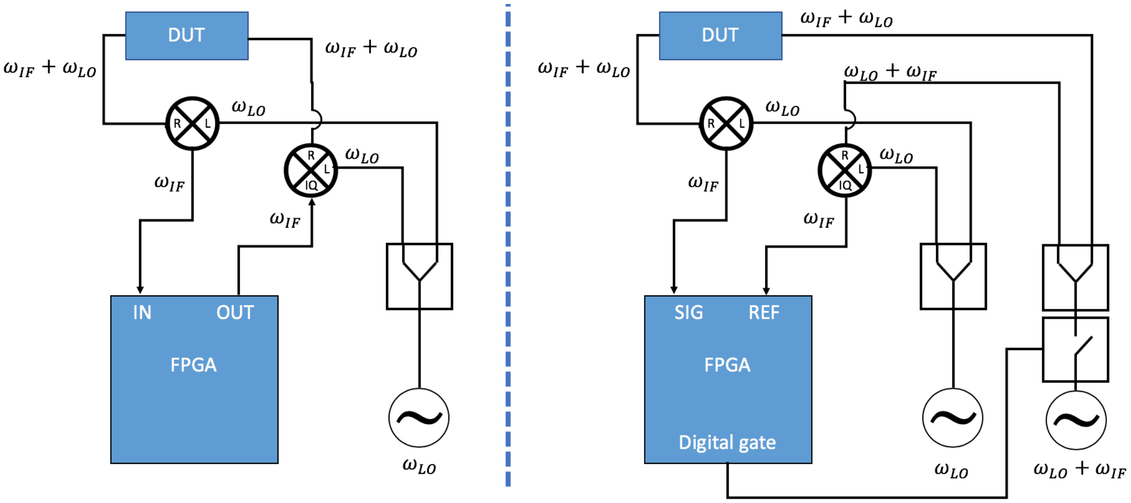


Figure D.6: **Schematic depicting two schemes for readout phase-locking**. Left, the phase is locked because the modulation and demodulation both come from the same source, namely the FPGA. In this method an output pair is consumed but gives flexibility in shaping the readout pulse. This method uses `dot_product_mode = "dup ch1"` (§D.4) as well as a `ModeReadout` object. Right, the phase is produced by the beating of two oscillators, detuned by ω_{IF}. In order to determine the phase, a reference signal must be passed to the FPGA (typically on input AD0). This method requires an additional RF source, but conserves analog outputs. To use this method, set `dot_product_mode = "dot product"` and use the `TriggeredReadout` object.

object plays a marker pulse which can be used to trigger a gated RF source. Alternatively, one can use an analog output pair (which allows for pulse shaping), by employing the `ModeReadout` object, which is itself also a subclass of `Mode`, and therefore inherits the properties discussed in §D.6.1. These two approaches require different approaches to phase locking, which are diagrammed in figure D.6.

D.7 The `fpga_lib.experiment` experiment class

While the `fpga_lib.dsl` sequencing language can be used on its own, in whatever context python programs can be run, there is some benefit gained by organizing experiment code using the `BaseExperiment` class or its subclasses, particularly `FPGAExperiment`. Using these classes facilitates the automatic saving of experiment data along with necessary meta-data such as instrument settings and experiment parameters. In addition, by using the experiment classes, one can take advantage of the `fpga_lib.gui` (§D.8), which provides automatic visualization and flexible parameter management.

While not strictly necessary, if one wishes to use the graphical interface, the preferred organization for experiment classes is to create a new file for each class, and to name the experiment object contained within the same name. For instance, a typical experiment might be declared as follows (in a file named `my_experiment.py`)

```
from init_script import *

class my_experiment(FPGAExperiment):
    def sequence(self):
        ...
```

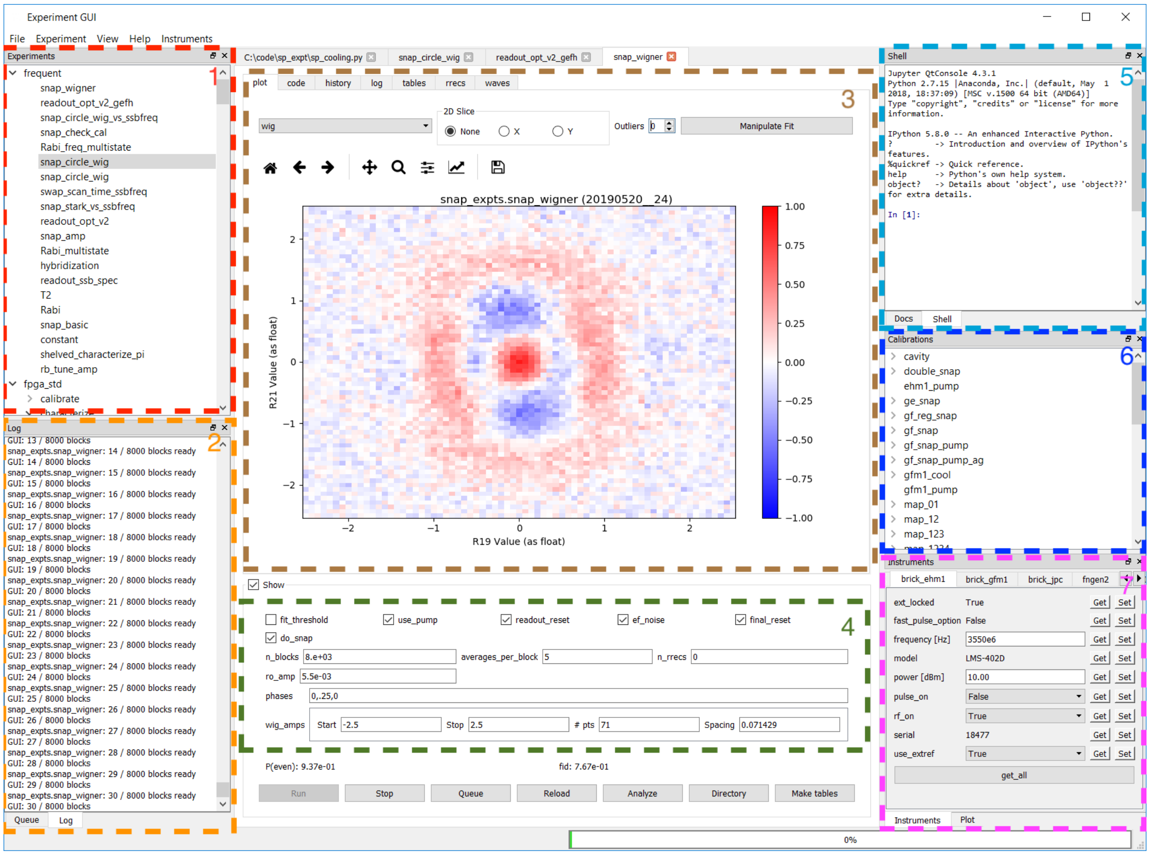
A typical experiment class will declare three things. First, a list of parameters, using methods from `fpga_lib.parameters`, such as `IntParameter()` or `StringParameter()`, which declare settings for the experiment which can be modified at the GUI level, and which are saved when the experiment is run. Next, one defines the `sequence` method. This method when called should produce the sequence using the functions defined in `fpga_lib.dsl`. Finally one can declare data-processing properties. A fit function can be declared using the top-level `fit_func` property which is in general a dictionary mapping names of datasets to names of fit functions. The available fit functions are defined in the `fpga_lib.analysis.fit` module. In addition, other analysis can be defined in the `process_data` method. Additionally, if the GUI is being used, one can define a `plot` method, which accepts a matplotlib figure, can be used to define custom plotting behavior. The advantage of writing the analysis and plotting functionality into the experiment class, is that it allows for these functions to be automatically run as soon as data becomes available, and re-run as data streams in.

D.8 The `fpga_lib.gui` graphical interface

The graphical interface consists of several components, the arrangement of which can be seen in figure D.7. Experiments can be selected from the experiment browser. The experiments shown in this window are those found from the modules specified in the list of paths `fpga_lib.config.exp_directories`. In order for experiments to be used from here, each experiment file must contain an instance of `BaseExperiment` with the same name as the file. Selecting experiments from this browser creates an instance of the experiment class, and opens an experiment widget corresponding to this instance as a tab in the central area.

The experiment widget has several components itself. It has an area for specifying parameters, where each parameter (defined using the `fpga_lib.parameters` methods) has an associated widget for data entry in this area. Below the parameter specification area is a set of buttons which control the experiment:

- Run: Compile the tables and start an acquisition, acquiring as many blocks as are indicated in the experiment `n_blocks` parameter.
- Stop: Halt the execution of the current experiment, stopping the FPGA output.
- Queue: Add the experiment with the current parameter settings to the queue. As soon as the currently running experiment is finished, the next item in the queue will be started, loading the settings which were present at the time that the experiment was queued.
- Reload: Close and open the experiment tab, allowing any changes to the experiment code to be applied. If the experiment code has changed since the experiment was loaded, this button is highlighted.



- frequent
  - snap_wigner
  - readout_opt_v2_gefh
  - snap_circle_wig_vs_ssbfreq
  - snap_check_cal
  - Rabi_freq_multistate
  - snap_circle_wig
  - snap_circle_wig
  - swap_scan_time_ssbfreq
  - snap_stark_vs_ssbfreq
  - readout_opt_v2
  - snap_amp
  - Rabi_multistate
  - hybridization
  - readout_ssb_spec
  - T2
  - Rabi
  - snap_basic
  - constant
  - shelved_characterize_pi
  - rb_tune_amp
- fpga_std
  - calibrate

```
Log
GUI: 13 / 8000 blocks
snap_expts.snap_wigner: 14 / 8000 blocks ready
GUI: 14 / 8000 blocks
snap_expts.snap_wigner: 15 / 8000 blocks ready
GUI: 15 / 8000 blocks
snap_expts.snap_wigner: 16 / 8000 blocks ready
GUI: 16 / 8000 blocks
snap_expts.snap_wigner: 17 / 8000 blocks ready
GUI: 17 / 8000 blocks
snap_expts.snap_wigner: 18 / 8000 blocks ready
GUI: 18 / 8000 blocks
snap_expts.snap_wigner: 19 / 8000 blocks ready
GUI: 19 / 8000 blocks
snap_expts.snap_wigner: 20 / 8000 blocks ready
GUI: 20 / 8000 blocks
snap_expts.snap_wigner: 21 / 8000 blocks ready
GUI: 21 / 8000 blocks
snap_expts.snap_wigner: 22 / 8000 blocks ready
GUI: 22 / 8000 blocks
snap_expts.snap_wigner: 23 / 8000 blocks ready
GUI: 23 / 8000 blocks
snap_expts.snap_wigner: 24 / 8000 blocks ready
GUI: 24 / 8000 blocks
snap_expts.snap_wigner: 25 / 8000 blocks ready
GUI: 25 / 8000 blocks
snap_expts.snap_wigner: 26 / 8000 blocks ready
GUI: 26 / 8000 blocks
snap_expts.snap_wigner: 27 / 8000 blocks ready
GUI: 27 / 8000 blocks
snap_expts.snap_wigner: 28 / 8000 blocks ready
GUI: 28 / 8000 blocks
snap_expts.snap_wigner: 29 / 8000 blocks ready
GUI: 29 / 8000 blocks
snap_expts.snap_wigner: 30 / 8000 blocks ready
GUI: 30 / 8000 blocks
```

2D Slice: ● None ○ X ○ Y Outliers 0

☑ Show

☐ fit_threshold ☑ use_pump ☑ readout_reset ☑ ef_noise ☑ final_reset
☑ do_snap
n_blocks 8.e+03 averages_per_block 5 n_rrecs 0
ro_amp 5.5e-03
phases 0,.25,0
wig_amps Start -2.5 Stop 2.5 # pts 71 Spacing 0.071429

P(even): 9.37e-01 fid: 7.67e-01

```
Jupyter QtConsole 4.3.1
Python 2.7.15 |Anaconda, Inc.| (default, May  1
2018, 18:37:09) [MSC v.1500 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more
information.

IPython 5.8.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's
features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??'
for extra details.

In [1]:
```

Calibrations
- cavity
- double_snap
- ehm1_pump
- ge_snap
- gf_reg_snap
- gf_snap
- gf_snap_pump
- gf_snap_pump_ag
- gfm1_cool
- gfm1_pump
- map_01
- map_12
- map_123

Instruments: brick_ehm1 | brick_gfm1 | brick_jpc | fngen2

ext_locked	True	Get	Set
fast_pulse_option	False	Get	Set
frequency [Hz]	3550e6	Get	Set
model	LMS-402D	Get	Set
power [dBm]	10.00	Get	Set
pulse_on	False	Get	Set
rf_on	True	Get	Set
serial	18477	Get	Set
use_extref	True	Get	Set

Figure D.7: **`fpga_lib.gui` experiment running interface**. The sections of the interface are divided as follows. (1) the experiment browser, listing the available experiment types. (2) The log, displaying information about the running experiment. (3) The current experiment's display widget, currently on the "plot" tab, which displays the most recently created dataset. There are several other tabs which display other information about the experiment. (4) The parameter input area for the current experiment. Below this is the buttons used to perform various actions, such as starting, stopping, and analyzing the experiment. (5) A python interpreter, which has access to the currently running experiment data, allowing for quick numerical investigations. (6) The "calibrations" editor, which allows the settings on any `Calibratable` instance to have its parameters modified (e.g. pulse unit amplitudes). (7) The instruments widget, which is connected to the `instrumentsserver` instance which maintains access to all connected instruments, e.g. RF sources and `Yngwie_FPGA` instrument objects.

- Analyze: Run the experiment `analyze()` method, which in turn, fits the data, and runs the `process_data()` method.
- Directory: Open the experiment's data directory in a file browser
- Make Tables: Compile the experiment tables *without* running the FPGA.

Above the experiment's parameter input area is a set of widgets accessible via a tabbed interface.

- plot: Browse and visualize the currently accessible datasets. Can create line plots and image plots, automatically label and annotate axes, plot data of higher dimension by adding sliders, plot fit results along with data, among other features.
- code: A full featured python code editor for the code for the selected experiment. After saving, the experiment can be reloaded to apply the given changes.
- history: A browser for historical data created by this experiment. Data is grouped in files by day, with different runs per day corresponding to numbered datasets within the file. A snapshot of the experiment plot widget at the time of experiment completion is used as a thumbnail for result in the history browser. Once selected, historical datasets can be visualized, properties such as parameters and calibrations can be viewed, and the corresponding calibrations and instrument settings can be loaded.
- log: A saved log of the experiment (text) output from all runs of the experiment.
- tables: A set of widgets for reading the compiled tables (§D.2.4) specifically, the master, analog, digital, and CPU tables.
- rrecs: A widget which allows generated result records to be browsed. When used in combination with `rrec_generate` $= 2^{13}$, allows for a sort of debugger view of the FPGA, in which the state of the system can be visualized step by step.
- waves: A widget which visualizes the contents of wave memory, showing the pulses which can be played by the various analog output modes as well as their associated addresses.

In addition to the main experiment widget, there are several auxiliary widgets not associated with any experiment. A widget with a python interpreter is provided. In this interpreter, one can run the command `exp()` to get a handle on the currently displayed experiment instance. A "calibrations" widget provides access to all of the parameters associated with instances of subclasses of `fpga_lib.parameters.Calibratable`, which are typically defined in the `init_script.py` file (§D.6. Whenever modifications are made to the parameters here, the values are cached to disk, allowing them to be re-loaded to the changed value upon re-initializing the system. Finally there is an instruments widget, which provides access to the instruments hosted by the `instrumentsserver`. The server needs to be separately created and running before the experiment GUI is launched. It will attempt to connect to a server running at the address specified by `fpga_lib.config.instrumentserver_addr`. If it cannot be found, it will attempt to create an instance, however this is only reasonable if the instruments are connected to the computer running the experiment GUI.

Appendix E

The `pygrape` optimal control package

The `pygrape` package was developed in parallel with the experiment discussed in chapter 6. While there are many other optimal control software packages available including SPINACH[1], DYNAMO[2] and QuTiP[3], working with `pygrape` allowed us to easily develop the types of modifications discussed in chapter 4, and to tailor the application to our use case. Therefore, the resulting package tries to be flexible, but its flexibility is tailored towards the anticipated application of controlling cavities coupled to transmons.

E.1 Overview

The `pygrape` package provides one main point of entry for running GRAPE-style pulse optimizations, namely the `run_grape` command. A typical invocation looks like this:

```
result = run_grape(init_controls, setups, penalty_fns=None, reporter_fns=None,
    outdir=None, dt=1, init_aux_params=None, shape_sigma=10,
    n_proc=1, discrepancy_penalty=0, dtype=None,
    save_data=0, term_fid=None, freq_range=None, bound_amp=None, **opts):
```

The returned result object is a report data dictionary (§E.5). The following describes the main available parameters.

- `init_controls`: A (real) array which has shape `n_ctrls`×`pulse_len` which is the initial guess for the pulse. Can be created using `pygrape.preparations.random_waves()` if a suitable initial guess is not known.
- `setups`: A list of setup objects (§E.2) which describe the quantum optimization problem.
- `penalty_fns`: Either `None` or a list of penalty functions (§E.3) which constrain the types of admissible pulses

[1] http://spindynamics.org/Spinach.php

[2] https://github.com/shaimach/Dynamo

[3] http://qutip.org/docs/4.0.2/guide/guide-control.html

- reporter_fns: Either None or a list of reporter functions (§E.4) which can be used to monitor the progress of ongoing optimizations
- outdir: The output directory where results will be stored. Per-run results are stored in a numbered sub-directory of outdir. If None, use the current directory.
- dt: The time between points in the control array. The total time of the pulse is dt × pulse_len.
- init_aux_params: The initial values for auxiliary parameters, i.e. parameters to be optimized which are not time-dependent controls. The current main application of auxiliary parameters are the coefficients used for gauge degrees of freedom (§E.2.1).
- shape_sigma: In order to encourage smooth start and stop to the pulse, the simulated pulse is actually the product of the control parameters and a "shape function." The shape function is defined to be $f(t) = 1 - e^{-t/\sigma} - e^{-(t_{\mathsf{max}}-t)/\sigma}$ where $\sigma = \texttt{shape\_sigma}$. If shape_sigma is 0, then $f(t) = 1$.
- n_proc: If multiple setups are being optimized against, the work of simulating each can be distributed across multiple processes. n_proc is an integer specifying how many processes to use.
- discrepancy_penalty: If multiple setups are being used, an extra term to the cost function can be added which penalizes any (pair-wise) difference between the reported fidelities of the various setups. The penalty is of the form discrepancy_penalty × $\sum_i \sum j > i|\mathcal{F}_i - \mathcal{F}_j|^2$, with $\mathcal{F}_i$ the fidelity of the i-th setup.
- save_data: Specify a period (once every save_data iterations) for saving report data (§E.5) to a file report_data.h5 in the output directory.
- term_fid: Specifies a fidelity at which to terminate the optimization.
- freq_range: If not None, then must be a tuple $(\texttt{min\_freq}, \texttt{max\_freq})$ which specifies the range of acceptable frequencies for the pulse. This works by effectively re-parameterizing the pulse in the frequency domain (see 4.4.2).
- bound_amp: If not None, put a hard limit the amplitude of the controls being optimized to be between (-bound_amp,+bound_amp). Incompatible with the freq_range option.
- **opts: Further keyword arguments are passed to the scipy.optimize.minimize, specifically those options available with the L-BFGS-B solver.[4] The most important option is gtol which specifies when the gradient is small enough to stop optimizing. Setting this value smaller causes the algorithm to "try harder" before giving up.

E.2 Setups

The first task is to describe the "quantum" problem being solved. We associate to each quantum problem an object called a Setup(). Each setup contains two main components: a

[4] https://docs.scipy.org/doc/scipy/reference/optimize.minimize-lbfgsb.html

description of the control system, and a description of fidelity metric which we wish to optimize. For each setup we describe the control system in terms of its "drift" Hamiltonian `H0` and a list of "control" Hamiltonians `Hcs`. Whenever we describe quantum objects (states, operators, Hamiltonians, etc.) the interface should accept either a `numpy` array or a `qutip.Qobj` object.

In the `StateTransferSetup()` we seek to optimize the fidelity of several simultaneous state transfers. We describe the initial states (`inits`) and the target final states (`finals`). Depending on the value of the `coherent` flag, we use either the metric from equation 4.15 (`coherent=True`) or equation 4.16 (`coherent=False`).

In the case that evolution of the entire Hilbert space is to be specified, one can instead use the `UnitarySetup()` which requires no input states, but only the final target unitary matrix.

Finally, if one wishes to work in an open-systems context, using the Lindblad-Markov master equation, instead of the Schrödinger equation, then one can employ the `LindbladSetup()`. In addition to the arguments provided to `StateTransferSetup()`, one can also provide `c_ops`, a list of "collapse" operators which define the Markovian non-unitary evolution. Going from a `StateTransferSetup` to a `UnitarySetup` squares the dimension of the system of equations, and therefore is much slower.

E.2.1 Gauge Operators

Each of the setups also has the ability to take in a list of operators (gauge_ops) describing "gauge degrees of freedom." These operators are described mathematically in section 4.3.3, specifically equation 4.30. For each gauge operator in the list, a new parameter is added, in addition to the `n_ctrls` × `pulse_len` parameters associated with the controls themselves. These controls are added to the "auxiliary parameters" (`aux_params`). The algorithm needs to compute the gradient with respect to these new parameters as well. Provide an initial guess for these parameters using the `init_aux_params` argument to `run_grape`.

E.3 Constraining the result with penalties

A *penalty*, as accepted by the `penalty_fns` argument to `run_grape`, can be any function which accepts a controls array on its input (of shape `n_ctrls` × `pulse_len`) and returns a tuple consisting of a cost and a gradient, where the gradient has the same shape as the input.

Several penalties are defined in `pygrape.penalties`. Many of these take the form of a function which returns a penalty function, specializing it with appropriate parameters.

- `make_lin_amp_cost(reg, iq_pairs=False)`: Makes a linear penalty (equation 4.36) with weighting coefficient `reg`. If `iq_pairs` is enabled, then the controls will be interpreted as $\mathtt{n\_ctrls}/2$ complex-valued controls, by identifying the controls as being alternating real and imaginary components, i.e. $[\mathrm{Re}\,\epsilon_1(t), \mathrm{Im}\,\epsilon_1(t), \ldots, \mathrm{Re}\,\epsilon_{\mathtt{n\_ctrls}/2}, \mathrm{Im}\,\epsilon_{\mathtt{n\_ctrls}/2}]$.
- `make_amp_cost(reg, thresh, iq_pairs=False)`: Makes a *non-linear* amplitude penalty in the form of equation 4.35.

- `make_lin_deriv_cost(reg, iq_pairs=False)`: Makes a linear derivative penalty in the form of equation 4.40.

- `make_deriv_cost(reg, thresh)`: Make a nonlinear derivative penalty in the form of equation 4.41.

- `make_l1_penalty(reg, alpha)`: Makes a sparsity-preferring penalty function, using an approximation to the L_1 norm, $f(\epsilon) = \sum_k |\epsilon_k|$. In the vicinity of $\epsilon > 1/\alpha$, the absolute value is replaced by a smooth continuation, so as to preserve the validity of the cost function's Hessian.

- `make_cplx_l1_penalty`: A version of `make_l1_penalty` which treats the controls as pairs corresponding to complex-valued controls.

E.4 Monitoring the progress with reporters

While not strictly contributing to the final result, the ability to monitor the progress of ongoing optimizations is a crucial step in solving convergence problems, saving time from being wasted, and generally understanding what is happening within the GRAPE system. For this purpose, one can create instances of `pygrape.reporters.Reporter`, and pass them to the `reporter_fns` argument of `run_grape`. Each instance can take a `spacing` argument on creation which defines how often (in units of iterations) the reporter should be called, e.g. `spacing=5` dictates that the reporter will be called once per five iterations.

- `print_costs()`: Print the fidelity associated with each setup as well as the cost associated with each penalty function

- `save_waves(wave_names)`: Save the controls into a file (named `waves.npz` by default) which can be loaded by `numpy.load`. The list of strings `wave_names` dictates the name associated with each control in the file.

- `plot_waves(wave_names, iq_pairs=False)`: Plot the controls using `matplotlib`. A separate axis is used for each control array. If `iq_pairs` is enabled, treat the controls as pairs of values corresponding to complex-valued controls, and plot the complex pair on a single axis.

- `plot_fidelity`: Create a plot of the fidelity as a function of the number of iterations.

- `verify_from_setup`: Simulate the system using a different setup. This is useful to, for instance, check that the fidelity remains unchanged by increasing the dimension of a cavity subsystem.

- `verify_master_equation`: Simulate the system using the `qutip.mesolve` master equation solver. One can provide a list of collapse operators (`c_ops`) to account for decoherence.

E.5 Report data

The report data, which is returned from a call to run_grape is a dictionary with several entries containing information about the preceding optimization. In addition, this data can be periodically saved to a file using the save_data option to run_grape. The report data contains the following entries:

- n_iter: Number of optimization iterations which have passed.
- sim_controls: The "simulated" controls, which are the product of the raw_controls and shape_func.
- aux_params: The auxiliary parameters, e.g. gauge degree of freedom coefficients.
- dt: The time step
- raw_controls: The controls parameters without the shape_func applied.
- shape_func: The tapering function dictated by the shape_sigma parameter to run_grape.
- setups: The list of setups being optimized
- props: The final (propagated initial) states for each setup.
- fids: The fidelity associated with each setup.
- fids_hist: The fidelity history for each setup, i.e. as a function of iteration number.
- fid_grads: The gradient of the fidelity with respect to the controls.
- aux_fid_grads: The gradient of the fidelity with respect to the auxiliary parameters.
- pen_costs: The cost associated with each penalty
- pen_grads: The gradient of each penalty with respect to the controls.
- pen_hist: The history of each penalty cost as a function of iteration number.
- tot_cost: The total cost (infidelity plus penalties)
- tot_grad: The gradient of the total cost with respect to the controls.
- outdir: The output directory

Appendix F

Optimal control pulse generation script

```python
from pygrape import run_grape
from pygrape.setups import StateTransferSetup, SubspaceSetup
from pygrape.penalties import make_amp_cost, make_lin_deriv_cost, make_tail_cost
from pygrape.preparations import make_hmt, make_target, random_waves, make_ops
from pygrape.preparations import unitary_to_states
from pygrape.reporters import *
from pygrape.grape import get_impulse_response
import numpy as np
import qutip
import sys
import os
import argparse
from glob import glob
from scipy.linalg import expm, eigh


def fourcat(nc, nq, iq, s):
    h = (qutip.coherent(nc, ALPHA) + PARITY * qutip.coherent(nc, -ALPHA)).unit()
    v = (qutip.coherent(nc, 1j*ALPHA) + PARITY * qutip.coherent(nc, -1j*ALPHA)).unit()
    if s == 0:
        c_state = (h + v).unit()
    elif s == 1:
        c_state = (h - v).unit()
    return np.squeeze(qutip.tensor(qutip.basis(nq, iq), c_state).full())


PLEN = 550
OPNAME = 'H' # Options are id,x,y,x2,y2,x2m,y2m,H,z,t,map_4cat,unmap_4cat
ALPHA = np.sqrt(3)
NC = 22
```

```
PARITY = +1
NQ = 3
T1q = 110e3
T2eq = 40e3
CHI = -2.199e-3
CHI_PRIME = -17.7e-6
KERR = -3.43e-6
ANHARM = -236.2e-3
QDRIVE = 14.78e-3
CDRIVE = 37.6e-3
MAX_AMP = 0.32
MAX_ITER = 1200
TERM_FID = 0.999
MAX_FREQ = 0.04
CNORM = .002
QNORM = .01
T1cav = 2.8e6
outdir = None


def make_setup(nc, nq, chi=CHI, delta_q=0, delta_c=0):
    H0, Hcs = make_hmt(
        nc, nq, chi, CHI_PRIME, KERR, ANHARM, QDRIVE, CDRIVE,
        delta_q=delta_q, delta_c=delta_c
    )

    if OPNAME == 'map_4cat':
        inits = [np.eye(nc*nq)[0],np.eye(nc*nq)[nc]]
        finals = [fourcat(nc, nq, 0, 0), fourcat(nc, nq, 0, 1)]
    elif OPNAME == 'unmap_4cat':
        finals = [np.eye(nc*nq)[0],np.eye(nc*nq)[nc]]
        inits = [fourcat(nc, nq, 0, 0), fourcat(nc, nq, 0, 1)]

    # Register operations
    else:
        if OPNAME == 'id':
            op = qutip.qeye(2)
        elif OPNAME == 'x':
            op = qutip.sigmax()
        elif OPNAME == 'y':
            op = qutip.sigmay()
        elif OPNAME == 'H':
            op = qutip.hadamard_transform()
        elif OPNAME == 'x2':
            op = (1j*np.pi/4*qutip.sigmax()).expm()
        elif OPNAME == 'x2m':
            op = (-1j*np.pi/4*qutip.sigmax()).expm()
        elif OPNAME == 'y2':
            op = (1j*np.pi/4*qutip.sigmay()).expm()
        elif OPNAME == 'y2m':
            op = (-1j*np.pi/4*qutip.sigmay()).expm()
        elif OPNAME == 'z':
            op = qutip.sigmaz()
        elif OPNAME == 't':
            op = (1j*np.pi/8*qutip.sigmaz()).expm()
        else:
            raise ValueError('Unknown pulse: ' + opname)

        inits = np.array([
            fourcat(nc, nq, 0, 0),
            fourcat(nc, nq, 0, 1),
        ])
        finals = op.full().dot(inits)

    setup = StateTransferSetup(H0, Hcs, inits, finals, coherent=True, use_taylor=True)
    setup.taylor_order = 20
    return setup


setups = [make_setup(NC, NQ), make_setup(NC+2, NQ)]
penalties = [
    make_amp_cost(1e-4, np.sqrt(.5)*MAX_AMP, iq_pairs=True),
    make_lin_deriv_cost(0.2),
]
wave_names = 'qI,qQ,cI,cQ'.split(',')
reporters = [
    print_costs(),
    save_waves(wave_names, 10),
    plot_waves(wave_names, 10, iq_pairs=True),
    plot_penalties(10),
    plot_fidelity(10),
    plot_states(5),
    verify_from_setup(make_setup(NC+3, NQ), 30)
]
init_ctrls = random_waves(4, PLEN, npoints=15)
init_ctrls[2:] *= CNORM
init_ctrls[:2] *= QNORM


ret = run_grape(
    init_ctrls, setups, penalties, reporters, outdir,
    iprint=0, maxcor=20, maxiter=MAX_ITER, n_proc=len(setups), dt=2, save_data=5,
    check_grad=25, term_fid=TERM_FID, discrepancy_penalty=1e4,
    freq_range=(-MAX_FREQ, MAX_FREQ)
)
```

Appendix G

Optimal Wigner displacement generation script

```
%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import numpy as np
from scipy.special import genlaguerre
from math import sqrt, factorial
from numpy.linalg import cond, svd
from scipy.optimize import fmin, check_grad, minimize
from IPython.display import display, clear_output
import time

# Number of photons
FD = 14
# Number of displacements
n_disps = FD**2 + 30

def wigner_mat_and_grad(disps, FD):
    ND = len(disps)
    wig_tens = np.zeros((ND, FD, FD), dtype=np.complex)
    grad_mat_r = np.zeros((ND, FD, FD), dtype=np.complex)
    grad_mat_i = np.zeros((ND, FD, FD), dtype=np.complex)

    B = 4 * abs(disps)**2
    pf = (2 / np.pi) * np.exp(−B/2)
    for m in range(FD):
        x = pf * np.real((−1) ** m * genlaguerre(m, 0)(B))
        term_r = −4 * disps.real * x
        term_i = −4 * disps.imag * x

        if m > 0:
            y = 8 * pf * (-1)**(m-1) * genlaguerre(m-1, 1)(B)
            term_r += disps.real * y
            term_i += disps.imag * y
        wig_tens[:, m, m] = x
        grad_mat_r[:, m, m] = term_r
        grad_mat_i[:, m, m] = term_i

        for n in range(m+1, FD):
            pf_nm = sqrt(factorial(m)/float(factorial(n)))
            x = pf * pf_nm * (-1)**m * 2 * (2*disps)**(n-m-1) * genlaguerre(m, n-m)(B)
            term_r = ((n - m) - 4*disps.real*disps) * x
            term_i = (1j * (n - m) - 4*disps.imag*disps) * x
            if m > 0:
                y = 8 * pf * pf_nm * (-1)**(m-1) * (2*disps)**(n-m) *\
                    genlaguerre(m-1, n-m+1)(B)
                term_r += disps.real * y
                term_i += disps.imag * y
            wig_tens[:, m, n] = disps * x
            wig_tens[:, n, m] = (disps * x).conj()
            grad_mat_r[:, m, n] = term_r
            grad_mat_r[:, n, m] = term_r.conjugate()
            grad_mat_i[:, m, n] = term_i
            grad_mat_i[:, n, m] = term_i.conjugate()
    return (
        wig_tens.reshape((ND, FD**2)),
        grad_mat_r.reshape((ND, FD**2)),
        grad_mat_i.reshape((ND, FD**2)
    )

def cost_and_grad(r_disps):
    N = len(r_disps)
    c_disps = r_disps[:N/2] + 1j*r_disps[N/2:]
    M, dM_rs, dM_is = wigner_mat_and_grad(c_disps, FD)
    U, S, Vd = svd(M)
    NS = len(Vd)
    cn = S[0] / S[-1]
    dS_r = np.einsum('ij,jk,ki->ij', U.conj().T[:NS], dM_rs, Vd.conj().T).real
    dS_i = np.einsum('ij,jk,ki->ij', U.conj().T[:NS], dM_is, Vd.conj().T).real
    grad_cn_r = (dS_r[0]*S[-1] - S[0]*dS_r[-1]) / (S[-1]**2)
    grad_cn_i = (dS_i[0]*S[-1] - S[0]*dS_i[-1]) / (S[-1]**2)
    return cn, np.concatenate((grad_cn_r, grad_cn_i))

best_cost = float('inf')
f, ax = plt.subplots(figsize=(5, 5))
def wrap_cost(disps):
    global best_cost
    cost, grad = cost_and_grad(disps)
    best_cost = min(cost, best_cost)
    ax.clear()
    ax.plot(disps[:n_disps], disps[n_disps:], 'k.')
    ax.set_title('Condition Number = %.1f' % (cost,))
    clear_output(wait=True)
    display(f)
    #print '\r%s (%s)' % (cost, best_cost),
    return cost, grad
init_disps = np.random.normal(0, 1, 2*n_disps)
init_disps[0] = init_disps[n_disps] = 0
ret = minimize(wrap_cost, init_disps, method='L-BFGS-B', jac=True, options=dict(ftol=1e-6))
print ret.message
new_disps = ret.x[:n_disps] + 1j*ret.x[n_disps:]
new_disps = np.concatenate(([0], new_disps))
```