Bosonic State Preparation Through Coupling With Semicon-spin Qubit Using Numerically Optimized Pulse Sequences

Final Year Project Report

Lin Zhonglin A0222183N

Supervisor: Associate Professor Ng Huikhoon

Co-supervisor: Dr Koh Teck Seng

Department of Physics

National University of Singapore

Apr 2024

Abstract

Quantum computing leverages quantum mechanics principles, using qubits that can exist in multiple states simultaneously, enabling parallel processing. Various physical implementations such as superconducting qubits, trapped ions, and quantum dots have emerged as possible quantum computing platforms. Besides traditional discrete quantum variable quantum computing where physical qubits represent logical qubits, an emerging continuous variable quantum computing uses usually infinite-dimensional quantum systems (like cavity states) to encode logical qubits, offering a different approach to error correction. [**RevModPhys.77.513**] One common platform for continuous variable quantum computing is cavity coupled to an auxiliary qubit for control. This is often termed "bosonic qubits in circuit QED".[**Joshi˙2021**] In this area of quantum computing, the common physical platform is cavity coupled to a transmon. In this report we consider a possible alternative, cavity coupled to a Double Quantum Dot (DQD). [**D˙Anjou˙2019**] Moreover, in order to do quantum computation on a physical system, being able to do state preparation on the physical system is crucial, as it initializes qubits for computation, with prepared state fidelity impacting the overall performance of quantum algorithms. [**Nielsen2010**] In this report I focus on simultaneous control pulse optimization for state preparation of a cavity cat state on a cavity-DQD physical system (as opposed to driving cavity and qubit sequentially). Pulse optimization for the overall composit state was successfully run for effective Hamiltonian (in restricted regime) of the system to arbitrarily high fidelity. Though intended, optimization for the cavity state alone was not achieved; cheking the fidelity of the optimized pulses when run using the full physical Hamiltonian couldn't be attained in reasonable computation time; optimization under noise was not achieved. Details on unreached goals and challenges faced can be seen in chapter **??**.

Acknowledgements

I would like to thank Associate Professor Ng Huikhoon and Dr Koh Tech Seng for this opportunity to gain invaluable experience in a quantum computation research group and also for their patience with me every Monday for the past eight months. I would also like to thank seniors Siyan and Chiyuan who have given me incredible support and guidance as I struggled through the project that is in a field very new to me. I would also like to thank all the other seniors in Huikhoon's research group, especially Lukas who gave me many programming supports. This project would not have been possible without the superb people mentioned.

Contents

Chapter 1

Introduction

Quantum computing has emerged as a paradigm with the potential to solve problems that are intractable for classical computers. By leveraging the principles of quantum mechanics, such as superposition and entanglement, quantum computers can perform certain computations exponentially faster than their classical counterparts [**Nielsen2010**]. This has led to significant interest in developing practical quantum computing systems and algorithms for various applications, ranging from cryptography and optimization to quantum simulation and machine learning [**Preskill2018**].

At the heart of quantum computing lies the qubit, the fundamental unit of quantum information. Unlike classical bits, which can only be in one of two states (0 or 1), qubits can exist in a superposition of multiple states simultaneously. This property, along with the ability to create entanglement between qubits, enables quantum computers to explore vast computational spaces in parallel [**Nielsen2010**]. However, realizing robust and scalable qubits is a significant challenge due to their susceptibility to noise and decoherence.

Various physical implementations of qubits have been proposed and experimentally demonstrated, each with its own advantages and limitations. Superconducting qubits, which utilize the collective behavior of cooper pairs in superconducting circuits, have emerged as a leading platform due to their strong nonlinearity, fast gate operations, and compatibility with existing microwave technology [**Kjaergaard2020**]. Trapped ion qubits, on the other hand, leverage the electronic states of ions confined in electromagnetic traps, offering long coherence times and high-fidelity gate operations [**Bruzewicz2019**]. Quantum dots, which are nanoscale structures that confine electrons or holes, have also shown promise as a scalable and easily controllable qubit platform [**Chatterjee2021**].

In addition to the traditional approach of encoding qubits in discrete two-level systems, there

has been growing interest in continuous variable quantum computing (CVQC). CVQC utilizes the infinite-dimensional Hilbert space of bosonic modes, such as those of electromagnetic fields or mechanical oscillators, to encode and process quantum information [**Braunstein2005**]. Such a scheme is desirable because it enables more elaborate encoding that encodes a single logical qubit on multiple physical qubits while still requiring control of a single physical system. One of the most prominent physical implementations of CVQC is the cavity-transmon system, where a superconducting transmon qubit is coupled to a microwave cavity. This hybrid approach combines the long coherence times of cavities with the strong nonlinearity and fast control of transmon qubits, enabling a range of quantum operations and error correction schemes [**Ofek2016**].

However, the cavity-transmon system also has its limitations, such as its relatively short coherence times of transmon qubits compared to other platforms. An alternative approach is to couple the cavity to a double quantum dot (DQD) instead of a transmon. The cavity-DQD system has the potential to combine the advantages of semiconductor quantum dots, such as long coherence times and compatibility with existing semiconductor technology, with the benefits of CVQC [**Mi2018**]. This hybrid platform opens up new possibilities for quantum information processing and provides a promising avenue for scalable quantum computing [**Burkard2020**]. (chapter **??**)

One of the key challenges in quantum computing is state preparation, which involves initializing the qubits in a desired quantum state before performing any computation. The state fidelity and efficiency of state preparation directly impact the overall performance of quantum algorithms and the reliability of quantum error correction schemes [**Shen2017**]. In CVQC, state preparation often involves creating non-classical states, such as Schrödinger cat states or squeezed states, which are essential for various logical qubit encoding schemes [**Joshi˙2021**]. However, preparing these states with high fidelity is a non-trivial task due to the presence of noise and the limitations of control techniques.

Pulse optimization has emerged as a powerful method for state preparation in quantum systems. By carefully designing the time-dependent control pulses applied to the qubits or cavities, one can steer the system towards the desired target state while minimizing the effects of noise and decoherence [**Khaneja2005**]. Quantum optimal control theory provides a framework for finding the optimal control pulses that maximize the fidelity of state preparation or gate operations [**Glaser2015**]. It has been shown by [**Krastanov2015**] which gave a construction for how to achieve arbitrary operations on dispersively coupled cQED systems (which includes my system of interest) using a set of two operations: displacements and selective number-dependent arbitrary phase (SNAP) operations. Hence, this project focuses on achieving cavity state preparation by first optimizing for selective qubit rotations (this report focuses on qubit flipping). My research group colleage Chiyuan has carried out a state preparation on the same physical system by first analytically solving then driving the caivty and qubit sequentially. Such an analytical is a good proof of concept. However, it is not necessarily the most gate time efficient approach when it comes to applying to actual physical state preparation as it is not utilizing the full optimization

landscape, but only a special solution. Hence, this project which allows for simultaneous driving of the cavity and qubit serves as an extension of Chiyuan's work. (chapter **??** and **??**)

In this project, we focus on exploring quantum optimal control techniques for state preparation in the cavity-DQD system. Our methodology involves first deriving the effective Hamiltonian that describes the dynamics of the coupled cavity-DQD system in the dispersive regime. We then formulate the state preparation problem as an optimization task, where the goal is to find the control pulses that maximize the fidelity between the evolved state and the target state. We employ various optimization algorithms, such as GRAPE (GRadient Ascent Pulse Engineering) and CRAB (Chopped RAndom Basis), to numerically search for the optimal pulses [**KHANEJA2005296**; **PhysRevLett.106.190501**]. (chapter **??** and **??**)

Here is a summary of optimization results:

where, initial and target states are given by:

$$
\begin{aligned}
&\text{vac2coherent}\\
&\qquad \psi_{i1} = |0\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}}\\
&\qquad \psi_{f1} = |\alpha\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}}\\
&\text{unselective}\\
&\qquad \psi_{i2} = \frac{1}{\sqrt{2}}(|0\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}} + |1\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}})\\
&\qquad \psi_{f2} = \frac{1}{\sqrt{2}}(|0\rangle_{\text{cavity}} \otimes |1\rangle_{\text{qubit}} + |1\rangle_{\text{cavity}} \otimes |1\rangle_{\text{qubit}})\\
&\text{selective}\\
&\qquad \psi_{i3} = \frac{1}{\sqrt{2}}(|0\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}} + |1\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}})\\
&\qquad \psi_{f3} = \frac{1}{\sqrt{2}}(|0\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}} + |1\rangle_{\text{cavity}} \otimes |1\rangle_{\text{qubit}})\\
&\text{vac2cat}\\
&\qquad \psi_{i1} = |0\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}}\\
&\qquad \psi_{f1} = |\alpha\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}} + |-\alpha\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}}
\end{aligned}
$$

The project aimed to optimize control pulses for state preparation of a cavity vacuum state to cavity cat state in a cavity-coupled double quantum dot system. The original goal was to first optimize for high state fidelity using an effective Hamiltonian of the system and make sure that the optimized pulses work well when using the original full Hamiltonian. If this is to be achieved,

name	physical system	initial state	target state	ops algo	N(Fock space trunca-tion)	time discre--tisation	pulse length (ns)	final fidelity (simulation)
single_qubit	single qubit	ground state	excited state	GRAPE	NIL	1000	18	0.9999
vaccuum2coherent	cavity coupled to a qubit	ψ_{i1}	ψ_{f1}	GRAPE	5	100	1	0.9999
unselective_spin_flip	cavity coupled to a qubit	ψ_{i2}	ψ_{f2}	GRAPE	16	500	1	0.9995
selective_spin_flip	cavity coupled to a qubit	ψ_{i3}	ψ_{f3}	GRAPE	16	100000	500	0.9994
selective_spin_flip _constraints	cavity coupled to a qubit	ψ_{i3}	ψ_{f3}	GRAPE	16	100000	500	0.9992
vacuum2cat	cavity coupled to a qubit	ψ_{i4}	ψ_{f4}	GRAPE	16	100000	5000	0.9529
vacuum2cat	cavity coupled to a qubit	ψ_{i4}	ψ_{f4}	KROTOV	16	100000	5000	0.9995

other potential goals include optimization under physical pulse constraints, optimization for a open quantum system (considering noise), optimization for other state optimizations, optimization for gates. However, several challenges were encountered that prevented reaching the initial goals within the given timeframe. Here I list the goals not achieved and outline the challenges faced. Firstly, QuTip's implementation made it difficult to optimize the fidelity of the cavity state alone by partial tracing out the qubit state, requiring a significant rewrite of the optimization code. Although the Krotov package seems to allow for customization of fidelity, this was not attempted due to time constraints. Secondly, the project did not fully consider all physical constraints on the control pulses, with only pulse amplitude constraints being implemented. The CRAB algorithm could be used to incorporate more constraints and optimize functional basis coefficients for smoother pulses, but requires more trial and error. This was also not attempted due to time constraints. Thirdly, the native oscillation frequency of the cavity caused fast oscillations in the optimized pulses, which could potentially be addressed but issues also came up. See chapter **??** for more details. Though it was unfortunate that many of the original goals were not achieved due to time constraints, all of these challenges seem to be technical difficulties. With more time, these technical programming issues could be resolved.

Chapter 2

Physical System and Hamiltonians: DQD-cavity

The growing interest in integrating cavity dynamics with double quantum dot (DQD) configurations is a significant development in the field of quantum computing. This interest stems from recent successes in achieving strong coupling between semiconductor spin qubits and superconducting microwave resonators. These breakthroughs offer the exciting possibility of combining the robust coherence properties of solid-state spin qubits with the extensive connectivity, fast control, and high-fidelity quantum-non-demolition readouts that are characteristic of superconducting qubit systems.

While our project focuses exclusively on state preparation within this physical system and does not address the readout phase, the potential applications and advancements serve as a compelling motivation for our investigation. The foundational physics and Hamiltonian formulations of the system have been comprehensively discussed in the existing literature, particularly by $\mathrm{D}_A njou_2 019 and Reinhold_2 019, whose explanations we adapt here for clarity.$

Our study revolves around a DQD defined within a double-well potential $V(z)$, with the wells separated by a distance $2d$. The primary energy orbitals of the right and left wells are represented by $|\widetilde{R}\rangle$ and $|\widetilde{L}\rangle$, respectively, each associated with its own energy levels ϵ_R and ϵ_L. The energy difference between these orbitals is denoted as $\epsilon = \epsilon_R - \epsilon_L$, and they are coupled through a tunneling term t_c. An external uniform magnetic field along the z-axis creates a Zeeman energy split between the spin states $|\widetilde{\uparrow}\rangle$ and $|\widetilde{\downarrow}\rangle$, while a position-dependent magnetic field along the x-axis helps hybridize the spin and charge states, enabling electrical manipulation and readout of the spin states.

$$H_d = H_m + H_Z, H_m = \frac{\epsilon}{2}\widetilde{\tau}_z + t_c\widetilde{\tau}_x, H_Z = \frac{B_z}{2}\widetilde{\sigma}_z + \frac{b_x}{2}\widetilde{\tau}_z\widetilde{\sigma}_x. \tag{2.1}$$

The Hamiltonian captures the interactions within the DQD, where $\widetilde{\tau}_i$ and $\widetilde{\sigma}_i$ represent the Pauli matrices in the orbital and spin basis, respectively. The eigenstates of the molecular Hamiltonian H_m are characterized by their energy separation Ω, which represents the molecular energy gap and plays a crucial role in understanding the quantum dynamics within the DQD.

In this system, the electric field of the resonator directly interacts with the electron's electric dipole moment, allowing the resonator photons to influence spin transitions. The Hamiltonian describing the combined resonator-DQD system is given by:

$$H = H_d + H_r + V, H_r = \omega_r a^\dagger a, V = g_c\tilde{\tau}_z(a + a^\dagger). \tag{2.2}$$

This setup enables us to investigate the quantum behavior of spins within a DQD under the influence of a microwave resonator. We employ a model effective Hamiltonian for a two-level system, expressed as:

$$\boldsymbol{H} = \omega_c \boldsymbol{a}^\dagger \boldsymbol{a} + \frac{\omega_q}{2}\boldsymbol{\sigma}_z + \frac{\chi}{2}\boldsymbol{a}^\dagger \boldsymbol{a}\boldsymbol{\sigma}_z, \tag{2.3}$$

with $\chi = 2g^2/\Delta$. This simplified representation is particularly useful in the dispersive regime, where the DQD can be effectively treated as a two-level system. The constants in this Hamiltonian, such as resonator frequency and cavity-qubit coupling strength, are typically derived from experimental data, although they may vary depending on the specific physical implementations. By focusing on optimization using the effective Hamiltonian, our project aims to deepen our understanding and control of these quantum systems.

Chapter 3

Methodology: Quantum Optimal Control

In quantum control we look to prepare some specific state, effect some state-to-state preparation, or effect some transformation (or gate) on a quantum system. For a given quantum system there will always be factors that effect the dynamics that are outside of our control. As examples, the interactions between elements of the system or a magnetic field required to trap the system. However, there may be methods of affecting the dynamics in a controlled way, such as the time varying amplitude of the electric component of an interacting laser field. And so this leads to some questions; given a specific quantum system with known time-independent dynamics generator (referred to as the drift dynamics generators) and set of externally controllable fields for which the interaction can be described by control dynamics generators:

1. What states or transformations can we achieve (if any)?
2. What is the shape of the control pulse required to achieve this?

These questions are addressed as controllability and quantum optimal control [**d2021introduction**]. The answer to question of controllability is determined by the commutability of the dynamics generators and is formalised as the Lie Algebra Rank Criterion and is discussed in detail in [**d2021introduction**]. The solutions to the second question can be determined through optimal control algorithms, or control pulse optimisation.

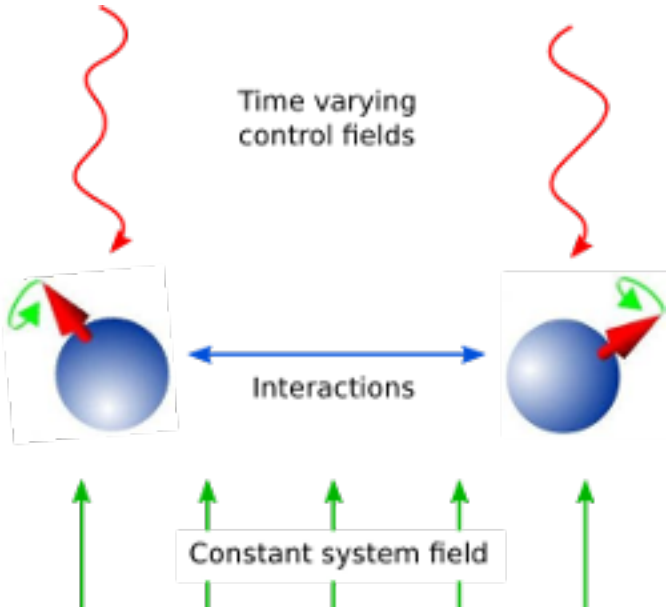

Figure 3.1: Schematic showing the principle of quantum control[**Johansson2013**]

Here we will first consider only finite-dimensional, closed quantum systems. In closed quantum systems the states can be represented by kets, and the transformations on these states are unitary operators. The dynamics generators are Hamiltonians. The combined Hamiltonian for the system is given by

$$H(t) = H_0 + \sum_{j=1} u_j(t) H_j$$

where H_0 is the drift Hamiltonian and the H_j are the control Hamiltonians. The u_j are time varying amplitude functions for the specific control.

The dynamics of the system are governed by Schrödingers equation.

$$\frac{d}{dt}|\psi\rangle = -iH(t)|\psi\rangle$$

Note we use units where $\hbar = 1$ throughout. The solutions to Schrödinger's equation are of the form:

$$|\psi(t)\rangle = U(t)\,|\psi_0\rangle$$

where ψ_0 is the state of the system at $t = 0$ and $U(t)$ is a unitary operator on the Hilbert space containing the states. $U(t)$ is a solution to the Schrödinger operator equation

$$\frac{d}{dt}U = -iH(t)U, \quad U(0) = \mathbb{1}$$

We can use optimal control algorithms to determine a set of u_j that will drive our system from $|\psi_0\rangle$ to $|\psi_1\rangle$, this is state-to-state preparation, or drive the system from some arbitary state to a given state $|\psi_1\rangle$, which is state preparation, or effect some unitary transformation U_{target} , called gate synthesis. The latter of these is most important in quantum computation.

3.1 GRAPE: Gradient Ascent Pulse Engineering

The GRadient Ascent Pulse Engineering was first proposed in [**KHANEJA2005296**]. Overview of GRAPE algorithm as implemented in QuTip is as follows. Solutions to Schrödinger's equation for a time-dependent Hamiltonian are not generally possible to obtain analytically. Therefore, a piecewise constant approximation to the pulse amplitudes is made. Time allowed for the system to evolve T is split into M timeslots (typically these are of equal duration), during which the control amplitude is assumed to remain constant.

In the GRAPE algorithm, the combined Hamiltonian is approximated as a piecewise constant function, where the control amplitudes are assumed to be constant within each timeslot. Mathematically, this can be expressed as $H(t) \approx H(t_k) = H_0 + \sum_{j=1}^{M} u_{jk} H_j$, where k is a timeslot index, j is the control index, and M is the number of controls. The variable t_k represents the evolution time at the start of the timeslot, and u_{jk} is the amplitude of control j throughout timeslot k. The total number of time steps, denoted by N, is equal to the total evolution time T divided by the timeslot duration δt. Though in my code in chapter **??**, I set number of time steps and evolution time, which then determines the time slot duration δt.

Given the piecewise constant approximation of the Hamiltonian, the time evolution operator, also known as the propagator, within each timeslot can be calculated using the matrix exponential: $U_k := e^{-iH(t_k)\Delta t_k}$. To obtain the overall evolution up to and including any timeslot k (including the full evolution at $k = M$), the propagators for each timeslot are multiplied together in chronological order: $U(t_k) := U_k U_{k-1} \cdots U_1 U_0$.

To quantify the effectiveness of the control pulses, a figure of merit or fidelity is introduced. This metric measures how close the achieved evolution is to the desired target state or transformation, based on the control amplitudes in the timeslots. For unitary systems, the typical figure of merit is the normalised overlap between the achieved evolution and the target. A more detailed discussion of the figure of merit can be found in chapter **??**.

The optimization problem in GRAPE can be formulated as follows. With the piecewise constant approximation, there are now $N \times M$ variables (the control amplitudes in each timeslot) that need to be optimized to minimize the figure of merit. This transforms the problem into a finite

multi-variable optimization problem, for which many established methods exist. These methods are discussed in more detail in chapter **??**. In the QuTiP Qtrl GRAPE implementation, the default optimization method is the L-BFGS-B algorithm, which is available through the Scipy library in Python.

3.2 CRAB: Chopped RAndom Basis

As opposed to GRAPE which optimises time-slice pulse amplitudes, the Chopped RAndom Basis (CRAB) algorithm optimizes the coefficients of a fintie set of functional basis of a chosen functional basis set. CRAB algorithm is particularly useful when the pulse complexity is very low [**PhysRevLett.106.190501**; **PhysRevA.84.022326**]. Here we are referring to relatively smooth pulses without singularities or sharp features. Such pulses are often more desirable for experiments as actuators at different levels from the pulse generators all the way to the controlled physical often require certain response time. In these cases, instead of optimizing the control amplitudes at each time slice, CRAB transforms the optimal control problem into a search over a few parameters by introducing a physically motivated function basis that constructs the pulse. This approach significantly reduces the number of optimization parameters compared to the number of time slices needed for accurate quantum dynamics simulations in gradient-based algorithms, often by orders of magnitude. As a result, CRAB can efficiently optimize smooth pulses while incorporating realistic experimental constraints. However, note that CRAB by itself does not guarantee smooth pulses. If a functional basis with basis elements containing discontinuities is chosen, the CRAB solution will inherit those discontinuities.

3.3 Figure of Merit

For state preparation case, we have the following evolved state fidelity as the figure of merit:

$$
\begin{aligned}
f(\vec{\epsilon}(t)) &= \mathcal{F}(\vec{\epsilon}(t)) \quad \text{state fidelity} \\
&\equiv |\langle\psi_{targ}| U(T, \vec{\epsilon}(t)) |\psi_{init}\rangle|^2 \\
&= \left| \langle\psi_{targ}| \mathcal{T} \exp\left\{ -\frac{i}{\hbar} \int_0^T dt H(\vec{\epsilon}(t)) \right\} |\psi_{init}\rangle \right|^2 \\
&\text{make the problem numerical, make } \vec{\epsilon}(t) \text{ a piece-wise constant function with } N = T/\delta t, \\
&\delta t \text{ is a parameter, usually set to time resolution of AWG.} \\
&= |\langle\psi_{targ}| U_N U_{N-1} ... U_1 |\psi_{init}\rangle|^2, \quad \text{where, } U_k = \exp\left\{ \frac{i\delta t}{\hbar} H(\vec{\epsilon}(k\delta t)) \right\}
\end{aligned}
$$

However, for our cavity-DQD composite system the composist system states need to be traced over the qubit state to obtain the cavity state fidelity.

3.4 Finite Variables Optimization Algorithms

Here we discuss a typical finite-variables optimization scheme.

$$H(\vec{\epsilon}(t)) = H_0 + \sum_{k=1}^{m} \epsilon_k(t) H_k \tag{3.1}$$

The goal of quantum optimal control is to find the optimal control field $\vec{\epsilon}(t)$ that optimizes a given function $f(\vec{\epsilon}(t)) = f(H(\vec{\epsilon}(t)))$, where $H(\vec{\epsilon}(t))$ represents the Hamiltonian of the quantum system under the influence of the control field. The optimization procedure involves several key steps and considerations.

To effectively optimize a large number of parameters, two important requirements must be met. First, an efficient method for calculating the gradients of the cost function with respect to the parameters is essential. Second, the optimization landscape should be such that sub-optimal local minima are either sufficiently unlikely or close to the global minimum, ensuring that the optimization algorithm can converge to a satisfactory solution.

Given methods to calculate the cost function and its gradients, two main classes of algorithms can be employed for performing the optimization. The first class is line-search methods, which

alternate between selecting a direction in the parameter space, radiating from the current point, and performing a one-dimensional minimization along this line to find the minimum. The most basic line-search method is gradient descent, which chooses the direction of steepest descent based on the gradient at the current point. More advanced methods, such as the Newton method, utilize both the gradient and the Hessian matrix to determine the search direction. Quasi-Newton methods, like the L-BFGS algorithm, approximate the Hessian matrix to improve convergence while reducing computational complexity. The L-BFGS algorithm is employed in the QuTiP Qtrl GRAPE implementation, and is available through the Scipy library in Python. The second class of optimization algorithms is trust-region methods, which define a region around the current point within which a quadratic approximation of the cost function is considered reliable, and the optimization is performed within this trust region.

The gradient of the cost function $f(\vec{\epsilon}(t))$ with respect to the control field $\vec{\epsilon}(t)$ plays a crucial role in many optimization algorithms. For an analytical function $f(\vec{\epsilon})$, the gradient is given by $\nabla f(\vec{\epsilon}) = \sum_{i=1}^{m} \frac{\partial f(\vec{\epsilon})}{\partial \epsilon_i} \hat{\epsilon}_i$, where m is the number of control parameters and $\hat{\epsilon}_i$ are unit vectors in the parameter space. When dealing with discrete control fields, the gradient can be approximated using the finite difference method. In some cases, calculating or approximating the gradient may be computationally expensive, and alternative optimization algorithms that do not rely on gradient information should be considered.

Once the gradient is calculated, the optimization algorithm takes a step in the direction of the negative gradient, with the step size being a tunable parameter. If the step size is too small, the optimization process may be slow, requiring many iterations to converge. On the other hand, if the step size is too large, the algorithm may overshoot the minimum and fail to converge to the true optimum. The gradient is then recalculated at the new point, and the process is repeated iteratively until a convergence criterion is met, indicating that the optimal control field has been found.

3.5 Pulse Constraints

When designing optimal control pulses for quantum systems, it is essential to consider the constraints imposed by the experimental setup, particularly those related to the arbitrary waveform generator (AWG). The AWG's amplitude and bandwidth limitations can significantly impact the feasibility and performance of the control pulses. To incorporate these constraints into the optimization process, one approach is to add a set of constraint terms to the cost function, as shown in the following equation:

$$f(\vec{\epsilon}(t)) = \mathcal{F}(\vec{\epsilon}(t)) + \sum_i \lambda_i g_i(\vec{\epsilon})$$

Here, $\mathcal{F}(\vec{\epsilon}(t))$ represents the original cost function, such as the fidelity of the desired quantum operation, while $g_i(\vec{\epsilon})$ are the constraint terms with their respective weights λ_i.

One of the most common constraints is the pulse amplitude, which is limited by the output power of the AWG. To ensure that the control pulses are feasible, we require $\epsilon(t) \leq \epsilon_{max}$ for all times t. There are two main approaches to enforce this constraint: hard cutoff and soft cutoff. The hard cutoff approach involves employing an optimization algorithm that naturally allows for such constraints, such as the L-BFGS-B algorithm available in the scipy.optimize module in Python. Alternatively, the soft cutoff approach involves parametrizing the optimization problem by introducing a new set of variables $\vec{x}$ and expressing the control amplitudes as $\epsilon_k = \epsilon_{max} \tanh(x_k)$. This ensures that the control amplitudes always remain within the allowed range.

Another important constraint is the pulse bandwidth, which can be addressed using various methods. Soft cutoff methods gradually penalize high-frequency components of the control pulses, while hard cutoff methods impose strict limits on the frequency content. Linear frequency-dependent penalties can also be employed, where the penalty term increases linearly with the frequency of the control pulse components.

In addition to the AWG constraints, the finite memory of classical computers poses a challenge when simulating quantum systems with infinite-dimensional Hilbert spaces, such as harmonic oscillators. To address this issue, we truncate the Hilbert space by choosing a maximum photon number n_{ph}, effectively replacing the infinite-dimensional oscillator with a finite-dimensional qudit. However, this truncation is only valid if all the relevant system dynamics for the desired state transfers occur within the subspace spanned by $|0\rangle, \cdots, |n_{ph} - 1\rangle$.

To ensure that the truncation does not introduce errors in the optimization process, we can modify the optimization problem to find a solution that operates identically under different values of n_{ph}. By writing the fidelity as a function of the truncation level, $F_{n_{ph}}$, we can formulate the optimization problem as follows:

$$\underset{\vec{\epsilon}}{\text{maximize}} \left(\sum_k F_{n_{ph}+k}(\epsilon(t)) \right) - \left(\sum_i \lambda_i g_i(\epsilon(t)) \right) \tag{3.2}$$

To enforce identical behavior across different truncations, we can add a penalty term that minimizes the discrepancy between the fidelities computed at different truncation levels:

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

This penalty term effectively symmetrizes the cost function with respect to n_{ph}. An alternative and more direct approach is to add a penalty term that minimizes the occupation of the highest

photon state ($|n_{ph} - 1\rangle$) in the truncated Hilbert space at any time during the evolution:

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

This penalty term encourages the optimization algorithm to find control pulses that keep the system dynamics within the truncated subspace, ensuring the validity of the simulation. By incorporating these constraints and penalty terms into the optimization problem, we can design control pulses that are not only effective in achieving the desired quantum operation but also feasible to implement experimentally, taking into account the limitations of the AWG and the finite memory of classical computers used for simulations.

3.6 Troubleshooting Optimization Algorithm

Here I provide a check list for Troubleshooting the optimization algorithm that I have accumulated during my project. However, this list is nonexhaustive and Reinhold provides his version in chapter 4.6, [**reinhold2019**].

1. Check optimization parameters:
   (a) Check that the time given $T = N\delta t$ is appropriate. T should be 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.
   (b) Check that truncations are sufficiently large. N value appropriate
2. check constraints: Completely remove all constraints and penalties, and make sure that the algorithm works in this context before re-introducing them
3. check termination conditions
   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. (gtol in scipy.minimize)
4. check initial guess: Avoid special initial guesses, this depends on the algorithm. For gradient-based algo, make sure initial gradient is not vanishing.

3.7 My Codes

This project is entirely conducted in python. The main quantum control libray in python is QuTiP. This project uses QuTiP library functions for the quantum simulation and GRAPE, CRAB optimization. Another python library based on QuTip, Krotov is also used in chapter **??**. All codes mentioned in chapter **??** follow a similar structure of defining the Hamiltonian, initial and target states, optimization parameters, then running the optimization algorithm by feeding the above defined variables in into QuTip or Krotov optimization functions and lastly checking optimization result via a forward simulation using both a library function or a self-implemented function. A python sample code can be seen in Appendix **??**. Note that the codes in Github are in Jupyter Notebook format, which is more readable and easier for dynamic scientific programming. Main effort went into determining the appropriate system and optimization parameters values rather than coding these scripts. Though with the above said, much effort was put into understanding QuTip source code and at some point correcting or modifying the source code to suit the needs of the project. For instance, much time was spent in trying to understand how QuTip implements the figure of merit for optimization and modify it to incorporate partial tracing out qubit into the fidelity function. However, this did not work out as will be discussed in chapter **??**. All the codes mentionded in are available in the GitHub repository [**lzl˙github˙repo**].

Chapter 4

State preparation pulse optimization

Here is a summary of optimization results (shown previously in overview chapter):

where, initial and target states are given by:

vac2coherent

$$\psi_{i1} = |0\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}}$$
$$\psi_{f1} = |\alpha\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}}$$

unselective

$$\psi_{i2} = \frac{1}{\sqrt{2}}(|0\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}} + |1\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}})$$
$$\psi_{f2} = \frac{1}{\sqrt{2}}(|0\rangle_{\text{cavity}} \otimes |1\rangle_{\text{qubit}} + |1\rangle_{\text{cavity}} \otimes |1\rangle_{\text{qubit}})$$

selective

$$\psi_{i3} = \frac{1}{\sqrt{2}}(|0\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}} + |1\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}})$$
$$\psi_{f3} = \frac{1}{\sqrt{2}}(|0\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}} + |1\rangle_{\text{cavity}} \otimes |1\rangle_{\text{qubit}})$$

vac2cat

$$\psi_{i1} = |0\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}}$$
$$\psi_{f1} = |\alpha\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}} + |-\alpha\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}}$$

name	physical system	initial state	target state	ops algo	N(Fock space trunca-tion)	time discre--tisation	pulse length (ns)	final fidelity (simulation)
single_qubit	single qubit	ground state	excited state	GRAPE	NIL	1000	18	0.9999
vaccuum2coherent	cavity coupled to a qubit	ψ_{i1}	ψ_{f1}	GRAPE	5	100	1	0.9999
unselective_spin_flip	cavity coupled to a qubit	ψ_{i2}	ψ_{f2}	GRAPE	16	500	1	0.9995
selective_spin_flip	cavity coupled to a qubit	ψ_{i3}	ψ_{f3}	GRAPE	16	100000	500	0.9994
selective_spin_flip _constraints	cavity coupled to a qubit	ψ_{i3}	ψ_{f3}	GRAPE	16	100000	500	0.9992
vacuum2cat	cavity coupled to a qubit	ψ_{i4}	ψ_{f4}	GRAPE	16	100000	5000	0.9529
vacuum2cat	cavity coupled to a qubit	ψ_{i4}	ψ_{f4}	KROTOV	16	100000	5000	0.9995

4.1 Starting With a Simple Diagnostic Pulse, Single Qubit System

To verify that I have a working code that can optimize a pulse, I will start with a simple qubit flip operation of a single-qubit system. The purpose of this chapter is to show that the algorigm is indeed running optimization by recovering analytically solvable state-to-state transer using pulse optimization.

In the following example, we consider using the CRAB algorithm as implemented in the qutip python library. In qutip, the ctrlpulseoptim.optimize_pulse_unitary function is used to optimize pulse shapes to minimize the fidelity error, which is equivalent maximising the fidelity to an optimal value of 1.

The Hamiltonian of a single qubit system with arbitrary control is give by,

$$H(t) = \frac{\hbar\omega_0}{2}\sigma_z + \epsilon_x(t)\sigma_x + \epsilon_y(t)\sigma_y \tag{4.1}$$

where, $\epsilon_x(t), \epsilon_y(t)$ are control pulses.

Consider a special case of the above general case.

$$H_1(t) = \frac{\hbar\omega_0}{2}\sigma_z + \frac{\hbar\omega_1}{2}(\sigma_x \cos\omega t + \sigma_y \sin\omega t) \tag{4.2}$$

From Valerio's note, via frame rotation, starting from $|\psi_{init}\rangle = |+\hat{z}\rangle$, we have evoloved state,

$$|\psi(t)\rangle = e^{-i\omega t/2}[\cos(\Omega t/2) - i\cos\theta\sin(\Omega t/2)]|+\hat{z}\rangle - ie^{i\omega t/2}\sin\theta\sin(\Omega t/2)|-\hat{z}\rangle \tag{4.3}$$

where,

$$\Omega = \sqrt{(\omega_0 - \omega)^2 + \omega_1^2}\cos\theta = \frac{\omega_0 - \omega}{\Omega}$$

Suppose we want to achieve a state-to-state preparation from $|+\hat{z}\rangle$ to $|-\hat{z}\rangle$,

$$P_{+\hat{z}\to-\hat{z}}(t) = \sin^2\theta\sin^2(\Omega t/2) = \left(\frac{\omega_1}{\Omega}\right)^2\sin^2(\Omega t/2)$$

With $H_1(t)$, $P_{+\hat{z}\to-\hat{z}}(t)$ only goes to 1 if

1. $\omega_1 = \Omega$, i.e. $\omega = \omega_0$.

2. $\omega_1 = \frac{\pi}{T}$ (with evolution time T fixed, smallest frequency needed)

Now we run the pulse optimization and compare with analytical result.
Let t = evo_time = T
Defining the time evolution parameters: - To solve the evolution the control amplitudes are considered constant within piecewise timeslots, hence the evolution during the timeslot can be calculated using U(t_k) = expm(-iH(t_k)dt). - Combining these for all the timeslots gives the approximation to the evolution from an initial state ψ_0 at t=0 to $U(T)\psi_0$ at the t=evo_time. The number of timeslots and evo_time have to be chosen such that the timeslot durations (dt) are small compared with the dynamics of the system. - set drift frequency ω_0 to some number, at resonance $\omega = \omega_0$, for state to state preparation to happen at the end of pulse, require $\omega_1 = \frac{\pi}{T}$.

Next, we set the initial guess pulses where the pulse optimization algorithm starts. During the each iteration of the optimization, the Nelder-Mead algorithm calculates a new set of coefficients that improves the currently worst set among all set of coefficients. For details see [1,2] and a textbook about static search methods. The algorithm continues until one of the termination conditions defined above has been reached. If undesired results are achieved, rerun the algorithm and/or try to change the number of coefficients to be optimized for, as this is a very crucial parameter.

The optimization parameters are:

1. Number of time slots, n_ts = 1000
2. Time allowed for the evolution, evo_time = 18
3. Fidelity error target, fid_err_targ = 1e-6
4. Maximum iterations for the optimization algorithm, max_iter = 10000
5. Maximum (elapsed) time allowed in seconds, max_wall_time = 120
6. initial pulse, p_type = 'SINE'

The optimized pulses are:

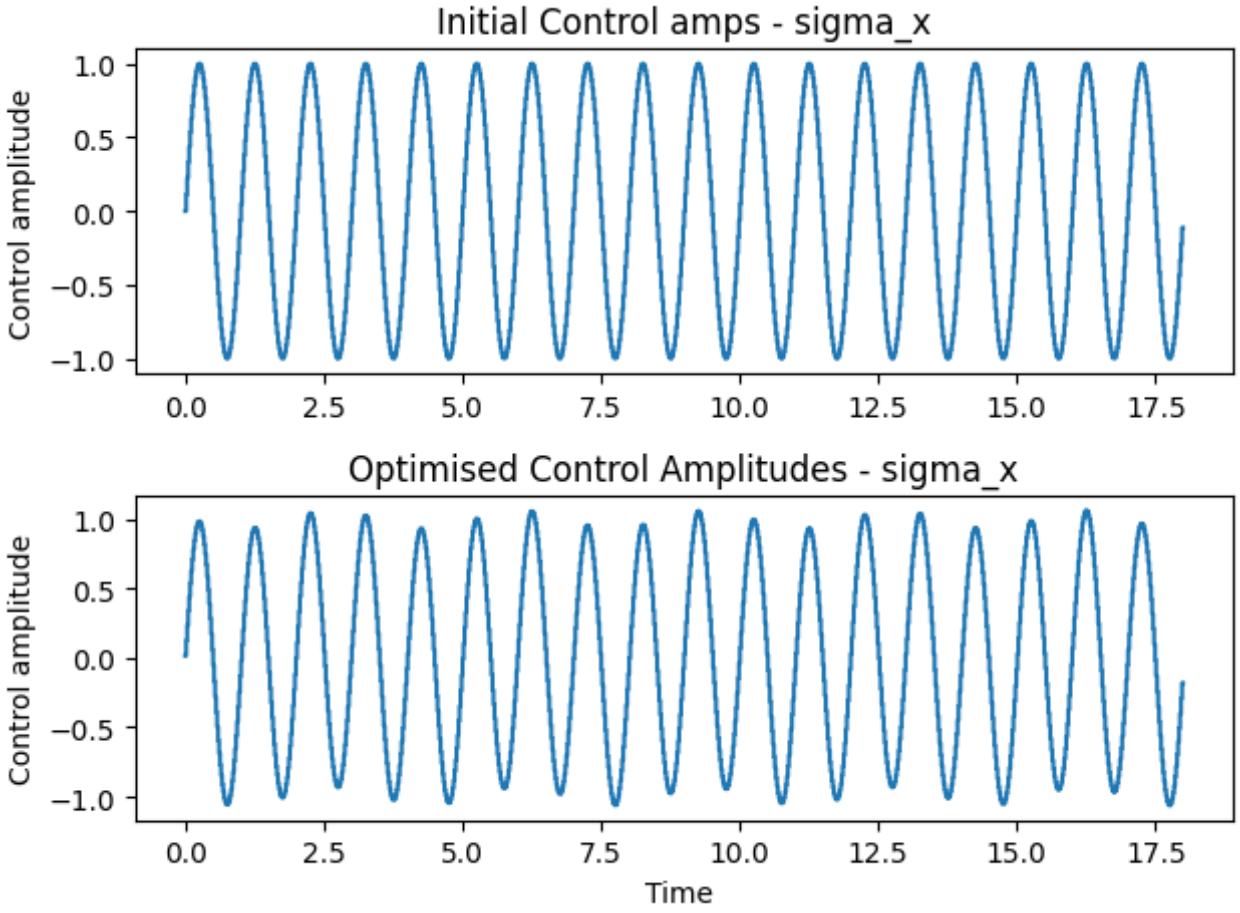

Figure 4.1: single qubit control 1 pulse

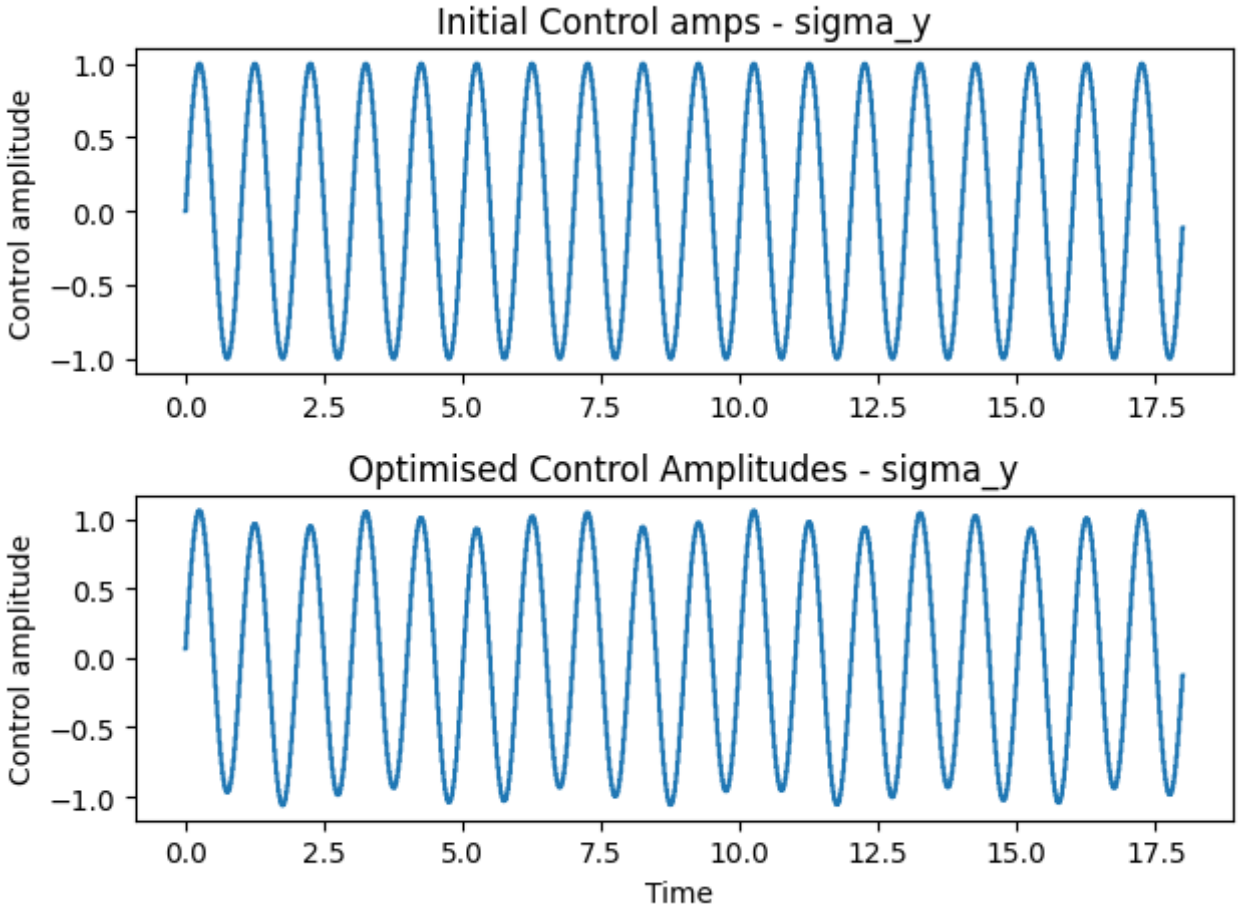

Figure 4.2: single qubit control 2 pulse

We can see that the optimized pulses are only slightly modified from the initial guess pulses. This is because firstly to drive a single qubit flip is but a simple rabi drive and the initial SINE pulse frequency has been chosen to match the qubit frequency. Secondly, the GRAPE algorithm optimizes by modifying time-slice pulse amplitudes, thus leading to small overall waveform change for a simple optimization task.

Though the exact analytical solution couldn't be recovered from numetical optimization as numerical optimization searches through a huge optimization landscape where the analytical solution is only a special case, the algorithm can be crudely verified to be running by a forward simulation which shows the optimized evolved state fidelity to be 0.9999367235643194.

4.2 Diagnostic Pulse: Cavity Vacuum to Cavity Coherent State

Having tested the code on a single qubit system, we now move on to a more complex system that I am eventually interested in. In this code, we consider a cavity coupled to a single qubit. For this physical system, we have drift Hamiltonian, from Reinhold PhD thesis (2.13) [**reinhold2019**]:

$$H_d = \omega_q a^\dagger a + \frac{\omega_z}{2}\sigma_z + \frac{\chi}{2}a^\dagger a \sigma_z \tag{4.4}$$

It can be shown that it's equivalent to:

$$\begin{aligned} H_d &= g\left(\hat{\sigma}_+\hat{\sigma}_- + \hat{a}^\dagger\hat{a}\hat{\sigma}_z\right) \\ &= g\left(\frac{1}{2}(1+\hat{\sigma}_x) + \hat{n}\hat{\sigma}_z\right) \end{aligned} \tag{4.5}$$

where, g is the cavity-qubit coupling strength which is set to be $50 * 2\pi MHz$ here, as given by experiment

For this physical system we have both cavity and auxiliary qubit contorl, in the form of control Hamiltonian:

$$H_c = \epsilon_c(t)\hat{a} + \epsilon_T(t)\hat{\sigma}_- + h.c. \tag{4.6}$$

where,

- $\epsilon_c(t)$ is the contorl signal to cavity
- $\epsilon_T(t)$ is the control signal to ancilliary qubit

with $\hat{\sigma}_- = \frac{1}{2}\hat{\sigma}_x - \frac{i}{2}\hat{\sigma}_y$, rewrite H_c as following for ease of implementation using QuTip:

$$H_c = \epsilon_c(t)\hat{a} + \epsilon_T(t)(\frac{1}{2}\hat{\sigma}_x - \frac{i}{2}\hat{\sigma}_y) + h.c. \tag{4.7}$$

Assuming that the control signals $\epsilon_c(t), \epsilon_T(t)$ are real, and ignoring scalar constants, we have

$$H_c = \epsilon_c(t)(\hat{a} + \hat{a}^\dagger) + \epsilon_T(t)\hat{\sigma}_x$$

Initial and target states:

$$\begin{aligned} |\psi_{\text{initial}}\rangle &= |\alpha\rangle \\ |\psi_{\text{final}}\rangle &= |\alpha\rangle + |-\alpha\rangle \end{aligned} \tag{4.8}$$

For testing the code, consider

$$\begin{aligned} |\psi_{\text{initial}}\rangle &= |0\rangle \\ |\psi_{\text{final}}\rangle &= |\alpha\rangle \end{aligned} \tag{4.9}$$

The optimization algorithm settings used in this example are:

- Number of time slots, n_ts = 100
- Time allowed for the evolution, evo_time = 1
- Fidelity error target, fid_err_targ = 1e-4
- Maximum iterations for the optisation algorithm, max_iter = 100000
- Maximum (elapsed) time allowed in seconds, max_wall_time = 10000
- optimization algorithm: GRAPE

The initial and optimised pulses:

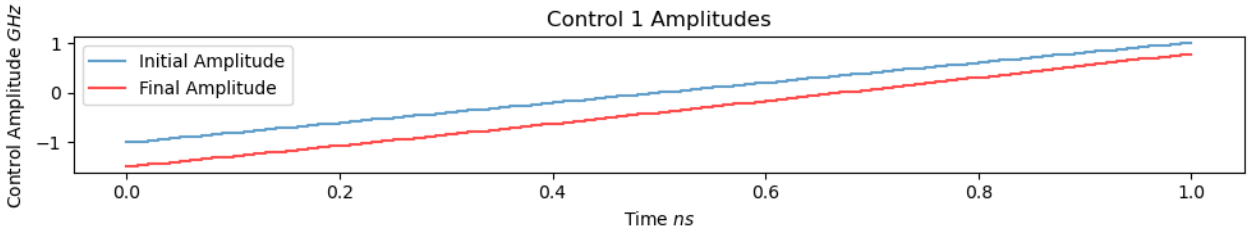

Figure 4.3: vacuum to coherent control 1 pulse

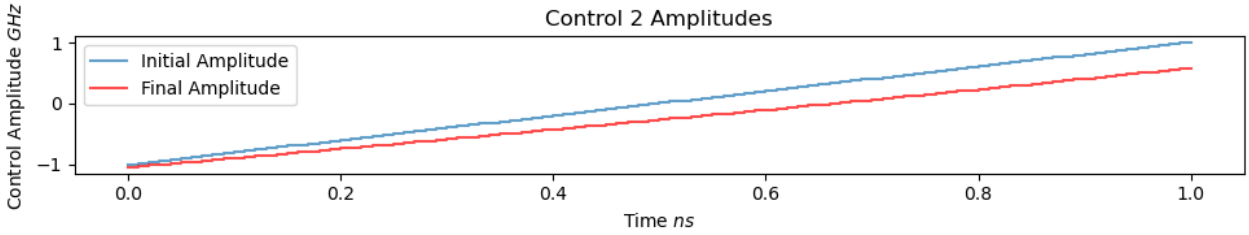

Figure 4.4: vacuum to coherent control 2 pulse

Forward simulation is as follows:

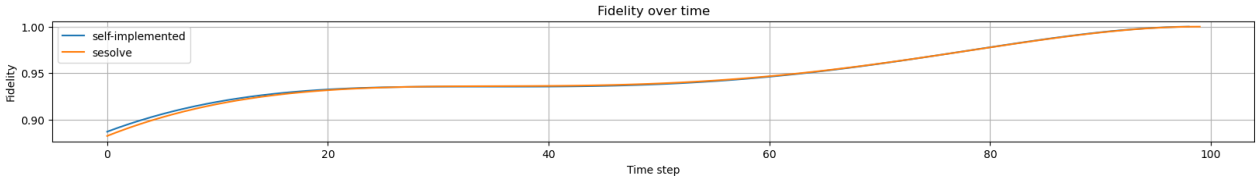

Figure 4.5: vacuum to coherent forward simulate

To verify whether the cavity state basis truncation N is large enough, i.e. whether the optimziation result has converged with respect to N, consider:

1. Run the optimization with the same algorithm settings and same initial guess, but with different N values. Then plots the optimized pulses ran with different N values on the same plot.

2. Simulate the evolution of the system with the optimized pulses, but with a range of higher N values.

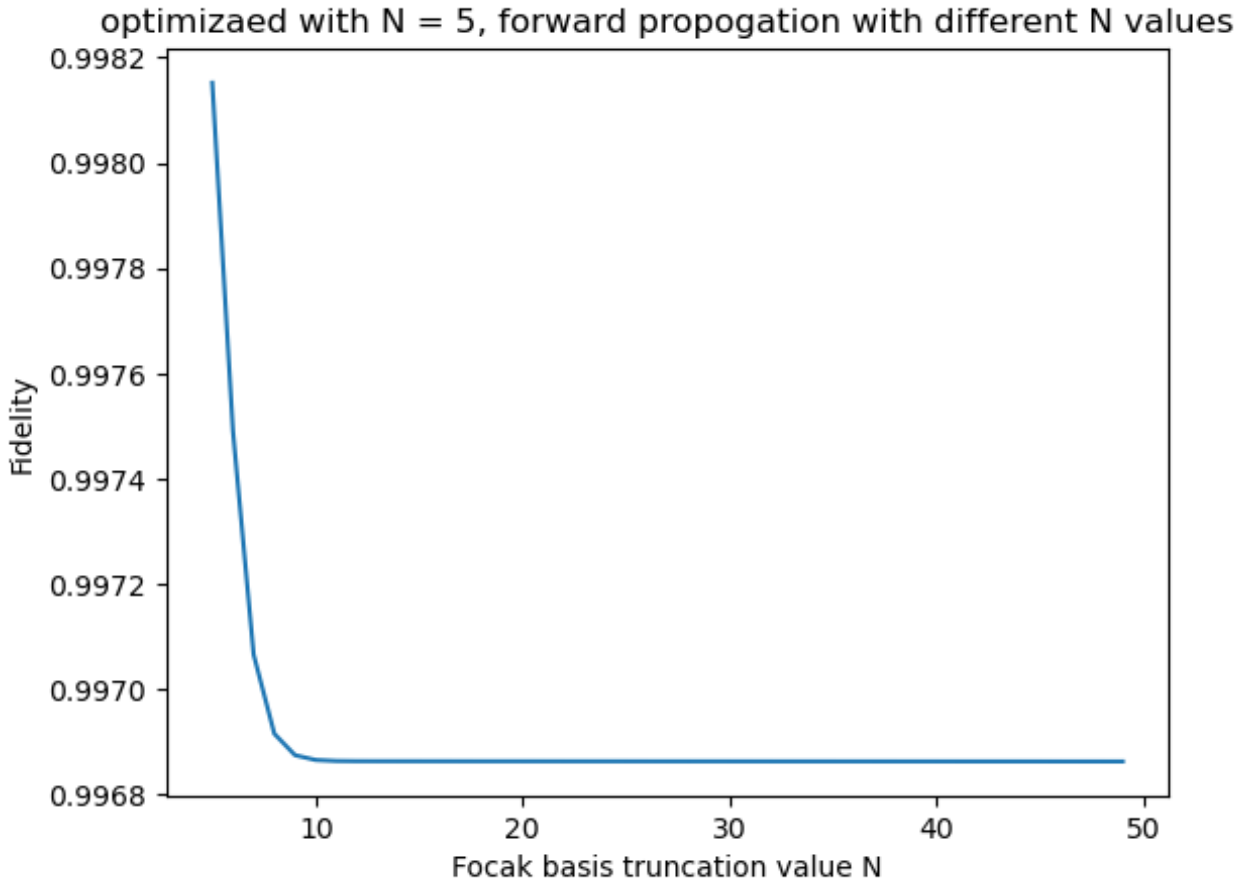

Figure 4.6: cavity state Fock basis truncation N convergence

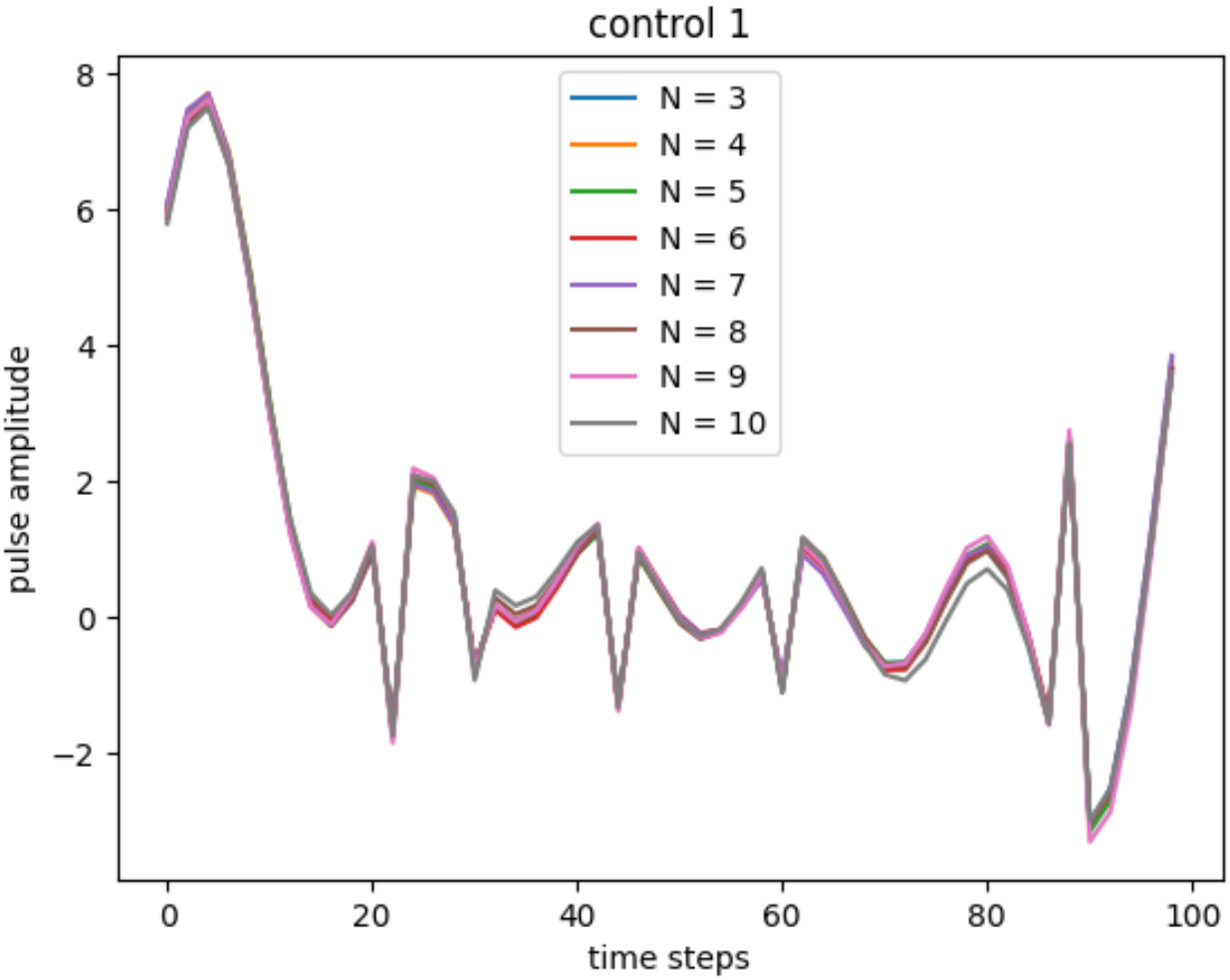

Figure 4.7: cavity state Fock basis truncation N convergence

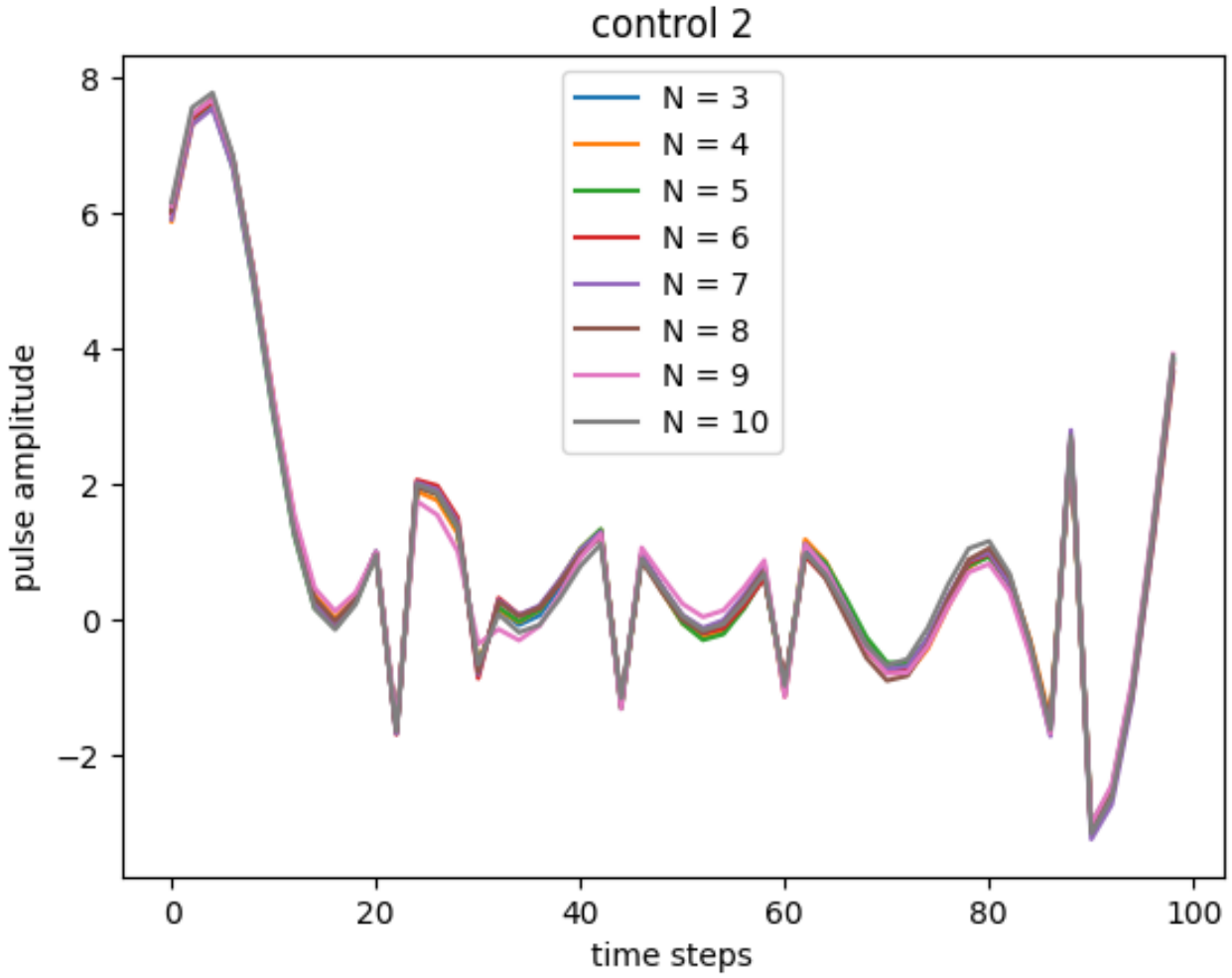

Figure 4.8: cavity state Fock basis truncation N convergence

4.3 Diagnostic Pulse: Unselective Spin Flip, Cavity Coupled With Single Qubit(effective)

Going from a cavity cavuum state to a cavity coherent state analytically requires only a displacement operation which requires only a control pulse to the cavity. Hence, in the previous disagnostic scenario the full Hamiltonia with all three (real) control channels were not used.

Here we consider the full system Hamiltonian (cavity + auxiliary qubit + coupling terms) drift

Hamiltoinan is given by,

$$H_d = \text{wr} * n_{\text{cavity}} + \text{Ez} * \sigma_z + \text{chi} * \sigma_z * (n_{\text{cavity}} + 1/2) \tag{4.10}$$

where parameters of the physical system are:

(notet that energy terms have unit GHz and time terms have unit ns)

- cavity frequency, wr = 2π
- qubit anharmonicity Ez = 0.44π
- qubit-cavity coupling strength chi = 0.007

Real control channels in the Hamiltonain are given by (each term comprises a channel):

$$H_c = a + a^\dagger + -1j * (a - a^\dagger) + \sigma_x \tag{4.11}$$

The initial and target states for this pulse optimization code are:

$$\psi_0 = \frac{1}{\sqrt{2}}(|0\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}} + |1\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}})$$
$$\psi_{\text{targ}} = \frac{1}{\sqrt{2}}(|0\rangle_{\text{cavity}} \otimes |1\rangle_{\text{qubit}} + |1\rangle_{\text{cavity}} \otimes |1\rangle_{\text{qubit}})$$

where the qubit state is flipped irrepsective of the cavity state

The final optimization algorithm parameters are:

- optimization algorithm: GRAPE
- Number of time slots, n_ts = 500
- Time allowed for the evolution, evo_time = 1
- Fidelity error target, fid_err_targ = 1e-3
- Maximum iterations for the optisation algorithm, max_iter = 1000
- Maximum (elapsed) time allowed in seconds, max_wall_time = 120

- initial guess pulse type, p_type = 'LIN'

The inital and optimized pulses are:

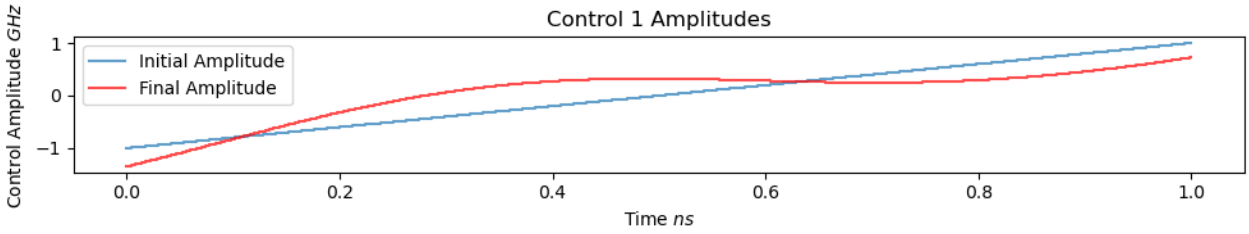

Figure 4.9: unselective spin flip control 1 pulse amplitudes

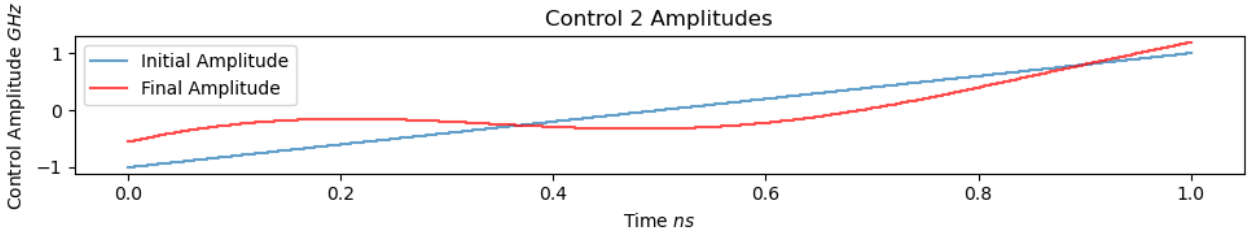

Figure 4.10: unselective spin flip control 2 pulse amplitudes

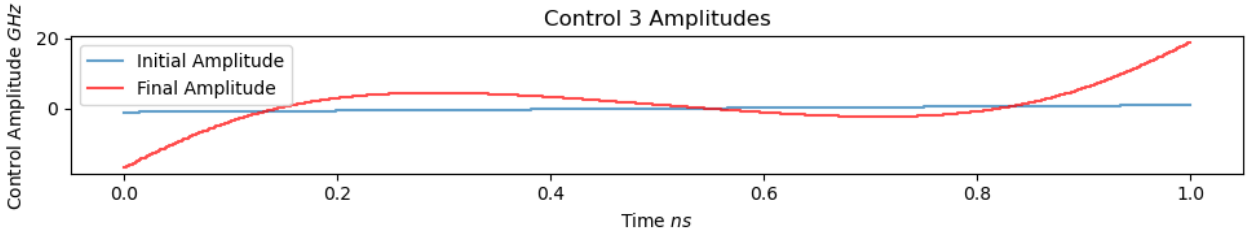

Figure 4.11: unselective spin flip control 3 pulse amplitudes

Forward simulation is given below to ensure that pulse optimization has run properly.

Simulation gives:

- sesolve final fidelity: 0.9995869703894827

- self-implemented final fidelity: 0.9995575205270999

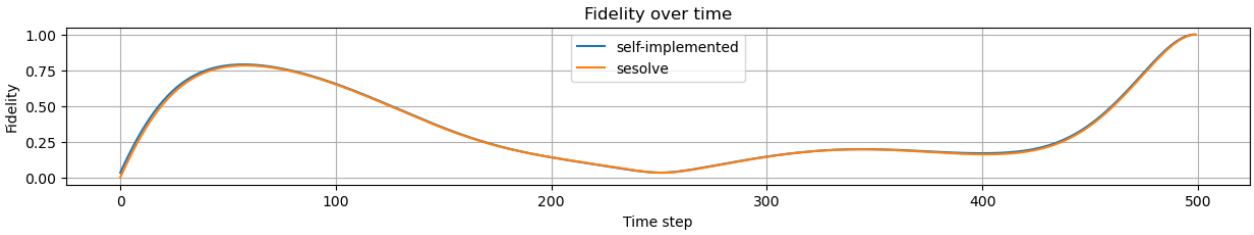

Figure 4.12: unselective spin flip forward simulation

4.4 Diagnostic Pulse: Selective Spin Flip, Cavity Coupled With Single Qubit (effective)

For this scenario, the drift and contorl Hamiltonian terms are same as in the previous chapter. We are interested in this state preparation because as achieved by Krastanov et al.(2015) [**Krastanov2015**] which gave a contruction for how to achieve arbitrary operation on dispersively coupled cQED systems using a set of two opeartions: 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

$$S(\vec{\theta}) = \sum_k e^{i\theta_k} |k\rangle \langle k| \tag{4.12}$$

Clearly, SNAP operations require selective spin rotations of which selective spin flip is a special case.

State preparation optimization target is as follow:

$$\psi_0 = \frac{1}{\sqrt{2}}(|0\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}} + |1\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}})$$
$$\psi_{\text{targ}} = \frac{1}{\sqrt{2}}(|0\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}} + |1\rangle_{\text{cavity}} \otimes |1\rangle_{\text{qubit}})$$

(where qubit spin flip is selective on cavity state)

The best optimization result that was obtained by tuning the optimization script from previous chapter is as follow.
The final optimization algorithm parameters are:

- optimization algorithm: GRAPE
- Number of time slots, n_ts = 100000
- Time allowed for the evolution, evo_time = 500
- Fidelity error target, fid_err_targ = 1e-8
- Maximum iterations for the optisation algorithm, max_iter = 10000
- Maximum (elapsed) time allowed in seconds, max_wall_time = 7200
- initial guess pulse type, p_type = 'SINE'

The inital and optimized pulses are:

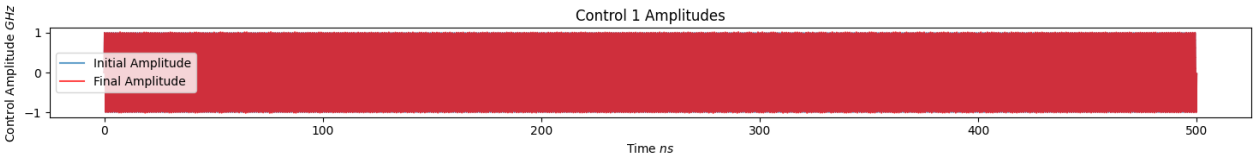

Figure 4.13: selective spin flip control 1 pulse amplitudes

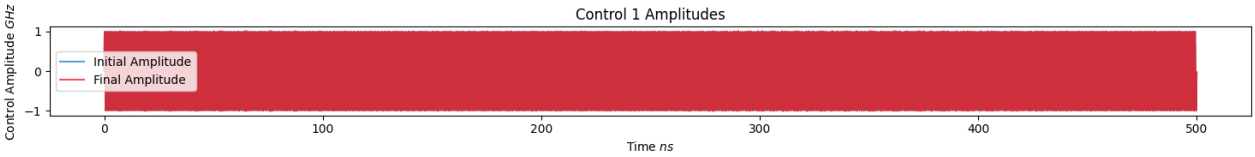

Figure 4.14: selective spin flip control 2 pulse amplitudes

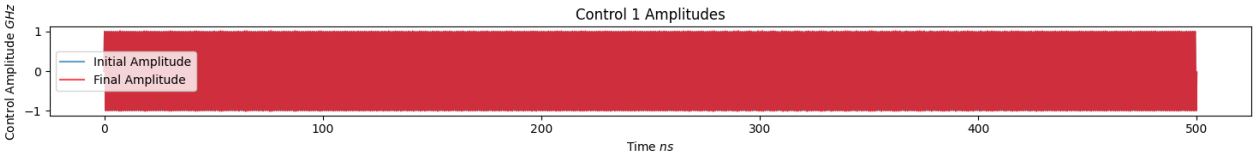

Figure 4.15: selective spin flip control 3 pulse amplitudes

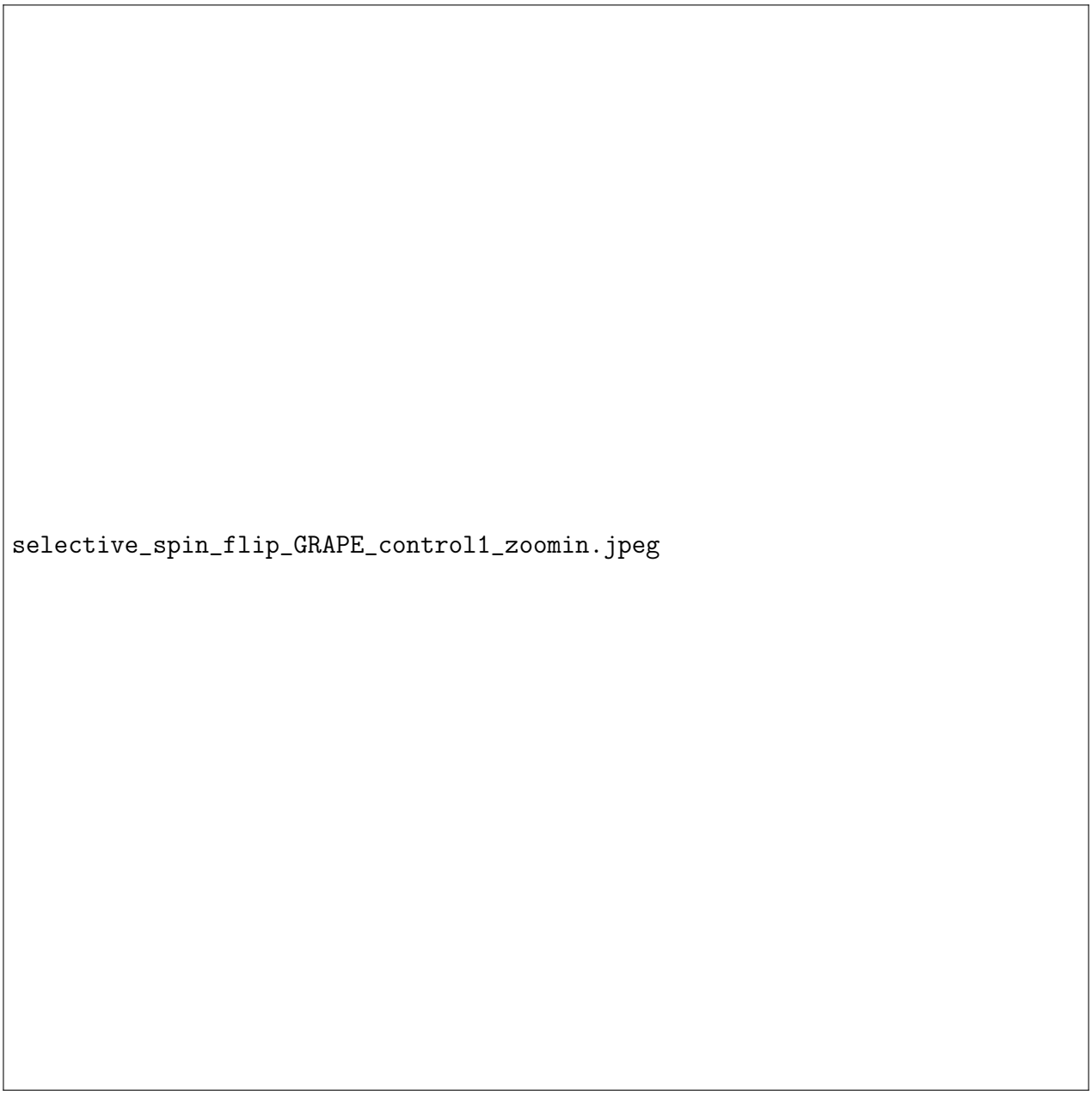

Figure 4.16: selective spin flip control 1 pulse amplitudes zoom in

The control pulses have very fast oscillations due to the large number of time slices. Zooming into control 1 pulse amplitudes, as shown in fig **??**, we see that the pulse is fairly smooth. However, this fast oscillation is still not desirable and preferably gotten rid of. Another reason that might have contributed to this fast oscillation is the background frequency of the cavity and qubit of which

the control pulses are trying to cancel out. One way to partially deal with this fast oscillation issue is to work with the Hamiltonian in the rotating frame which will be tried out in the next chapter when cavity vacuum state to cavity cat state is being optimized.

Forward simulation is given below to ensure that pulse optimization has run properly.

Simulation gives:

- sesolve final fidelity: 0.9994876128337873
- cavity ptrace fidelity: 0.9997184822307845
- qubit ptrace fidelity: 0.999999719275808
- self-implemented final fidelity: 0.9997005493228243

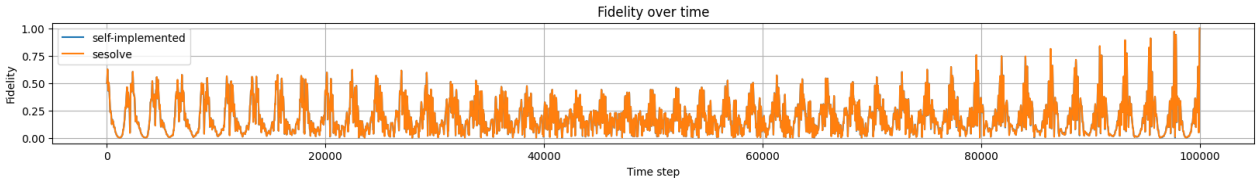

Figure 4.17: selective spin flip forward simulation

A particular analytical solution as given by Chiyuan already gives fidelity of 0.98. We expect a higher fidelity from a numerical optimization solution.

However, this numerical solution has the folowing issues. The optimzied pulses go beyong the amplitude constrains that required for the effective Hamiltonian to be valid. When the pulse amplitdes constrained are supplied to the optimziation algorithms, the results are not very satisfactory.

The following run has two modifications:

- initial guess pulse is changed from sindusoidal to linear to see whether this improves the fast oscillation issue
- Pulse amplitude constraints are added to the optimization algorithm

However, as can be seen from the optimized pulses below, the fast oscillation issue is not resolved well. This is especially so for control 3.

The inital and optimized pulses are:

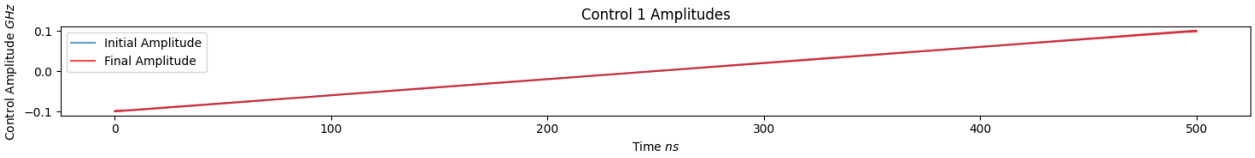

Figure 4.18: selective spin flip control 1 pulse amplitudes

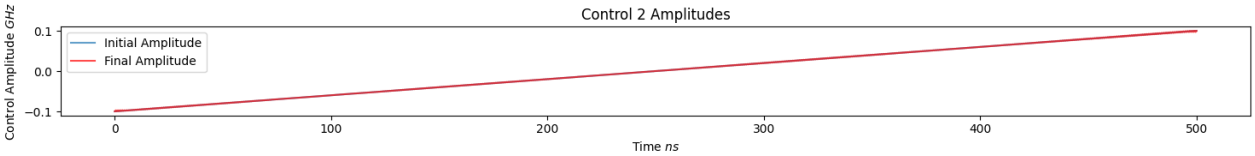

Figure 4.19: selective spin flip control 2 pulse amplitudes

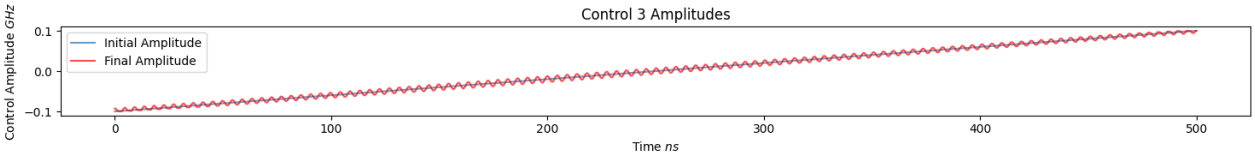

Figure 4.20: selective spin flip control 3 pulse amplitudes

Forward simulation is given below to ensure that pulse optimization has run properly. Simulation gives:

- sesolve final fidelity: 0.9992259750482354
- cavity ptrace fidelity: 0.9992525011412332
- qubit ptrace fidelity: 0.9992706948956261
- self-implemented final fidelity: 0.9999937323329505

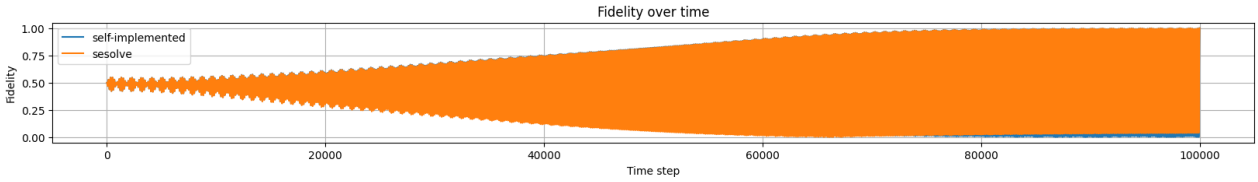

Figure 4.21: selective spin flip forward simulation

At this point, we continued varying the parameters (puse length, number of slices, initial guess pulse, etc.). However, none gave optimization results where fidelity, pulse amplitude, pulse smoothness and fast oscillation issue are all accounted for satisfactorily.

4.5 Vacuum to Cat State, Cavity Coupled With Single Qubit (effective)

In this chapter, we finally get to do a state preparation optimization of cavity vacuum state to cavity cat state. The drift and control Hamiltonian terms are same as in the previous chapter.

The initial and target states for this pulse optimization code are:

$$\psi_{\text{initial}} = |0\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}} \tag{4.13}$$

$$\psi_{\text{target}} = |\alpha\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}} + |-\alpha\rangle_{\text{cavity}} \otimes |0\rangle_{\text{qubit}} \tag{4.14}$$

$$\tag{4.15}$$

The most efficient way to do this is to optimize for the cavity state alone by tracing out the qubit state when calculating state fidelity. However, the QuTip package does not support this feature. I attemped to modify QuTip source code to allow for this feature but failed. One might naively think that QuTip calculates state fidelity at the end of each optimization iteration. Intead, QuTip calculates the state fidelity throughout the entire control pulse evolution for some caching purposes that improves the overall performance of the optimization algorithm. Here, a problem arises as tracing out qubit state in the middle of a pulse is not valid.

Hence, composite state of the cavity and qubit was still used for optimization which restricts the search space to smaller than ideal. However, the optimization results as shown below were not very satisfacotry.
The final optimization algorithm parameters are:

- optimization algorithm: GRAPE
- Number of time slots, n_ts = 100000
- Time allowed for the evolution, evo_time = 5000
- Fidelity error target, fid_err_targ = 1e-8
- Maximum iterations for the optimization algorithm, max_iter = 10000
- Maximum (elapsed) time allowed in seconds, max_wall_time = 21600
- initial guess pulse type, p_type = 'SIN'

The inital and optimized pulses are:

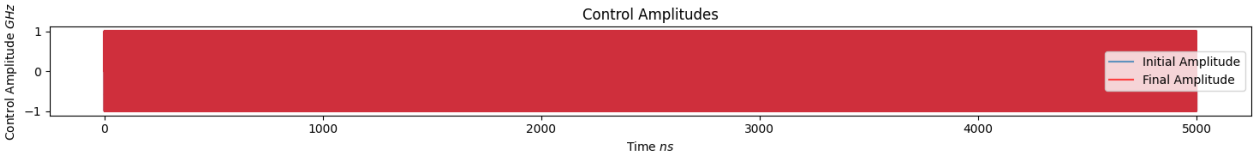

Figure 4.22: vacuum to cat state control 1 pulse amplitudes

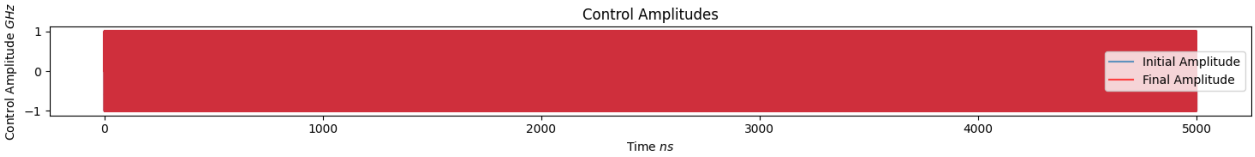

Figure 4.23: vacuum to cat state control 2 pulse amplitudes

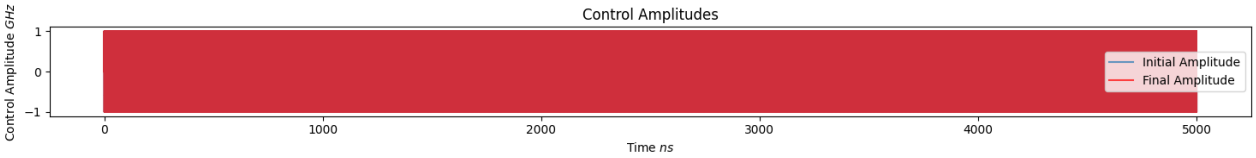

Figure 4.24: vacuum to cat state control 3 pulse amplitudes

Forward simulation is given below to ensure that pulse optimization has run properly.

Simulation gives:

- sesolve final fidelity: 0.4390558062780975
- cavity ptrace fidelity: 0.44320315443027297
- qubit ptrace fidelity: 0.9871678390509413
- self-implemented final fidelity: 0.9529111010142354

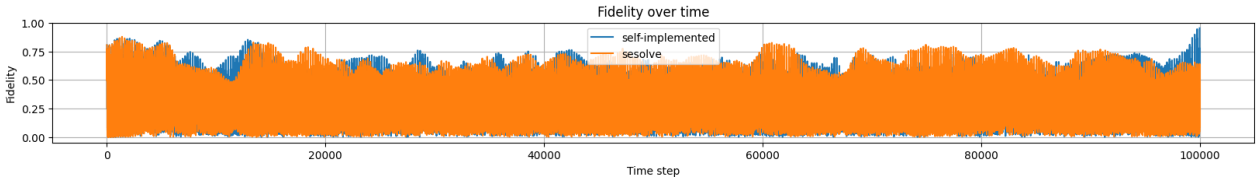

Figure 4.25: vacuum to cat state forward simulation

It can be seen that the optimization results are not very satisfactory. Many tunings were attempted but none gave satisfactory results.

At this point, we turned to look at a new python quantum control optimziation library, Krotov which builds on top of QuTip, to see whether it can give better optimization results. Krotov's optimization method is a gradient-based optimization algorithm like GRAPE. Krotov's method distinguishes itself by guaranteeing monotonic convergence for near-continuous control fields. Besides the difference in optimization algorithm used, the Krotov package has the following incentives to be tried:

- Krotov is much better implemented than QuTip. For instance, Krotov allows display of optimization progress for every iteration.
- Krotov allows inbuilds better customizability than QuTip. For instance, Krotov allows for the optimization of the fidelity to be optimized.

Besides switching from QuTip to Krotov, the Hamiltonians are also redefined to be an interaction picture. The drift Hamiltonian (H_0) and control Hamiltonians (H_1, H_2, H_3) are defined as follows:

1. **Drift Hamiltonian** (H_0):

$$H_0 = \hbar\omega_r a^\dagger a + \frac{E_s}{2}\sigma_z + \frac{E_m}{2}\tau_z - \chi_m\tau_z\left(a^\dagger a + \frac{1}{2}\right) - \chi_s\sigma_z\left(a^\dagger a + \frac{1}{2}\right) - \frac{\chi_0}{2}\sigma_z\tau_z, \tag{4.16}$$

where ω_r is the resonator frequency, a and $a^\dagger$ are the annihilation and creation operators of the cavity, E_s and E_m are the energy levels of the qubit and the molecular orbital, σ_z and τ_z are the Pauli Z operators for the qubit and the molecular orbital, respectively, and χ_m, χ_s, and χ_0 are the dispersive shifts.

2. **Control Hamiltonians**: - H_1: This control Hamiltonian corresponds to the real part of the cavity drive. It is given by

$$H_1 = \left(a + a^\dagger\right), \tag{4.17}$$

where a and $a^\dagger$ are the annihilation and creation operators of the cavity.

- H_2: This control Hamiltonian corresponds to the imaginary part of the cavity drive. It is given by

$$H_2 = -i\left(a - a^\dagger\right), \tag{4.18}$$

where a and $a^\dagger$ are the annihilation and creation operators of the cavity.

- H_3: This control Hamiltonian corresponds to the drive on the qubit. It is given by

$$H_3 = \sigma_x, \tag{4.19}$$

where σ_x is the Pauli X operator for the qubit.
The optimized results are as follow.

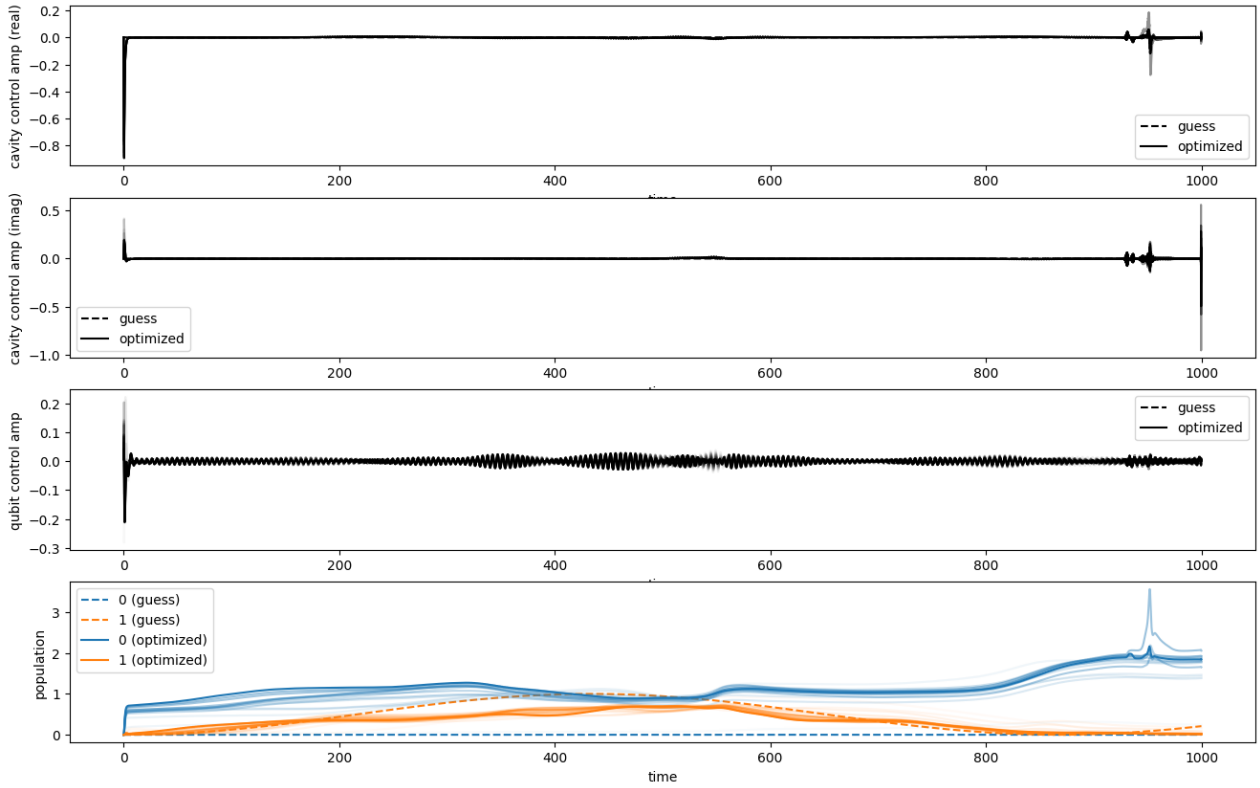

Figure 4.26: (1st,2nd) The optimized control fields for the cavity drive (real and imaginary parts) and the qubit drive.
(3rd) qubit control
(4th) qubit population

Forward simulation is given below to ensure that pulse optimization has run properly.
Simulation gives:

- final fidelity: 0.9992259750482354

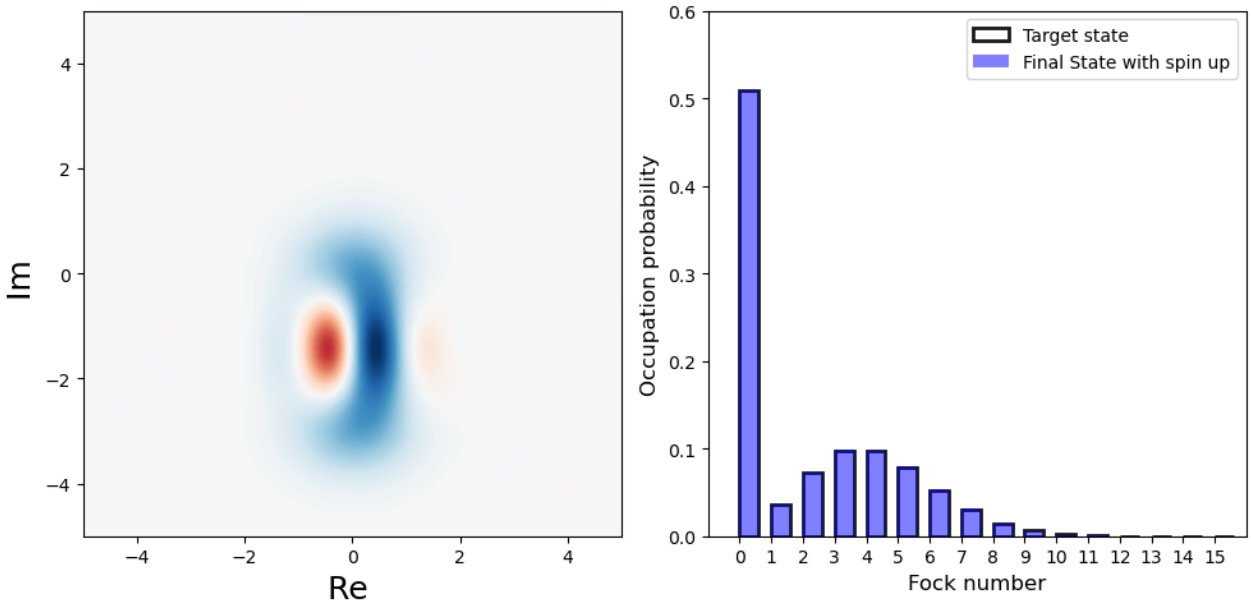

Figure 4.27: (left) Wigner function representation of target cavity state and final cavity state (right) target cavity state and final cavity state in Fock state basis

4.6 Further Discussions

There are a few points to be made regrading choosing the time parameters for optimization, evolution time (evo_time) and number of time slices (n_ts). Evolution time (evo_time), or equivalently the pulse length, can be estimated Number of time slices (n_ts) that is too small will lead to large error. n_ts that is too large will result in long computation time; and also lead to fast oscillations in optimized pulses when optimization is using GRAPE. Hence an appropriate n_ts value needs to be tested via trial and error.

It is unfortunate that there was not enough time to reach the initial goal of the project due to programming issues regrading QuTip and some other unexpected complications. Firstly, it is understood that optimization should have been run with the fidelity of the cavity state alone, i.e. partial tracing out the qubit state when calculating fidelity. However, implementation of QuTip doesn't make this task easily achievable. QuTip doesn't calculate the fidelity of the current final state evolved from the current optimized pulse for each optimization iteration. Instead, it has a caching system that tries to minimize redundant calculations as the forward simulation process is very computation intensive due to the many matrix multiplications involved (which

is proportional to number of time slices and the cavity state space truncation N). As such, to implement partial trace fidelity for optimization using QuTip, one effectively needs to rewrite the bulk of the optimization code up to calling scipy optimization function. Attempt was made in this direction but was given up half way due to project time constraint. Besides, it was latered discovered that the Krotov package, a package based on top of QuTip, implements the optimziation process different from QuTip in such a was that allows for customization of fidelity. See How-to chapter, "How to define a new optimization functional" in Krotov documentation for details. However, this was not attempted due to time constraint.

Secondly, the project did not yet get to consider all physical constraints on the control pulses as discussed in chapter **??**. The only constraint that was considered was the pulse amplitude constraint. To incorporate more physical constraints into pulse optimization, CRAB algorithm could be used which allows for the incorporation of physical constraints into the optimization process. Moreover, CRAB optimizes for functional basis coefficients rather than time-slice amplitudes of control pulses. This allos for arbitrary smoothness of the physical control pulses. However, running CRAB requires more trial and error with choosing optimzation time parameters and functional basis parameters. From experience, GRAPE behaves much robustly in the QuTip implementation. One idea is to first run CRAB, aiming for a 90% fidelity and pass this optimized pulse into GRAPE for further optimizaiton. This is because CRAB usually runs faster than GRAPE, with only a small number of coefficients to be optimized. However, CRAB also has a smaller optimization landscap, making GRAPE a better choice for finer fidelity optimization.

Thirdly, one issue that was realised at the end was that the native oscillation frequency of the cavity seems to be causing the fast oscillations in the optimized pulses as seen in Fig **??**. One idea was to transform the Hamiltonian to the rotating frame at the native oscillation frequency of the cavity to alleviate this issue. However, going to the rotating frame tags a rotating phase factor to the annihilation and creation operators of the cavity. As QuTip and Krotov only optimizes for real control pulses, so far in the control Hamiltonians that I used, control signal on the cavity has been separated into real and imaginary parts. With different rotating phase factors taged to the annihilation and creation operators of the cavity, the real and imaginary parts of the control pulses cannot be separated any more. This issue was not further looked into due to time constraint.

With more time, all three above stated problems should be solvable. All three problems come from technical programming issues. With these settled, the next step would be to use the final optimized control pulses to simulate the evolution of the original full Hamiltonian and compare the state fidelity. This is to check whether the optimized pulses violate the constraints for the effective Hamiltonian to be valid. Although the above three issues were not settled, this task of checking with the full Hamiltonian was still attempted. (the same code as the Krotov code) However, the computation couldn't be completed in any reasonable amount of time. One idea is to make use of packages such as TensorFlow and CuPy to speed up matrix multiplications by through GPU acceleration. This idea was attempted but also dropped half way due to time constraints. Besides,

this idea didn't seem promising as it seemed that even with GPU acceleration, the computation time would still be unreasonable. If this task is completed and a fidelity generated. A good fidelity will signify that the control pulse is within the valid limits of the effective Hamiltonian. If the fidelity is not good, more constraints need to be added during pulse optimization.

Beyond state preparation for a close quantum system, the next step would be to consdier state preparation in an open and noisy system. Apart from switching from the Schrödinger equation to the Lindblad master equation or Liouville-von Neumann equation, this is a very similar optimization process.

Beoyond state preparations in close or open systems, a related next task would be to do control pulse optimizations for unitary gates, often called gate design. Both single-mode and multi-mode gates can be potential targets. Though many of the optimization lessons learned and implemented in this project can be re-used for such a purpose, this project did not have the time to reach this point.

Chapter 5

Conclusion

In conclusion, this report explored quantum optimal control for state preparation in a cavity-coupled double quantum dot system. Pulse optimization was successfully performed using the effective Hamiltonian to achieve high fidelity state transfers, demonstrating the potential of numerical methods for designing control protocols. However, challenges arose when applying the optimized pulses to the full physical Hamiltonian, indicating the need for further refinement of the optimization process and the inclusion of additional physical constraints. Future work should focus on improving computational efficiency, incorporating open quantum system dynamics, and extending the techniques to gate design for quantum computing applications.

Appendix A

Sample Code

```
# %% [markdown]
# # selective spin flip, full Hamiltonian
# Here we consider the full system Hamiltonian (cavity +
   auxiliary qubit + coupling terms)
# drift Hamiltoinan is given by,
# $$
# H_d = \text{wr} * n_{\text{cavity}} + \text{Ez} * \sigma_z +
   \text{chi} * \sigma_z * (n_{\text{cavity}} + 1/2 )
# $$
# where,
# - wr: resonator frequency
# - Ez: qubit anharmonicity
# - chi: qubit-cavity coupling strength
#
# Real control channels in the Hamiltonain are given by:
# $$
# H_c = a + a^{\dagger} + -1j*(a - a^{\dagger}) + \sigma_x
# $$
# state preparation optimization target:
# \begin{align*}
#     \psi_0 &= \frac{1}{\sqrt{2}} (\ket{0}_{\text{cavity}}
   \otimes \ket{0}_{\text{qubit}}
```

```
#                  + \ket{1}_{\text{cavity}} \otimes
   \ket{0}_{\text{qubit}})\\
#     \psi_{\text{targ}} &= \frac{1}{\sqrt{2}}
   (\ket{0}_{\text{cavity}} \otimes \ket{0}_{\text{qubit}}
#                  + \ket{1}_{\text{cavity}} \otimes
   \ket{1}_{\text{qubit}})\\
# \end{align*}
# where qubit spin flip is selective on cavity state

# %%
!pip install qutip

# %%
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import datetime
from qutip import *
import random
import qutip.logging_utils as logging
logger = logging.get_logger()
#Set this to None or logging.WARN for 'quiet' execution
log_level = logging.INFO
#QuTiP control modules
import qutip.control.pulseoptim as cpo

# %% [markdown]
# ## Physical system parameters

# %%
# time = 1 -> GHz and ns
# time = 1000 -> MHz and us
time = 1
# spin flip, from chiyuan, pulse length is ns scale
# vac2cat, from chiyuan, pulse length is ms scale
g = 0.05*2*np.pi *time # cavity qubit coupling strength
```

```
N = 16 # Fock space truncation
wr = 1.0*2*np.pi *time# resonator frequency
Ez = 0.22*2*np.pi *time# qubit anharmonicity
chi = 0.007 *time # qubit-cavity coupling strength

#### set up the operators ####
a = tensor(destroy(N),  qeye(2))
n_cavity = tensor(num(N), qeye(2))
Sigmaz = tensor(qeye(N),sigmaz())
Sigmax = tensor(qeye(N),sigmax())
Sigmam = tensor(qeye(N),  destroy(2))
Id = tensor(qeye(N),  qeye(2))
Sigmap = Sigmam.dag()

# H_d = wr * n_cavity + Ez * Sigmaz + chi * Sigmaz * (n_cavity +
   Id/2)
# corrected:
H_d = wr * n_cavity + Ez/2 * Sigmaz - chi * Sigmaz * (n_cavity +
   Id/2 )

H_c = [a + a.dag(),
       -1j*(a - a.dag()),
       Sigmax]


#### set Hamiltonian ####
n_ctrls = len(H_c)

H_labels = [r'$ReD$',
            r'$ImD2$',
            r'$R$']

#### initial and target states ####
alpha = np.sqrt(1) # coherent state complex amplitude

# vacuum to cat
```

```
# psi_0 = tensor(fock(N, 0), basis(2, 0))
# psi_targ = tensor((coherent(N, alpha)+coherent(N,
   -alpha)).unit(),  basis(2, 0))

# unselective spin rotation
# psi_0 = tensor(fock(N, 0), basis(2, 0))
# psi_targ = tensor(fock(N, 0), basis(2, 1))

# selective spin flip
psi_0 = (tensor(fock(N, 1), basis(2, 0)) + tensor(fock(N, 0),
   basis(2, 0))).unit()
psi_targ = (tensor(fock(N, 1), basis(2, 0)) + tensor(fock(N, 0),
   basis(2, 1))).unit() # selective wrt cavity state
U = psi_targ * psi_0.dag()

# %% [markdown]
# # Optimization parameters

# %%
# Number of time slots
n_ts = 100_000
# Time allowed for the evolution
evo_time = 500

# Fidelity error target
fid_err_targ = 1e-8
# Maximum iterations for the optisation algorithm
max_iter = 10000
# Maximum (elapsed) time allowed in seconds
max_wall_time = 7200

# pulse type alternatives: RND|ZERO|LIN|SINE|SQUARE|SAW|TRIANGLE|
# for GRAPE, this is initial pulse type
p_type = 'SINE'

# Set to None to suppress output files
```

```
# f_ext = "{}_n_ts{}_ptype{}.txt".format('v2cat', n_ts, p_type)
f_ext = None

# %% [markdown]
# ## Run the optimisation

# %%
# GRAPE
result = cpo.optimize_pulse_unitary(H_d, H_c, Id, U,
                 n_ts, evo_time,
                 fid_err_targ=fid_err_targ,
                 max_iter=max_iter, max_wall_time=max_wall_time,
                 method_params={'xtol':1e-3},
                 init_pulse_type=p_type,
                 out_file_ext=f_ext,
                 log_level=log_level, gen_stats=True)

# result  = cpo.optimize_pulse_unitary(H_d, H_c, U_0, U_targ,
#                                      num_tslots=None,
    evo_time=None, tau=None, amp_lbound=None, amp_ubound=None,
#                                      fid_err_targ=1e-10,
    min_grad=1e-10, max_iter=500, max_wall_time=180,
#                                      alg='GRAPE',
    alg_params=None, optim_params=None, optim_method='DEF',
    method_params=None,
#                                      optim_alg=None,
    max_metric_corr=None, accuracy_factor=None,
    phase_option='PSU',
#                                      dyn_params=None,
    prop_params=None, fid_params=None,
#                                      tslot_type='DEF',
    tslot_params=None, amp_update_mode=None,
#                                      init_pulse_type='DEF',
    init_pulse_params=None, pulse_scaling=1.0, pulse_offset=0.0,
#                                      ramping_pulse_type=None,
    ramping_pulse_params=None, log_level=0, out_file_ext=None,
```

```
    gen_stats=False)

# %%
result.stats.report()
print("Final evolution\n{}\n".format(result.evo_full_final))
print("********* Summary *****************")
print("Final fidelity error {}".format(result.fid_err))
print("Final gradient normal {}".format(result.grad_norm_final))
print("Terminated due to {}".format(result.termination_reason))
print("Number of iterations {}".format(result.num_iter))
print("Completed in {} HH:MM:SS.US".format(
        datetime.timedelta(seconds=result.wall_time)))

# %% [markdown]
# ## Plot control pulses

# %%
def plot_amp(result,i):
    # i is the index of the control amplitude
    plt.figure(figsize=(15, 2))
    # title and labels
    plt.title(f"Control {i+1} Amplitudes")
    plt.xlabel("Time $ns$")
    plt.ylabel("Control Amplitude $GHz$")

    # Plot the initial control amplitudes in blue
    plt.step(result.time,
             np.hstack((result.initial_amps[:, i],
                result.initial_amps[-1, i])),
             where='post',alpha = 0.7, label='Initial Amplitude')

    # Plot the final control amplitudes in red
    plt.step(result.time,
             np.hstack((result.final_amps[:, i],
                result.final_amps[-1, i])),
```

```
            where='post', color='red',alpha = 0.7, label='Final␣
               Amplitude')

    # Add a legend to the plot
    plt.legend()

    plt.tight_layout()
    plt.show()

plot_amp(result,0)
plot_amp(result,1)
plot_amp(result,2)

# %% [markdown]
# # Simulation

# %%
def cavityFidelity(psi1, psi2):
    '''
    psi1, psi2: Qobj
    '''
    return fidelity(ptrace(psi1, 0), ptrace(psi2, 0))

def qubitFidelity(psi1, psi2):
    '''
    psi1, psi2: Qobj
    '''
    return fidelity(ptrace(psi1, 1), ptrace(psi2, 1))

# %%
# H_c amplitudes (from pulse optimization)
amplitudes_0 = result.final_amps[:, 0]
amplitudes_1 = result.final_amps[:, 1]
amplitudes_2 = result.final_amps[:, 2]
```

```
tlist = np.linspace(0, evo_time, n_ts)
H_array_form =QobjEvo([H_d, [H_c[0], amplitudes_0], [H_c[1],
   amplitudes_1],[H_c[2], amplitudes_2]], tlist=tlist)

# res = qt.mesolve(H_array_form, psi_0, tlist)
res = sesolve(H_array_form, psi_0,
   tlist,options=Options(nsteps=1000000))


fidelities1 = []
for i in range(0, n_ts):
    fidelities1.append(qutip.metrics.fidelity(res.states[i].unit(),
       psi_targ.unit()))

print("sesolve final fidelity: ", fidelities1[-1])

print('cavity ptrace fidelity: ', cavityFidelity(res.states[-1],
   psi_targ))
print('qubit ptrace fidelity: ', qubitFidelity(res.states[-1],
   psi_targ))

### self-implemented forward propagation ###
from scipy.linalg import expm

psi_init = psi_0.full()
psi_end = psi_targ.full()
fidelities = []
for n in range(len(tlist) - 1):
    H = H_d + sum([H_c[i] * result.final_amps[n, i] for i in
       range(n_ctrls)])
    H = H.full()
    dt = tlist[n + 1] - tlist[n]
    o = expm(-1j * H * dt)
    psi_init = np.matmul(o,psi_init)
    fid =
       np.abs(np.matmul(np.transpose(np.conjugate(psi_init)),psi_end))
```

```
    #
    fidelities.append(fid[0])
# print(shape(psi_init))

print("self-implemented final fidelity: ", fidelities[-1][0])

# %%
# Create a new figure
plt.figure(figsize=(20, 2))

# Plot fidelities
plt.plot(fidelities, label='self-implemented')
plt.plot(fidelities1, label='sesolve')

# Set the title and labels for the plot
plt.title('Fidelity over time')
plt.xlabel('Time step')
plt.ylabel('Fidelity')

# Add a legend
plt.legend()

plt.grid(True)
# Show the plot
plt.show()
```

Code A.1: selective spin flip optimization code