Quantum Correlations and Energy Transport in Trapped Ions

by

Michael Ramm

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Physics

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Hartmut Häffner, Chair
Professor Dmitry Budker
Professor Jeffrey Bokor

Spring 2014

Quantum Correlations and Energy Transport in Trapped Ions

Abstract

Quantum Correlations and Energy Transport in Trapped Ions

by

Michael Ramm

Doctor of Philosophy in Physics

University of California, Berkeley

Professor Hartmut Häffner, Chair

We present experimental results of using trapped ions for the study of energy transport and quantum correlations in many-body systems. We investigate energy propagation by locally exciting one end of an ion chain and then observing the subsequent dynamics. The experimental results agree with the presented normal mode model of ion motion, and signify the first steps towards realizing theoretical proposals of studying many-body physics with the motional degree of freedom of trapped ions. Additionally, we present a method for detecting correlations between an open quantum system and its environment by acting only locally on the open system. We implement this method using a single trapped ion and discuss our current work towards extending the results to larger systems. Lastly, we demonstrate how the presented method for local detection of quantum correlations can be used for detection of quantum phase transitions.

To my parents Mark and Sofya and my dear Alexa

whose support made it all possible.

Contents

List of Figures

List of Tables

Acknowledgments

The past six years of graduate school at Berkeley have been a remarkable journey and I have many to thank along the way. First and foremost, my advisor Hartmut Häffner, for the opportunity to join his group, his endless patience, creativity, and supply of ideas, and the freedom to pursue the topics I found exciting. Thanks to Thaned (Hong) Pruttivarasin for working together with me on the experiment: everything from the design, first trapping, debugging, and making the final measurements. Hong has been an amazing collaborator and a great friend. Axel Kreuter was instrumental in getting the experiment up and running, and setting the right tone for the future. It was a pleasure to collaborate with Manuel Gessner and have him in lab during his stay in Berkeley. Thanks to Nikos Daniilidis for tons of useful advice and to Christopher Reilly and Mark Kokish for their contributions to the experimental control. A particular thanks to Soenke Moeller for helping the lab run smoothly. Over the years I have had the pleasure to work with many other talented labmates: Sankar, Dylan, Philipp, Ishan, Erick, Sebastian, Tony, Crystal, Greg, Gerbhard, Oliver, Boyan, Josselin, and Ahmed. I wish them the best of luck going forward. Finally, I also would like to acknowledge support from the DOE-SCGF fellowship.

Chapter 1

Introduction

Chains of trapped ions are among rare physical systems that allow for control of many interacting particles at a single quantum level. The original experiments demonstrating confinement of charged particles by Wolfgang Paul in 1950s using oscillating electric fields and trapping of a single electron by Hans Dehmelt in 1973 using a combination of static magnetic and electric fields have led to inestimable progress in the field of atomic physics. The invention of laser cooling by the groups of Dave Wineland and Werner Neuhauser in 1978 allowed to dramatically reduce the kinetic energies of the trapped particles. The development of sideband cooling allowed to further reduce the motional energy, reaching the ground state of motion, improving the ability to manipulate the internal state of the atom with laser fields. These advancements led to remarkable refinement of precision measurements of fundamental constants, spurred laser cooling experiments with neutral atoms, and laid the groundwork for emerging fields such as quantum information processing and quantum simulation.

In particular, technological advancements in the field of atomic physics have made it possible to experimentally study the role of quantum effects in the dynamics of many-body systems. This topic has been the subject of a multitude of theoretical pursuits covering far-ranging topics from the connection of quantum correlations and quantum phase transitions to consequences of quantum effects for the efficiency of energy propagation in photosynthetic complexes. There have been several experiments successfully realizing a quantum phase transition with trapped ions. However, many theoretical ideas concerning the presence of quantum correlations during these transitions and, more generally, the effect of the quantum correlations on the system dynamics remain unexplored.

Even in the classical regime where quantum correlations do not play a role, the motional dynamics of long ion chains presents a compelling subject of study. By increasing the number of particles from tens to hundreds, one may begin to observe the emergence or deviations from expected thermodynamic quantities, thus, experimentally validating the fundamental origins of statistical mechanics. There have been no experimental measurements of the thermodynamic quantities due to the difficulties of controlling long ion chains.

The next five chapters of this dissertation provide a detailed description of several experiments performed in pursuit of these goals. Chapter 2 reviews the theory of ion trapping

and laser-ion interaction. Then Chapter 3 describes the experimental setup for trapping and controlling long chains of ions. Chapter 4 focuses on classical measurements of energy transport across the chain. Chapter 5 then demonstrates a technique for detecting quantum correlations between two quantum systems with only access to one of the two systems. Finally the ideas of the previous chapters are combined in Chapter 6, where we make a connection between dynamics of the chain motion and the detected quantum correlations over the course of the evolution. The appendix covers numerous technical details.

Chapter 2

Ion Trapping and Laser Interaction

In this Chapter we review the principles of ion trapping and the interaction of the confined ions with laser light. In the first section, we describe the use of oscillating electric fields to confine charged particles in a three-dimensional harmonic potential. In the second section, we describe the interaction between the confined ions and laser light, covering the processes of Doppler cooling using a dipole transition and the coherent manipulation of energy levels of a quadrupole transition. These topics have been extensively described in many textbooks, so we focus on the main results that will be needed for the experiment described in the thesis.

2.1 Ion Trapping

According to Earnshaw's theorem, point charges can not be confined in a stable equilibrium using static electric fields. To trap ions we, therefore, require dynamic or time-varying electromagnetic fields. Following the treatment of Leibfried et al. [2], we consider a quadrupole potential ϕ that is composed of static and time-varying components:

$$\phi(x,y,z,t) = \frac{U}{2}\left(\alpha x^2 + \beta y^2 + \gamma z^2\right) + \frac{\tilde{U}}{2}\cos\left(\omega_{\mathrm{rf}} t\right)\left(\alpha' x^2 + \beta' y^2 + \gamma' z^2\right) \ , \tag{2.1}$$

where U and $\tilde{U}$ are the amplitudes of the static and dynamic components, and ω_{rf} is the angular frequency of the oscillating electric field. The typical configuration of Paul trap electrodes and the generated time-varying potential are depicted in Figure 2.1. We analyze the motion of the ion in the trap by considering the classical equations of motions along the x axis for a particle of mass m and charge $Z|e|$:

$$m\ddot{x} = -Z|e|\frac{\partial \phi}{\partial x} = -Z|e|\left(U\alpha + \tilde{U}\alpha'\cos\left(\omega_{\mathrm{rf}} t\right)\right) x \ . \tag{2.2}$$

This differential equation is equivalent to the Mathieu equation:

$$\frac{d^2x}{d\xi^2} + \left(a_x - 2q_x\cos\left(2\xi\right)\right) x = 0 \ , \tag{2.3}$$

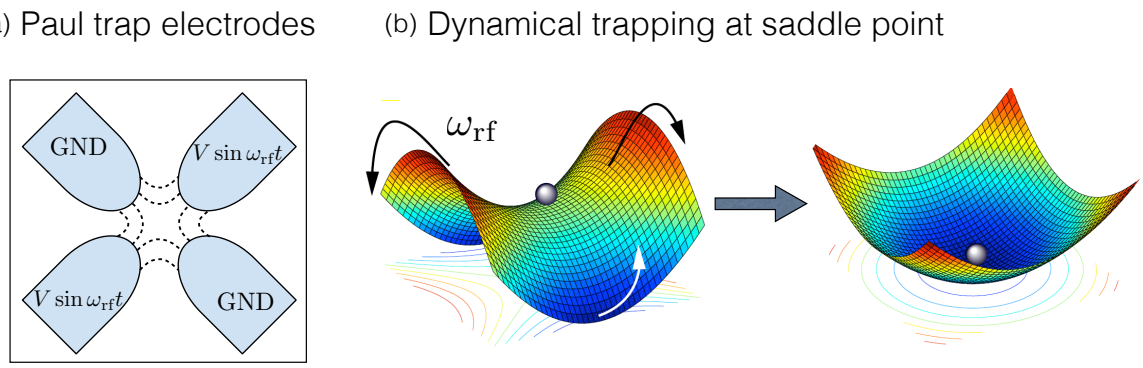

Figure 2.1: On the left is the standard Paul trap electrode configuration. An oscillating voltage with frequency $\omega_{\rm rf}$ is applied to one pair of electrodes while the other pair is held at ground. Dashed lines indicate the resultant electric field of the generated quadrupole potential. The oscillating quadrupole potential is depicted on the right. The ion is trapped at the saddle point of the potential where the electric field strength is zero. The diagram is adapted from [1].

where ξ , a_x and q_x are dimensionless variables representing time, strength of static and oscillating fields, respectively:

$$\xi = \frac{\omega_{\rm rf} t}{2} \ , \tag{2.4}$$

$$a_x = \frac{4Z|e|U\alpha}{m\omega_{\rm rf}^2} \ , \tag{2.5}$$

$$q_x = \frac{2Z|e|\tilde{U}\alpha'}{m\omega_{\rm rf}^2} \ . \tag{2.6}$$

The solution to the Mathieu equation is studied by Floquet theory. While the most general solution has no closed form,

$$x(\xi) = Ae^{i\beta\xi} \sum_{n=-\infty}^{\infty} C_{2n} e^{i2n\xi} + Be^{-i\beta\xi} \sum_{n=-\infty}^{\infty} C_{2n} e^{-i2n\xi} \ , \tag{2.7}$$

the lowest-order approximation to the ion motion in the stability region is given by [2]:

$$x(t) = 2AC_0 \cos(\beta_x \frac{\omega_{\rm rf}}{2} t)(1 - \frac{q_x}{2} \cos(\omega_{\rm rf} t)) \ , \tag{2.8}$$

where β_x is defined as:

$$\beta_x = \sqrt{a_x + \frac{q_x^2}{2}} \ . \tag{2.9}$$

We observe that the ion exhibits oscillatory motion at the secular angular frequency $\omega = \beta_x \omega_{\mathrm{rf}}/2$ superimposed with oscillation at the driving frequency ω_{rf}. The latter is called micromotion because its amplitude is reduced by the factor $q_x/2$. Disregarding the driven motion at the frequency ω_{rf}, the ion's secular motion corresponds to confinement in a harmonic potential with the natural frequency ω. This result may also be derived by making the pseudo potential approximation [3]. The effective potential is given by

$$U = \frac{Q^2}{2m\omega_{\mathrm{rf}}^2}\langle E_x^2 \rangle \ , \tag{2.10}$$

where Q is the total charge $Q = Z|e|$, and $\langle E_x^2 \rangle$ is the time-averaged square of the electric field.

2.2 Ion-Laser Interactions

2.2.1 Coherent operations

In this section, we consider the interaction of a two-level confined ion with a traveling light wave. This analysis applies to the laser at 729 nm interacting with the narrow $S_{1/2} - D_{5/2}$ transition of Ca^+. The transition will be employed in the experiments for determining the state of the ion motion and preparing quantum correlations between the ion's electronic state and its motion.

As shown in section 2.1, the ion is effectively confined in a harmonic potential with the oscillation frequency ω. Additionally, we assume that the ion has two electronic energy levels with the atomic transition frequency ω_a. The bare Hamiltonian H_0 is then given by

$$H_0 = \frac{p^2}{2m} + \frac{1}{2}m\omega^2 x^2 + \frac{1}{2}\hbar\omega_a\sigma_z \ , \tag{2.11}$$

where p is the ion momentum and σ_z is the third Pauli spin matrix. When the ion is interacting with the traveling light wave, the total Hamiltonian is the sum of the bare Hamiltonian H_0 and the coupling Hamiltonian H_c [4, 5, 2]:

$$H_c = \frac{1}{2}\hbar\Omega_0 \left(\sigma_+ + \sigma_-\right)\left(e^{i(kx-\omega_L t+\phi)} + e^{-i(kx-\omega_L t+\phi)}\right) \ , \tag{2.12}$$

where ω_L, k, and ϕ are the frequency, wave vector, and phase of the traveling wave, and σ_+ and σ_- are the raising and lowering operators. The Rabi frequency Ω_0 describes the interaction strength in the presence of micromotion. The ion motion in the trap is quantized in the standard way:

$$x = \sqrt{\frac{\hbar}{2\omega m}}(a^\dagger + a) \ , \tag{2.13}$$

$$p = i\sqrt{\frac{\hbar}{2\omega m}}(a^\dagger - a) \ , \tag{2.14}$$

using the raising and lowering harmonic oscillator operators $a^\dagger$ and a. The total number of vibrational quanta, or phonons, is defined as the expectation value of the number operator $N = a^\dagger a$. Additionally, we define the Lamb-Dicke η parameter as the ratio of the traveling wave wavelength and the extent of the ion's ground state wave function:

$$\eta = k\sqrt{\frac{\hbar}{2m\omega}} \ . \tag{2.15}$$

We calculate the interaction Hamiltonian by performing the transformation $H_{\text{int}} = U^\dagger H_c U$ with $U = \exp[-iH_0 t/\hbar]$. Performing the rotating-wave approximation yields [2]

$$H_{\text{int}} = \frac{1}{2}\hbar\Omega_0 \left(\sigma_+ e^{i\eta(ae^{-i\omega t}+a^\dagger e^{i\omega t})} e^{i(\phi-\delta t)} + \sigma_- e^{-i\eta(ae^{-i\omega t}+a^\dagger e^{i\omega t})} e^{-i(\phi-\delta t)}\right) \ , \tag{2.16}$$

where $\delta = \omega_L - \omega_a$ is the detuning between the laser and the atomic transition. Controlling the laser detuning allows to resonantly couple the ion's electronic and motional states. Setting the detuning $\delta \approx l\omega$ for some integer l couples the states in the form $|g\rangle|n\rangle$ and $|e\rangle|n+l\rangle$ with the Rabi frequency $\Omega_{n,n+l}$, where $|g\rangle$ and $|e\rangle$ are the electronic states of the atom and $|n\rangle$ is the number state with n phonons in the trap. The coupling strength $\Omega_{n,n+l}$ is given by

$$\Omega_{n,n+l} = \Omega_0 |\langle n+l|e^{i\eta\left(a+a^\dagger\right)}|n\rangle| \ , \tag{2.17}$$

where the matrix element may be expressed in terms of a Laguerre polynomial [2]. The transitions can be driven individually in the resolved sideband regime: small detuning $\delta - l\omega \ll \omega$, sufficiently weak coupling $\Omega_{n,m} \ll \omega$ for all possible n and m, and narrow excited state linewidth $\Gamma \ll \omega$. The integer l is called the order of the sideband. The first blue sideband corresponds to $l = +1$ and the first red sideband is $l = -1$ while $l = 0$ is known as the carrier excitation. The ion trapping experiments are typically conducted in the Lamb-Dicke regime where the spatial extent of the ion's wavepacket is much less than the wavelength of the atomic transition, $\eta \ll 1$. In this regime, the interaction Hamiltonian may simplified by expanding for small η. Particularly for the first red sideband $l = -1$, $H_{\text{int}} \equiv H_{\text{rsb}}$ reproduces the Jaynes-Cummings Hamiltonian:

$$H_{\text{rsb}} = \frac{1}{2}\hbar\Omega_0\eta \left(a\sigma_+ e^{i\phi} + a^\dagger \sigma_- e^{-i\phi}\right) \ . \tag{2.18}$$

This interaction is used to perform sideband cooling, as described in Figure 2.2. The first blue sideband $l = 1$ gives rise to the anti-Jaynes-Cummings interaction:

$$H_{\text{bsb}} = \frac{1}{2}\hbar\Omega_0\eta \left(a^\dagger\sigma_+ e^{i\phi} + a\sigma_- e^{-i\phi}\right) \ . \tag{2.19}$$

The red and blue sideband interactions are depicted in Figure 2.3.

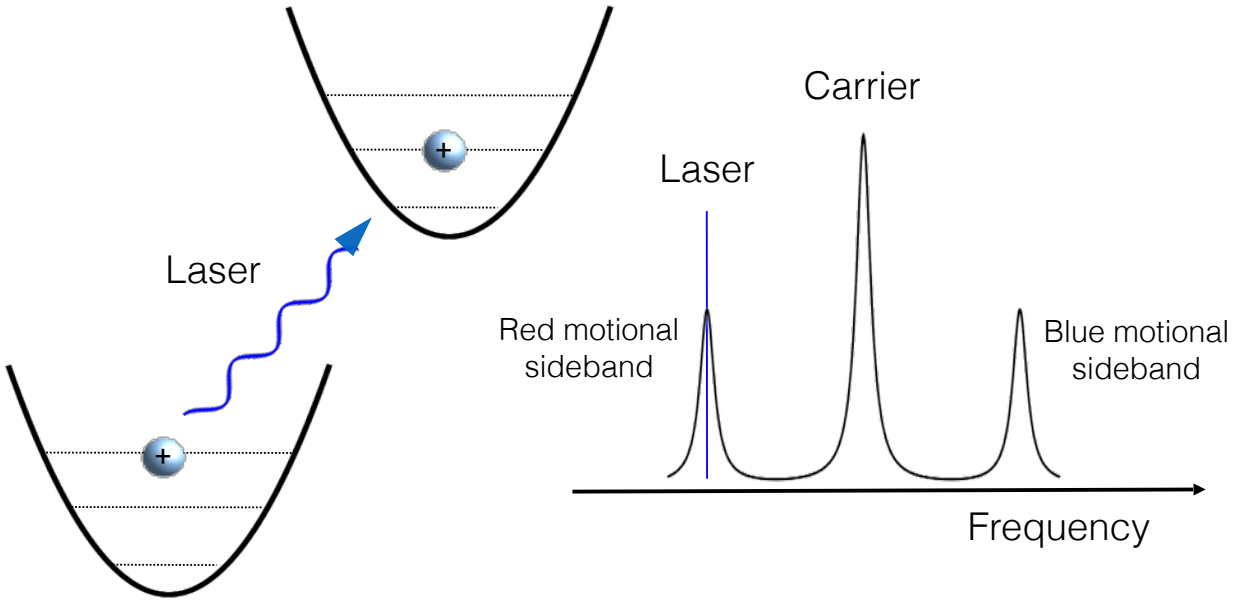

Figure 2.2: Sideband cooling is performed in the regime of resolved motional sidebands. The laser frequency is tuned to resonance with the first red motional sideband: the laser frequency is equal to the frequency of the carrier transition minus the trap frequency. When the laser promotes the ion from the ground state to the excited electronic state, the number of phonons decreases by 1. The ion then spontaneously decays back to the electronic ground state, or is repumped back to the ground state with an additional laser.

2.2.2 Doppler cooling

We perform Doppler cooling to reduce the kinetic energy of the trapped ions and conduct experiments in the Lamb-Dicke regime. We use the $S_{1/2}$–$P_{1/2}$ transition with a short lifetime where the resolved-sideband condition does not hold due to the large transition linewidth $\Gamma \gg \omega$. The intuitive scheme of Doppler cooling is presented in Figure 2.4. Mathematically, the process is treated semi-classically. We assume that the two-level atom is interacting with a laser detuned by Δ from the atomic transition. Each scattered photon transfers momentum onto the atom, resulting in the radiation pressure force F,

$$F = \hbar k \Gamma \rho_{\mathrm{ee}} \ , \tag{2.20}$$

where ρ_{ee} is the excited state probability,

$$\rho_{\mathrm{ee}} = \frac{s/2}{1 + s + (2\delta_{\mathrm{eff}}/\Gamma)^2} \ , \tag{2.21}$$

with the saturation parameter $s = 2\Omega^2/\Gamma$ and the effective detuning $\delta_{\mathrm{eff}} = \Delta - kv$. The excited state probability is greatest when the laser is on resonance $\delta_{\mathrm{eff}} = 0$. For a red

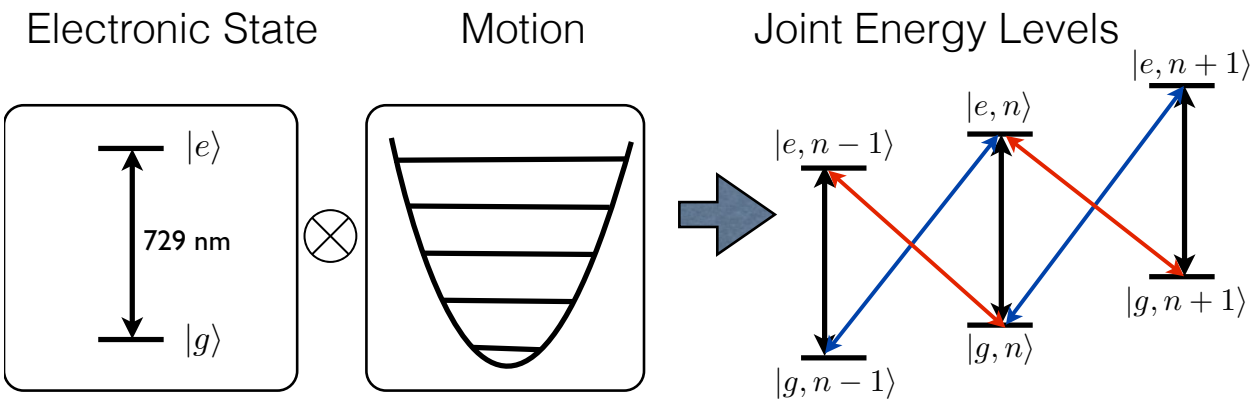

Figure 2.3: The laser at 729 nm couples the electronic and the motional states of the ion. When the laser is tuned to the frequency of the carrier transition (black), the motion is not affected. Tuning the laser to the red motional sideband (red) couples the states in the form $|g, n\rangle$ and $|e, n-1\rangle$. This process is used to perform sideband cooling. Tuning the laser to the blue sideband (blue) couples the states in the form $|g, n\rangle$ and $|e, n+1\rangle$. This interaction will be used to create quantum correlations between the electronic state and the motional degree of freedom.

detuning Δ, the atom is more likely to absorb a photon when it's moving towards laser than away from it, resulting in a net cooling effect. For small velocities, the force may be linearized $F = F_0(1 + \kappa v)$ where the prefactor $F_0 = F(v = 0)$ and the viscosity coefficient $\kappa = \frac{\partial F}{\partial v}(v = 0)$. The viscous force reduces the atom's kinetic energy, but the random nature of the scattering events provides a heating mechanism. The heating and the cooling are balanced at the equilibrium temperature known as the Doppler limit,

$$T_{\min} = \frac{\hbar\Gamma\sqrt{1+s}}{4k_{\mathrm{B}}}(1+\xi) \ , \tag{2.22}$$

where k_{B} is the Boltzmann's constant and ξ is the geometric projection of an emission recoil kick onto the considered axis, $\xi = 2/5$ for dipole emission.

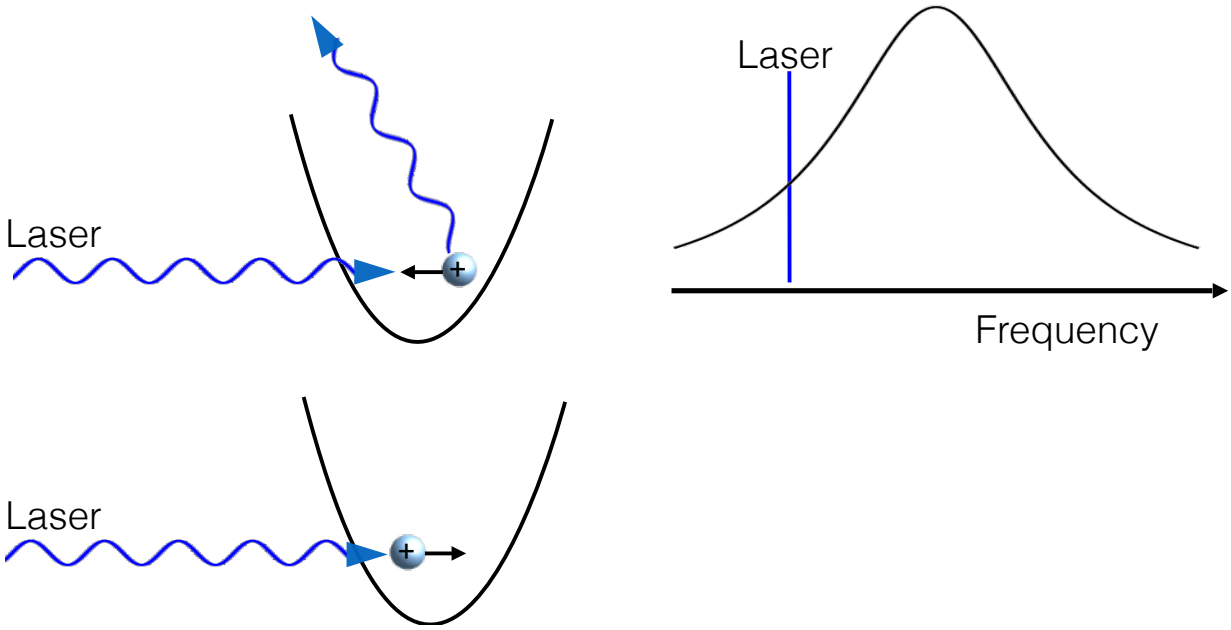

Figure 2.4: The schematic of the Doppler cooling process. The laser frequency is set to be below the atomic transition. Due to the Doppler effect, the ion is more likely to absorb a photon when it is moving in the direction opposite to the laser wave vector. The absorbed photon is reemitted in a random direction. The net effect leads to a decrease in the ion's kinetic energy.

Chapter 3

Experimental Setup

In this section we describe the ion trapping apparatus used for the presented experiments. The design goals for the setup include construction of a general-purpose ion trap to study energy transport in ion chains, and an in-vacuum optical cavity with the purpose of localizing the ions in the standing wave and performing quantum simulations of oscillator chain models. We, first, describe design and operation of the ion trap constructed to accommodate these goals. We also present the ultra-high-vacuum (UHV) chamber that houses the trap and provides optical access for ion-laser interactions. The next section focuses on the improved design for the next generation of the trap and the chamber, both currently under construction. The last sections describe imaging the ions, the electronics used in the experiment, and gives an overview of experimental control software.

3.1 Trap Design

Selecting a suitable ion trap design was a crucial consideration in planning the entire experiment. To perform experiments with ion chains, it was clear that we had to use a linear ion trap to generate the suitable confining potential. Additionally, our goal was to overlap the chain of confined ions with a standing light wave. To achieve the best stability of the standing wave with respect to the ions, we wanted to generate the standing wave with an optical cavity placed inside the vacuum chamber. Both the trap and the cavity mirrors would be subject to the same vibrations, minimizing their relative movement. The selected ion trap, therefore, had to fulfill additional requirements: it had allow for optical access in the direction along the ion chain, and had to provide the means of precisely positioning the standing wave with respect to the ions.

The main choice for the trap geometry consisted of either using three-dimensional trap geometry similar to Paul's original design or using a surface ion trap, which were a relatively new technology at the time of experiment planning. The main advantage of the surface design is that the electrodes are precisely patterned using standard lithography techniques. The exact positions of the ions is known ahead of time, making it easier to overlap the standing

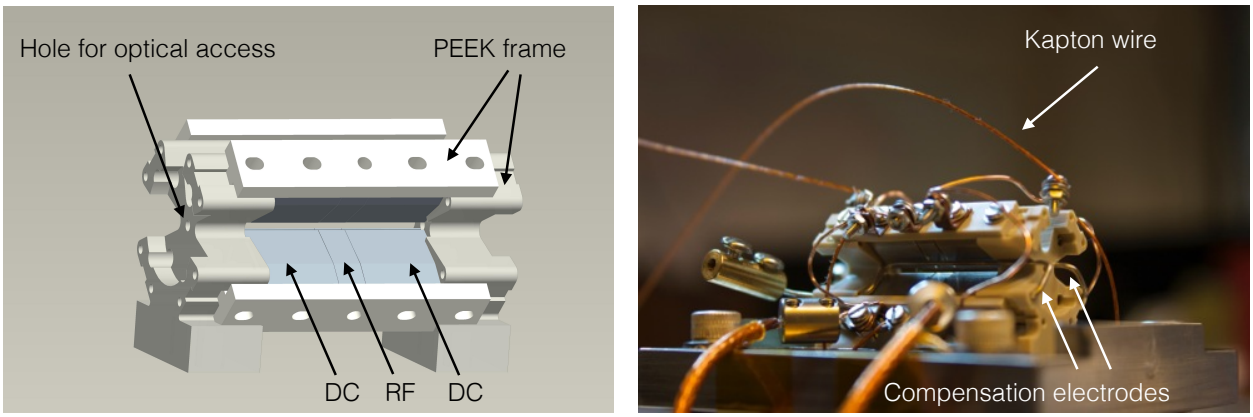

Figure 3.1: On the left is the schematic for the linear ion trap showing the DC and RF stainless steel electrodes supported by a PEEK frame. An additional hole in the frame provides axial optical access. The gap between the electrodes is 1 mm and the width of the RF electrode is 3 mm. On the right is the assembled and wired trap as installed in the experiment.

light wave. The ion is trapped hundreds of microns above the surface, providing optical access from all directions. However, due to the short distance between the ions and the electrodes, the surface traps were known to exhibit surface charging effects and anomalously high heating rates.

In contrast to surface traps, the technology of three-dimensional traps was well established. They had been used for decades in a number of laboratories around the world, and were known to be reliable in operation. In the three-dimensional designs, the ions are trapped farther from the nearest electrodes (typically on the order of 500 μm), avoiding the undesired surface effects. The main disadvantage is that these traps consist of many delicate parts that require precise machining. This makes it challenging to prealign the optical cavity to the eventual ion position. Typically the axial confinement is provided by endcap electrodes which block optical access along the chain of trapped ions.

Above all, we wanted the experiment to operate reliably, and so after taking all of the presented considerations into account, we chose to use a three-dimensional trap. We based the design on the trap used at the University of Innsbruck [6], altering the model to make it compatible with our requirements.

The technical drawing and the constructed trap are shown on Figure 3.1. The electrodes are made of 316L stainless steel, while the supporting frame is constructed from PEEK, an UHV-compatible plastic material. After machining, the stainless steel electrodes were electropolished in sulfuric acid using recipe SS-4 in [7]. The gap between the electrodes is

1 mm and the center blades are 3 mm in length. The end pieces have a central hole for optical access along the ion chain. The shape of the endcap electrodes of the Innsbruck design was modified to match the blade geometry of the RF electrodes. The electrodes are squeezed together and we use Kapton tape to insulate them electrically. The voltage breakdown between adjacent electrodes was tested under vacuum for DC voltages up to 4 kV. The voltage values for trap operation are stated in section 3.6. We typically operate the trap with the radial confinement frequency of 3 MHz and much lower axial confinement of 200 kHz.

The standard linear ion trap is operated with a pair of radio frequency (RF) electrodes and a pair of electrodes held at ground. Due to the finite length of the electrodes, this configuration results an oscillating potential along the length of the trapping axis. This oscillating potential results in a driven axial motion of the ions called micromotion. In order to avoid this effect, we supply the voltage in an out-of-phase configuration. One pair of electrodes is still supplied with RF while the other pair is supplied with the same RF 180 degrees out of phase. The produced potentials cancel along the trap axis, avoiding the axial micromotion.

3.2 Vacuum Chamber

The optical cavity is designed for the wavelength of 405 nm, far detuned form the $S_{1/2} - P_{1/2}$ transition. The mirrors have reflectivity of 99.5% and the radius of curvature of 38 mm. The resultant cavity was set up in a near-concentric configuration with the waist of 25 μm, and finesse of 600. Both the ion trap and the optical cavity were mounted inside a spherical octagon[1] vacuum chamber, as shown on Figure 3.2. The cavity mirrors were glued to a U-shaped stainless steel piece, which was attached to an XYZ manipulator[2] and free to move relative to the trap. For attaching the cavity mirrors, we used ultra-violet curable, low outgassing epoxy[3]. The plan was to first trap the ions and then position the cavity to overlap the waist of the cavity with the ion position. Once overlapped, the cavity could be secured in place with a screwdriver inside the vacuum chamber attached to a wobble stick[4].

We conducted several preparatory bakes to clean the parts and the chamber as much as possible before the final pumpdown. All of the stainless steel parts that were machined in a machine shop were cleaned in acetone and isopropanol and baked under vacuum at 300°C for one week to remove all the oil residue and other contaminants. We conducted the same bakeout procedure for all of the vacuum chamber parts that could withstand the high temperature, using blank flanges instead of viewports. The glued cavity was baked to confirm the UHV properties of employed epoxy and make sure that it stays aligned despite the temperature ramps.

[1] Kimball Physics MCF600-SO200800
[2] Thermionics EC-1.39-2-B600
[3] Dymax OP-67-LS
[4] Thermionics FWS-44R-275-2

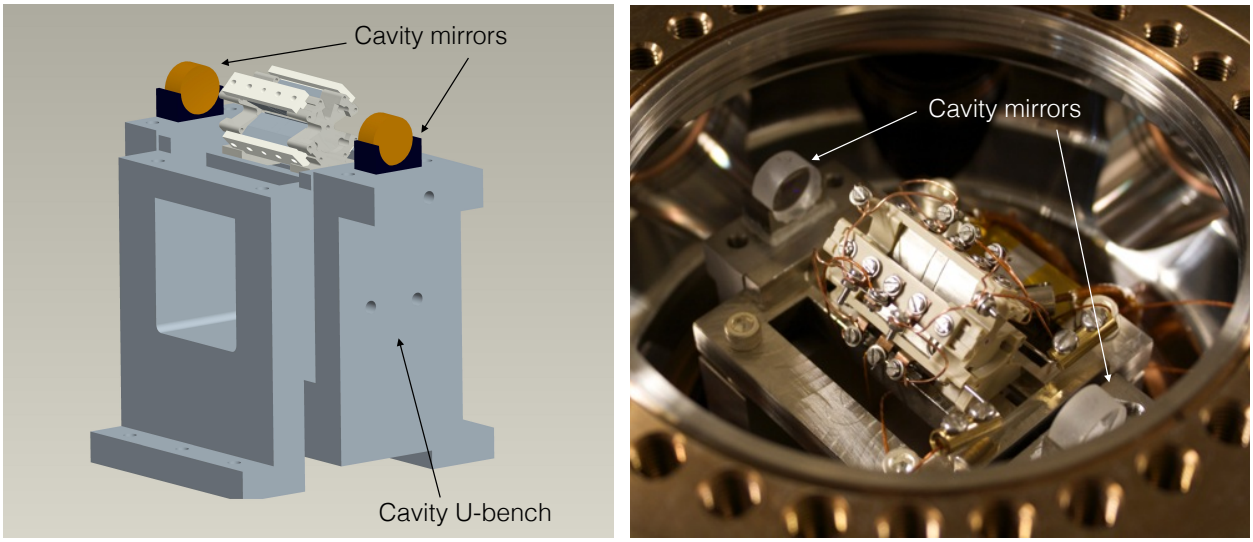

Figure 3.2: On the left is the schematic of the ion trap together with the optical cavity at 405 nm. The cavity mirrors are attached to the U-shaped stainless steel piece that is free to move independently from the ion trap. On the right are the ion trap and the cavity placed inside the UHV chamber. The distance between the mirrors is close to 76 mm.

The final assembly was baked for three weeks at a moderate temperature of 200°C compatible with the XYZ manipulator and the wobble stick. After the final bake, the pressure inside the chamber reached $\sim 2 \times 10^{-9}$ torr. At this point we also activated the Titanium sublimation pump to pump out the present hydrogen. We discovered that the pressure was limited by a small leak in the angle valve. In the process of leak testing we sprayed the problematic flange with water, managing to seal the leak and reducing the pressure by one order of magnitude to $\sim 2 \times 10^{-10}$. Since then, the experiment has been under vacuum for four years with the pressure improving beyond the measurement limit of the ion gauge. The current pressure inside is less than $\sim 5 \times 10^{-11}$ torr, leading to very few observed collisions of trapped ions with the background gas even while working with long ion chains. We typically observe a single collision-induced excitation to a dark state per hour in a chain of 10 ions.

3.3 Design Improvements

The presented chamber was our first assembled in Berkeley and we have been fortunate to have conducted the first generation of experiments without ever breaking vacuum. In this section, we describe the shortcomings of the current design and our plans for the second generation trap and chamber, currently under construction.

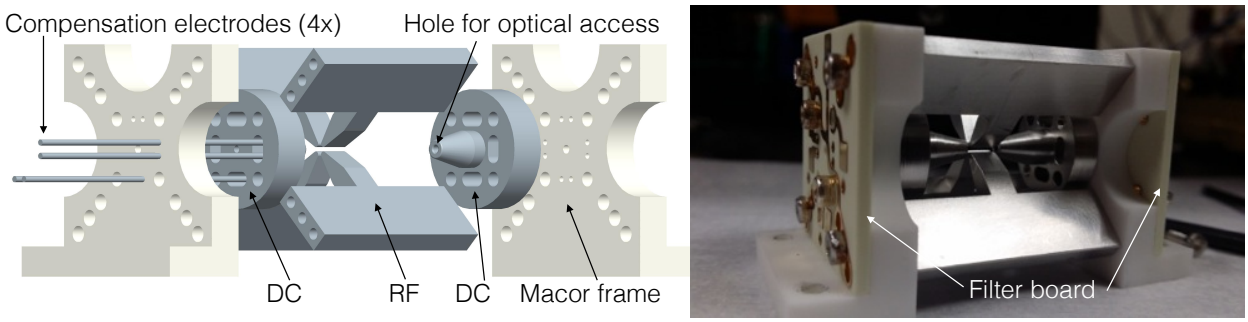

Figure 3.3: On the left is the schematic for the improved linear ion trap. All of the electrodes are supported by a Macor frame. The DC electrodes have holes for optical access. The design includes four compensation electrodes to circumvent shielding effects. On the right is the assembled trap with the attached filter boards. The width of the RF blade is 4 mm.

3.3.1 Trap

The PEEK material used for the insulating parts of the chamber was found to be too soft to allow precise positioning of the trap electrodes. In the new design of the trap, it has been replaced with Macor. Ensuring relative alignment between the DC and RF blades was particularly problematic so in the new design we modified the endcap electrodes. They are nearly cylindrically-symmetric and have a hole in the center to allow for axial optical access. While all of the stainless steel electrodes in the new design are secured with screws to the Macor support pieces, their placement is controlled with tightly fitting dowel pins.

The current trap design included only two compensation electrodes, as shown on Figure 3.1. We assumed that the common voltage on the two electrodes would produce a vertical electric field at the ion position, while the differential voltage would produce a horizontal electric field, thus allowing to compensate micromotion in both of the directions. In practice we found that even for common voltages up to 2 kV on the compensation electrodes, the ion displacement in the vertical direction was minimal. We attribute this to the shielding effect of the stainless steel blades. The new design will have two compensation electrodes on the bottom and two on the side, allowing to compensate micromotion in both directions even if the produced electric fields are shielded. The design also includes a mounted filter-board to filter out electrical noise as close as possible to the electrodes.

3.3.2 Chamber

In the current setup, we have not been able to in couple laser light into the assembled cavity. This prevented us from conducting experiments realizing the Frenkel-Kontorova model in

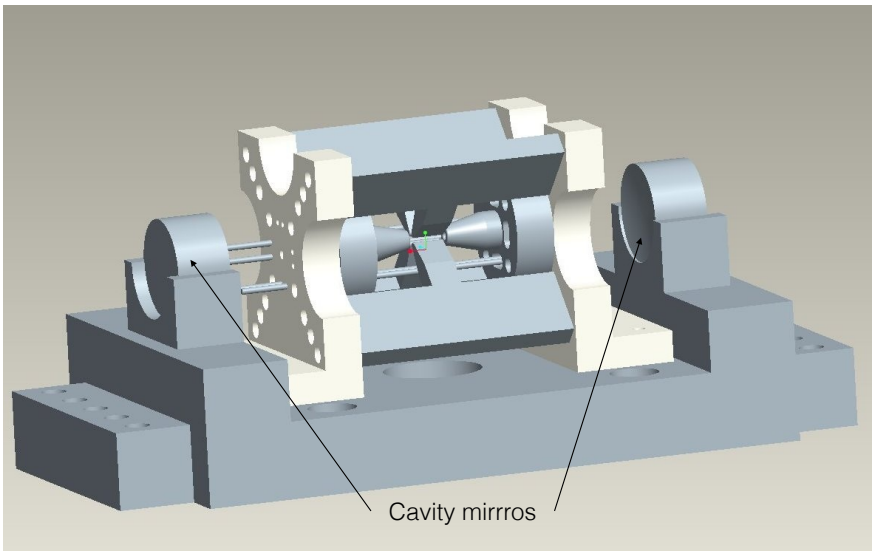

Figure 3.4: The new trap assembly shows two cavity mirrors attached to the same support piece as the ion trap. The distance between the mirrors is 10 cm.

the current apparatus[5]. There are several possible explanations for the cavity misalignment. Even though the optical cavity was tested to withstand the bakeout temperatures, it could still have misaligned during the final pumpdown. After the bake, the XYZ translation stage holding the cavity became coarser and more difficult to maneuver. It's also possible that the beam of neutral calcium from the oven was not sufficiently well collimated and covered the cavity mirrors. The actual cause of the problem will only become known when the current setup is opened, but the new design of the chamber already addresses the most likely explanations.

In order to make cavity less prone to misalignment during temperature ramps, the new design calls for a near confocal instead of a near concentric configuration. The larger resultant waist and the corresponding decrease of the light intensity at the ion position will be compensated by higher cavity finesse resultant from higher reflectivity of the cavity mirrors. The new design also involves a simpler construction: the cavity will have a larger waist, making it easier to prealign it to the trap. The cavity will be glued directly on the same mount as the trap. There is no longer a need for the added complexity of the in-vacuum XYZ translation stage and the wobble stick.

The new design of the chamber introduces several improvements. With the removal of the wobble stick and component rearrangements, there will be better optical access with seven available viewports. The chamber is electrically insulated from the optical table, reducing the noise pickup. The octagon is electrically insulated from the grounded ion pump via an

[5]For more on the proposed experiments to detect the sliding to pinned phase transition in the Frenkel-Kontorova model see Refs. [8, 9].

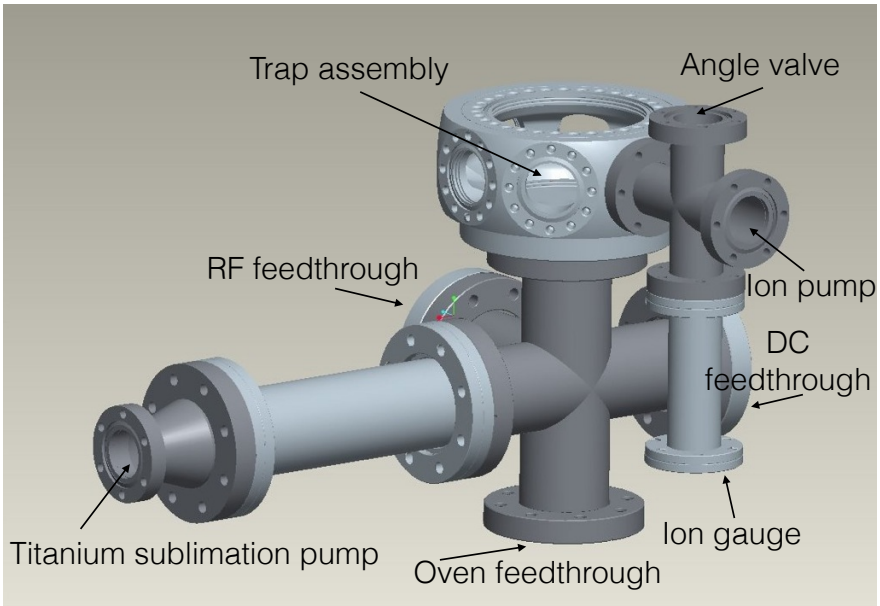

Figure 3.5: The new UHV Chamber improves on the design of the current setup. The feedthroughs for the RF and DC electrodes are separated. The trap assembly is placed inside the octagon, providing seven viewports for optical access.

insulating nipple. The DC and RF feedthroughs are moved to separate flanges, allowing to filter the DC wires as close as possible to the chamber. Finally a new oven construction with an additional pinhole improves collimation of the neutral calcium beam.

3.4 Level Structure and Lasers

For the experiments we use a number of different laser sources. They, along with the configuration of the double passes, are described in detail in Thaned Pruttivarasin's thesis [9] so here we provide a general summary. For calcium photoionization we use laser light at 422 nm, generated by frequency doubling a diode laser at 844 nm, and a 375 nm laser. The level scheme of ionized calcium is shown in Figure 3.6. We perform Doppler cooling with the diode laser at 397 nm to address the $S_{1/2} - P_{1/2}$ transition. The lasers at 866 nm and 854 nm repump the ion from $D_{3/2}$ and $D_{5/2}$ states, respectively. The 397 nm laser and both repumping lasers are referenced to optical cavities in order to stabilize their frequencies. The diode laser at 729 nm is locked to a high finesse cavity and addresses the narrow $S_{1/2} - D_{5/2}$ transition.

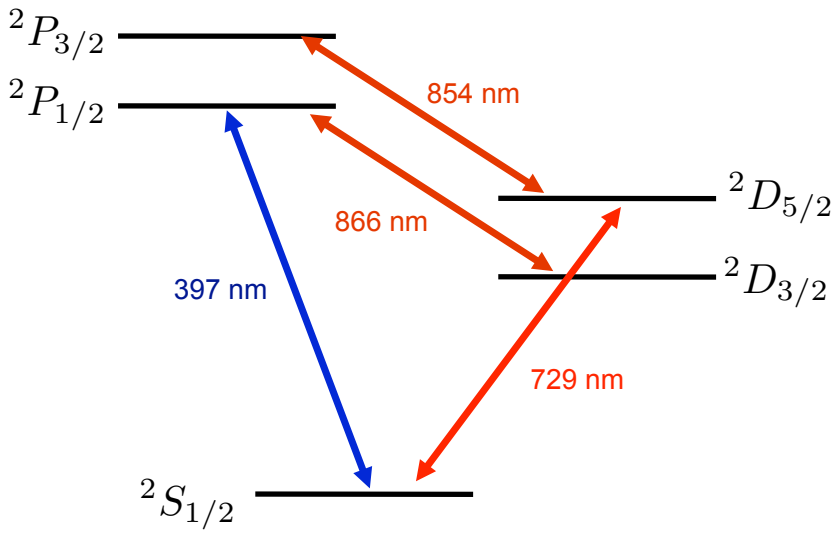

Figure 3.6: Level scheme $^{40}Ca^+$. We perform Doppler cooling on the $S_{1/2}$–$P_{1/2}$ transition. The laser at 729 nm addresses the narrow $S_{1/2}$–$D_{5/2}$ transition. The electron shelving method allows to detect with high fidelity whether the ion is in the ground state $S_{1/2}$ or in the excited state $D_{5/2}$.

3.5 Imaging

The purpose of the imaging system is to collect as much fluorescence as possible from the trapped ions, and in addition, to narrowly focus onto the ions the laser beams at 729 nm. The fluorescence is collected through the top viewport of the chamber. The imaging diagram is shown in Figure 3.7. The ion fluorescence is focused with a custom-made objective[6] designed to correct for chromatic aberrations at 397 nm and 729 nm [10]. The objective is approximately 70 mm away from the trapped ions. It can be precisely positioned with a manual XYZ translation stage[7]. At the working distance of the objective, the image is focused approximately 75 cm away. The collected light is reflected by a 2" silver mirror chosen to be highly reflective at both 397 nm and 729 nm. After the mirror, the light is sent into the imaging box. Inside, it is split with a 55% / 45% pellicle beamsplitter[8] between a

[6] Silloptics
[7] Thorlabs PT3/M
[8] Thorlabs BP245B5

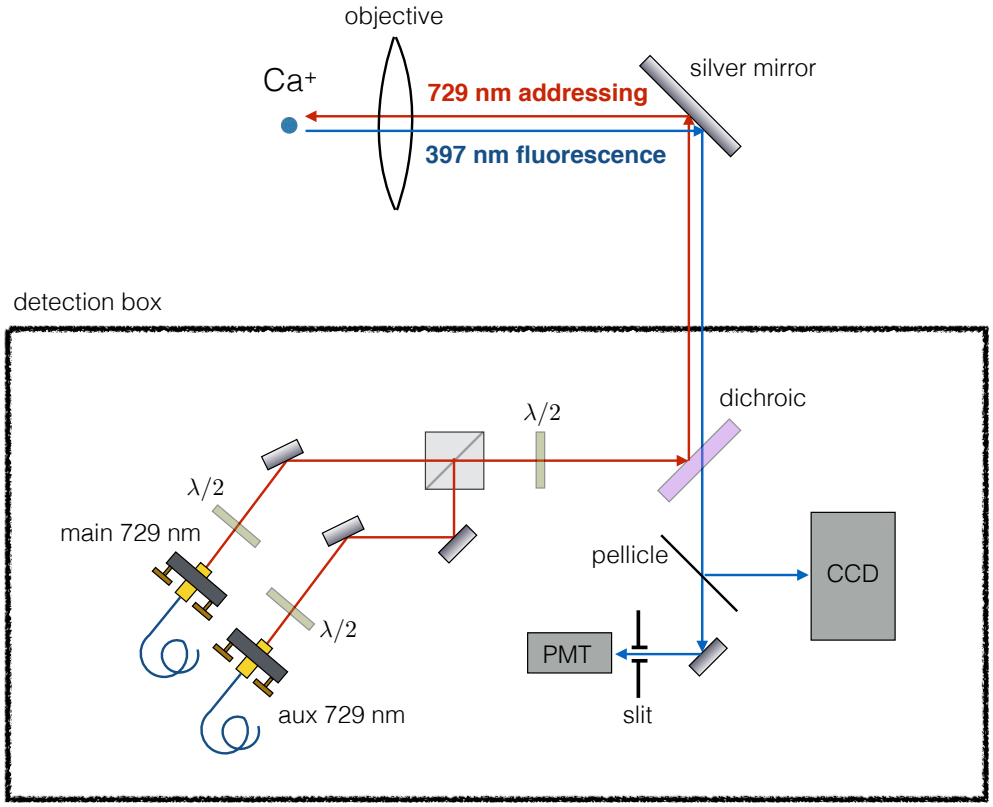

Figure 3.7: The imaging diagram from Thaned Pruttivarasin's thesis [9]. The fluorescence from the ion is collected with an objective and then split among a CCD and a PMT. Two laser beams at 729 nm are overlapped on a beam cube and are counter-propagated along the imaging system.

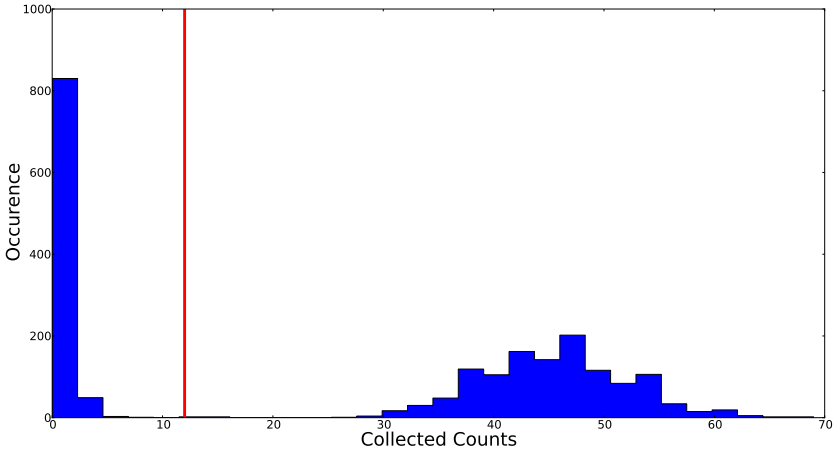

Figure 3.8: Photons collected by the PMT in 3ms. The two distinct distribution arise from the cases when the ion is in the $D_{5/2}$ state (left) and the $S_{1/2}$ ground state (right).

photomultiplier tube (PMT)[9] and an electron-multiplying (EM) CCD Camera[10,11]. Light at other wavelengths than 397 nm is filtered out with two band-pass filters[12]. As much of the optical path as possible is enclosed with plastic 2" tube to prevent stray light. We typically collect a maximum of $\sim 30,000$ counts per ion per second with the PMT. The overall detection efficiency for the setup was measured to be 1.2×10^{-3} by using a series of population transfers [11].

Two laser light beams at 729 nm are overlapped inside the imaging box. They are counter-propagated along the imaging system using a dichroic mirror in order to produce a tight focus at the ion position. The direction of both beams can be adjusted using electrically actuated mirror mounts[13].

Using the imaging system, it is possible to detect with high fidelity whether the ion is in the ground state $S_{1/2}$ or the excited states $D_{5/2}$. If the ion is in the ground state when we switch on Doppler cooling, it will continuously scatter photons at 397 nm. If the ion is in the D state, it will remain dark with the Doppler cooling beams on. The two cases result in the histogram of counts collected on the PMT as shown on Figure 3.8. By setting the discrimination threshold in between the two distributions, we can determine the probability of the ion being in the state $D_{5/2}$.

[9] Sens Tech P25PC

[10] Andor Luca

[11] Depending on the experiment, the beamsplitter is removed and all of the light is directed to either of the detectors

[12] Semrock FF01-377/50-25 and FF01-417/60-25

[13] Newport Picomotor 8821

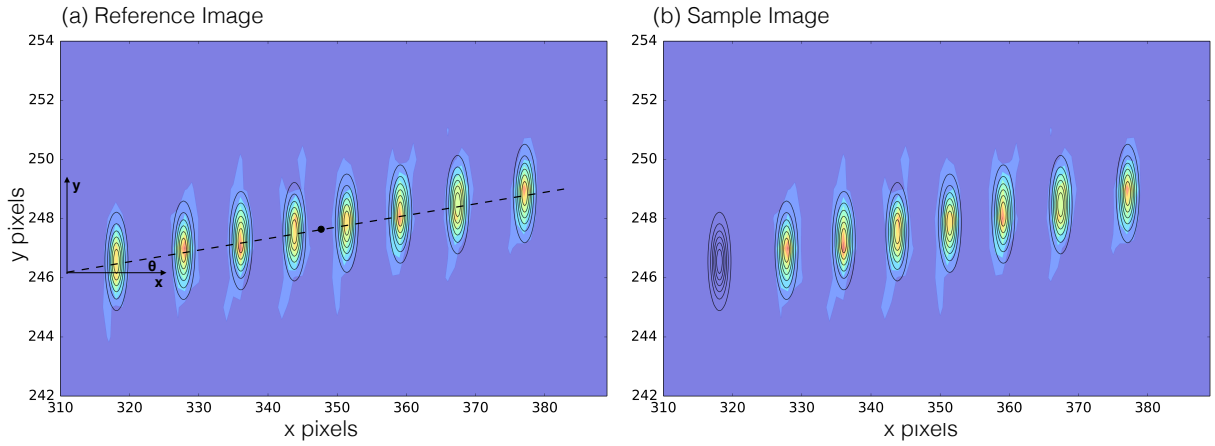

Figure 3.9: (a) The reference image of an 8-ion chain shown along with the chain rotation angle θ. The black circle is the center of the chain with the coordinates (x_0, y_0). (b) The sample image to test the state detection algorithm – the ion state is determined to be $\vec{r} = [1, 0, 0, 0, 0, 0, 0, 0]$. Note that the images are stretched vertically in order to demonstrate the rotation angle.

3.5.1 Ion State Detection CCD

In this section, we describe the procedure used to perform state detection with the CCD camera. The CCD is advantageous for detecting the state of multiple ions in the trap. While the PMT has a higher detection efficiency of $\sim 25\%$ at 397 nm, it only collects the total emitted photons and does not provide information about which particular ion is bright or dark. The CCD allows to obtain this information at the expense of additional complications: the CCD method requires a longer exposure time of 10 ms, a fitting procedure and a reference image are needed to determine the ion state, and it is more difficult to establish the statistical confidence of the result.

When using the CCD for state readout, we, first, specify the region of interest – the range of camera pixels that fully contain the trapped ions. Reducing the region of interest improves both the transfer rate of the image to the computer and the speed of the fitting algorithm. These are important considerations because hundreds of images need to be taken to precisely determine the expectation values of the ion state. We then specify the exposure parameters: EM gain, pixel binning and the exposure time. By varying the readout parameters, we found that the state detection worked best with 2×2 pixel binning and EM gain set to 200 out of the maximum of 255. The former improves signal-to-noise while the latter avoids many "hot" pixels appearing at the maximum gain setting. These optimized settings allowed us to reduce the required exposure time to 10 ms. This is still much longer compared to 1.5 ms required for the PMT, pointing to the large difference in detection efficiencies.

With all of the settings optimized, we take a reference image as shown in Figure 3.9. This is done with all of the ions bright at the same laser parameters and for the same exposure at the eventual state readout. We usually average several hundred images, each exposed for the state readout duration, to produce the reference image. The reference image is then fitted with the model below to extract all of the relevant parameters about the ion positions and intensity[14]:

$$I(x,y) = I_0 + A\sum_{i=0}^{i<N} \mathcal{N}(x_i, y_i, \sigma^2) \ , \tag{3.1}$$

where $I(x,y)$ is the pixel intensity of the reference image, I_0 is the background level, A is the ion intensity, N is the total ion number of ions, and $\mathcal{N}(x_i, y_i, \sigma^2)$ is the Gaussian distribution centered at the position of ion i, (x_i, y_i), where every Gaussian has the same standard deviation σ:

$$\mathcal{N}(x_i, y_i, \sigma^2) = \exp\left[-\frac{(x-x_i)^2 + (y-y_i)^2}{2\sigma^2}\right] \ . \tag{3.2}$$

We place additional constraints on the ion positions by assuming that the ions form a linear chain. In this case, the relative ion positions are determined from the equilibrium between the Coulomb potential and the trap confinement. The positions are given by:

$$x_i = x_0 + s_{i,N} l \cos\theta \ , \tag{3.3}$$

$$y_i = y_0 + s_{i,N} l \sin\theta \ , \tag{3.4}$$

where (x_0, y_0) are the center coordinates of the chain, θ is the rotation angle of the chain axis relative to the coordinate system, $s_{i,N}$ is the relative position of ion i, and l is the length scale converting the relative position to distance in pixels. The relative positions $s_{i,N}$ are calculated numerically ahead of time for the chain of the given length N. For a listing of the precalculated relative positions see Table 1 in Ref. [12].

The free parameters of the model are the background intensity I_0, the ion intensity A, the chain center (x_0, y_0), the chain rotation angle θ, and the length scale l. The number of ions is held constant and is specified beforehand. The fitting procedure is fully automated: all of the initial guesses for the parameters are extracted directly from the reference image, requiring no user intervention.

We briefly describe the methods for automated parameter extraction that worked well in practice. The guess background intensity I_0 is found by taking the mean of the pixel values along the edge of the image - where we assume no ions are located. We also calculate the standard deviation of intensity of these background pixels. All of the pixels three standard deviations above the background are assumed to belong to the ions. If no such pixels are found, the threshold is progressively lowered. The mean of the positions of the ion pixels

[14]For consistency with the software implementation, we use the Python indexing convention, denoting the left-most ion as ion 0.

are used to guess the center of the ion chain (x_0, y_0) and the mean value to guess the ion intensity A. The variance of the positions is also used to guess the length scale l[15].

The width of the Gaussian, σ is assumed to be 1 since the intensity of each ion is typically contained in an area of 2×2 pixels. The rotation angle θ is guessed to be zero because it's known to be small, $\theta \sim 3°$. In fact, to make the algorithm more robust, the fitting is first done with the angle fixed at $\theta = 0$ to optimize the other parameters, and only then θ is varied.

Once the reference image is fitted, we are ready to perform state detection. For each collected image, the goal is to determine which of the ions, if any, are dark. The ion state is represented by an N-long vector $\vec{r}$ where each entry r_i is either 0 (bright) or 1 (dark), $r_i \epsilon \{0, 1\}$, see Fig. 3.9. The expected intensity for this combination of bright and dark ions is:

$$I_{\vec{r}}(x, y) = I_0 + A \sum_{i=0}^{i<N} (1 - r_i) \mathcal{N}(x_i, y_i, \sigma^2) \ . \tag{3.8}$$

We determine the state vector by finding one that minimizes the mean-squared error to the state readout image. In the current implementation, we do this by sampling over all of the 2^N possible state combinations[16]. Additionally, we extract the confidence of the fit by comparing the errors from the best and the second best combinations. We found empirically that state direction has a very high fidelity when the error of the second best combination is at least 20% larger than the best combination. It is also useful to monitor the confidence level: a drop in confidence indicates that the ions drifted away from the previous positions and another reference needs to be taken.

[15]It is important to have an accurate initial guess for the spacing l to order to avoid local minima of the minimization procedure. For example while fitting a five-ion chain, if the initial guess for spacing is too long then the ions to the left and to the right of the center will get overlapped with the extreme ions of the image. The spacing guess is also extracted from the image: consider the chain of N ions to lie along the x axis, centered at $x = 0$ with the spacing length scale l. The position of the i-th ion $x_i = s_{i,N} l$. Each ion is assumed to be 1 pixel wide to make the analysis easier. Once the bright pixels are identified, we can compute the variance of their positions:

$$\text{Var} = E(x^2) - E(x)^2 = \frac{1}{N} \sum_{i=1}^{N} x_i^2 - \left(\frac{1}{N} \sum_{i=1}^{N} x_i \right)^2 \tag{3.5}$$

$$= \frac{l^2}{N} \sum_{i=1}^{N} s_{i,N}^2 \ , \tag{3.6}$$

since $E(x) = 0$. Therefore the length scale is related to the variance through

$$l = \sqrt{\frac{\text{Var} \times N}{\sum_{i=1}^{N} s_{i,N}^2}} \ . \tag{3.7}$$

[16]While the procedure of sampling over all the combinations is prohibitively long for longer chains of $N > 8$ ions, it can easily modified to only make site-by-site state determination. The modified procedure would not take into account the cross talk about the ion intensities due to imaging aberrations.

3.6 Electronics

In this section, we provide a brief description of the electronics used in the experiment. The trap is driven with an oscillating RF voltage of 30 MHz generated with a Rohde Schwarz SMB100A synthesizer, amplified with a 20 W amplifier[17]. The amplified signal is stepped up in a half-wave helical resonator described in [9] to produce two out-of-phase oscillating voltages needed for the out-of-phase trap drive. The DC voltages for the endcaps in the range between 0 V and 40 V are generated with a custom-built DAC controlled with an Opal Kelly FPGA. One of the DAC channels applies a DC bias on the RF voltage using a bias tee. The compensation voltages of up to 2000 V are produced with a commercial high voltage power supply Iseg SHQ222m. All of the DC channels are heavily filtered to minimize electrical noise at the ion.

The radio frequency voltages for the acousto-optic modulators (AOMs) are synthesized on custom-made direct digital synthesis (DDS) boards. These are programmed by the computer via an FPGA board named the Pulser [9]. The voltage for each AOM is amplified with a MiniCircutis 2W or 5W amplifiers. All of the DDS boards are referenced to a common 800 MHz clock. Additionally, there are PID circuits for laser intensity stabilization and a high current power supply to produce 12 A to heat up the calcium oven.

3.6.1 Electrode Diagonalization

In general, the voltages applied to specific trap electrodes produce the electric field in all three directions at the ion position: E_x, E_y, and E_z. In order to improve usability, it is desirable to find linear combinations of electrode voltages which change the electric field only in a single direction. The linear combination is determined both by trap design and by geometric imperfections when the trap is machined.

In the trap geometry we have two compensation electrodes two endcaps. Suppose that to these electrodes we apply voltages $C1$, $C2$, $D1$ and $D2$ respectively. The most general relationship between the applied voltages and the resultant electric fields at the ion position can be expressed as a sum for some unknown matrix M_{ij}:

$$E_i = \sum_j M_{ij} V_j \ , \tag{3.9}$$

where V_j enumerates over the applied voltages: $\vec{V} = (C1, C2, D1, D2)$.

The problem is underdetermined since we apply 4 different voltages but need to fix only 3 electric fields. In order to have a unique solution, we constraint the problem further by requiring that the axial trap frequency remains constant as we change the electric fields.

We start by recognizing that in the axial direction, the influence of the endcaps is much stronger than that of the compensation electrodes. Therefore as the fist step it's important to differentiate between a common mode increase in the endcap voltages D_+ corresponding

[17] Mini Circuits ZHL-20W-13

to the increased axial trap frequency ω_z and a differential voltage D_- corresponding to displacing the ion along the axis. We introduce the rotation angle θ_d to account for the asymmetry between the electrodes:

$$\begin{pmatrix} D_- \\ D_+ \end{pmatrix} = \begin{pmatrix} \cos\theta_d & -\sin\theta_d \\ \sin\theta_d & \cos\theta_d \end{pmatrix} \begin{pmatrix} D1 \\ D2 \end{pmatrix} \tag{3.10}$$

or, equivalently,

$$\begin{pmatrix} D1 \\ D2 \end{pmatrix} = \begin{pmatrix} \cos\theta_d & \sin\theta_d \\ -\sin\theta_d & \cos\theta_d \end{pmatrix} \begin{pmatrix} D_- \\ D_+ \end{pmatrix} \ . \tag{3.11}$$

For a symmetric trap $\theta_d = 45°$ but experimentally we measured $\theta_d \approx 53°$. This value was determined by measuring the pairs of voltages ($D1$,$D2$) for which the ion remains fixed at a given point along the axis.

The next step is to decouple the influence of the compensation electrodes, including their possible projection on the z axis. This can be expressed, in general, as:

$$\begin{pmatrix} C1 \\ C2 \\ D_- \end{pmatrix} = \begin{pmatrix} \ddots & \ddots & \ddots \\ \ddots & \ddots & \ddots \\ \ddots & \ddots & \ddots \end{pmatrix} \begin{pmatrix} \tilde{E}_x \\ \tilde{E}_y \\ \tilde{E}_z \end{pmatrix} \ . \tag{3.12}$$

The tilde notation as in $\tilde{E}^x$ indicates that the result is proportional to the electric field E_z where the proportionality constant converts the units of the applied Volts to the produced electric field.

Experimentally we measured sets of three voltages ($C1$,$C2$,D_-) for which the ion remains fixed on the camera. These points can be fit to a parameterized line in three dimensions:

$$\vec{r} = \vec{r_0} + t\hat{v} \ . \tag{3.13}$$

The direction unit vector $\hat{v}$ defines the axis along which only $\tilde{E}_y$ is changing while $\tilde{E}_x$ and $\tilde{E}_z$ are held constant. The intercept $\vec{r_0}$ determines the unimportant fixed values $\tilde{E}_x$ and $\tilde{E}_z$. The parameter t is proportional to the electric field: $t = \tilde{E}_y$ where we are free to ignore any possible offset. Thus, for a change of the electric field $\Delta\tilde{E}_y$, the voltages have to be adjusted according to

$$\Delta C1 = \hat{v}_0 \Delta\tilde{E}_y \ , \tag{3.14}$$

$$\Delta C2 = \hat{v}_1 \Delta\tilde{E}_y \ , \tag{3.15}$$

$$\Delta D_- = \hat{v}_2 \Delta\tilde{E}_y \ , \tag{3.16}$$

or in the matrix form:

$$\begin{pmatrix} C1 \\ C2 \\ D_- \end{pmatrix} = \begin{pmatrix} \ddots & \hat{v}_0 & \ddots \\ \ddots & \hat{v}_1 & \ddots \\ \ddots & \hat{v}_2 & \ddots \end{pmatrix} \begin{pmatrix} \tilde{E}_x \\ \tilde{E}_y \\ \tilde{E}_z \end{pmatrix} \ . \tag{3.17}$$

On the practical note, the slopes $m_0 = \frac{\hat{v}_0}{\hat{v}_1}$ and $m_2 = \frac{\hat{v}_2}{\hat{v}_1}$ can be easily ascertained from the data be rewriting eq. 3.13 as two linear equations:

$$x = \left(\frac{\hat{v}_0}{\hat{v}_1}\right) y + \left(x_0 - \frac{\hat{v}_0 y_0}{\hat{v}_1}\right) = m_0 y + \text{const} \ , \tag{3.18}$$

$$z = \left(\frac{\hat{v}_2}{\hat{v}_1}\right) y + \left(z_0 - \frac{\hat{v}_2 y_0}{\hat{v}_1}\right) = m_2 y + \text{const} \ . \tag{3.19}$$

Then it follows that

$$\hat{v}_0 = \frac{m_0}{\sqrt{m_0^2 + m_1^2 + 1}} \ , \quad \hat{v}_1 = \frac{1}{\sqrt{m_0^2 + m_1^2 + 1}} \ , \quad \hat{v}_2 = \frac{m_2}{\sqrt{m_0^2 + m_1^2 + 1}} \ . \tag{3.20}$$

The other columns of the coupling matrix have to be perpendicular to $\hat{v}$ and to each other. Our strategy is to arbitrary choose two such vectors u_0 and w_0 and allow for their joint rotation about the $\hat{v}$ axis. Particularly we pick:

$$\vec{u_0} = (-v_1, v_0, 0) \ , \tag{3.21}$$

$$\vec{w_0} = \hat{v} \times \vec{u_0} = (-v_0 v_2, -v_1 v_2, v_0^2 + v_1^2) \ . \tag{3.22}$$

We use the Rodriguez formula to define the rotation matrix about the axis $\hat{v}$ by an angle θ:

$$R_v(\theta) = \begin{pmatrix} \cos\theta - v_0^2(\cos\theta - 1) & -v_0 v_1(\cos\theta - 1) - v_2 \sin\theta & v_1 \sin\theta - v_0 v_2(\cos\theta - 1) \\ v_2 \sin\theta - v_0 v_1(\cos\theta - 1) & \cos\theta - v_1^2(\cos\theta - 1) & -v_1 v_2(\cos\theta - 1) - v_0 \sin\theta \\ -v_0 v_2(\cos\theta - 1) - v_1 \sin\theta & v_0 \sin\theta - v_1 v_2(\cos\theta - 1) & (v_0^2 + v_1^2)(\cos\theta - 1) + 1 \end{pmatrix} . \tag{3.23}$$

Rotating the vectors u_0 and w_0 by the angle θ produces:

$$R_v(\theta) u_0 = u_\theta = \begin{pmatrix} -v_1 \cos\theta - v_0 v_2 \sin\theta \\ v_0 \cos\theta - v_1 v_2 \sin\theta \\ (v_0^2 + v_1^2) \sin\theta \end{pmatrix} \ , \tag{3.24}$$

$$R_v(\theta) w_0 = w_\theta = \begin{pmatrix} -v_0 v_2 \cos\theta + v_1 \sin\theta \\ -v_1 v_2 \cos\theta - v_0 \sin\theta \\ (v_0^2 + v_1^2) \cos\theta \end{pmatrix} \ . \tag{3.25}$$

All together we determine:

$$\begin{pmatrix} C1 \\ C2 \\ D_- \end{pmatrix} = \begin{pmatrix} -v_1 \cos\theta - v_0 v_2 \sin\theta & \hat{v}_0 & -v_0 v_2 \cos\theta + v_1 \sin\theta \\ v_0 \cos\theta - v_1 v_2 \sin\theta & \hat{v}_1 & -v_1 v_2 \cos\theta - v_0 \sin\theta \\ (v_0^2 + v_1^2) \sin\theta & \hat{v}_2 & (v_0^2 + v_1^2) \cos\theta \end{pmatrix} \begin{pmatrix} \tilde{E}_x \\ \tilde{E}_y \\ \tilde{E}_z \end{pmatrix} \ . \tag{3.26}$$

Combining the result with the initial endcap rotation we arrive at the final expression:

$$\begin{pmatrix} C1 \\ C2 \\ D1 \\ D2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\theta_d & \sin\theta_d \\ 0 & 0 & -\sin\theta_d & \cos\theta_d \end{pmatrix} \begin{pmatrix} -v_1 \cos\theta - v_0 v_2 \sin\theta & \hat{v}_0 & -v_0 v_2 \cos\theta + v_1 \sin\theta & 0 \\ v_0 \cos\theta - v_1 v_2 \sin\theta & \hat{v}_1 & -v_1 v_2 \cos\theta - v_0 \sin\theta & 0 \\ (v_0^2 + v_1^2) \sin\theta & \hat{v}_2 & (v_0^2 + v_1^2) \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \tilde{E}_x \\ \tilde{E}_y \\ \tilde{E}_z \\ D_+ \end{pmatrix} \ . \tag{3.27}$$

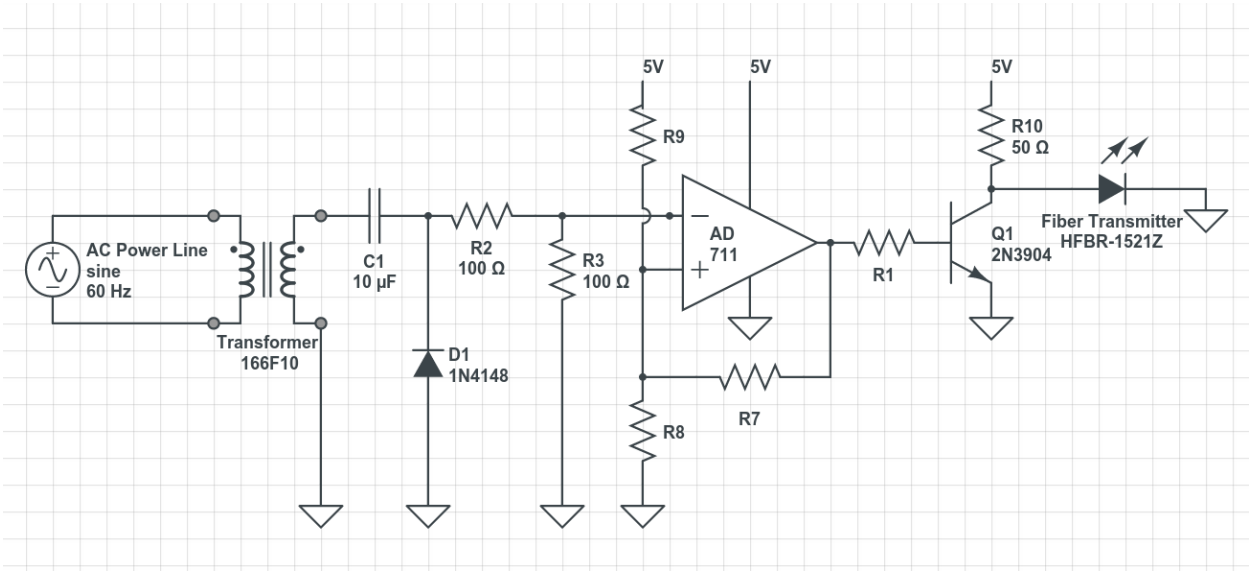

Figure 3.10: The circuit used to trigger the pulse sequences on the AC line. The power line AC voltage is stepped down with a transformer, and rectified. The signal then reaches a Schmitt trigger, which drives a fiber-coupled LED connected to the Pulser FPGA [9].

All of the rotations are parameterized by the two rotation angles θ_d and θ and two independent components of the unit vector $\hat{v}$.

3.6.2 Line Triggering

In our early measurements, the spectral width of the 729 nm carrier transition lines was limited by magnetic field fluctuations. These fluctuations broaden the observed transition because the line centers depend on the Zeeman shift produced by the instantaneous magnetic field. Repeating the experiment many times then samples over the varying line centers. One of the dominant source of such fluctuations is technical noise at the electrical grid frequency of 60 Hz. To circumvent fluctuations at this frequency, we implemented the ability to trigger the experiment sequences on the AC line. Triggering on the power line allows us to perform the 729 nm spectroscopy pulses at the same phase of the AC voltage.

The line triggering circuit is presented on Figure 3.10. It samples the power line and then generates a train of TTL pulses at a certain phase of the AC line that are sent to the Pulser FPGA responsible for executing pulse sequences [9]. The TTL pulses are transmitted over a fiber to provide electrical isolation and avoid ground loops.

The effect of line triggering can be seen on Figure 3.11. It shows two spectra of the $|S_{1/2}, m_j = -1/2\rangle - |D_{5/2}; m_j = -5/2\rangle$ transition recorded at different phases of the AC line. Without line triggering, the spectroscopy pulse is performed at a random phase, broad-

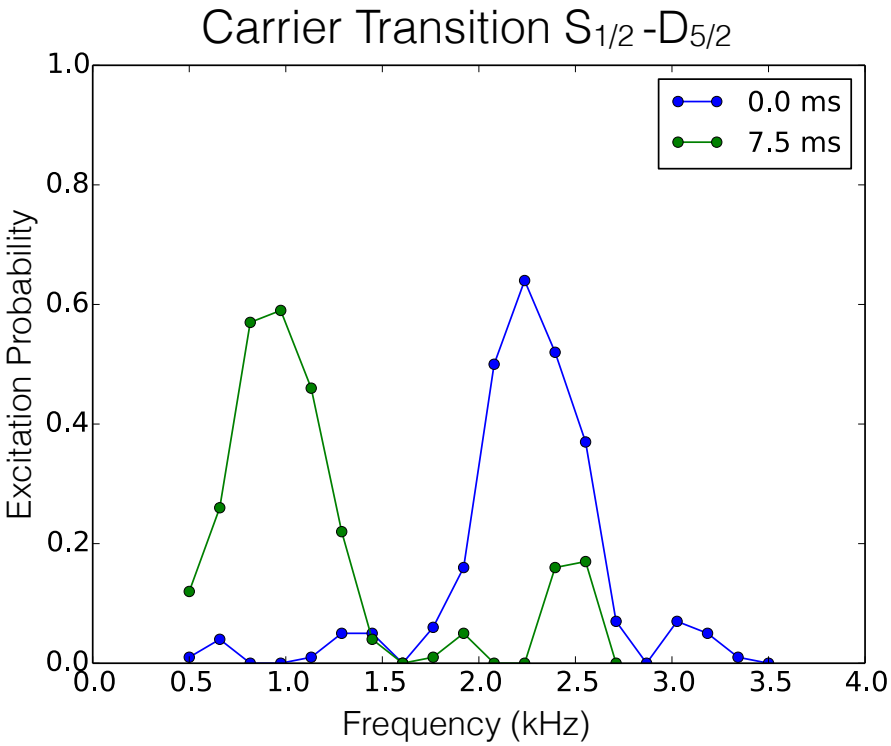

Figure 3.11: The frequency of the $|S_{1/2}, m_j = -1/2\rangle - |D_{5/2}; m_j = -5/2\rangle$ transition measured for offsets of 0.0ms and 7.5ms from the line triggering TTL pulse. The corresponding phases of the AC line correspond to the two extrema of the magnetic field fluctuations.

ening the line to the width between the recorded peaks, $\sim$2 kHz. With line triggering the spectroscopic pulse is performed at the same instantaneous magnetic field and the measured linewidth is reduced to $\sim$500 Hz[18]. Additionally, we have measured that line triggering also improves the T2 coherence time on the $|S_{1/2}, m_j = -1/2\rangle - |D_{5/2}; m_j = -5/2\rangle$ transition to 1.5 ms from below 500 μs

3.7 Experimental Control

Computer control of ion trapping experiments requires interacting with a large number of individual hardware devices ranging from controlling the voltages supplied to the reference laser cavities, to synthesizing pulse sequences of laser pulses applied to the ion to perform experiments and analyzing ion images collected with a CCD camera. Some of the controlled devices are commercially available with provided APIs or can be controlled with NI-VSA

[18]The reduced linewidth is likely limited by the laser spectral broadening due to phase noise in the fiberoptic cables.

drivers, while much electronics hardware is custom-made and tailored for the specific experiments. In the past, the standard way of controlling laboratory-scale experiments with LabView has been found to not scale well with the complexity of the ion trapping experiments. Once the code base reached a certain level of complexity, it became difficult to make even small additions and to see the meaning and the structure behind LabView's visual code representation. For controlling the experiments in our laboratory, we have chosen to use LabRAD - an open source experimental control software developed by John Martinis group at UCSB who faced the same scalability challenges in the quantum computing experiments.

The main strength of LabRAD is the modularity of individual components making up the experimental control. Instead of a single central program controlling all of the devices, the experimental control is a network of individual programs called servers that each controls a specific device or type of device. The servers abstract away the procedural steps of how to communicate with devices or perform certain tasks. The available tasks for each server are listed as commands callable by other programs connected to the LabRAD network. Performing an experiment then simply requires executing a script that issues a sequence of commands to the servers performing the required actions of setting hardware parameters, data collection, saving etc. The servers support multiple simultaneous connections from other programs. They are written in standard programming languages making use of object-oriented programming paradigm.

The modular nature of LabRAD makes the individual components much easier to modify and debug as they can be improved and tested separately from the rest of the system. The individual servers controlling specific hardware devices are responsible for a singular task and are easily reusable across different laboratories using the same hardware. We use GitHub version control to keep track of code changes across the experiments in order to maintain a single code base for every server. If there is no need to make improvements, the individual servers can act as black boxes where it is possible to make use of the functionality without ever reading through the server's source code. In addition, LabRAD runs on all of the major operating systems, providing an effective interface to devices that are restricted to work on a specific platform.

At its core, LabRAD provides a protocol for how the separate programs making up the experimental control communicate over TCP. The individual components, thus, do not have to all run on the same computer as long as they belong to the same local area network. The programs can be written in any language that confirms to the protocol, making it possible to write the code using the object-oriented programming paradigm in stark contrast to the LabView approach. We have primarily utilized the Python implementation of the LabRAD protocol called pylabrad.

The technical descriptions of the experimental control software is provided in the Appendix A. We, first, describe the structure of all the servers used in the experiments and their interactions. We then explain in detail several of our additions to the LabRAD experimental control: the ScriptScanner Framework for scheduling, pausing or stopping individual experiments, and creation of graphical user interfaces (GUIs) within the LabRAD framework. For a thorough introduction to LabRAD please refer Markus Ansmann's thesis [13]

who co-developed LabRAD at UCSB with Matthew Neely. For more information, see the GitHub Repository Wiki and links to LabRAD guides and tutorials therein.

3.8 Experimental Procedures

In this section we give a brief description of procedures commonly employed in the experiments. The techniques of Doppler cooling and sideband cooling have become routine for ion trapping experiments. Instead of reviewing those, here we focus on less standard procedures.

3.8.1 Frequency-resolved Optical Pumping

With an applied magnetic field, the ground state $|S_{1/2}\rangle$ splits into two Zeeman sublevels: $|S_{1/2}; m_j = -1/2\rangle$ and $|S_{1/2}; m_j = +1/2\rangle$. The goal of optical pumping is to prepare the ion in one of the sublevels, namely $|S_{1/2}; m_j = -1/2\rangle$. Typically this is done by applying σ-polarized laser light at 397 nm to optically pump the ion to the desired state. This requires an additional laser beam, as linearly polarized 397 nm light is used for Doppler cooling. The procedure of frequency-resolved optical pumping avoids this requirement by employing the available laser light at 729 nm for optical pumping. The laser is detuned to be resonant to the narrow transition between $|S_{1/2}; m_j = +1/2\rangle$ and $|D_{5/2}; m_j = -3/2\rangle$. Simultaneously, we switch on the repumping laser at 854 nm. The ion returns to the ground state via the short-lived $P_{3/2}$ level. When it decays to the ground state it may end up in either of the Zeeman substates, increasing the likelihood of it being found in $|S_{1/2}; m_j = -1/2\rangle$. We typically switch on the optical pumping laser beams for a total duration of 2 ms.

3.8.2 Auto-crystallization

Trapped chains of ions occasionally melt, likely due to collisions with background gas or instability in the governing equations of motion. The term melting refers to the analogy between the ordered solid when the ions are in the chain and a disordered liquid. When the ions melt, they lose their structural order and instead form an atomic cloud. The auto-crystallization procedure is used to restore the ions back to the ordered state. When the ions melt, their average kinetic energy rises, leading to a Doppler-broadened linewidths of the atomic transitions. We detect the melting event by monitoring the fluorescence of 397 nm light tuned closely to the resonance between $S_{1/2}$ and $P_{1/2}$ energy levels. The sudden increase in the linewidth occurring during a melting event is detected via a drop in the measured fluorescence. The events may also detected using the CCD camera by monitoring the total intensity of the pixels corresponding to the ion positions within the chain. Once the melting event is detected, the ions are crystallized again by shining 397 nm 220 MHz red-detuned from the $S_{1/2} - P_{1/2}$. The red-detuned light provides effective Doppler cooling of the broad linewidth and effectively crystallizes the chain. Crystallization may also be achieved by

lowering the power of the confining RF voltage. However, we typically avoid this to prevent thermal contraction and expansion of the trap electrodes.

Chapter 4

Energy Transport

4.1 Introduction

Models of coupled oscillator chains are ubiquitous in physics as they provide a way to study many-body behavior emerging from interactions among well-understood constituent particles. The diverse range of phenomena described by such models include Bose-Einstein condensation, equilibration in dissipative quantum systems, heat transport, friction in nanoscale systems, and many others [14, 15, 16, 17, 8].

Historical interest in behavior of oscillator chain models was spurred by the Fermi-Pasta-Ulam paradox [18, 19]. When a chain of oscillators has a linear nearest neighbor coupling term, then the time evolution of the system can be found by describing the behavior in terms non-interacting normal modes. The energy of the modes remains constant and the system does not thermalize. Enrico Fermi, John Pasta, Stanislaw Ulam, and Mary Tsingou conjectured that addition of a non-linearity to the coupling potential would give rise to ergodicity and lead the system to eventually thermalize. They performed numerical simulations of the model and found otherwise. Further work on the problem has led to discoveries of soliton solutions in related non-linear systems and the concept of dynamical chaos.

The Fermi-Pasta-Ulam approach introduced a new tool of experimental mathematics: performing numerical experiments to study the model dynamics when an analytical solution is not attainable. This is effective for classical coupled oscillators and one can numerically simulate time evolutions of hundreds of coupled oscillators. However in the quantum regime, this is, in general, is no longer possible as the size of the possible solution space – the Hilbert space – grows exponentially with the number of particles. As a solution to this issue, Richard Feynman proposed using a well-controllable quantum system to simulate behavior of the quantum model. The experimenter tunes the interactions of the controlled system to match those of the model and simply observes the ensuing dynamics. This approach called quantum simulations is promising for understanding quantum magnetism, superconductivity and other topics.

Chains of trapped ions are one of the leading candidate systems for performing quantum

simulations and studying quantum phenomena [20, 21, 22, 23, 24]. They can be well isolated from the environment, exhibiting long coherence times. The interactions between the ions, the amount of present non-linearity and decoherence can be precisely controlled with additional optical potentials. In fact, chains of ions are already used to study quantum magnetism [21, 23, 24]. These experiments utilize the internal degree of freedom. Controlling the motional degree of freedom, however, would allow to treat trapped ions an oscillator chain model and to explore the rich associated physics. There have been numerous theoretical proposals exploring oscillator chain models by using ion motion and inducing phonon-phonon interactions. Here we focus on the proposals studying heat transport and thermalization: topics similar to the ones pursued by Fermi-Pasta-Ulam but with additional focus on the quantum regime.

Bermudez et al. [16] study how heat propagates across the ion chain when the ions on the ends are coupled to heat baths of different temperatures. They find that depending on the amount of disorder and dephasing in the system, the resultant temperature distribution will deviate from the expected linear profile expected from Fourier's law. A similar non-linear temperature distribution is found in [15] where the authors also study how the system equilibrates. Both [25] and [26] demonstrate the thermodynamic properties of ion chains are controlled by the amount of present non-linearity induced by the linear to zigzag structural phase transition. Finally, in [8], it is shown that a standing optical wave superimposed with the ion chain significantly modifies the normal mode structure, thereby altering the energy transport characteristics. The heat conduction in the resultant Frenkel-Kontorova model was is studied in [27] with a periodic driving force.

Despite the richness of the exhibited physics, there have been no experimental realizations of the proposed models. Observing the phenomena requires long chains of ions, where it becomes difficult to control the motion. In this Chapter we describe the first experiment aimed at realizing the aforementioned models, providing an additional level of detail compared to our publication [28]. Specifically, we observe how energy propagates in an ion chain by performing a Newton cradle-like experiment. We prepare an out-of-equilibrium state of the chain by quickly adding energy to the ion on one end of the chain. We then observe how the energy propagates throughout the chain.

We begin the Chapter by reviewing the theoretical framework of local modes of ion motion used in the experiment and the method used to measure the ion energy. In the following section we describe numerical simulations used to model ion excitation and energy propagation. We then interpret the experimental results and conclude by considering future possible pursuits.

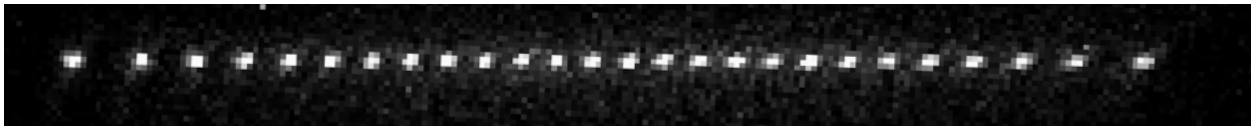

Figure 4.1: The CCD image of a long linear ion chain. All of the ions are confined axially in a single harmonic well. Due to the Coulomb repulsion among the ions, the spacing decreases in the center of the chain.

4.2 Theory

4.2.1 Normal Mode Structure

In this section we consider the normal mode structure of the ion chain and calculate the subsequent evolution after a single ion on end of the chain is displaced from its equilibrium position. When a chain of N ions is confined in the harmonic potential of a Paul trap with harmonic frequencies ω_x, ω_y, and ω_z, the total potential energy V is given by the sum of harmonic trap potential and and the Coulomb interaction:

$$V = \sum_{i=1}^{N} \frac{1}{2} m\omega_x^2 x_i^2 + \frac{1}{2} m\omega_y^2 y_i^2 + \frac{1}{2} m\omega_z^2 z_i^2 + \frac{1}{2} \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j\neq i}}^{N} \frac{e^2}{4\pi\epsilon_0} \frac{1}{\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2}} , \tag{4.1}$$

where e is the electron charge and m is the ion mass, and we have included a factor of $\frac{1}{2}$ in the Coulomb term to prevent double counting of ion pairs. Minimizing potential energy V with respect to the coordinates of every ion i, (x_i, y_i, z_i), yields the equilibrium positions (x_i^0, y_i^0, z_i^0). The CCD image of a trapped linear chain of ions is shown on Figure 4.1. Considering the motion in only one of the radial directions, x, the potential energy V can be expanded for small displacements about the equilibrium, $q_i = x_i - x_i^0$.

$$V \approx V_0 + \sum_i \frac{\partial V}{\partial q_i} q_i + \frac{1}{2} \sum_{i,j} \frac{\partial^2 V}{\partial q_i \partial q_j} q_i q_j \ . \tag{4.2}$$

The constant offset V_0 is unimportant for equations of motion and will be ignored. At the equilibrium position, the first derivative $\frac{\partial V}{\partial q_i} = 0$. The third derivative $\frac{\partial V^3}{\partial q_i^3}$ is also zero by the symmetry of the Coulomb potential. We then obtain:

$$\frac{\partial^2 V}{\partial q_k \partial q_l}\Big|_{\text{equil}} = \begin{cases} m\omega_x^2 - \sum_{\substack{j\prime=1 \\ j\prime\neq k}}^{N} \frac{e^2}{4\pi\epsilon_0} \frac{1}{|z_k^0 - z_{j\prime}^0|^3}, & \text{if } k = l. \\ \frac{e^2}{4\pi\epsilon_0} \frac{1}{|z_k^0 - z_l^0|^3}, & \text{if } k \neq l. \end{cases} , \tag{4.3}$$

matching eq. 7 in [20], where we assumed that the radial trap frequencies are much stronger than the axial, $\omega_x \gg \omega_z$, $\omega_y \gg \omega_z$, and the ions are arranged as a linear chain along the z-axis with $y_i^0 = x_i^0 = 0$. The potential energy can then be written as

$$V = \sum_{i=1}^{N} \frac{1}{2} m \left(\omega_x^2 - \sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{e^2}{4\pi\epsilon_0 m} \frac{1}{|z_i^0 - z_j^0|^3} \right) q_i^2 + \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{1}{2} \frac{e^2}{4\pi\epsilon_0} \frac{1}{|z_i^0 - z_j^0|^3} q_i q_j \ . \tag{4.4}$$

We define the local oscillation frequency $\omega_{i,\mathrm{loc}}$ of ion i as

$$\omega_{i,\mathrm{loc}}^2 = \omega_x^2 - \sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{e^2}{4\pi\epsilon_0 m} \frac{1}{|z_i^0 - z_j^0|^3} \ . \tag{4.5}$$

The local oscillation frequency differs from the radial trap confinement frequency ω_x by the repulsive force from the other ions in the chain. This effect drops off as the cube of the inter-ion distance and is, particularly, important in the middle of the chain where the ions are closest. Thus, the local radial oscillation frequency will be minimal for the middle ion. Furthermore, when the local oscillation frequency becomes zero, this soft mode leads to a structural phase transition to the zigzag regime of the chain [29].

It is clear from eq. 4.4 that the local displacements q_i are coupled. The system may be diagonalized in terms of the N radial normal modes of the ion chain [12]. We enumerate the normal modes $\vec{v}_n$ in the order of decreasing eigenfrequencies ω_n such that $\vec{v}_1$ always refers to the center-of-mass mode with the corresponding eigenfrequency of $\omega_1 = \omega_x$. We consider the normal mode decomposition when the leftmost is radially displaced from its equilibrium position. The unit displacement can decomposed as the sum of radial normal modes with coefficients c_n:

$$\vec{q} = [1, 0, 0, \ldots, 0] = \sum_{i=1}^{N} c_n \vec{v}_n. \tag{4.6}$$

The normal mode decomposition is presented on Figure 4.2. We see that for long chains, the higher order modes are not significantly excited. The excited eigenmodes will oscillate at the corresponding eigenfrequencies, determining the subsequent time evolution of the ion chain:

$$\vec{q}(t) = \sum_{i=1}^{N} c_n \vec{v}_n \cos(\omega_n t) \ . \tag{4.7}$$

The oscillation energy of the displaced ion will decrease once the excited normal modes add destructively. At the rephasing time of the eigenmodes, the energy will come back to the displaced ion.

The coupling times among the local ion excitations are easily determined by considering the quantized version of the system [20, 30, 31] written in terms of local phonon creation

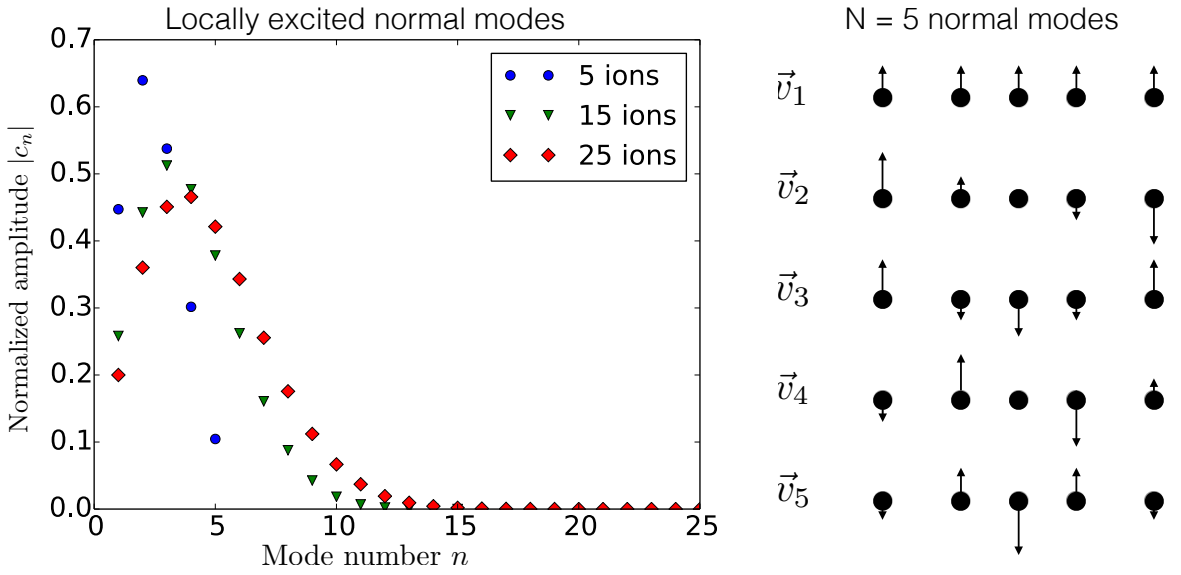

Figure 4.2: We decompose a unit radial displacement of the leftmost ion in terms of the radial normal modes of chain motion. Not all of the modes are equally excited. For longer chains, only lower-ordered have significant excitation. Shown on the right are the radial normal modes for a $N = 5$ chain. We can see that only the first three have a significant projection onto motion of the leftmost ion.

and annihilation operators at site i denoted $a_i^\dagger$, a_i:

$$H = \sum_{i=1}^{N} \hbar\omega_{x,i} a_i^\dagger a_i + \hbar \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j<i}}^{N} t_{ij} \left(a_i^\dagger a_j + a_i a_j^\dagger \right) \ , \tag{4.8}$$

where we have neglected fast-rotating non-energy conserving terms. The site-dependent oscillation frequency and the tunneling amplitude are given by:

$$\omega_{x,i} = \omega_x - \frac{1}{2} \sum_{\substack{j=1 \\ j\neq i}}^{N} \frac{1}{m\omega_x} \frac{e^2}{4\pi\epsilon_0} \frac{1}{|z_i^0 - z_j^0|^3} \ , \tag{4.9}$$

$$t_{ij} = \frac{1}{2} \frac{1}{m\omega_x} \frac{e^2}{4\pi\epsilon_0} \frac{1}{|z_i^0 - z_j^0|^3} = \frac{1}{2} \frac{\omega_z^2}{\omega_x} \frac{1}{\Delta\bar{z}^3} \ , \tag{4.10}$$

where $\Delta\bar{z} \sim O(1)$ is a dimensionless separation of ions in the units of the characteristic length scale l [12]

$$l^3 = \frac{Z^2 e^2}{4\pi\epsilon_0 m\omega_z^2} \ , \tag{4.11}$$

with $Z = 1$ referring to the number of elementary charges of the ion.

The cubic dependency of tunneling t_{ij} on the inter-ion distance makes the length of the chain an important factor because for longer chains trapped at the same trap frequencies, the leftmost ion is closer to its neighbor. We find that for our experimental trap frequencies, the tunneling matrix element between the leftmost ion and its neighbor, t_{12}, ranges from $2\pi \times 6.7$ kHz for a chain of 5 ions to $2\pi \times 21.1$ kHz for a chain of 25 ions.

The normal mode picture is valid when the non-linearities in the potential are insignificant in the course of ion motion. To verify that this is the case for the regime of our experiments, we performed numerical molecular dynamics simulations integrating the ion equations of motion. The simulations are described in Appendix C.1. They do not rely on the normal mode expansion as they include the complete Coulombic interaction as well as driven motion of ions present in Paul traps. The time evolution predicted by the simulations matches those given by equation 4.7, verifying the validity of the treatment. We explicitly calculate the amount of non-linearity present in ion chains in Appendix C.2.

4.2.2 Fast Excitation

In order to create an out-of-equilibrium state of the ion chain by displacing the leftmost ion, we need to add energy to the ion much faster that the coupling time, $2\pi/\sum_{j\neq 1} t_{1j}$. This will prevent the energy flowing out from the ion before we have even completed the excitation. For this, we have chosen the technique of pulsed excitation. The laser resonant with a short-lived transition is switch off and on resonantly with the trap frequency [32, 33]. In this section we elaborate on a similar calculation we performed in [8] to calculate the expected excitation rates for pulsed excitation methods and other available techniques.

We estimate the rate of energy addition with pulsed laser excitation by considering a classical ion trajectory with initial amplitude y_0 oscillating around $y = 0$ in a harmonic potential $V = \frac{1}{2}m\omega_y^2 y^2$. Switching on a laser with saturation s, wavevector k resonant with a transition with linewidth Γ produces a constant force F due to the laser light's radiation pressure:

$$F = \hbar k \Gamma \rho_{ee} \ , \tag{4.12}$$

where the probability of finding the atom in the excited state ρ_{ee} is given by

$$\rho_{ee} = \frac{s/2}{1 + s + (2\Delta/\Gamma)^2} \ . \tag{4.13}$$

Since $\rho_{ee} \to \frac{1}{2}$ for $s \to \infty$, the scattering of a highly saturated laser is given by $F = \frac{1}{2}\hbar k\Gamma$. When the force is applied to the ion, the equilibrium position of oscillation is displaced to $y_{\text{eq}} = F/m\omega_y^2$. If the laser is switched on when the ion is at the leftmost point in its trajectory, $-y_0$, then after half of trap cycle it will reach the position $y_1 = y_0 + 2y_{\text{eq}}$. At this point, the laser is switched off to be switched on again half cycle later when the ion is at position $-y_1$. This process leads to a linear increase in oscillation amplitude and a quadratic rise in energy with the number of pulses. After n pulses, the amplitude is

$y_n = y_0 + 2ny_{\text{eq}}$ with the corresponding energy $E_n = \frac{1}{2}m\omega_y^2(y_0 + 2ny_{\text{eq}})^2$. For the trap frequency of $\omega_y = 2\pi \times 2$ MHz and a saturated laser at 397 nm addressing the $S_{1/2}$-$P_{1/2}$ transition with $\Gamma = 2\pi \times 22.4$ MHz, the equilibrium shift $y_{eq} = 11$ nm. During the first pulse, the energy will rise by approximately 2 motional quanta: $\frac{1}{2}m\omega_y^2(2y_{\text{eq}})^2 = 2.0\hbar\omega_y$ and then quickly grow quadratically with the number of pulses. In the first microsecond, of heating already 8 quanta will have been added.

We compare the result for pulsed excitation to heating due the random photon scattering and due to anti-viscous force of a blue-detuned laser. The heating rate due to the random timing of absorption events and random directionality of spontaneous emissions is given in the low velocity $v \sim 0$ limit by [2]:

$$\dot{E}_h = \frac{1}{2m}(\hbar k)^2\Gamma\rho_{ee}(v=0)(1+\xi) \ , \tag{4.14}$$

where the emission directionality factor $\xi = 2/5$ for dipole radiation. This term is maximal for a highly saturated beam $\rho_{ee} = 1/2$, at which point $\dot{E}_h \sim 1.5\hbar\omega_y/\mu$s for the same trap parameters at above. To estimate the effect of Doppler heating we expand the scattering force for low velocities: $F = F_0\,(1 + \kappa v)$ with the prefactor F_0 and the viscosity coefficient κ are given by:

$$F_0 = \hbar k\Gamma \frac{\frac{s}{2}}{1+s+(\frac{2\Delta}{\Gamma})^2} \ , \tag{4.15}$$

$$\kappa = \frac{\frac{8k\Delta}{\Gamma^2}}{1+s+(\frac{2\Delta}{\Gamma})^2} \ . \tag{4.16}$$

Only the velocity-dependent part of the force contributes to Doppler heating:

$$\dot{E}_{\text{dop}} = F_0\kappa\langle v^2\rangle \ . \tag{4.17}$$

For an initial temperature T, $\langle v^2\rangle = \frac{k_bT}{m}$ while the term $F_0\kappa$ is maximal when $s = 1 + 4\frac{\Delta^2}{\Gamma^2}$ and $\Delta = \frac{\Gamma}{2}$. At these parameters,

$$\dot{E}_{\text{dop}} = \frac{\hbar k^2}{2}\frac{k_bT}{m} \ . \tag{4.18}$$

If the initial energy is $\bar{n} = 5$ with $k_bT = \hbar\omega\bar{n}$, we have

$$\dot{E}_{doppler} = \frac{\hbar k^2}{2m}\bar{n} \sim 1\hbar\omega/\mu s \ . \tag{4.19}$$

We have seen that the initial rates of energy addition for pulsed excitation are much higher than for heating through random kicks, and Doppler heating. This calculation is confirmed with the molecular dynamics code detailed in C.1 plotted in Figure 4.3. Pulsed excitation is straightforward to implement, as there is no need to progressively adjust the laser detuning or laser saturation to maintain optimal heating conditions.

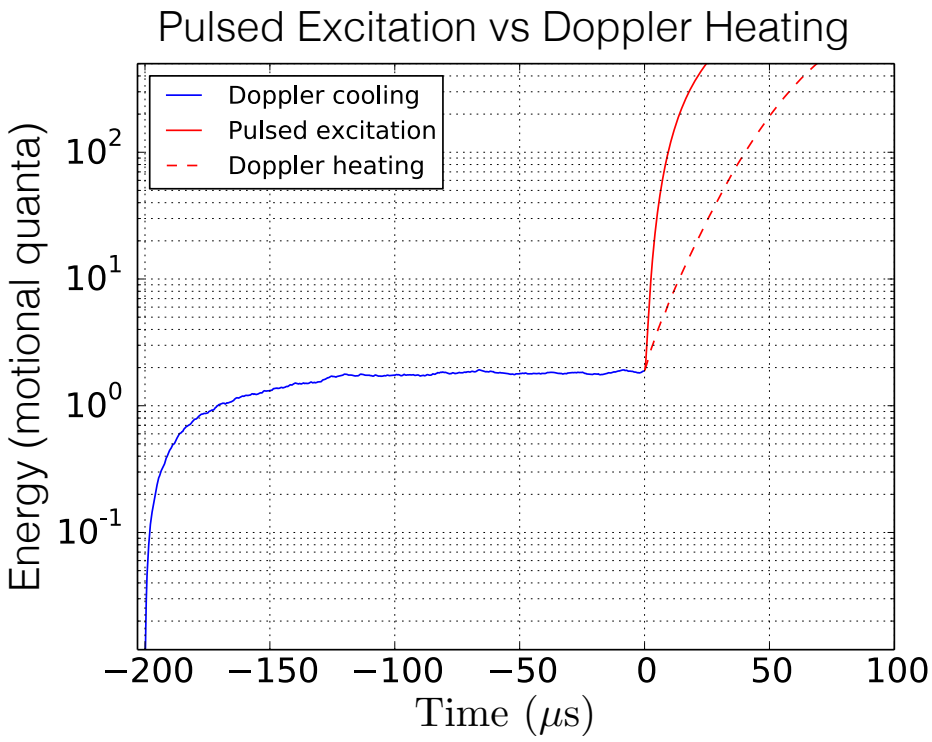

Figure 4.3: Results of a molecular dynamics simulation demonstrating that pulsed excitation adds energy much faster that Doppler heating. This single-ion simulation starts with ion stationary and located at its equilibrium position. Then we simulate the process of Doppler cooling for 200 μs to achieve a realistic energy given by the Doppler temperature. From that point on, we compare the ion kinetic energy after pulsed excitation and Doppler heating. Due to the stochastic nature of laser-ion interaction, we average over 500 individual simulations.

4.2.3 Energy Readout

In this section we discuss the method of quantifying the energy of the ion motion. Similar to the need for fast excitation, energy readout needs to be fast compared to the coupling in order to accurately capture the dynamics. In this regime, we can ignore the coupling among the ions and consider them as N independent harmonic oscillators:

$$H = \sum_{i=1}^{N} \hbar\omega_{x,i} a_i^\dagger a_i \ . \tag{4.20}$$

After performing Doppler cooling, the local radial mode of every ion is in a thermal state with a temperature $\bar{n}$. The density matrix of the ion i, ρ_i^{th}, can be expressed in terms of the

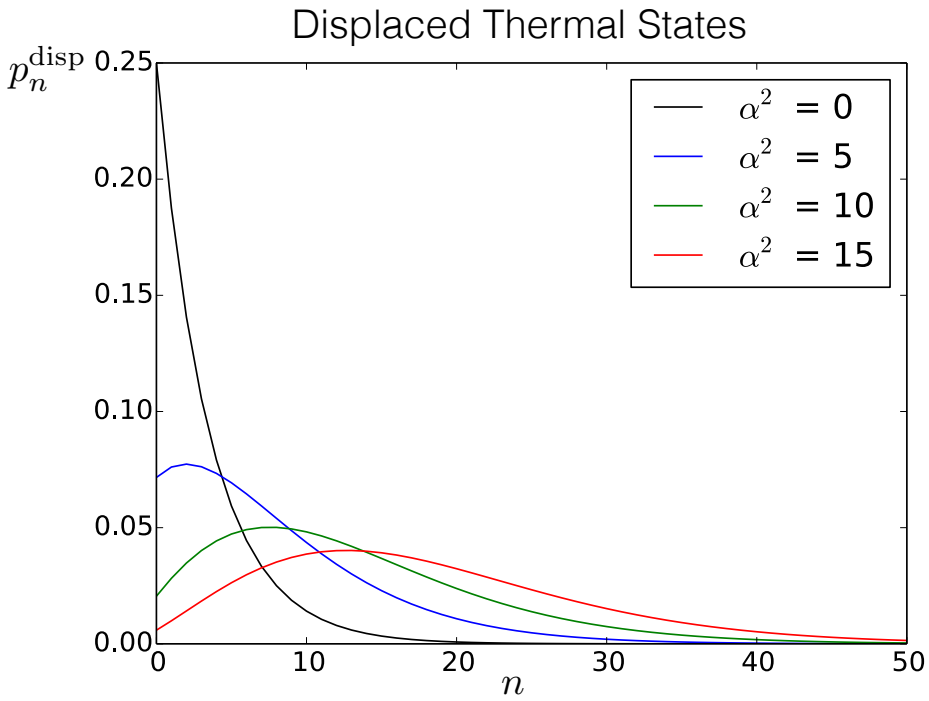

Figure 4.4: The occupational probabilities for displaced thermal states of $\bar{n} = 3$, typical of a Doppler-cooled ion. For displacement $\alpha = 0$, the distribution reduces back to a thermal state. For large displacements $\alpha^2 \gg n$, it approaches a coherent distribution centered at α^2.

local phonon number basis $|n_i\rangle$:

$$\rho_i^{\text{th}} = \frac{1}{\bar{n}+1}\left(\frac{\bar{n}}{\bar{n}+1}\right)^n |n_i\rangle\langle n_i| \ . \tag{4.21}$$

The method of pulsed excitation is modeled as a displacement operator $D(\alpha)$ applied onto the leftmost ion ($i = 1$) in the chain [34] for some complex amplitude $\alpha = |\alpha|e^{i\phi}$ resulting in a displaced thermal state of motion. Experimentally we do not control the phase of the pulsed laser ϕ, yielding a diagonal density matrix of the first ion ρ_1 after averaging over ϕ:

$$\rho_1 = \frac{1}{2\pi}\int d\phi \left(D(\alpha)\rho_1^{\text{th}} D(\alpha)^\dagger\right) = \sum_n p_n^{\text{disp}} |n_1\rangle\langle n_1| \ , \tag{4.22}$$

with the occupational probabilities given by [35]:

$$p_n^{\text{disp}} = \left(\frac{1}{\bar{n}+1}\right)\left(\frac{\bar{n}}{\bar{n}+1}\right)^n e^{-\frac{|\alpha|^2}{\bar{n}+1}} L_n\left(-\frac{|\alpha|^2}{\bar{n}(\bar{n}+1)}\right) \ , \tag{4.23}$$

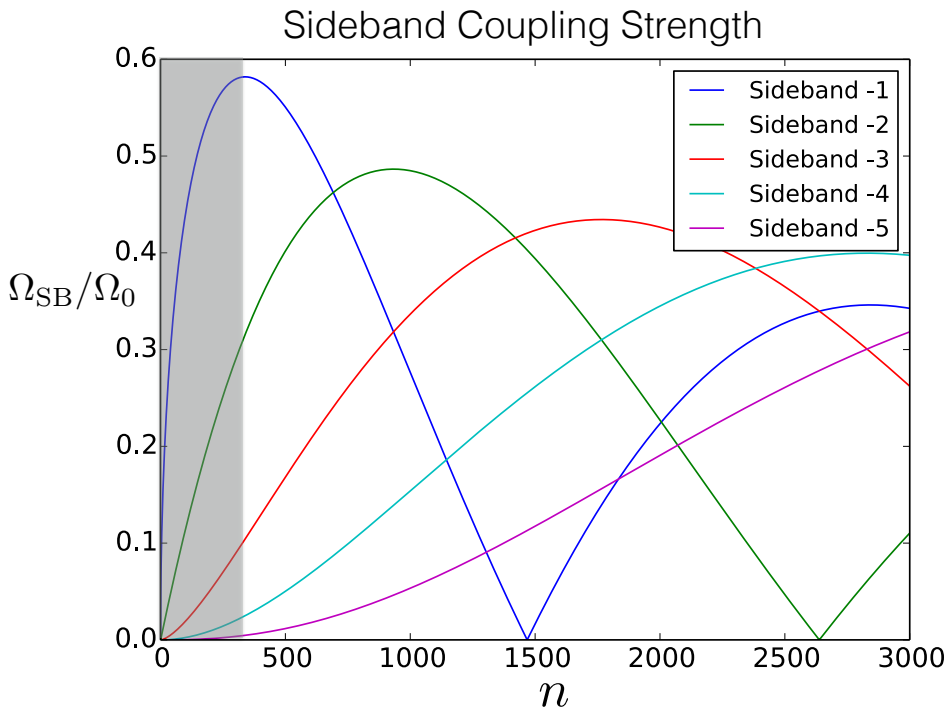

Figure 4.5: The sideband coupling strength $\Omega_{\mathrm{SB}} = \Omega_{n,n-l}$ in units of the carrier Rabi frequency Ω_0 for $n = 0$ shown for various red sidebands $l = -1$ to $l = -5$ as a function of the motional mode number n with the Lamb-Dicke parameter $\eta = 0.05$. For every sideband order the strength monotonically increases with n until turning over, reducing back to zero and reviving. The experiments discussed here are conducted for the range of motional modes shown with the shaded region.

where L_n is the Laguerre polynomial of the n^{th} degree. The occupational probabilities for several displaced thermal states are shown in Figure 4.4. We measure the displacement $|\alpha|$ of the ion i by driving the red motional sideband of the transition between $|g\rangle$, and $|e\rangle$. This interaction couples the electronic and motional states of the ion in the form $|g\rangle|n_i\rangle$ and $|e\rangle|n_i - 1\rangle$ with Rabi frequency $\Omega_{n,n-1}$, which depends on the particular motional state n [2]:

$$\Omega_{n,n-l} = \Omega_0 |\langle n - l | e^{i\eta\left(a+a^\dagger\right)} | n \rangle| \ , \tag{4.24}$$

where Ω_0 is the scale of the coupling strength and η is the Lamb-Dicke parameter. The sideband coupling strengths are shown in Figure 4.5. In the regime of our experiment, the Rabi frequency $\Omega_{n,n-1}$ increases with the n, as shown by the shaded region of the Figure. In this regime, the energy can be determined by monitoring the strength of the sideband interaction. Specifically, we measure the probability to find the ion in the electronic ground

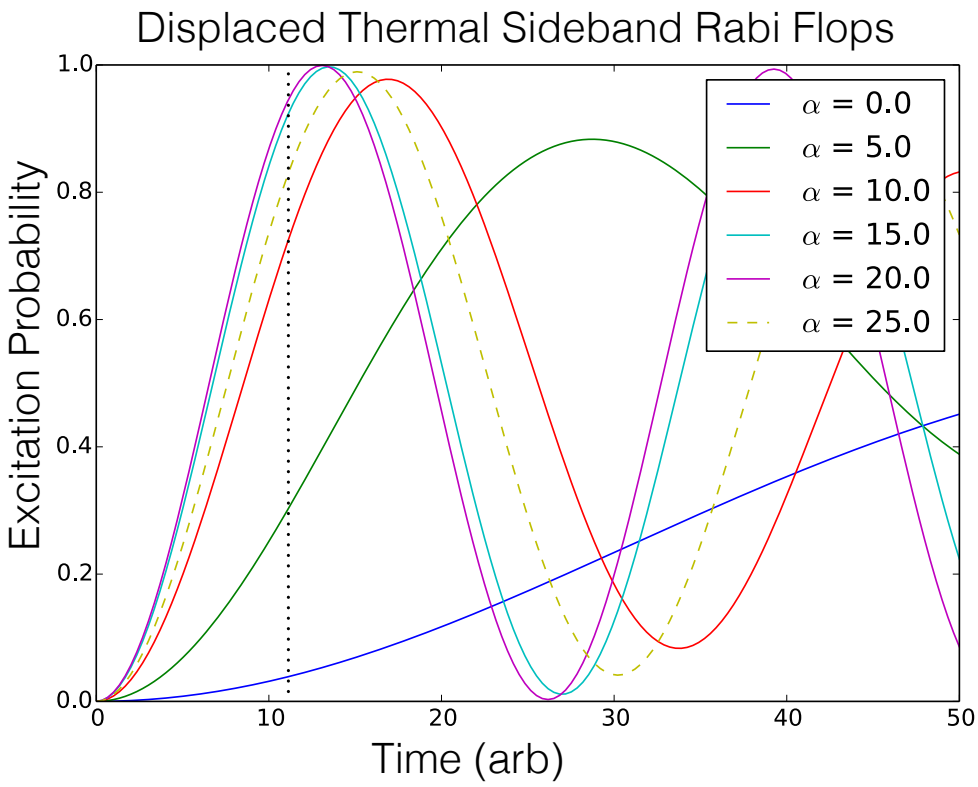

Figure 4.6: Shown are Rabi Flops on the first red sideband for displaced thermal states with various displacements α. For low enough displacements, the strength of the sideband excitation measured at a fixed time along the black dotted line can be directly converted to α. When the displacement excites motional modes outside of the shaded region of Figure 4.5, the sideband strength drops and the displacement can no longer be uniquely determined (dashed yellow line). In these cases, one can monitor the strength of higher order sidebands to determine the displacement.

state $|g\rangle$:

$$P_g(t) = \frac{1}{2}\left[1 + \sum_{n=0}^{\infty} p_n^{\text{disp}} \cos(\Omega_{n,n-1} t)\right] \quad . \tag{4.25}$$

First, we extract the initial temperature, $\bar{n}$, without pulsed excitation. The knowledge of $\bar{n}$, the laser intensity, and the trapping parameters allows us to numerically invert Eq. 4.25. This way, as shown in Figure 4.6 we directly convert the measured ground state probability $P_g(t)$ to the displacement $|\alpha|$.

4.3 Experimental Results

In this section we describe the energy transport experiment, employing the theoretical tools presented earlier to analyze the results. The experiment proceeds as follows: a chain of N $^{40}Ca^{+}$ ions is confined in a harmonic potential of a linear Paul trap with trap frequencies $(\omega_x, \omega_y, \omega_z) = 2\pi \times (2.25, 2.0, 0.153)$ MHz. A laser at 397 nm is red-detuned with respect to the $S_{1/2}$-$P_{1/2}$ transition to perform Doppler cooling of the whole chain. An intense beam at 397 nm is tightly focused on one end of the chain, as shown in Fig. 4.7 for pulsed excitation of a single ion[1] . After the system evolves freely for a time τ, the ion energy is locally measured with a 729 nm laser tuned to the red motional sideband. The pulsed excitation time is fixed at $t = 7.5$ μs while the duration of sideband interaction for energy readout is 10 μs. Both are short compared to the characteristic coupling time of the leftmost ion $2\pi/t_{12}$, as calculated in section 4.2.1, to address only the local mode of motion.

The results for the 5-ion chain and the comparison to theory are presented in Fig. 4.8. The experimental data are in agreement with the normal mode dynamics presented in equation 4.7 where the leftmost ion is initially displaced in the radial direction. The theory has no free parameters and uses independently measured trap frequencies as inputs.

We repeat the experiment with progressively longer chains. The energy of the rightmost ion during the sequence of experiments is presented in Fig. 4.9. We observe clear revivals of the energy even for long chains. With the increase of the length of the ion chain, only a few additional eigenmodes are populated with the local excitation. Once again, the eigenfrequencies of the populated eigenmodes rephase resulting in observed revivals.

For all ion numbers, the time it takes for the excitation to propagate across the entire chain is comparable to the coupling rate between nearest neighbors, $2\pi/t_{12}$. This is explained by the fact that the excitation does not simply hop from one ion to another along the chain. The intuition about excitation hopping does not apply because all of the ions are coupled to each other due to the long range of the Coulomb interaction. It is more appropriate to analyze the dynamics in terms of a normal mode picture. The initial excitation creates a superposition of the normal modes, and the energy is transferred across the chain after rephasing of some of the excited modes of motion and is, thus, comparable to the normal-mode frequency splitting.

Interestingly, we observe that the timescale of the revivals is similar even for long ion chains. This is explained by the fact that the eigenfrequencies of the populated 10 normal modes have only a weak dependence on the ion number. For example, the splitting between the eigenfrequencies of the modes $\vec{v}_1$ and $\vec{v}_5$ increases by $\sim 3\%$ as the length is increased from $N = 5$ to $N = 25$. It can be seen that the revival features become sharper for longer chains: more normal modes participate in the dynamics and the ions are spaced closer

[1]The pulsing is implemented by using an RF switch (Minicircuits ZYSWA-2-50DR) to switch off and on the 397 nm laser beam passing through an AOM double pass. The switch is driven with a function generator (Rigol DG4062). The advanced features of Rigol DG4062 make it possible to control the pulsing time with an external TTL pulse as well as the phase of the generated square wave. The response time of the AOM was minimized by reducing the waist of the laser beam.

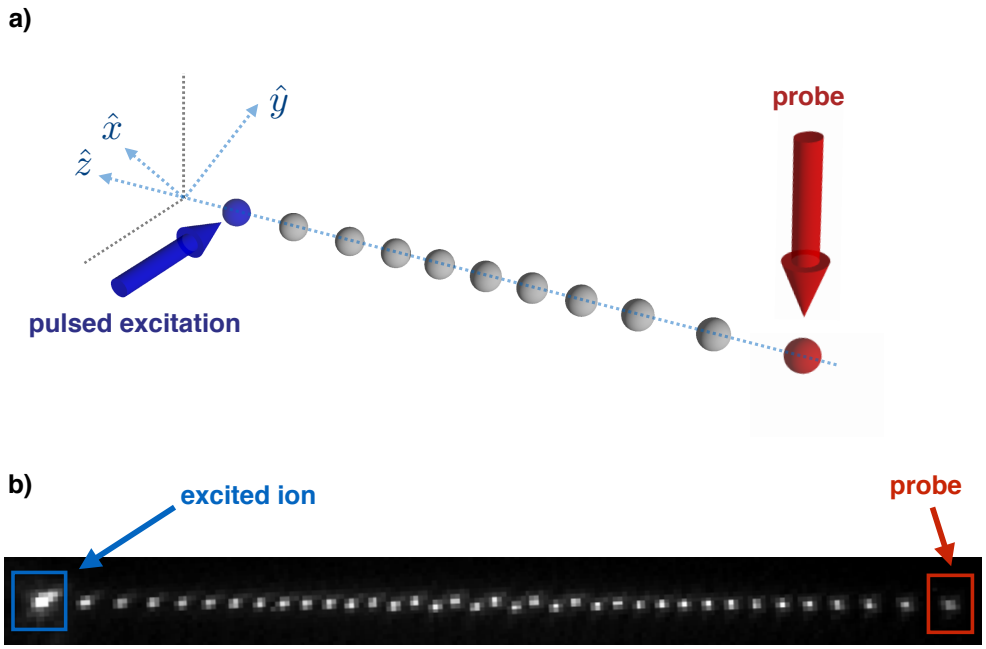

Figure 4.7: The schematic of the energy transport experiment. (a) The pulsed excitation beam (blue) rapidly adds energy to the ion chain. After a free time evolution, the energy can be read out anywhere along the chain with a probe at 729 nm (red). Both beams are at 45° with respect to the radial modes of motion. (b) The CCD image of the ion chain with 37 ions with the tightly focused excitation beam.

together increasing the coupling rate. The faster coupling also leads to a higher energy of the rightmost ion for evolutions times $\tau \sim 0$ μs: some energy has already transferred from the excited ion to the rightmost ion by the time the pulsed excitation is complete.

Even for very long chains, the energy revivals persist for a long time compared to the coupling strength. This is illustrated by Fig. 4.10. This measurement was performed with 37 ions in a partially zigzag configuration as shown in Fig. 4.7b. The energy revivals continue even after an evolution time of $\tau = 40$ ms. For times longer than 40 ms the dynamics wash out, likely due to the instability of the trapping frequencies. Consecutive measurements apart by 12 minutes for the evolution time $\tau = 40$ ms revealed a 20 μs shift in the position of the revival peak, corresponding to 5×10^{-4} change in the coupling strength, ω_z^2/ω_x. The trap frequencies, particularly the radial frequency ω_x, are not expected to be stable at this level.

The measurement presented in Fig. 4.10 shows a large difference in the excitation between adjacent ions: the rightmost ion in the chain and its neighbor. While the energy efficiently transfers from the leftmost kicked ion to the rightmost ion, the ion second from the right does not get as energetic. This phenomenon follows from the normal mode decomposition

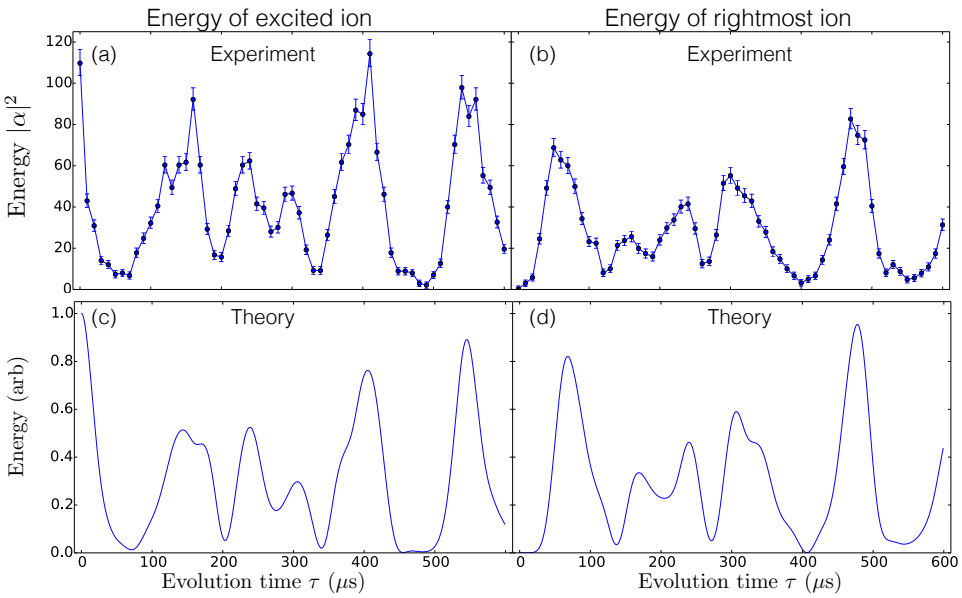

Figure 4.8: Energy transport in a chain of 5 ions. The leftmost ion is given a kick and its energy is measured after a subsequent evolution time τ as shown in (a). The energy revivals occur at the rephasing times of the eigenfrequencies and approach the energy of the initial excitation. Plot (b) shows the energy of the rightmost ion: initially unexcited the energy from the kick is rapidly transferred across the chain. The energy of both ions evolves according to the populated normal modes of motion. Each point is a result of 500 measurements resulting in the denoted statistical error bars. The points are connected to guide the eye. Plots (c) and (d) show the kinetic energies of the ions from the molecular dynamics simulations.

of the initial excitation: the excited normal modes have a small projection onto the motion of the ion second from the right. In the local mode picture, it can also be seen that the efficient transfer of energy occurs because the kicked and the rightmost ion have the same local trap frequency leading to an on-resonant coupling. However, the local frequency of the ion second from the right is different (by approximately the next neighbor coupling), leading to an off-resonant excitation.

4.4 Future Pursuits

The presented results demonstrate the first steps towards realizing the theoretical proposals relying on the motional degree of freedom of long ion chains. The results are in agreement with the theoretical model of normal mode dynamics, demonstrating a sufficient level of control of the experimental parameters. The system is well isolated from the environment –

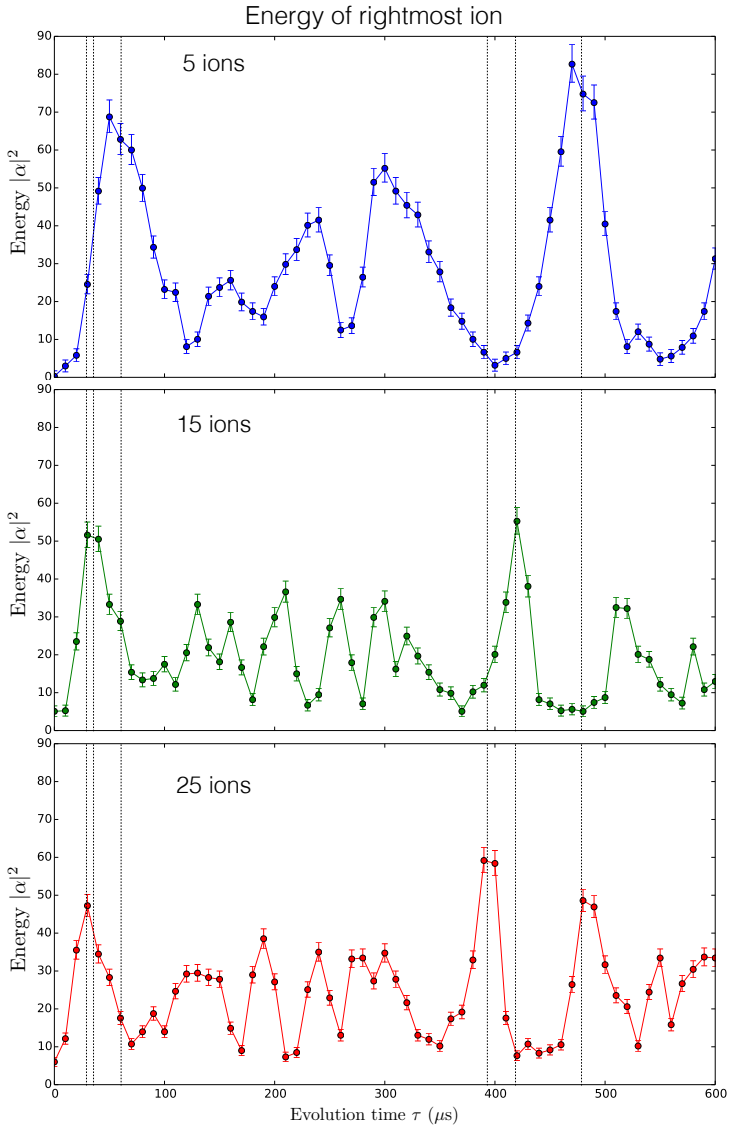

Figure 4.9: The energy of the rightmost ions measured for progressively longer chains. As seen by position the first peak, the rate of energy transfer across the chain increases slightly due to the reduction in the inter-ion distance. Due to the faster coupling for longer chains, the feature size of the revivals decreases, as can be observed from the width of the first revival peak. Also, the average measured energy drops for longer chains: with a greater number of participating normal modes, the excitation is distributed among more ions.

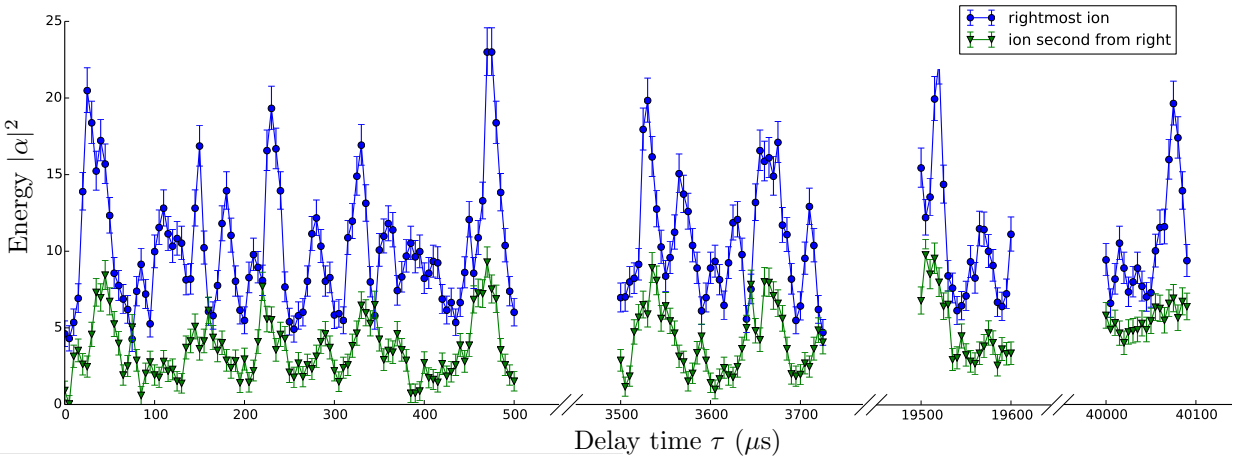

Figure 4.10: The energy revivals measured for the rightmost ion (circles) and ion second from the right (triangles) in a 37-long ion chain partially in the zigzag configuration. The dynamics extend up 40 ms and are likely limited by trap frequency instabilities. The energy from the kicked ion does not efficiently transfer to the ion second from the right. The general rise in the measured energy corresponds to background heating during the free evolution.

an important ingredient for the proposed models – as the observed energy transport dynamics go on for up 40 ms.

With the addition of the second cooling beam, it should be possible to realize a steady state energy distribution, and the measure the temperature profile along the chain. Similar to the requirement of heating, the cooling mechanism also has to be fast compared to the coupling. While it may be difficult to speed up the typical time scale for Doppler cooling on the order of 1 ms, the requirement may be satisfied instead by the reduction in the ion-ion coupling with a lower axial confinement frequency.

The question arises whether it is possible to go beyond the normal mode dynamics. It has been shown theoretically than non-linearities may significantly alter the energy transport properties of the system. Non-linearities in the potential can lead to additional resonances and coupling among the normal modes. The mode coupling involves energy redistribution among the excited normal modes and, perhaps, eventual thermalization of the ion chain. However, as we show in appendix C.2, the intrinsic non-linearity due to the Coulomb interaction in a linear chain is small. A stronger non-linearity may be engineered with an additional standing light wave inside a laser [8]. Additionally it may be possible to repeat the presented experiments while exciting the soft mode present in the structural transition to the zigzag mode. In that situation the non-linearity is much stronger compared to the low local confinement frequency of the soft motional mode.

One disadvantage of the present pulsed heating technique is that it requires scattering photons, heating the ion during the generated displacement operation. This can be avoided

by exciting the ion with a traveling wave resonant with the trap frequency and generated either with Raman beams detuned from the $S_{1/2} - P_{1/2}$ transition or by simultaneously driving both the red and the blue sideband using the 729 nm laser. Another advantage of exciting the ion with the 729 nm traveling wave is that the phase of the energy readout pulse can be controlled relative to the phase of the initial excitation. Such phase control is the basis of the recently proposed method to implement multidimensional spectroscopy with trapped ion chains [36].

Chapter 5

Detection of Quantum Correlations

5.1 Introduction

Detection of quantum correlations between two quantum systems typically requires control and access of both of the systems. This is well exemplified by the procedure of quantum state tomography. Each qubit needs to be measured in three orthogonal bases of the Bloch sphere, corresponding to the control requirement of performing single qubit rotations. Every subsystem needs to be accessible because the density matrix of a N-qubit system requires measurements in every combination of the basis vectors of the constituent qubits. Performing the full procedure of state tomography requires a total of 3^N measurements, which scales rather unfavorably for large quantum systems.

A natural question to pose is whether it is possible to prove the presence of quantum correlations between two subsystems of a bipartite quantum system if the experimenter does not have access to one of the subsystems. The term access here refers to the ability to perform single qubit rotations or measurements. This situation arises in a multitude of scenarios: the experimenter may be presented with a large ensemble of quantum spins, for instance, where it is not technically possible to perform operations on all of the spins. In the context of quantum communication protocols, each party only has access to their part of the shared quantum state. Most generally, whenever a quantum system interacts with its environment, it is considered an open quantum system. By definition, the system's environment is not accessible, yet the presence of quantum correlations between the system and the environment can play a large role in the system dynamics.

At first brush, it seems implausible to be able to observe quantum correlations with limited access. It is well known, for instance, that in the case of a Bell-state $|\Psi\rangle = |00\rangle + |11\rangle$ nothing can be said about the presence of entanglement from local measurements of only one particle. Tracing out the second spin results in a mixed state of the first particle: $\rho_0 = |0\rangle\langle 0| + |1\rangle\langle 1|$ and, hence, in the absence of additional knowledge, local measurements yield no information about the entanglement. In this Chapter we present a protocol, proposed by Manuel Gessner and Heinz-Peter Breuer [37, 38] to accomplish this very task:

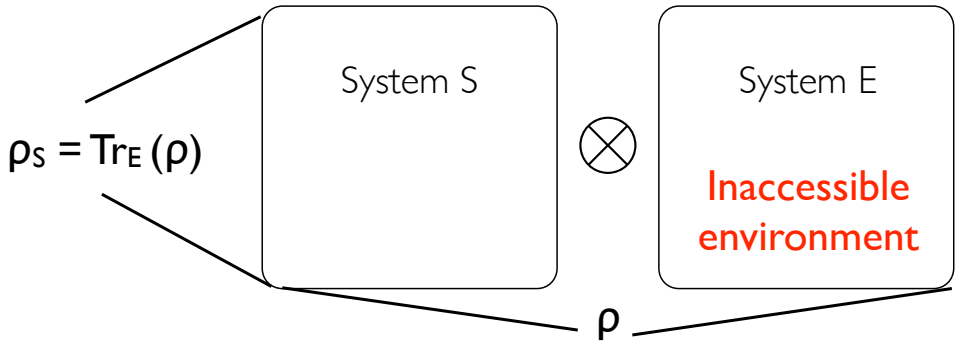

Figure 5.1: The open quantum system S is interacting with an inaccessible environment E. Together, they form a closed quantum system described by the density matrix ρ. The reduced density matrix of S is calculated by tracing out the environment E.

witness quantum correlations between two quantum systems without access to one of them. The method relies on observing the dynamics of the accessible system in the presence of a unitary time evolution that depends on the presence of quantum correlations between the two quantum systems. After proposing the protocol, Manuel Gessner worked together with us to realize the ideas experimentally.

In this Chapter, we first review the protocol for local detection of quantum correlations and then describe the experimental implementation of the protocol using two degrees of freedom of a single trapped ion [39]. The intention is to elaborate on the published results, providing more detail and intuition.

5.2 Local Detection Protocol

We present the protocol by employing the formalism of open quantum systems [40]. As shown in Fig. 5.1, we consider two quantum systems S and E where the system E is the inaccessible environment of the open quantum system S. The goal is to determine whether or not there exist quantum correlations between S and E. Since the environment is not accessible, we are only allowed to perform operations and measurements locally on S, hence the name of local detection protocol.

The protocol is depicted in Fig. 5.2. We start with the joint state of both quantum systems S and E. The state may contain quantum correlations at time $t = 0$ and is described by the density matrix $\rho(0)$. The protocol establishes the presence of quantum correlations by comparing the time evolution of the accessible system S with quantum correlations between S and E with the situation when the initial correlations are removed. The protocol consists of three steps: The first step is to subject the system to a unitary time evolution for a time t, $U(t)$ such that $\rho(t) = U(t)\rho(0)U(t)^\dagger$. The local part of the system, S, at times 0 and t are found by tracing out the environment: $\rho_S(0) = \mathrm{Tr}_E\{\rho(0)\}$ and $\rho_S(t) = \mathrm{Tr}_E\{U(t)\rho(0)U(t)^\dagger\}$.

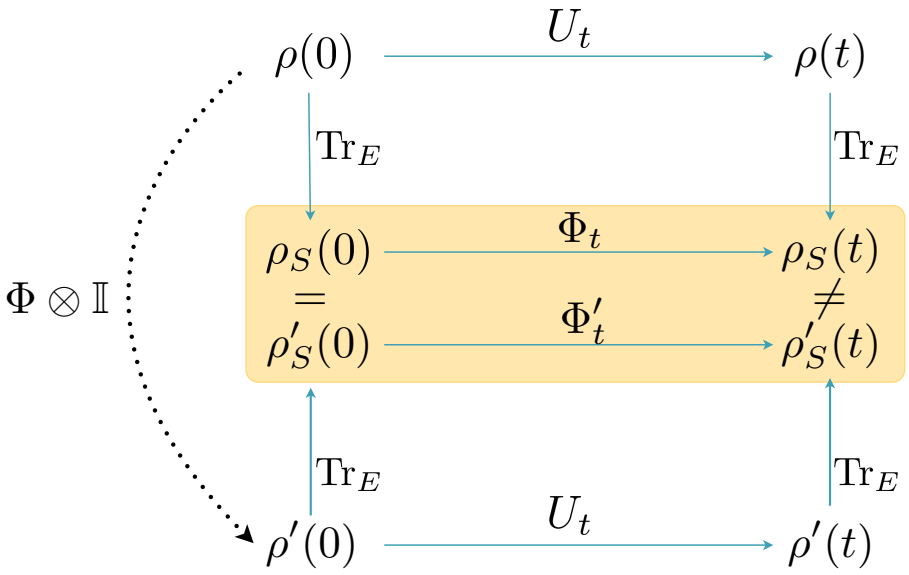

Figure 5.2: The protocol for local detection of quantum correlations. The density matrix ρ of the joint system of S and E is subjected to the unitary time evolution $U(t)$. Additionally, the dephasing operation Φ removes all quantum correlations from ρ producing a state $\rho\prime$. In both cases, the state of the accessible system ρ_S is found by tracing out the inaccessible environment. The presence of quantum correlations in ρ is established if the dephasing operation modifies the time evolution of ρ_S.

The evolution of ρ_S may not be unitary as some information may be flowing to or from the environment. In the case where S and E are statistically independent at time $t = 0$, the evolution of ρ_S will be described by a dynamical map, which is a convex-linear, completely positive and trace-preserving quantum operation [40]. In this case, it is common to use master-equation methods to predict the time evolution of the open quantum system. If there are initial quantum correlations, then these methods can no longer be employed. In general, the time evolution of ρ_S will depend on the presence of initial quantum correlations between S and E. The local detection protocol exploits this property in order to witness the presence of initial quantum correlations.

The second step of the protocol is to apply an operation Φ to the accessible system S. This operation is called the dephasing operation and will be discussed in more detail below. The dephasing operation removes all quantum correlations between S and E, producing a new joint density matrix $\rho\prime(0)$ but it does not alter the initial reduced density matrices of either quantum system: $\rho\prime_S(0) = \rho_S(0)$, $\rho\prime_E(0) = \rho_E(0)$. The dephased system is subjected to the same time evolution $U(t)$, producing a new reduced density matrix $\rho\prime_S(t)$ after the interaction time t.

The third and final step of the protocol is to compare the two resultant density matrices

$\rho_S(t)$ and $\rho\prime_S(t)$. If the two are equal, then nothing can be concluded about the presence of initial quantum correlations in ρ. However, if the reduced density matrices differ, then the initial state contained quantum correlations. To see why, consider that the reduced system S started in the same initial state $\rho_S(0) = \rho\prime_S(0)$ and yet became different after evolving for time t. Therefore the dephasing operation must have been responsible for this change. Since its only effect is to remove quantum correlations from the state ρ, quantum correlations must have been present in the initial state ρ, when the dephasing operation was applied.

We have not yet discussed how it is possible to perform the dephasing operation Φ, which removes all quantum correlations between the environment E and the open quantum system S without altering either reduced density matrix. This is accomplished by performing state tomography of the open quantum system S to measure $\rho_S(0)$ and calculating the eigenvectors π_μ. Then the dephasing operation Φ is defined as a projection onto the calculated set of eigenectors: $\rho\prime = \sum_\mu (\pi_\mu \otimes I)\, \rho\, (\pi_\mu \otimes I)$. Defined this way, the dephasing operation has several important properties. It is a local operation of the accessible system S as shown explicitly by applying the identity operation I onto the environment. The dephasing operation does not alter the state of S because it is a projection onto its own eigenbasis. However, as explicitly proven in [38], the dephasing operation removes all quantum correlations between S and E. The procedure of projection onto the set of eigenvectors of $\rho_S(0)$ can be interpreted as a non-selective measurement. In a non-selective measurement, the experimenter performs a quantum measurement of the system ρ_S to project the system onto the measurement eigenbasis. This may require rotating the state of the system such that the measurement eigenbasis coincides with the eigenvectors of $\rho_S(0)$. We will discuss the experimental realization of this operation in detail in the next section.

We illustrate the action of the dephasing operation by considering a simple example of a quantum-correlated state. Suppose that the initial quantum correlated state $|\Psi\rangle$[1], where the first qubit is the system S and the second is E, is:

$$|\Psi\rangle = \frac{3}{5}|00\rangle + \frac{4}{5}|11\rangle. \tag{5.1}$$

The density matrix ρ is readily computed,

$$\rho = |\Psi\rangle\langle\Psi| = \frac{1}{25}\begin{pmatrix} 9 & 0 & 0 & 12 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 16 \end{pmatrix} \tag{5.2}$$

with the off-diagonal terms are the quantum coherences between the system S and E. The

[1]This state is chosen rather than a simpler Bell state $|\Psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ to avoid the technical issue of degenerate eigenvalues of the reduced density matrix ρ_S. This theoretical point is of little importance to the experimental realization of the protocol.

reduced density matrices are calculated to be the same due to the symmetry of the state:

$$\rho_S = \mathrm{Tr}_E\{\rho\} = \frac{1}{25}\begin{pmatrix} 9 & 0 \\ 0 & 16 \end{pmatrix} \tag{5.3}$$

$$\rho_E = \mathrm{Tr}_S\{\rho\} = \frac{1}{25}\begin{pmatrix} 9 & 0 \\ 0 & 16 \end{pmatrix} \tag{5.4}$$

The diagonal form of ρ_S means that the eigenvectors are given by the bases vectors $\pi_1 = |0\rangle$ and $\pi_2 = |1\rangle$. The dephasing operation applied to the system S is then defined as a projection operation onto these eigenvectors:

$$\rho\prime = (|0\rangle\langle 0| \otimes I)\,\rho\,(|0\rangle\langle 0| \otimes I) + (|1\rangle\langle 1| \otimes I)\,\rho\,(|1\rangle\langle 1| \otimes I) \tag{5.5}$$

$$= \frac{1}{25}\begin{pmatrix} 9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 16 \end{pmatrix} . \tag{5.6}$$

We see that indeed the dephasing operation has removed all the quantum correlations. However, it has not affected the local reduced density matrices:

$$\rho\prime_S = \frac{1}{25}\begin{pmatrix} 9 & 0 \\ 0 & 16 \end{pmatrix} = \rho_S \ , \tag{5.7}$$

$$\rho\prime_E = \frac{1}{25}\begin{pmatrix} 9 & 0 \\ 0 & 16 \end{pmatrix} = \rho_E \ . \tag{5.8}$$

We have seen that by observing a difference in the evolution of the accessible system ρ_S with and without dephasing, we are able to establish the presence of quantum correlations in the initial state ρ. The protocol never makes use of the inaccessible environment, all of the required operations, including the dephasing operation, are performed on the accessible system S. It is important to emphasize that if we observe no difference between ρ_S and $\rho\prime_S$, it does not mean that there were no initial quantum correlations. It is possible that the chosen unitary time evolution $U(t)$ was simply not sensitive to the presence of quantum correlations.

The protocol never assumes that the systems are in a pure quantum state. However, if the system were in a pure state, then the detected quantum correlations correspond to entanglement between the two states. For mixed states, the detected correlations correspond to the concept of quantum discord. In the next section, we present the experimental implementation of the local detection protocol using a single trapped ion. The experiments were performed for both quasi-pure and mixed states, confirming the validity of the protocol in both regimes.

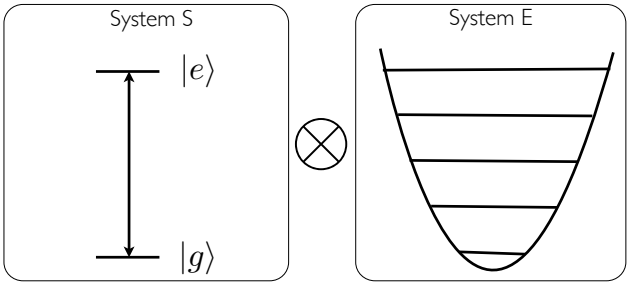

Figure 5.3: The ion's electronic state is chosen to be the accessible system S while ion's harmonic motion in the trap plays the role of the inaccessible environment E.

Protocol	Implementation
Open quantum system S	Ion electronic state
Inaccessible environment E	Ion motional state
Dephasing operation Φ	AC Stark shift with 397 nm laser
Unitary evolution $U(t)$	Blue sideband interaction

Table 5.1: Summary of the experimental implementation of the local detection protocol.

5.3 Single Ion Demonstration

The goal of the experiment was to implement the local detection protocol described above. The realization uses a well-studied and precisely controlled system of a a single trapped ion. Verifying the protocol's validity using a well-understood system paves the way for its use in more complex scenarios.

5.3.1 Choice of Physical Systems

A number of choices had to be made regarding the protocol's implementation. The summary of these selections is presented in Table 5.1. The protocol requires two quantum systems S and E, where the quantum state of the system S is subject to arbitrary operations and measurements. The choices for these are shown in Fig. 5.3. We chose the electronic state of the ion to represent the system S. It is a two level system consisting of the ground state $|g\rangle = |\mathrm{S}_{1/2}, m_j = -1/2\rangle$ and the long-lived excited state $|e\rangle = |\mathrm{D}_{5/2}, m_j = -5/2\rangle$. All of the state operations required for the protocol have been demonstrated with high fidelities in the context of quantum information processing [41]. The inaccessible environment E is modeled by the one-dimensional harmonic oscillator motion of the ion in the trap.

We need to generate quantum correlations between the chosen quantum states in order to detect their presence. This is done with the blue sideband interaction of the Jaynes-Cummings Hamiltonian produced with the 729 nm laser tuned to the first blue motional

sideband [2],

$$U(t) = \frac{\hbar}{2}\Omega_0\eta\left(\hat{a}^\dagger\sigma_+ e^{i\phi} + \hat{a}\sigma_- e^{-i\phi}\right) \ , \tag{5.9}$$

where Ω_0 is the carrier Rabi frequency, and η is the Lamb-Dicke parameter. The same interaction is chosen for the unitary time evolution $U(t)$ that distinguishes between the dephased and the undephased systems. Aside from the unitary interaction, we abstain from performing any operations on the ion motion, as we are not allowed to access the environment in the course of the protocol.

According to the protocol, the dephasing operation requires performing a non-selective measurement in the eigenbasis of the electronic system. Performing a non-selective measurement involves reading out the electronic state of the ion, a process that may involve scattering many photons off the ion and affecting the inaccessible ion motion. We have, therefore, chosen a different way of dephasing the ion by using AC Stark shift generated by a far-detuned laser. Mathematically equivalent to a non-selective measurement, the dephasing via the AC Stark shift will scatter photons with a very small probability and will not disturb the motional degree of freedom[2].

5.3.2 Dephasing with AC Stark Shift

To perform the dephasing operation in the $\{|g\rangle,|e\rangle\}$ basis, we use a laser near 397 nm tightly focused onto the ion and far-detuned from the $S_{1/2} - P_{1/2}$ transition, $\Delta = +2\pi \times 400\text{GHz}$ as shown in Figure 5.4. With the coupling strength characterized in terms of the Rabi frequency Ω, the effective Hamiltonian for the AC Stark shift in the far-detuned regime is [42]:

$$H_{\text{ac}} = -\frac{\hbar\Omega^2}{4\Delta}\left(|p\rangle\langle p| - |g\rangle\langle g|\right) \ , \tag{5.10}$$

where $|p\rangle$ denotes the $P_{1/2}$ electronic state. We analyze this interaction using the following bases of three relevant energy levels:

$$|e\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \ , \quad |p\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \ , \quad |g\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \ . \tag{5.11}$$

Assuming that the $P_{1/2}$ electronic state is not populated and there are no initial coherences between the $P_{1/2}$ energy level and the qubit, we compute the effect of applying the

[2]The dephasing operation could still be implemented by measuring the electronic state but the procedure requires some additional complexity. To dephase the system in the $\{|g\rangle,|e\rangle\}$ basis, we need to be able to project onto those electronic states. Projecting the ion onto the dark state $|e\rangle$ does not affect the ion motion as no photons are scattered during state readout. Projection onto the ground state $|g\rangle$ may be implemented as a π-pulse to rotate the quibit to $|e\rangle$, state readout to project the system into $|e\rangle$ followed by another π rotation back to the ground state. The subsequent time evolution of the system would need to be weighted according to the obtained measurement probabilities.

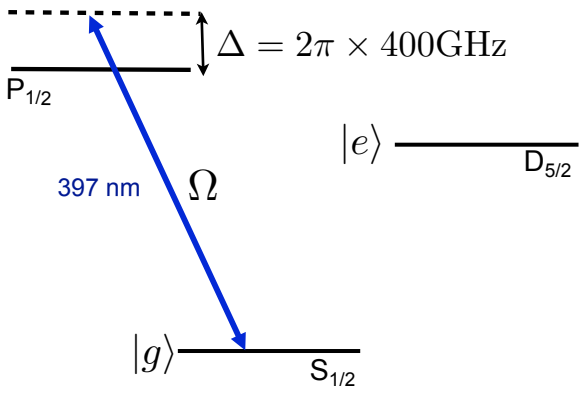

Figure 5.4: The AC Stark shift is generated with the laser at 397 nm blue-detuned from the $S_{1/2} - P_{1/2}$ transition by $\Delta = +2\pi \times 400\text{GHz}$.

Hamiltonian H_{ac} for a time t on a general density matrix ρ representing the three-level ion,

$$\rho = \begin{pmatrix} \rho_{ee} & 0 & \rho_{eg} \\ 0 & 0 & 0 \\ \rho_{ge} & 0 & \rho_{gg} \end{pmatrix} . \tag{5.12}$$

The density matrix evolves as follows:

$$\rho(t) = e^{-iH_{\text{ac}}t/\hbar}\rho e^{iH_{\text{ac}}t/\hbar} \tag{5.13}$$

$$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & e^{\frac{i\Omega^2}{4\Delta}t} & 0 \\ 0 & 0 & e^{-\frac{i\Omega^2}{4\Delta}t} \end{pmatrix} \begin{pmatrix} \rho_{ee} & 0 & \rho_{eg} \\ 0 & 0 & 0 \\ \rho_{ge} & 0 & \rho_{gg} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & e^{-\frac{i\Omega^2}{4\Delta}t} & 0 \\ 0 & 0 & e^{\frac{i\Omega^2}{4\Delta}t} \end{pmatrix} \tag{5.14}$$

$$= \begin{pmatrix} \rho_{ee} & 0 & \rho_{eg}e^{\frac{i\Omega^2}{4\Delta}t} \\ 0 & 0 & 0 \\ \rho_{ge}e^{-\frac{i\Omega^2}{4\Delta}t} & 0 & \rho_{gg} \end{pmatrix} . \tag{5.15}$$

When the AC Stark shift is applied, the populations of the electronic states $|g\rangle\langle g|$ and $|e\rangle\langle e|$ are not affected but the coherence evolve with the frequency $\Omega^2/4\Delta$. The precession frequency $\Omega^2/4\Delta \sim 2\pi \times 40\text{kHz}$ was measured by performing a Ramsey-type experiment with varying duration t of the AC Stark shift interaction, see Figure 5.5. From this, we calculate the coupling strength, $\Omega \sim 2\pi \times 250$ MHz corresponding to the saturation parameter $s \sim 255$ where the saturation parameter is defined as $s = 2\Omega^2/\Gamma^2$ where $\Gamma = 1/\tau$ is the linewidth of the $P_{1/2}$ energy level with the lifetime $\tau \sim 7.1\text{ns}$. The saturation parameter s can be expressed in terms of the transition saturation intensity $s = I/I_{\text{sat}}$ [43] with the saturation

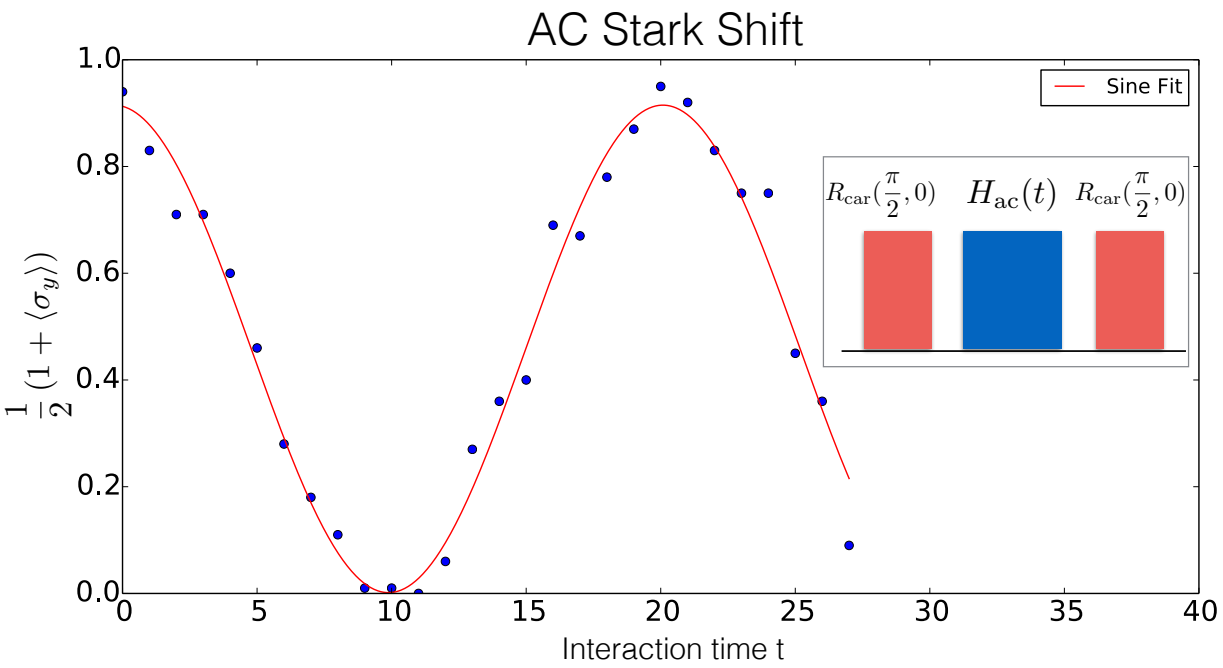

Figure 5.5: We perform a Ramsey-type experiment to measure the precession frequency of the Bloch vector $\frac{\Omega^2}{4\Delta}$ due to the AC Stark shift from a far-detuned laser. The initial $\frac{\pi}{2}$ pulse on the carrier transition creates a superposition state $\Psi = \frac{1}{\sqrt{2}}(|g\rangle + i|e\rangle)$. Then the far-detuned laser at 397 nm is applied to the atom for a variable interaction time t, after which the y-component is the Bloch vector is measured with an additional carrier $\frac{\pi}{2}$ pulse. In this realization, the precession frequency is extracted from a sinusoidal fit to be $\frac{\Omega^2}{4\Delta} \sim 2\pi \times 49$kHz. The contrast of 0.91 reflects imperfections in state preparation and qubit rotations.

intensity $I_{\rm sat}$:

$$I_{\rm sat} = \frac{\pi}{3}\frac{hc}{\lambda^3 \tau} \ , \tag{5.16}$$

where c is the speed of light and the transition wavelength is $\lambda = 397$ nm, yielding $I_{\rm sat} \sim 46.8$ mW/cm^2. The total applied laser power was $P \sim 500$ μW, indicating that the beam waist was approximately 5 μm.

In order to perform the dephasing operation, we sample over $n = 10$ different time AC Stark shift durations t_k evenly spaced between $t_1 = 0$ and $\frac{\Omega^2}{4\Delta} t_n = 2\pi$ with the maximum interaction time $t_n \sim 25$ μs. Averaging over these durations removes the off-diagonal coherence terms in the density matrix, yielding the dephased state $\rho\prime_S(0)$:

$$\rho\prime_S(0) = {\rm Tr}_E \left[\frac{1}{n} \sum_k \left(e^{-iH_{\rm ac} t_k/\hbar} \otimes I \right) \rho(0) \left(e^{iH_{\rm ac} t_k/\hbar} \otimes I \right) \right] \ . \tag{5.17}$$

The time-evolved dephased density matrix $\rho\prime_S(t)$ is calculated as the average time evolution of the n samples:

$$\rho\prime_S(t) = \mathrm{Tr}_E \left[\frac{1}{n} \sum_k U(t) \left(e^{-iH_{\mathrm{ac}} t_k/\hbar} \otimes I \right) \rho(0) \left(e^{iH_{\mathrm{ac}} t_k/\hbar} \otimes I \right) U(t)^\dagger \right] . \tag{5.18}$$

An intuitive way of picturing the dephasing implementation is that the AC Stark shift rotates the state vector in the $x-y$ plane of the Bloch sphere. By averaging over all of the rotation angles between 0 and 2π, we calculate the dephased state corresponding to the center of the Bloch sphere.

The far detuning of the applied laser ensures that the motional degree of freedom remains nearly unaffected. The expected number of scattered photons during the maximum interaction time of $t_n \sim 25$ μs is $\Gamma \rho_{pp} t_n \sim 3.5 \times 10^{-4}$ with the probably of finding the ion in the $P_{1/2}$ state, ρ_{pp} is given by [2]:

$$\rho_{pp} = \frac{s/2}{1 + s + (2\Delta/\Gamma)^2} , \tag{5.19}$$

where Γ is the width of the $P_{1/2}$ energy level, $\Gamma = 1/\tau \sim 2\pi \times 22.4$ MHz.

5.3.3 Experimental Results

The experiments realizing the local detection protocol proceed as follows: first the motional state of the ion is prepared either in a thermal state with Doppler cooling or close to the ground state with additional sideband cooling. Then we use the blue sideband interaction $U(t)$ for a preparation time t_0 to create quantum correlations between the qubit and the ion's motion. At this point, we perform state tomography of the electronic state to determine the correct eigenbasis for the dephasing operation. We then apply the same blue sideband interaction $U(t)$ for the excitation duration time t_1 to compare the time evolution of the electronic state with and without dephasing.

The results for the sideband-cooled case with preparation time of a $\frac{\pi}{2}$ pulse in the blue sideband, $t_0 \eta \Omega_0 = \frac{\pi}{2}$ presented in Figure 5.6. State tomography of the state after the preparation pulse indicates that the density matrix is diagonal in the $\{|g\rangle, |e\rangle\}$, basis. This is expected for the blue sideband interaction. For example, after tracing out the motional degree of freedom of the prepared state $\Psi = |g, 0\rangle + i|e, 1\rangle$, the density matrix $\rho_S(0)$ will have no off-diagonal terms. The diagonal form of $\rho_S(0)$ means that for dephasing we can use the AC Stark shift method without needing additional single qubit rotations. We also performed state tomography after the dephasing operation to confirm that the measured density matrices and the dephasing operation does not disturb the locally accessible system: $\rho_S(0) = \rho\prime_S(0)$.

We executed the local detection protocol with varying length of preparatory pulses, as shown in Figure 5.7 for preparation times of $t_0 \eta \Omega_0 = \pi$ and $t_0 \eta \Omega_0 = 3\pi/2$. In the case of

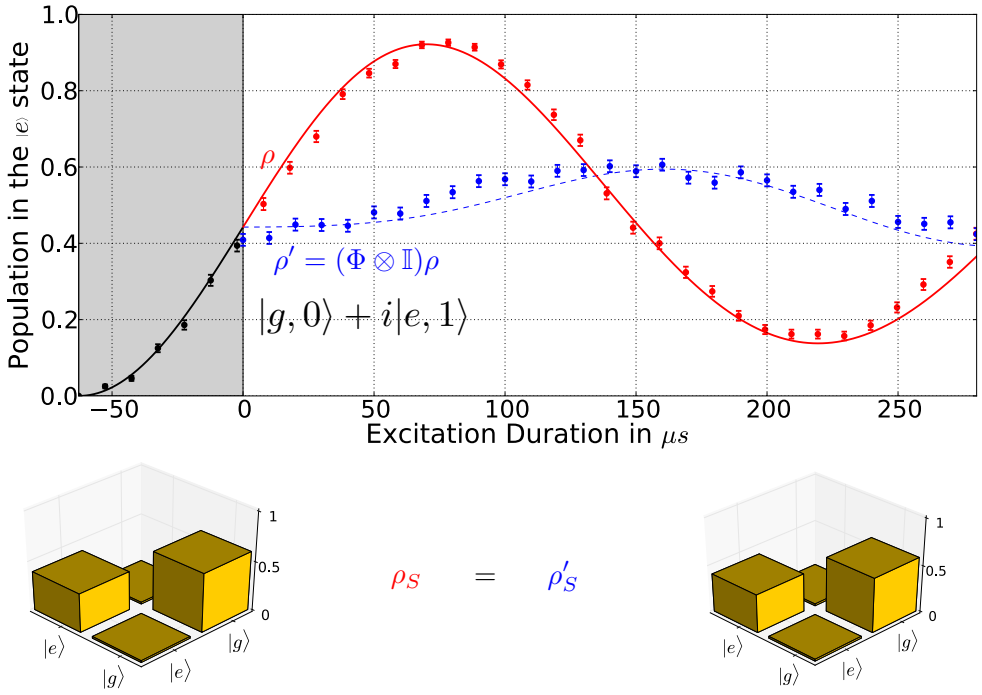

Figure 5.6: Comparing the dephased (blue points) and the undephased time (red points) evolutions when starting close to the motional ground state $\bar{n} = 0.2$. The error bars are the statistical errors for $n = 1000$ readouts, $\sigma = \sqrt{p(1-p)/n}$. The deviation between the two time evolutions proves the presence of quantum correlations in the prepared state. The undephased curve is fitted to a theoretical model for Rabi flopping (red curve) and the fitted parameters are used to predict the time evolution of the dephased curve under the same blue sideband interaction (dashed blue line). For more details, please refer to the experimental publication including the supplementary information [39].

a perfect π pulse on the sideband, there should be no quantum correlations between the electronic and the motional states. The small measured deviation between the dephased and the undephased curves is due to the finite initial temperature: if the motion is not all initially in the ground state then some correlations remain.

The protocol makes no assumptions whether the system or the environment are in a pure quantum state. In fact, we confirmed that the protocol reveals the presence of quantum correlations for mixed states motion as shown in Figure 5.8. In this case, we did not perform sideband cooling and started with the thermal state of the environment, $\bar{n} = 5.9$.

It is possible to use the measurements to extract quantitative information about the present quantum correlations. As shown in [38], the maximum of the trace distance between the dephased and the undephased states (equivalent to the maximum difference in the population of the excited states) provides the lower bound for quantum discord. Interestingly,

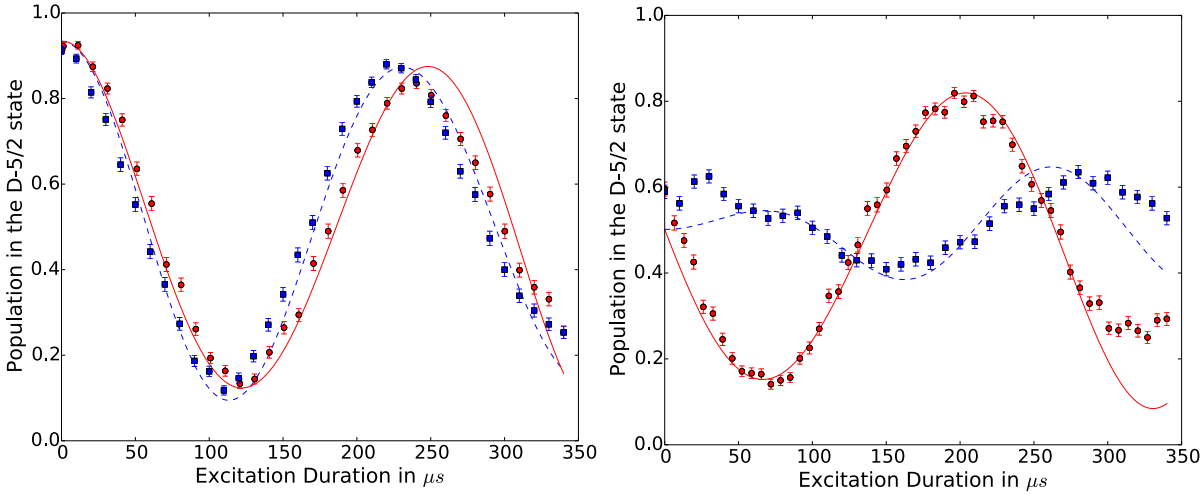

Figure 5.7: Comparing the dephased and undephased time evolutions after a preparatory π pulse (left) and $3\pi/2$ pulse (right) on the blue sideband. After the π pulse, there should be little quantum correlations present, consistent with little deviations between the two curves. The quantum correlations are again present after the $3\pi/2$ pulse and are revealed by the protocol. The plots follow the conventions of Figure 5.6.

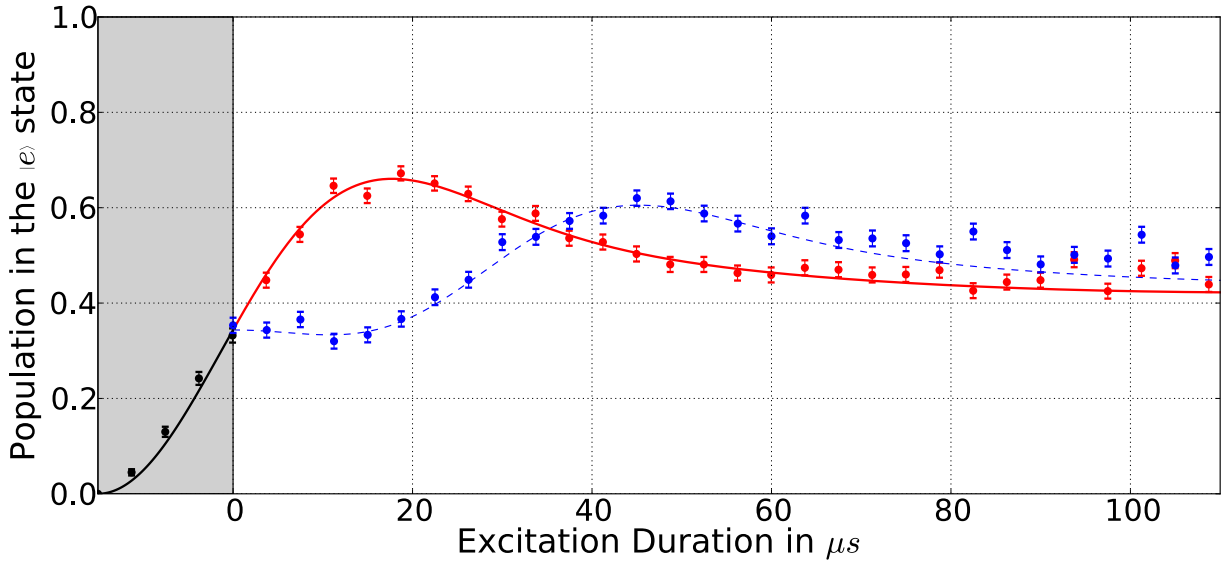

Figure 5.8: Local detection protocol applied with a thermal state of motion still reveals the presence of quantum correlations. The plot follows the conventions of Figure 5.6.

the detected lower bound comes close to actual amount of correlations expected from theory. The reader can refer to [39] for a complete discussion.

5.4 Summary

In this section we have presented the first experimental realization of the local detection protocol, verifying that it reveals quantum correlations in both pure and mixed states. We have experimentally shown that the presence of initial quantum correlations can play a large role in determining open system dynamics. The main strength is of the protocol is its ability to detect quantum correlations when the environment is not accessible. One of the most important presented features is that the measurements provide quantitative information about quantum correlations, making a direct connection between experimental results and the theoretical notion of quantum discord. The maximum measured deviation between the dephased and the undephased time evolutions directly gives the lower bound for the amount of discord initially present between the open quantum system and the environment.

It is important to emphasize that should the protocol not reveal the presence of quantum correlations, it does not mean quantum correlations are absent. The measurements are only a lower bound. Any deviation in the time evolution of the dephased and the undephased states reveals the presence of quantum correlations. Only a single measurement is required in contrast to performing 3^N operations for state tomography. In this respect, the protocol is similar to constructing a particular witness for quantum correlations instead of measuring the entire density matrix.

The experiment used one of the simplest possible representations for the environment, a single harmonic oscillator. This raises an important question as to how the protocol would perform with a larger environment - we will discuss an experiment aimed at answering this question in the next Chapter. In addition, we will build on this proof-of-principle experiment and discuss a direct application of the protocol for the purposes of detecting quantum phase transitions.

Chapter 6

Dephasing with Larger Environments

6.1 Long Trapped Ion Chains

The experiment described in Chapter 5 used the one-dimensional harmonic motion of the ion in the trap to represent the environment E of the open quantum system S represented by the electronic state of the ion. This environment is non-Markovian as there are clear memory effects in the system's time evolution. Here, we extend the experiment by considering a larger environment of multiple harmonic oscillators. The goal is to answer the question of whether or not the protocol still reveals the presence of quantum correlations for sufficiently large environments and to explore the connection with the system's degree of non-Markovianity [44, 45, 46]. In this Chapter, we present the first experimental steps undertaken towards this goal.

The experiment incorporates the local detection protocol with aspects of the energy transport in trapped ion chains presented in Chapter 4. To study transport of energy, we prepared an out-of-equilibrium state by rapidly imparting momentum onto one of the ions in the chain. Similarly, here we use a short sideband pulse of 729 nm to create quantum correlations between the electronic state of the ion and its local motion in the trap. The duration of the entangling pulse t_{SB} has to be short enough such that the pulse's Fourier bandwidth encompasses all of the radial normal modes, see Figure 6.1. Since the local motional mode can be decomposed in terms of the normal modes of the chain, we effectively create quantum correlations between the electronic state and a larger environment containing all of the participating radial normal modes of motion.

We then perform the dephasing operation and compare the time evolution of the dephased and the undephased states under a unitary interaction $U(t)$ for a time t_{max}. The unitary $U(t)$ is chosen to complete the underlying Ramsey experiment. The time evolution is shown in Figure 6.2. If $t_{\mathrm{max}} \leq t_{\mathrm{SB}}$, then the unitary is simply the continuation of the fast sideband pulse used to create the correlations. This is exactly the same as we performed in case of the single ion. If the maximum unitary interaction time is longer than the pulse time, $t_{\mathrm{max}} > t_{\mathrm{SB}}$, then we let the system evolve freely before for a time $t_{\mathrm{max}} - t_{\mathrm{SB}}$ and then apply the second

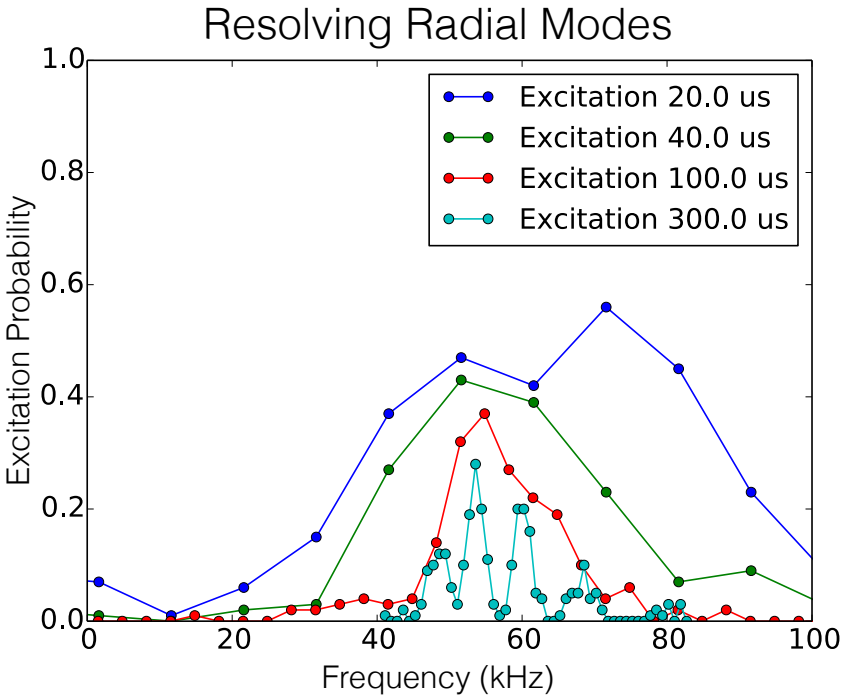

Figure 6.1: The spectra of the red radial motional modes for a five ion chain are shown for various excitation times with the 729 nm laser. For short excitations, the Fourier bandwidth is much larger than the mode splitting and the local mode of motion is excited. With sufficiently long excitations and reduced laser intensity, we resolve the normal modes of ion motion. The spectra are shifted at high laser intensities due to the AC Stark shift effect.

sideband pulse. In either case, the short duration of the sideband pulse interacts with the local mode of ion motion.

The preliminary results of such an experiment conducted with a chain of five ions is shown in Figure 6.3. All of the normal modes of ion motion were prepared in a thermal state with Doppler cooling. We observe that the dephased and the undephased time evolutions differ and, hence, the protocol is successful as detecting the present quantum correlations. The data was collected at the same conditions as the energy transport experiment where the energy supplied to the leftmost ion was transferred to the rightmost ion in $t \sim 70$ μs. Here we see that at that point in the time evolution, the dephased and the undephased curves match. Our interpretation is that the electronic state of the leftmost ion is now correlated with the local motion of the rightmost ion and quantum correlations can no longer be detected locally.

Similar to the energy transport results, the quantum correlations then come back to the leftmost ion: the larger environment is still non-Markovian. Then the protocol once again is able to detect the presence of quantum correlations. For longer time, the difference between the dephased and the undephased time evolutions vanishes due to the limited motional

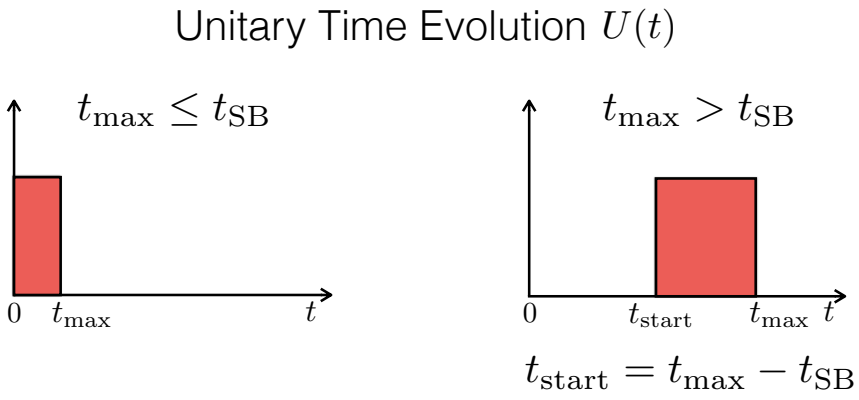

Figure 6.2: Shown is the time evolution $U(t)$ for two different maximum interaction times $t_{\max}$. Sideband pulses are depicted in red. For short time evolutions, the pulse begins immediately after dephasing $t = 0$. For long evolutions, the pulse only starts after a waiting period.

coherence time.

We are currently exploring the systematic effect of laser detuning of the radial sideband pulses. The challenge is that at Doppler temperature, it is no longer possible to perform a brute-force numerical simulation of the dynamics. If we need to consider 10 motional levels for each of the 5 normal modes, then the density matrix representing of the environment has dimension 10^5. Furthermore, similar to the energy transport experiment, the exact same experimental procedure can be repeated with a much larger number of ions.

6.2 Local Signature of Quantum Phase Transitions

In this section, we demonstrate how the local detection protocol can be applied for detection of quantum correlations during a quantum phase transition. There is a wide body of literature exploring the interplay between quantum entanglement and quantum phase transitions. While quantum phase transitions of quantum magnetism have already been realized with trapped ions, [23, 24] there has been no measurements demonstrating the presence of quantum correlations during the quantum phase transition. The advantage of the protocol is that it requires operations only on one subsystem, for example only one spin of a quantum magnet. This is particular pertinent as the experiments have been approaching system sizes for which measuring the full density matrix is no longer feasible.

To demonstrate the use of the protocol, we considering the example of a ferromagnetic quantum Ising model realized in [23]. Our study of quantum correlations between a single site and the rest of the system during the quantum phase transition is particularly compelling

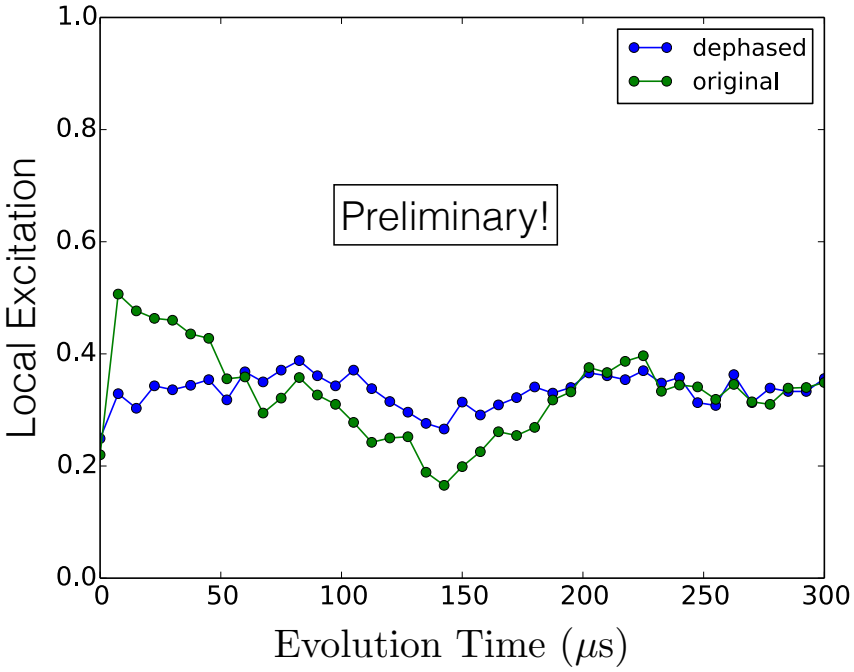

Figure 6.3: Preliminary data applying the local detection protocol to a chain of trapped ions. The leftmost ion is initially entangled with the local mode of motion with a fast sideband excitation pulse. The evolution time consists of a waiting period followed about another fast sideband excitation. The difference between the dephased and the undephased time evolution proves the presence of quantum correlations.

because of a known correspondence between the behavior of entanglement and the critical point in the transverse Ising model [47]. The Hamiltonian H is given by the sum of the Ising coupling and the magnetic field terms:

$$H = \sum_{i<j} -J_{ij}\sigma_x^{(i)}\sigma_x^{(j)} - B\sum_i \sigma_y^{(i)} \ , \tag{6.1}$$

where $\sigma_x^{(i)}$ and $\sigma_y^{(i)}$ are Pauli spin operators applied to the i^{th} spin and B is the strength of the magnetic field along the y direction. We consider the approximate Ising coupling J_{ij} [48] where the coupling strength falls off with distance $|i-j|$ as:

$$J_{ij} \approx \frac{J_0}{|i-j|^\alpha} \ . \tag{6.2}$$

The experimental quantum simulation begins by initializing all the spins to point along the y direction. This is the ground state of the Hamiltonian $H_0 = -B\sum_i \sigma_y^{(i)}$ and an approximate

ground state of H for very strong magnetic fields: $B \gg J_0$. After the preparation step, the magnetic field B is adiabatically ramped down to $B = 0$ such that the system remains in the ground state of H during the ramp. While the prepared initial state is a product state containing has no quantum correlations, the final state of the simulation is an entangled state. More precisely, for $B = 0$ the ground state of H is a superposition of the states $|\uparrow\uparrow \ldots \uparrow\rangle$ and $|\downarrow\downarrow \ldots \downarrow\rangle$ where $|\uparrow\rangle$ and $|\downarrow\rangle$ are the eigenstates of σ_x.

Let us consider the reduced density matrix of only one spin within the chain during the course of the quantum simulation. It is initialized with the Bloch vector pointing along the y direction and terminates as the maximally mixed state with no net magnetization along y. The magnetization during the course of the phase transition is plotted in Figure 6.5. This type of net magnetization measurement has been performed in the experiments confirming that the system has gone through the phase transition. However, the net magnetization does not reveal anything about the presence of quantum correlations during the evolution.

The method of local detection probes the emergence of the quantum correlations between the one spin and the rest of the system during the ramp. We performed numerical simulations using the QuTip framework [49] to confirm that the local detection method can be readily applied to the given system. For the simulation, we used $N = 5$ spins and the coupling coefficient $\alpha = 1.0$. The simulations consist of preparing a correlated state by ramping the magnetic field down to a particular value B, then comparing the time evolution of the dephased and the unperturbed system while the magnetic field remains fixed at the chosen value. The dephasing operation has to be performed in the σ_y basis, which, as described in section 5.3.2, may be implemented with an AC-Stark shift but with additional unitary rotations.

The results of the simulation for various dephasing point along the magnetic field ramp are plotted in Figure 6.4. We see that for early dephasing times, $B/J_0 \gg 1$, the dephased and the undephased time evolutions match, demonstrating that quantum correlations are either not present or do not play a significant role in the ensuing time evolution. For later dephasing times, there is an appreciable difference between the two time evolutions, and it seems to be largest close to the quantum critical point $B/J_0 = 1$. The difference in time evolutions of the dephased and the undephased states has a straightforward interpretation. For a sufficiently slow ramp, the system will remain in the ground state over the course of the ramp. With the magnetic field fixed, the evolution of the undephased state is trivial as the system remains in the ground state of the Hamiltonian. The dephasing operation, however, projects the ground state onto a superposition of the eigenstates. Therefore, the single spin begins to oscillate at the corresponding eigenfrequencies.

To quantify the effect of quantum correlations on the dynamics, we plot the maximum trace distance between the dephasing and the undephased time evolutions as a function of the dephasing point, see Figure 6.5. The maximal trace distance peaks near the quantum critical point. According to the protocol, the maximum trace distance is only the lowest bound on the present quantum correlations. Interestingly here, we see that the peak attainable lowest bound is indicative of the quantum critical point. Experimentally, it is not possible to let the system evolve indefinitely to obtain the largest possible trace distance. But the results

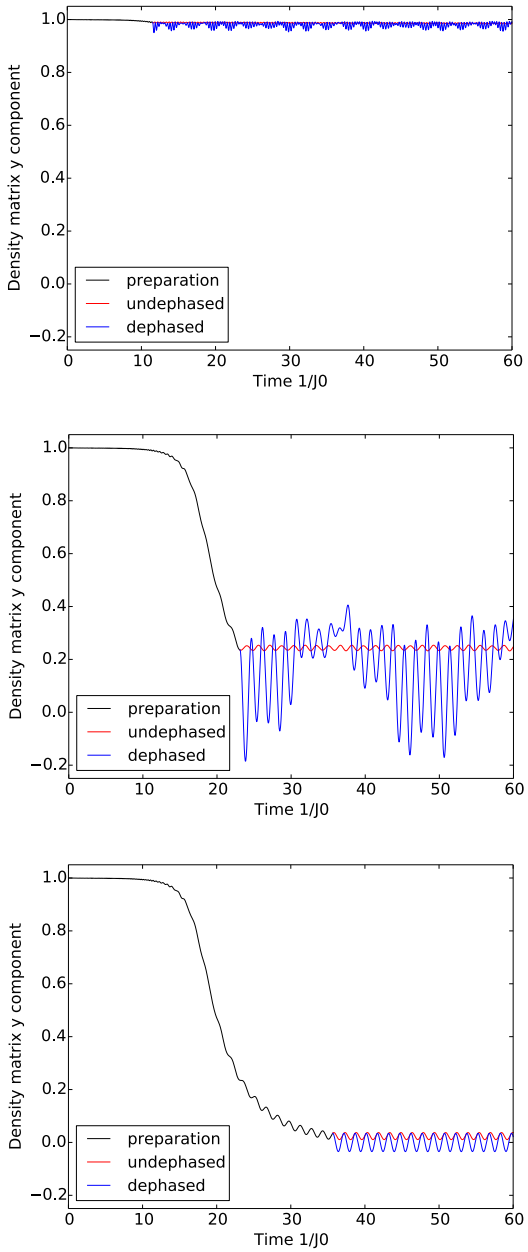

Figure 6.4: The time evolutions of the dephased and the undephased states are shown for three dephasing points along the quantum phase transition: $B/J_0 = 5$ (top), $B/J_0 = 0.5$ (center), and $B/J_0 = 0.05$ (bottom). The numerical simulation takes into account the finite speed of the ramp: the magnetic field is lowered from the initial value of $B/J_0 = 50$ to $B/J_0 = 0.01$ as a decaying exponential with the time constant of $5/J_0$.

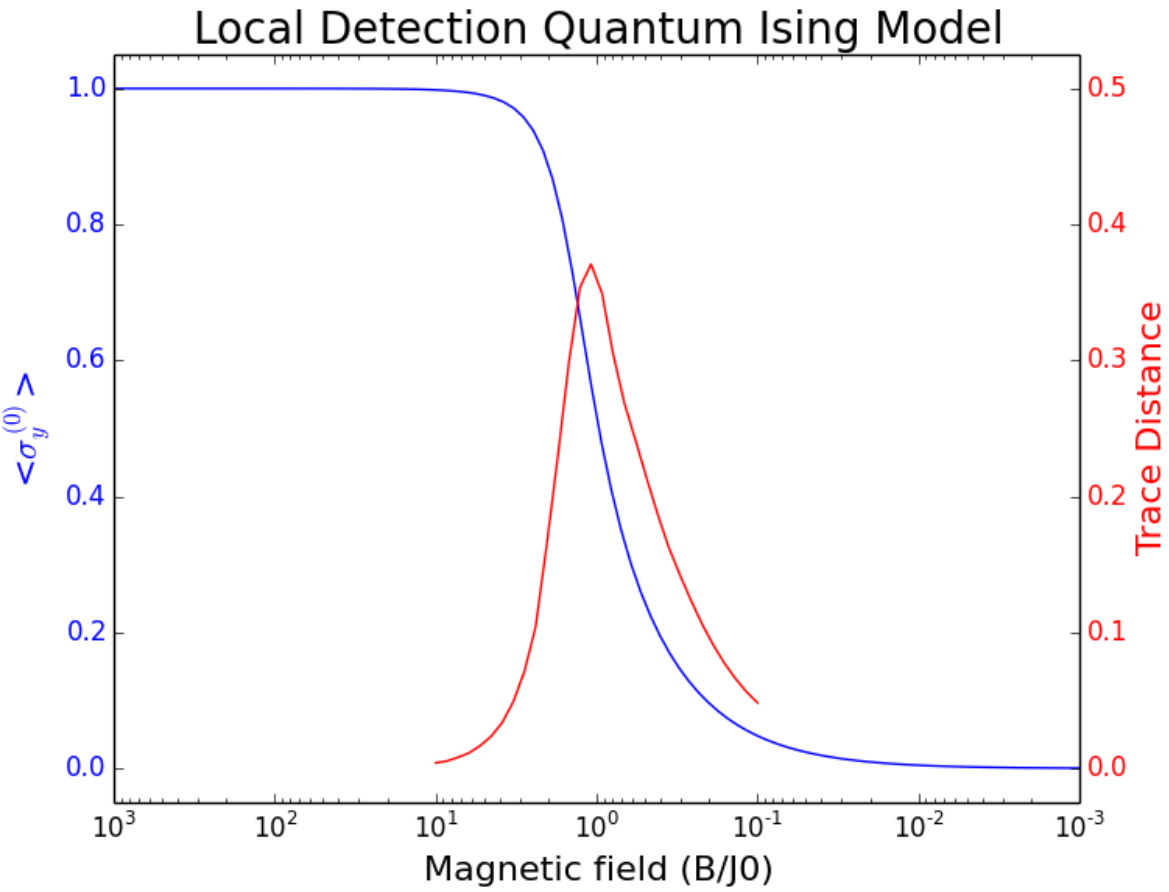

Figure 6.5: Shown in blue is the y component of the Bloch vector of the single spin as the magnetic field is ramped down. For this simulation, we assume an infinitely slow ramp, allowing us to simply compute the ground state of H for every value of B. The x and z components have no preferred direction over the course of the simulation. Shown in red is the maximal local trace distance between the subsequent dephased and the undephased time evolutions when the dephasing operation is applied at the particular field B. The simulation shows that there is a large deviation between the time evolutions, which we expect to be experimentally detectable.

should still be observable for reasonable decoherence timescales, as presented in [50]. For small magnetic fields no quantum correlations are detected with this method even though the spins are in a large superposition state.

While the method for detection of quantum correlations over the course of quantum phase transition was presented in the context of quantum magnetism simulations, we expect it to be broadly applicable to a variety of systems. For instance, we envision applying it to the Jaynes-Cummings-Hubbard model realized with a system of two ions by Toyoda et al. [51]. The protocol may serve as a conclusive experimental verification of a quantum phase transition. The standard definition of a quantum phase transition is that it occurs at zero temperature and is driven by quantum, rather than thermal fluctuations. Such a definition is difficult to verify experimentally. The presented protocol introduces a new experimental definition: for a phase transition to be quantum, the present quantum correlations must play a role in the system's time evolution.

Chapter 7

Summary

In this Chapter we give a brief review of the presented material. The introductory Chapter described the motivation for the conducted work. In Chapter 2, we reviewed the fundamentals of ion trapping and the ion-laser interactions, describing how time-varying electric fields in a Paul trap confine the ions. We then focused on coherent interaction of the laser addressing a narrow electronic transition and gave an overview of Doppler cooling.

In Chapter 3, we presented the experimental apparatus for trapping ions. We reviewed the motivations for selecting a three-dimensional Paul trap for the experiments, and described the design, assembly, and construction of the first-generation trap and the optical cavity. The trap was used to conduct all of the experiments in this thesis, but the cavity was misaligned after the final pumpdown and bakeout. We then presented our design modifications for the next-generation trap and cavity, which are currently under construction. The new designs built on our experience with the first trap and should correct the known shortcomings. The Chapter also described the imaging system for collecting ion fluorescence using a CCD and a PMT and the employed experimental procedures of frequency-resolved optical pumping and auto-crystallization.

In the following Chapter on energy transport, we reported on conducted experiments studying the dynamics of ion motion within long chains. We performed a Newton cradle-like experiment by rapidly imparting momentum onto a single ion at the end of the chain with the technique of pulsed excitation. After letting the system evolve for a variable delay time, we measured the energy of the ions on the extreme ends of the chain. This allowed us to observe the dynamics of the initial energy excitation as it propagated between the observed ions. The results agreed with the normal-mode model of ion motion, demonstrating a good degree of control over ion motion in long chains. The results represented the first experimental steps towards realizing existing theoretical proposals considering ion chains as a model system of coupled oscillators.

In Chapter 5, we presented the technique of detecting quantum correlations between two quantum systems with only access to one of the two subsystems. We used a single trapped ion to implement this technique of local detection. The ion's energy level represented the accessible system while the ion's motion in the trap played the role of the inaccessible

environment. We described the implementation of the dephasing operation, which removes the quantum correlations between the two quantum systems without disturbing either of the systems. The protocol relied on detecting the difference in the time evolution of the accessible system with and without the presence of quantum correlations. If the time evolution differed with and without the dephasing operations, we concluded that quantum correlations must have been present when the dephasing operation was applied. The protocol was shown to successfully detect quantum correlations for both quasi-pure and thermal states of ion motion.

The ideas of energy transport in long ion chains and the protocol for locally detecting quantum correlations coalesced in Chapter 6. There, we described the first experiments for detecting quantum correlations between the electronic level of a single ion and a larger environment consisting of many radial normal modes of chain motion. The results were relevant for the general study of the role of quantum correlations for dynamics of many-body systems. Lastly, we focused on how the local detection protocol may be used for detection of quantum phase transitions.

Bibliography

[1] Gebhard Littich. "Electrostatic Control and Transport of Ions on a Planar Trap for Quantum Information Processing". MA thesis. ETH Zurich, 2011.

[2] D. Leibfried et al. "Quantum dynamics of single trapped ions". In: *Rev. Mod. Phys.* 75 (Mar. 2003), pp. 281–324. DOI: 10.1103/RevModPhys.75.281.

[3] R. E. March and J. F. Todd. *Quadrupole Ion Trap Mass Spectrometry*. 2nd. Wiley-IEEE, 2005. ISBN: 9780471488880.

[4] C. A. Blockley, D. F. Walls, and H. Risken. "Quantum Collapses and Revivals in a Quantized Trap". In: *EPL* 17.6 (1992), p. 509. DOI: 10.1209/0295-5075/17/6/006.

[5] Christian Roos. "Controlling the quantum state of trapped ions". PhD thesis. University of Innsbruck, 2000.

[6] Stephan Gulde. "Experimental realization of quantum gates and the Deutsch-Josza algorithm with trapped $^{40}Ca^{+}$-ions". PhD thesis. University of Innsbruck, 2003.

[7] F. Rosebury. *Handbook of Electron Tube and Vacuum Techniques*. American Vacuum Society Classics. American Inst. of Physics, 1992. ISBN: 9781563961212.

[8] Thaned Pruttivarasin et al. "Trapped ions in optical lattices for probing oscillator chain models". In: *New J. Phys.* 13.7 (July 2011), p. 075012. DOI: 10.1088/1367-2630/13/7/075012.

[9] Thaned Pruttivarasin. "Spectroscopy, fundamental symmetry tests and quantum simulation with trapped ions". PhD thesis. University of California, Berkeley, 2014.

[10] Jan Benhelm. "Precision spectroscopy and quantum information processsing with trapped calcium ions". PhD thesis. University of Innsbruck, 2008.

[11] Michael Ramm et al. "Precision Measurement Method for Branching Fractions of Excited $P_{1/2}$ States Applied to $^{40}Ca^{+}$." In: *Phys. Rev. Lett.* 111 (July 2013), p. 023004. DOI: 10.1103/PhysRevLett.111.023004.

[12] D.F.V. James. "Quantum dynamics of cold trapped ions with application to quantum computation". In: *Applied Physics B* 66.2 (1998), pp. 181–190. DOI: 10.1007/s003400050373.

[13] Markus Ansmann. "Benchmarking the Superconducting Josephson Phase Qubit: The Violation of Bell's Inequality". PhD thesis. University of California, Santa Barbara, 2009.

[14] D. Porras and J. I. Cirac. "Bose-Einstein Condensation and Strong-Correlation Behavior of Phonons in Ion Traps". In: *Phys. Rev. Lett.* 93 (26 Dec. 2004), p. 263602. DOI: 10.1103/PhysRevLett.93.263602.

[15] G-D Lin and L-M Duan. "Equilibration and temperature distribution in a driven ion chain". In: *New J. Phys.* 13.7 (July 2011), p. 075015. DOI: 10.1088/1367-2630/13/7/075015.

[16] A. Bermudez, M. Bruderer, and M. B. Plenio. "Controlling and Measuring Quantum Transport of Heat in Trapped-Ion Crystals". In: *Phys. Rev. Lett.* 111 (4 July 2013), p. 040601. DOI: 10.1103/PhysRevLett.111.040601.

[17] A. Benassi, A. Vanossi, and E. Tosatti. "Nanofriction in cold ion traps." In: *Nat. Commun.* 2 (Jan. 2011), p. 236. DOI: 10.1038/ncomms1230.

[18] E Fermi, J Pasta, and S Ulam. "Studies of nonlinear problems". In: *Los Alamos Report LA-1940* (1955).

[19] G. P. Berman and F. M. Izrailev. "The Fermi-Pasta-Ulam problem: Fifty years of progress". In: *Chaos* 15.1, 015104 (2005). DOI: 10.1063/1.1855036.

[20] D. Porras and J. I. Cirac. "Effective Quantum Spin Systems with Trapped Ions". In: *Phys. Rev. Lett.* 92 (20 May 2004), p. 207901. DOI: 10.1103/PhysRevLett.92.207901.

[21] A. Friedenauer et al. "Simulating a quantum magnet with trapped ions". In: *Nat Phys* 4.10 (Oct. 2008), pp. 757–761. DOI: 10.1038/nphys1032.

[22] Ch Schneider, Diego Porras, and Tobias Schaetz. "Experimental quantum simulations of many-body physics with trapped ions". In: *Reports on Progress in Physics* 75.2 (2012), p. 024401.

[23] R Islam et al. "Onset of a quantum phase transition with a trapped ion quantum simulator." In: *Nat. Commun.* 2 (2011), p. 377. DOI: 10.1038/ncomms1374.

[24] R. Blatt and C. F. Roos. "Quantum simulations with trapped ions". In: *Nat. Phys.* 8 (2012), pp. 277–284. DOI: 10.1038/nphys2252.

[25] Nahuel Freitas, Esteban Martinez, and Juan Pablo Paz. *Heat transport through ion crystals*. 2013. eprint: arXiv:1312.6644.

[26] Antonia Ruiz et al. *Tuning heat transport in trapped-ion chains across a structural phase transition*. 2014. eprint: arXiv:1401.5480.

[27] Bao-quan Ai, Dahai He, and Bambi Hu. "Heat conduction in driven Frenkel-Kontorova lattices: Thermal pumping and resonance". In: *Phys. Rev. E* 81 (3 Mar. 2010), p. 031124. DOI: 10.1103/PhysRevE.81.031124.

[28] Michael Ramm, Thaned Pruttivarasin, and Hartmut Häffner. *Energy Transport in Trapped Ion Chains*. 2013. eprint: arXiv:1312.5786.

[29] H Landa et al. "Structure, dynamics and bifurcations of discrete solitons in trapped ion crystals". In: *New Journal of Physics* 15.9 (2013), p. 093003.

[30] P. A. Ivanov et al. "Simulation of a quantum phase transition of polaritons with trapped ions". In: *Phys. Rev. A* 80.6 (Dec. 2009), p. 060301. DOI: 10.1103/PhysRevA.80.060301.

[31] X.-L. Deng, D. Porras, and J. I. Cirac. "Quantum phases of interacting phonons in ion traps". In: *Phys. Rev. A* 77 (3 Mar. 2008), p. 033403. DOI: 10.1103/PhysRevA.77.033403.

[32] Kevin Sheridan et al. "All-optical broadband excitation of the motional state of trapped ions". In: *Eur. Phys. J. D* 66.11 (Nov. 2012), p. 289. DOI: 10.1140/epjd/e2012-30377-8.

[33] Y.-W. Lin, S Williams, and B C Odom. "Resonant few-photon excitation of a single-ion oscillator". In: *Phys. Rev. A* 87.1 (Jan. 2013), p. 011402. DOI: 10.1103/PhysRevA.87.011402.

[34] Frank Ziesel et al. "Experimental creation and analysis of displaced number states". In: *J. Phys. B: At. Mol. Opt. Phys.* 46.10 (May 2013), p. 104008. DOI: 10.1088/0953-4075/46/10/104008.

[35] Hiroki Saito and Hiroyuki Hyuga. "Relaxation of Schrödinger Cat States and Displaced Thermal States in a Density Operator Representation". In: *J. Phys. Soc. Jpn.* 65.6 (1996), pp. 1648–1654. DOI: 10.1143/JPSJ.65.1648.

[36] M. Gessner et al. *Nonlinear Spectroscopy of Many-Body Quantum Systems with Single Site Addressability*. 2013. eprint: arXiv:1312.3365.

[37] Manuel Gessner and Heinz-Peter Breuer. "Detecting Nonclassical System-Environment Correlations by Local Operations". In: *Phys. Rev. Lett.* 107 (18 Oct. 2011), p. 180402. DOI: 10.1103/PhysRevLett.107.180402.

[38] Manuel Gessner and Heinz-Peter Breuer. "Local witness for bipartite quantum discord". In: *Phys. Rev. A* 87 (4 Apr. 2013), p. 042107. DOI: 10.1103/PhysRevA.87.042107.

[39] M. Gessner et al. "Local detection of quantum correlations with a single trapped ion". In: *Nat. Phys.* 10.2 (Dec. 2013), pp. 105–109. DOI: 10.1038/nphys2829.

[40] H.P. Breuer and F. Petruccione. *The Theory of Open Quantum Systems*. OUP Oxford, 2007. ISBN: 9780199213900.

[41] H. Häffner, C.F. Roos, and R. Blatt. "Quantum computing with trapped ions". In: *Physics Reports* 469.4 (2008), pp. 155–203. DOI: 10.1016/j.physrep.2008.09.003.

[42] D. F. James and J. Jerke. "Effective Hamiltonian theory and its applications in quantum information". In: *Can. J. Phys.* 85.6 (2007), pp. 625–632. DOI: 10.1139/P07-060.

[43] Christopher J. Foot. *Atomic Physics (Oxford Master Series in Atomic, Optical and Laser Physics)*. 1st ed. Oxford University Press, USA, Feb. 10, 2005. ISBN: 0198506961.

[44] Heinz-Peter Breuer, Elsi-Mari Laine, and Jyrki Piilo. "Measure for the Degree of Non-Markovian Behavior of Quantum Processes in Open Systems". In: *Phys. Rev. Lett.* 103 (21 Nov. 2009), p. 210401. DOI: 10.1103/PhysRevLett.103.210401.

[45] Ángel Rivas, Susana F. Huelga, and Martin B. Plenio. "Entanglement and Non-Markovianity of Quantum Evolutions". In: *Phys. Rev. Lett.* 105 (5 July 2010), p. 050403. DOI: 10.1103/PhysRevLett.105.050403.

[46] E.-M. Laine, J. Piilo, and H.-P. Breuer. "Witness for initial system-environment correlations in open-system dynamics". In: *EPL* 92.6 (2010), p. 60010.

[47] Tobias J. Osborne and Michael A. Nielsen. "Entanglement in a simple quantum phase transition". In: *Phys. Rev. A* 66 (3 Sept. 2002), p. 032110. DOI: 10.1103/PhysRevA.66.032110.

[48] R. Islam et al. "Emergence and Frustration of Magnetism with Variable-Range Interactions in a Quantum Simulator". In: *Science* 340.6132 (2013), pp. 583–587. DOI: 10.1126/science.1232296.

[49] J.R. Johansson, P.D. Nation, and Franco Nori. "QuTiP 2: A Python framework for the dynamics of open quantum systems". In: *Computer Physics Communications* 184.4 (2013), pp. 1234–1240. DOI: 10.1016/j.cpc.2012.11.019.

[50] Manuel Gessner et al. *Observing a Quantum Phase Transition by Measuring a Single Spin*. 2014. eprint: arXiv:1403.4066.

[51] Kenji Toyoda et al. "Experimental Realization of a Quantum Phase Transition of Polaritonic Excitations". In: *Phys. Rev. Lett.* 111 (16 Oct. 2013), p. 160501. DOI: 10.1103/PhysRevLett.111.160501.

[52] Martin Sepiol. "Frequency stabilization of a 729 nm diode laser to an external high finesse reference cavity". MA thesis. ETH Zurich, 2012.

[53] G. Tommaseo et al. "The g_J-factor in the ground state of Ca^+". In: *The European Physical Journal D - Atomic, Molecular, Optical and Plasma Physics* 25.2 (2003), pp. 113–121. DOI: 10.1140/epjd/e2003-00096-6.

[54] M. Chwalla et al. "Absolute Frequency Measurement of the $^{40}Ca^+$ $4s^2S_{1/2} - 3d^2D_{5/2}$ Clock Transition". In: *Phys. Rev. Lett.* 102 (2 Jan. 2009), p. 023002. DOI: 10.1103/PhysRevLett.102.023002.

[55] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information (Cambridge Series on Information and the Natural Sciences)*. 1st ed. Cambridge University Press, Jan. 1, 2004. ISBN: 0521635039.

[56] Mark E. Tuckerman. *Statistical Mechanics: Theory and Molecular Simulation (Oxford Graduate Texts)*. Oxford University Press, USA, Apr. 19, 2010. ISBN: 0198525265.

[57] Claude Cohen-Tannoudji, Jacques Dupont-Roc, and Gilbert Grynberg. *Atom - Photon Interactions: Basic Process and Appilcations*. Wiley-VCH, 2008. ISBN: 9783527617197.

[58] Ali H Nayfeh and Dean T Mook. *Nonlinear Oscillations*. Wiley, 2008.

Appendix A

Detailed Experimental Control

A.1 LabRAD Structure

The LabRAD manager is the centerpiece program in controlling the experiments. It keeps track of and manages connections to all of the connected programs. The manager also contains the Registry - a simple database that is used to store permanent information about the control. Individual devices are controlled by LabRAD servers that may also perform more abstract tasks. The Data Vault server, for example, is responsible for saving collected experimental data to disk, while the Node server is able to launch all of the needed servers as subprocesses. The servers are written with an asynchronous programming framework (e.g twisted library in Python), allowing them to simultaneously respond to multiple connections.

In addition to servers, there are also clients. Clients come in two types: synchronous clients are just scripts that issue a sequence of commands to the servers. Asynchronous clients are used for GUI applications. Similar to scripts, they issue commands to the servers when the user interacts with their interface, but are also able to receive signals from the servers when an external update has been made. The signaling allows the GUIs to remain up to date with the server's information when the updates are issued externally.

The structure of all the servers used in our experiment is shown on Figure A.1. They span three separate computers: Laser Room Windows machine controlling laser cavities and the wavemeter, Imaging Windows machine communicating with the Andor CCD camera and GPIB devices, and the Linux machine running the majority of the servers. The Laser Room machine runs its own LabRAD Manager allowing access to other experiments in the laboratory even when the Linux and the Imaging Machines are switched off. All of the experiments access the machine using its static IP address on the network. The Laser Room machine communicates with the Wavemeter (HighFinesse WS7 Super Precision) using the API provided with the installation drivers. The multiplexer, switching the lasers measured by the wavemeter, and the DAC, setting the voltages of the reference laser cavities, are controlled by the corresponding servers via the serial interface managed with the Serial Server.

The Imaging Windows machine is responsible for all of the devices requiring the Windows operating system: the Andor CCD Camera, and all of the GPIB devices. Specifically it runs the Andor Server, which communicates to the Andor Luca camera via the Andor API as well as the GPIB bus that interfaces with all of the GPIB devices on the experiment.

The LabRAD manager and the bulk of the servers run on the Linux machine. In terms of hardware it uses the DAC and the Pulser servers to communicate with the corresponding FPGAs via the Opal Kelly API. The Pulser server is used to control DDS and TTL channels of the FPGA and to synthesize sequences of laser pulses, as described in the subsection A.1.4. The ADC server controls the analog-to-digital converter using the Serial Server interface. The servers for the controlled GPIB devices also run on the Linux machine. These communicate to the devices through the GPIB Device Manager.

There is a number of servers that perform abstract tasks: Data Vault stores all of the experimental data, Fitter allows to fit the collected data to predefined functions. The Normal PMT Flow server collects the PMT data from the Pulser and directs it to Data Vault. Finally the electrode diagonalization server compensates for the physical asymmetries in the trap electrodes, and the crystallizer assists with keeping ion chains from melting. Drift Tracker keeps track of the drifts of the high finesse laser cavity and the magnetic field, as described in section A.1.3. The Parameter Vault and ScriptScanner servers make up the ScriptScanner framework for scheduling, pausing, and stopping experiments, which will also be explained later in section A.1.2. Every machine runs a Node Server which handles starting and stopping all the other servers running on that computer.

In the following subsections, we provide a brief description of selected servers and clients with the emphasis on new describing the new LabRAD developments.

A.1.1 Camera Server

The Camera Server interfaces with the Andor Luca CCD. This server is unique in several respects: the communication is handled via the provided camera DLL loaded into memory with the Python ctypes library. The server also provides its own GUI window for displaying the collected images live. This is done to alleviate the burden of transmitting the collected stream of images via TCP. The camera server also provides methods for to request single and kinetic series of images, change triggering type, and specify the region of interest for image readout. All of the image processing required to identify the ions is done in the script requesting the images.

A.1.2 ScriptScanner Framework

Within our use of the LabRAD framework, the individual experiments are performed using Python scripts. Early on, we simply launched the individual scripts from a console window, however, this approach was limiting. First, there is no way of gracefully pausing and then resuming an executing script. There is no way to accommodate scheduling: over the course of complex measurement a particular calibration script may need to be execute every few

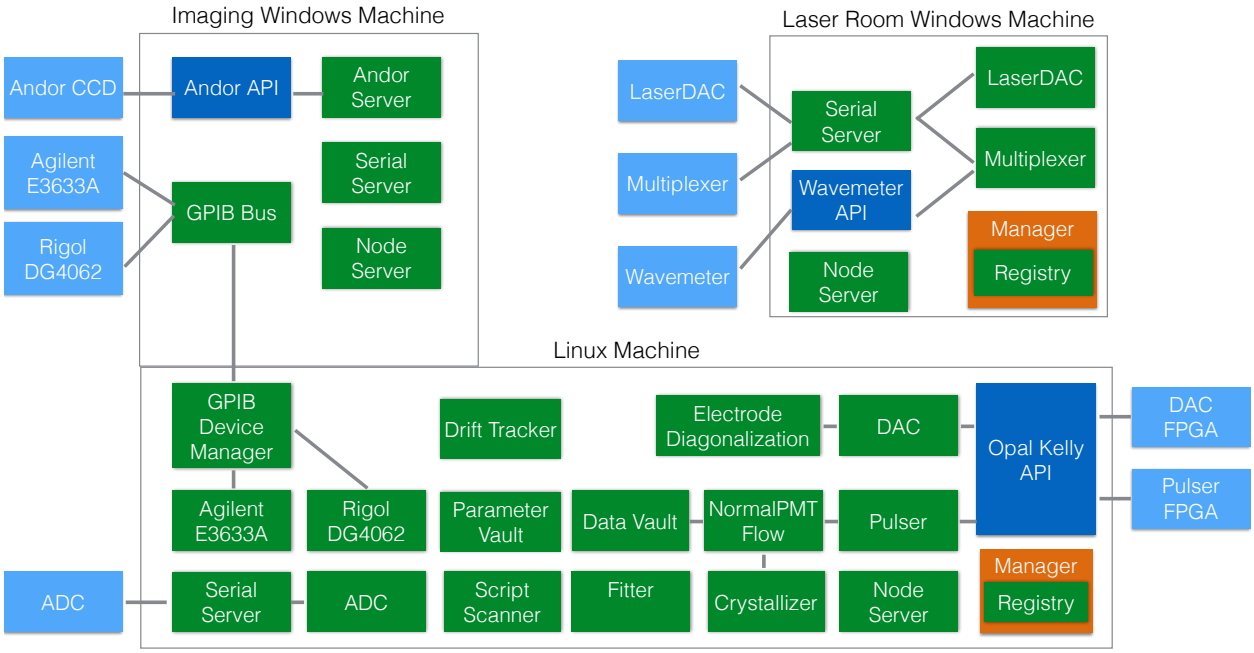

Figure A.1: The structure of LabRAD servers making up the experimental control spanning three computers. The servers are presented in green boxes, API protocols are in dark blue while the physical hardware is in light blue and the LabRAD managers are in orange.

minutes. The calibration script would then need to pause the measurement, complete calibration, and then resume the conducted measurement. Finally, each running script has to look up a multitude of experimental parameters. These should be stored in a single place, so that they are easily shared among multiple conducted experiments. In order to accommodate all of these requirements, we have implemented the ScriptScanner Framework.

The ScriptScanner Framework consists of two servers: Parameter Vault which stores the experimental parameters, and ScriptScanner which manages the launch and schedule of experiment execution. The ScriptScanner allows the running scripts to be paused and resumed. We illustrate their functionality by considering the frequently utilized Ramsey drift tracker experiment. The theory behind the experiment is explained in sections B.5 and A.1.3, so here we focus on the Python code. Below is the code for the top-level experiment:

```
from common.abstractdevices.script_scanner.scan_methods import experiment
from drift_tracker_ramsey_oneline import drift_tracker_ramsey_oneline
from labrad.units import WithUnit
from treedict import TreeDict
import numpy as np

class drift_tracker_ramsey(experiment):

    name = 'DriftTrackerRamsey'
    dt_required_parameters = [
                        ('DriftTracker','line_selection_1'),
```

```
                        ('DriftTracker','line_selection_2'),
                        ('DriftTrackerRamsey','line_1_pi_time'),
                        ('DriftTrackerRamsey','line_1_amplitude'),
                        ('DriftTrackerRamsey','line_2_pi_time'),
                        ('DriftTrackerRamsey','line_2_amplitude'),
                        ('DriftTrackerRamsey','error_sensitivity'),
                        ]

    @classmethod
    def all_required_parameters(cls):
        parameters = set(cls.dt_required_parameters)
        parameters = parameters.union(set(drift_tracker_ramsey_oneline.
            all_required_parameters()))
        parameters = list(parameters)
        #removing parameters we'll be overwriting, and they do not need to be loaded
        parameters.remove(('DriftTrackerRamsey','line_selection'))
        parameters.remove(('DriftTrackerRamsey','pi_time'))
        parameters.remove(('DriftTrackerRamsey','amplitude'))
        parameters.remove(('DriftTrackerRamsey','detuning'))
        return parameters

    def initialize(self, cxn, context, ident):
        self.ident = ident
        self.drift_tracker = cxn.sd_tracker
        self.ramsey_dt = self.make_experiment(drift_tracker_ramsey_oneline)
        self.ramsey_dt.initialize(cxn, context, ident)

    def run(self, cxn, context):
        dt = self.parameters.DriftTracker
        ramsey_dt = self.parameters.DriftTrackerRamsey
        if dt.line_selection_1 == dt.line_selection_2:
            raise Exception ("The_two_Drift_Tracking_lines_can_not_be_the_same")
        replace_1 = TreeDict.fromdict({
                                       'DriftTrackerRamsey.line_selection':dt.
                                           line_selection_1,
                                       'DriftTrackerRamsey.pi_time':ramsey_dt.line_1_pi_time
                                           ,
                                       'DriftTrackerRamsey.amplitude':ramsey_dt.
                                           line_1_amplitude,
                                       'DriftTrackerRamsey.detuning':WithUnit(0,'Hz'),
                                       })
        replace_2 = TreeDict.fromdict({
                                       'DriftTrackerRamsey.line_selection':dt.
                                           line_selection_2,
                                       'DriftTrackerRamsey.pi_time':ramsey_dt.line_2_pi_time
                                           ,
                                       'DriftTrackerRamsey.amplitude':ramsey_dt.
                                           line_2_amplitude,
                                       'DriftTrackerRamsey.detuning':WithUnit(0,'Hz')
                                       })

        replace_1,replace_2 = np.random.permutation([replace_1,replace_2])
        self.ramsey_dt.set_parameters(replace_1)
        self.ramsey_dt.set_progress_limits(0, 50.0)
        frequency_1,excitation = self.ramsey_dt.run(cxn, context)
        error_sensitivity = ramsey_dt.error_sensitivity
        if not 0.5 - error_sensitivity <= excitation <= 0.5 + error_sensitivity:
            raise Exception("Incorrect_Excitation_{}".format(replace_1.DriftTrackerRamsey.
                line_selection))
        self.ramsey_dt.set_parameters(replace_2)
        self.ramsey_dt.set_progress_limits(50.0, 100.0)
        frequency_2,excitation = self.ramsey_dt.run(cxn, context)
        if not 0.5 - error_sensitivity <= excitation <= 0.5 + error_sensitivity:
```

```
            raise Exception("Incorrect Excitation {}".format(replace_2.DriftTrackerRamsey.
                line_selection))
        self.submit_centers(replace_1,frequency_1,replace_2,frequency_2)

    def submit_centers(self, replace_1, center1, replace_2, center2):
        if center1 is not None and center2 is not None:
            submission = [
                        (replace_1.DriftTrackerRamsey.line_selection, center1),
                        (replace_2.DriftTrackerRamsey.line_selection, center2),
                        ]
            self.drift_tracker.set_measurements(submission)

    def finalize(self, cxn, context):
        self.ramsey_dt.finalize(cxn, context)

if __name__ == '__main__':
    import labrad
    cxn = labrad.connect()
    scanner = cxn.scriptscanner
    exprt = drift_tracker_ramsey(cxn = cxn)
    ident = scanner.register_external_launch(exprt.name)
    exprt.execute(ident)
```

The first thing to observe is that the **drift_tracker_ramsey** class inherits from the **experiment** class, which is imported from the ScriptScanner **scan_methods** file. While inhering from the parent class, we need to subclass several methods needed by all experiments: **def all_required_parameters**, **def initialize**, **def run** and **def finalize**. The **def all_required_parameters** method returns a list of all the parameters required by the experiment. Here, this consists of the constants specified in **dt_required_parameters** list in addition to all of the required methods needed by the **drift_track_ramsey_oneline** subexperiment. Notice that some of the subexperiment's required parameters are excluded from the list as these will be explicitly overwritten in the course of executing the experiment. The list of parameters returned by **def all_required_parameters** will be automatically looked up from the ParameterVault server when the experiment is executed. Once looked up, these parameters will be available in the **self.parameters** tree dictionary.

The execution of the experiment can be performed in two ways. It is possibly to run the Python code directly with the code below **if __name__ == ´__main__':**. In this case, the experiment will be executed in the local shell but the running experiment will be registered with the ScriptScanner server. While this method of launching experiments is better for debugging the code, it does not allow ScriptScanner to start the experiment on its own (while scheduling it to run every few minutes, for instance). Thus, the preferred way to run the experiment is to import it into the list of ScriptScanner available experiments. The imported experiment will execute in a separate thread of the ScriptScanner server and could be launched through the ScriptScanner GUI or by accessing one of the ScriptScanner settings. However the experiment is launched, its execution will look up the required parameters, and then call the **def initialize**, **def run**, and **def finalize** methods in that order.

The idea of this experiment is that we execute the subexperiment **drift_track_ramsey_oneline** twice, once for each laser transition. Then the results are submitted to the Drift Tracker server detailed in section A.1.3. To replace the parameters, we use the **self.ramsey_dt.set_parameters**

method with the argument of a tree dictionary containing the replacement parameters. Each time we also set the progress limits, so that they are correctly reported to the ScriptScanner server by the subexperiment. The source code is below:

```
from common.abstractdevices.script_scanner.scan_methods import experiment
from lattice.scripts.scriptLibrary.common_methods_729 import common_methods_729 as cm
from excitations import excitation_ramsey
from treedict import TreeDict
from labrad.units import WithUnit
from numpy import arcsin, pi
import time

class drift_tracker_ramsey_oneline(experiment):

    name = 'DriftTrackerRamseyOneLine'
    dt_required_parameters = [
                            ('DriftTrackerRamsey','line_selection'),
                            ('DriftTrackerRamsey','gap_time'),
                            ('DriftTrackerRamsey','pi_time'),
                            ('DriftTrackerRamsey','amplitude'),
                            ('DriftTrackerRamsey','detuning'),
                            ('DriftTrackerRamsey','readouts'),
                            ('DriftTrackerRamsey','optical_pumping_enable_DT'),

                            ('StateReadout','camera_primary_ion'),
                            ('StateReadout','use_camera_for_readout'),
                            ]


    @classmethod
    def all_required_parameters(cls):
        parameters = set(cls.dt_required_parameters)
        parameters = parameters.union(set(excitation_ramsey.all_required_parameters()))
        parameters = list(parameters)
        #removing parameters we'll be overwriting, and they do not need to be loaded
        parameters.remove(('Ramsey','ramsey_time'))
        parameters.remove(('Ramsey','second_pulse_phase'))
        parameters.remove(('Excitation_729','rabi_excitation_amplitude'))
        parameters.remove(('Excitation_729','rabi_excitation_frequency'))
        parameters.remove(('Tomography','iteration'))
        parameters.remove(('Tomography','rabi_pi_time'))
        parameters.remove(('Tomography','tomography_excitation_amplitude'))
        parameters.remove(('Tomography','tomography_excitation_frequency'))
        parameters.remove(('TrapFrequencies','axial_frequency'))
        parameters.remove(('TrapFrequencies','radial_frequency_1')),
        parameters.remove(('TrapFrequencies','radial_frequency_2')),
        parameters.remove(('TrapFrequencies','rf_drive_frequency')),
        #will be disabling sideband cooling automatically
        parameters.remove(('SidebandCooling','sideband_cooling_enable')),
        parameters.remove(('SidebandCooling','frequency_selection')),
        parameters.remove(('SidebandCooling','manual_frequency_729')),
        parameters.remove(('SidebandCooling','line_selection')),
        parameters.remove(('SidebandCooling','sideband_selection')),
        parameters.remove(('SidebandCooling','sideband_cooling_type')),
        parameters.remove(('SidebandCooling','sideband_cooling_cycles')),
        parameters.remove(('SidebandCooling','
            sideband_cooling_duration_729_increment_per_cycle')),
        parameters.remove(('SidebandCooling','sideband_cooling_frequency_854')),
        parameters.remove(('SidebandCooling','sideband_cooling_amplitude_854')),
        parameters.remove(('SidebandCooling','sideband_cooling_frequency_866')),
        parameters.remove(('SidebandCooling','sideband_cooling_amplitude_866')),
        parameters.remove(('SidebandCooling','sideband_cooling_amplitude_729')),
        parameters.remove(('SidebandCooling','sideband_cooling_optical_pumping_duration')),
```

```
        parameters.remove(('SidebandCoolingContinuous','sideband_cooling_continuous_duration
            ')),
        parameters.remove(('SidebandCoolingPulsed','sideband_cooling_pulsed_duration_729')),
        parameters.remove(('SidebandCoolingPulsed','sideband_cooling_pulsed_cycles')),
        parameters.remove(('SidebandCoolingPulsed','sideband_cooling_pulsed_duration_repumps
            ')),
        parameters.remove(('SidebandCoolingPulsed','
            sideband_cooling_pulsed_duration_additional_866')),
        parameters.remove(('SidebandCoolingPulsed','
            sideband_cooling_pulsed_duration_between_pulses')),
        #will be enable optical pumping automatically
        parameters.remove(('OpticalPumping', 'optical_pumping_enable'))
        return parameters

    def initialize(self, cxn, context, ident):
        self.ident = ident
        self.drift_tracker = cxn.sd_tracker
        self.excitation = self.make_experiment(excitation_ramsey)
        self.excitation.initialize(cxn, context, ident)
        self.phases = [WithUnit(90.0, 'deg'), WithUnit(-90.0, 'deg')]
        self.dv = cxn.data_vault

    def setup_data_vault(self):
        line_name = self.parameters.DriftTrackerRamsey.line_selection
        #navigate to the directory
        localtime = time.localtime()
        dirappend = [ time.strftime("%Y%b%d",localtime)]
        directory = ['','Experiments']
        directory.extend([self.name])
        directory.extend(dirappend)
        self.dv.cd(directory ,True)
        #try opening the existing dataset
        datasetname = 'RameyDriftTrack_{}'.format(line_name)
        datasets_in_folder = self.dv.dir()[1]
        names = sorted([name for name in datasets_in_folder if datasetname in name])
        if names:
            #dataset with that name exist
            self.dv.open_appendable(names[0])
        else:
            #dataset doesn't already exist
            self.dv.new(datasetname,[('Time', 'Sec')],[('Excitation','Average','percent'),('
                Excitation','Deviation','percent')])
            window_name = ['Ramey_Drift_Track_{0}'.format(self.parameters.DriftTrackerRamsey
                .line_selection)]
            self.dv.add_parameter('Window', window_name)
            self.dv.add_parameter('plotLive', True)

    def run(self, cxn, context):
        self.setup_data_vault()
        dt = self.parameters.DriftTrackerRamsey
        excitations = []
        frequency = cm.frequency_from_line_selection('auto', None , dt.line_selection, self.
            drift_tracker)
        frequency = frequency + dt.detuning
        for iter,phase in enumerate(self.phases):
            replace = TreeDict.fromdict({
                                        'Ramsey.first_pulse_duration':dt.pi_time / 2.0,
                                        'Ramsey.second_pulse_duration':dt.pi_time / 2.0,
                                        'Ramsey.ramsey_time':dt.gap_time,
                                        'Ramsey.second_pulse_phase':phase,
                                        'Excitation_729.rabi_excitation_amplitude':dt.
                                            amplitude,
                                        'Excitation_729.rabi_excitation_frequency':
```

```
                                                frequency,
                                            'Tomography.iteration':0.0,
                                            'StateReadout.repeat_each_measurement':dt.
                                                readouts,
                                            'SidebandCooling.sideband_cooling_enable':False,
                                            'OpticalPumping.optical_pumping_enable':dt.
                                                optical_pumping_enable_DT,
                                            })
                self.excitation.set_parameters(replace)
                self.update_progress(iter)
                if not self.parameters.StateReadout.use_camera_for_readout:
                    #using PMT
                    excitation_array, readout = self.excitation.run(cxn, context)
                    excitation = excitation_array[0]
                else:
                    primary_ion = int(self.parameters.StateReadout.camera_primary_ion)
                    excitation_array, readout = self.excitation.run(cxn, context)
                    excitation = excitation_array[primary_ion]
                excitations.append(excitation)
            print excitations
            detuning, average_excitation = self.calculate_detuning(excitations)
            corrected_frequency = frequency + detuning
#            print corrected_frequency, average_excitation
            return corrected_frequency,average_excitation

    def calculate_detuning(self, excitations):
        dt = self.parameters.DriftTrackerRamsey
        if not dt.optical_pumping_enable_DT:
            #if we are not doing optical pumping during drift tracking, then need to double
                the measured excitations
            excitations[0] = excitations[0]*2.0
            excitations[1] = excitations[1]*2.0
        average = (excitations[0] + excitations[1]) / 2.0
        deviation = (excitations[0] - excitations[1])
        detuning = arcsin(deviation) / (2.0 * pi * dt.gap_time['s'])
        detuning = WithUnit(detuning, 'Hz')
        self.dv.add([time.time(), average, deviation])
        return detuning, average

    def update_progress(self, iteration):
        progress = self.min_progress + (self.max_progress - self.min_progress) * float(
            iteration + 1.0) / len(self.phases)
        self.sc.script_set_progress(self.ident,  progress)

    def finalize(self, cxn, context):
        self.excitation.finalize(cxn, context)

if __name__ == '__main__':
    import labrad
    cxn = labrad.connect()
    scanner = cxn.scriptscanner
    exprt = drift_tracker_ramsey_oneline(cxn = cxn)
    ident = scanner.register_external_launch(exprt.name)
    exprt.execute(ident)
```

Again we see that the subexperiment inherits from the same **experiment** class. It can be directly executed just like the top level experiment. In this case, the measured spectral line is looked up directly from the ParameterVault and not provided by the top level experiment. When **drift_track_ramsey_oneline** runs, it will execute another subexperiment called **excitation_ramsey** with two different phases of the second ramsey pulse, see B.5. The subexperiment **excitation_ramsey** will return the probability of the ion being in the

excited state after repeating the Ramsey pulse sequence the number of times specified in StateReadout collection of the ParameterVault. Then these excitations will be used to calculate the detuning from the line. The **update_progress** method reports the completion percentage to the ScriptScanner so that the user is aware of the measurement progress.

The subexperiment is short and so it does not implement the ability to pause or stop: the execution will always continue until completion. If this were needed, we would use the **def pause_or_stop** method implemented by the **experiment** class. When this method is called, and returns **True** then the ScriptScanner requires the experiment to stop execution. The response of **False** means continue execution. Pausing is accomplished by the ScriptScanner delaying to answer the query until the experiment is unpaused. Whenever the experiment is paused, stopped, resumed, or finished, it will update the ScriptScanner of the latest status.

For completeness, we include the code for the **excitation_ramsey** experiment, which executes the **base_excitation** experiment measuring a single excitation percentage of the **ramsey** pulse sequence. The **base_excitation** allows to perform state readout with either CCD or the PMT. For a tutorial on pulse sequence writing, see section A.1.4

```
from base_excitation import base_excitation

class excitation_ramsey(base_excitation):
    from lattice.scripts.PulseSequences.ramsey import ramsey
    name = 'ExcitationRamsey'
    pulse_sequence = ramsey
```

```
from common.abstractdevices.script_scanner.scan_methods import experiment
from lattice.scripts.scriptLibrary.common_methods_729 import common_methods_729 as cm
from lattice.scripts.experiments.Camera.ion_state_detector import ion_state_detector
from labrad.units import WithUnit
import numpy
import time

class base_excitation(experiment):
    name = ''
    excitation_required_parameters = [
                            ('OpticalPumping','frequency_selection'),
                            ('OpticalPumping','manual_frequency_729'),
                            ('OpticalPumping','line_selection'),

                            ('OpticalPumpingAux','aux_op_line_selection'),
                            ('OpticalPumpingAux','aux_op_enable'),

                            ('SidebandCooling','frequency_selection'),
                            ('SidebandCooling','manual_frequency_729'),
                            ('SidebandCooling','line_selection'),
                            ('SidebandCooling','sideband_selection'),
                            ('TrapFrequencies','axial_frequency'),
                            ('TrapFrequencies','radial_frequency_1'),
                            ('TrapFrequencies','radial_frequency_2'),
                            ('TrapFrequencies','rf_drive_frequency'),

                            ('StateReadout', 'repeat_each_measurement'),
                            ('StateReadout', 'state_readout_threshold'),

                            ('StateReadout', 'use_camera_for_readout'),
                            ('StateReadout', 'state_readout_duration'),

                            ('IonsOnCamera','ion_number'),
```

```
                            ('IonsOnCamera','vertical_min'),
                            ('IonsOnCamera','vertical_max'),
                            ('IonsOnCamera','vertical_bin'),
                            ('IonsOnCamera','horizontal_min'),
                            ('IonsOnCamera','horizontal_max'),
                            ('IonsOnCamera','horizontal_bin'),

                            ('IonsOnCamera','fit_amplitude'),
                            ('IonsOnCamera','fit_background_level'),
                            ('IonsOnCamera','fit_center_horizontal'),
                            ('IonsOnCamera','fit_center_vertical'),
                            ('IonsOnCamera','fit_rotation_angle'),
                            ('IonsOnCamera','fit_sigma'),
                            ('IonsOnCamera','fit_spacing'),
                           ]
    pulse_sequence = None

    @classmethod
    def all_required_parameters(cls):
        params = set(cls.excitation_required_parameters)
        params = params.union(set(cls.pulse_sequence.all_required_parameters()))
        params = list(params)
        params.remove(('OpticalPumping', 'optical_pumping_frequency_729'))
        params.remove(('SidebandCooling', 'sideband_cooling_frequency_729'))
        params.remove(('OpticalPumpingAux', 'aux_optical_frequency_729'))
        return params

    def initialize(self, cxn, context, ident):
        self.pulser = cxn.pulser
        self.drift_tracker = cxn.sd_tracker
        self.dv = cxn.data_vault
        self.total_readouts = []
        self.readout_save_context = cxn.context()
        self.histogram_save_context = cxn.context()
        self.readout_save_iteration = 0
        self.setup_sequence_parameters()
        self.setup_initial_switches()
        self.setup_data_vault()
        self.use_camera = self.parameters.StateReadout.use_camera_for_readout
        if self.use_camera:
            self.initialize_camera(cxn)

    def initialize_camera(self, cxn):
        self.total_camera_confidences = []
        p = self.parameters.IonsOnCamera
        from lmfit import Parameters as lmfit_Parameters
        self.camera = cxn.andor_server
        self.fitter = ion_state_detector(int(p.ion_number))
        self.camera_initially_live_display = self.camera.is_live_display_running()
        self.camera.abort_acquisition()
        self.initial_exposure = self.camera.get_exposure_time()
        exposure = self.parameters.StateReadout.state_readout_duration
        self.camera.set_exposure_time(exposure)
        self.initial_region = self.camera.get_image_region()
        self.image_region = [
                             int(p.horizontal_bin),
                             int(p.vertical_bin),
                             int(p.horizontal_min),
                             int(p.horizontal_max),
                             int(p.vertical_min),
                             int(p.vertical_max),
                             ]
        self.fit_parameters = lmfit_Parameters()
```

```
        self.fit_parameters.add('ion_number', value = int(p.ion_number))
        self.fit_parameters.add('background_level', value = p.fit_background_level)
        self.fit_parameters.add('amplitude', value = p.fit_amplitude)
        self.fit_parameters.add('rotation_angle', p.fit_rotation_angle)
        self.fit_parameters.add('center_x', value = p.fit_center_horizontal)
        self.fit_parameters.add('center_y', value = p.fit_center_vertical)
        self.fit_parameters.add('spacing', value = p.fit_spacing)
        self.fit_parameters.add('sigma', value = p.fit_sigma)
        x_axis = numpy.arange(self.image_region[2], self.image_region[3] + 1, self.
            image_region[0])
        y_axis = numpy.arange(self.image_region[4], self.image_region[5] + 1, self.
            image_region[1])
        xx,yy = numpy.meshgrid(x_axis, y_axis)
        self.fitter.set_fitted_parameters(self.fit_parameters, xx, yy)
        self.camera.set_image_region(*self.image_region)
        self.camera.set_acquisition_mode('Kinetics')
        self.initial_trigger_mode = self.camera.get_trigger_mode()
        self.camera.set_trigger_mode('External')

    def setup_data_vault(self):
        localtime = time.localtime()
        self.datasetNameAppend = time.strftime("%Y%b%d_%H%M_%S",localtime)
        dirappend = [ time.strftime("%Y%b%d",localtime) ,time.strftime("%H%M_%S", localtime)
            ]
        self.save_directory = ['','Experiments']
        self.save_directory.extend([self.name])
        self.save_directory.extend(dirappend)
        self.dv.cd(self.save_directory ,True, context = self.readout_save_context)
        self.dv.new('Readout_{}'.format(self.datasetNameAppend),[('Iteration', 'Arb')],[('
            Readout_Counts','Arb','Arb')], context = self.readout_save_context)

    def setup_sequence_parameters(self):
        op = self.parameters.OpticalPumping
        optical_pumping_frequency = cm.frequency_from_line_selection(op.frequency_selection,
             op.manual_frequency_729, op.line_selection, self.drift_tracker, op.
            optical_pumping_enable)
        self.parameters['OpticalPumping.optical_pumping_frequency_729'] =
            optical_pumping_frequency
        aux = self.parameters.OpticalPumpingAux
        aux_optical_pumping_frequency = cm.frequency_from_line_selection('auto', WithUnit(0,
            'MHz'), aux.aux_op_line_selection, self.drift_tracker, aux.aux_op_enable)
        self.parameters['OpticalPumpingAux.aux_optical_frequency_729'] =
            aux_optical_pumping_frequency
        sc = self.parameters.SidebandCooling
        sideband_cooling_frequency = cm.frequency_from_line_selection(sc.frequency_selection
            , sc.manual_frequency_729, sc.line_selection, self.drift_tracker, sc.
            sideband_cooling_enable)
        if sc.frequency_selection == 'auto':
            trap = self.parameters.TrapFrequencies
            sideband_cooling_frequency = cm.add_sidebands(sideband_cooling_frequency, sc.
                sideband_selection, trap)
        self.parameters['SidebandCooling.sideband_cooling_frequency_729'] =
            sideband_cooling_frequency

    def setup_initial_switches(self):
        self.pulser.switch_manual('crystallization', False)
        #switch off 729 at the beginning
        self.pulser.output('729DP', False)

    def plot_current_sequence(self, cxn):
        from common.okfpgaservers.pulser.pulse_sequences.plot_sequence import
            SequencePlotter
        dds = cxn.pulser.human_readable_dds()
```

```
        ttl = cxn.pulser.human_readable_ttl()
        channels = cxn.pulser.get_channels().asarray
        sp = SequencePlotter(ttl.asarray, dds.aslist, channels)
        sp.makePlot()

    def run(self, cxn, context):
        threshold = int(self.parameters.StateReadout.state_readout_threshold)
        repetitions = int(self.parameters.StateReadout.repeat_each_measurement)
        pulse_sequence = self.pulse_sequence(self.parameters)
        pulse_sequence.programSequence(self.pulser)
#         self.plot_current_sequence(cxn)
        if self.use_camera:
            #print 'starting acquisition'
            self.camera.set_number_kinetics(repetitions)
            self.camera.start_acquisition()
        self.pulser.start_number(repetitions)
        self.pulser.wait_sequence_done()
        self.pulser.stop_sequence()
        if not self.use_camera:
            #get percentage of the excitation using the PMT threshold
            readouts = self.pulser.get_readout_counts().asarray
            self.save_data(readouts)
            if len(readouts):
                perc_excited = numpy.count_nonzero(readouts <= threshold) / float(len(
                    readouts))
            else:
                #got no readouts
                perc_excited = -1.0
            ion_state = [perc_excited]
#             print readouts
        else:
            #get the percentage of excitation using the camera state readout
            proceed = self.camera.wait_for_kinetic()
            if not proceed:
                self.camera.abort_acquisition()

                self.finalize(cxn, context)
                raise Exception ("Did not get all kinetic images from camera")
            images = self.camera.get_acquired_data(repetitions).asarray
            self.camera.abort_acquisition()
            x_pixels = int( (self.image_region[3] - self.image_region[2] + 1.) / (self.
                image_region[0]) )
            y_pixels = int(self.image_region[5] - self.image_region[4] + 1.) / (self.
                image_region[1])
            images = numpy.reshape(images, (repetitions, y_pixels, x_pixels))
            readouts, confidences = self.fitter.state_detection(images)
            ion_state = 1 - readouts.mean(axis = 0)
            #useful for debugging, saving the images
#             numpy.save('readout {}'.format(int(time.time())), images)
            self.save_confidences(confidences)
        return ion_state, readouts

    @property
    def output_size(self):
        if self.use_camera:
            return int(self.parameters.IonsOnCamera.ion_number)
        else:
            return 1

    def finalize(self, cxn, context):
        if self.use_camera:
            #if used the camera, return it to the original settings
            self.camera.set_trigger_mode(self.initial_trigger_mode)
```

```
            self.camera.set_exposure_time(self.initial_exposure)
            self.camera.set_image_region(self.initial_region)
            if self.camera_initially_live_display:
                self.camera.start_live_display()

    def save_data(self, readouts):
        #save the current readouts
        iters = numpy.ones_like(readouts) * self.readout_save_iteration
        self.dv.add(numpy.vstack((iters, readouts)).transpose(), context = self.
            readout_save_context)
        self.readout_save_iteration += 1
        self.total_readouts.extend(readouts)
        if (len(self.total_readouts) >= 500):
            hist, bins = numpy.histogram(self.total_readouts, 50)
            self.dv.cd(self.save_directory ,True, context = self.histogram_save_context)
            self.dv.new('Histogram_{}'.format(self.datasetNameAppend),[('Counts', 'Arb')],[(
                'Occurence','Arb','Arb')], context = self.histogram_save_context )
            self.dv.add(numpy.vstack((bins[0:-1],hist)).transpose(), context = self.
                histogram_save_context )
            self.dv.add_parameter('Histogram729', True, context = self.
                histogram_save_context )
            self.total_readouts = []

    def save_confidences(self, confidences):
        '''
        saves confidences readings for the camera state detection
        '''
        self.total_camera_confidences.extend(confidences)
        if (len(self.total_camera_confidences) >= 300):
            hist, bins = numpy.histogram(self.total_camera_confidences, 30)
            self.dv.cd(self.save_directory ,True, context = self.histogram_save_context)
            self.dv.new('Histogram_Camera_{}'.format(self.datasetNameAppend),[('Counts', '
                Arb')],[('Occurence','Arb','Arb')], context = self.histogram_save_context )
            self.dv.add(numpy.vstack((bins[0:-1],hist)).transpose(), context = self.
                histogram_save_context )
            self.dv.add_parameter('HistogramCameraConfidence', True, context = self.
                histogram_save_context )
            self.total_camera_confidences = []
```

A.1.3 Drift Tracker Server

The narrow transition between the $S_{1/2}$ and $D_{5/2}$ states requires precise control of the 729 nm laser frequency and the local magnetic field at the ion position. While the laser frequency is referenced to a high finesse cavity [52, 9], the cavity length may drift due to thermal expansion and contraction of the ultra low expansion (ULE) material. The stability of the local magnetic field is determined by the fluctuations in the current circulating in the magnetic field coils, thermal effects affecting their position, and external magnetic field fluctuations.

The purpose of the Drift Tracker server is to facilitate tracking drifts of both laser frequency and the local magnetic field. Both quantities are determined by measuring frequencies of two carrier transitions between chosen magnetic sublevels of $S_{1/2}$ and $D_{5/2}$. Once the frequencies of two carrier transitions are measured, typically by using a Ramsey technique described in Appendix B.5, the frequencies of the transitions as well as their names are submitted to Drift Tracking Server:

```
cxn.sd_tracker.set_measurements(
[('S-1/2D+3/2', WithUnit(-16.813, 'MHz')), ('S+1/2D+5/2', WithUnit(-15.68, 'MHz'))]
)
```

The submission may be done through an experimental script or a GUI interface. It is also possible to submit the frequency of only one line, assuming that only the magnetic field has drifted. The server uses the submitted information to calculate the magnetic field B and the zero-field splitting E_0, which measures the drift in the cavity frequency:

$$E_{S\leftrightarrow D} = E_0 + \mu_B \left(m_S g_{S1/2} - m_D g_{D5/2}\right) \tag{A.1}$$

where μ_B is the Bohr magneton, m_S and m_D as the magnetic sublevels of the $S_{1/2}$ and $D_{5/2}$ energy levels. See [10] for more details. While the corresponding Lange g-factors may be computed to be $g_{S1/2} = 2$ and $g_{D5/2} = 1.2$ using the well-known relationship [43],

$$g_J = \frac{3}{2} + \frac{S(S+1) - L(L+1)}{2J(J+1)} \tag{A.2}$$

it is more precise to use the experimentally measured values of $g_{S1/2} = 2.00225664(9)$ [53] and $g_{D5/2} = 1.2003340(3)$ [54], reflecting the necessary relativistic and QED corrections to the formula.

The server performs a linear fit to the magnetic field and cavity center drifts. This allows to extrapolate the trend into the future: whenever a new experiment is run, the server will provide the predicted values for all of the required transition frequencies.

A.1.4 Pulse Sequence Synthesis

Fundamentally, every pulse sequence is a just collection of TTL and DDS pulses. Every pulse has a start time and a duration. The DDS pulses additionally specify, frequency, amplitude, and an optional phase. These pulses are programmed to the Pulser FPGA before execution. The FPGA hardware is described in detail in Thaned Pruttivarasin's thesis [9]. While pulse sequences for manipulating the state of the ion may contain hundreds of individual pulses, there is usually no need to individually specify their timing and parameters. Pulse sequences consist of logical components: typically they involve Doppler cooling, optical pumping, sideband cooling, etc. If these logical components or subsequences are made modular, they become easily reusable in a variety of composite sequences. This minimizes efforts of creating a new complex pulse sequence and greatly reduces the errors in the process. Here we provide a tutorial on our implementation of the pulse sequence synthesis. It is accurate as of **pulse_sequence** version 1.1, and is, of course, subject to future modifications. For the latest description, see the common GitHub Repository Wiki.

To achieve the desired modularity, every sequence inherits from the Python base class **pulse_sequence**. The methods contained in the base class allow the user to easily modify and interact with the pulse sequences. When inheriting from the **pulse_sequence**, the user has to overwrite several constants describing the new sequence: **required_parameters**

lists the parameters required by the new sequence, **required_subsequences** lists the subsequences added to the new sequence, and **replaced_parameters** is a dictionary of which parameters of the added subsequences will be overwritten. Finally, the user has to implement the method **def sequence(self):** where the new sequence is contained. As an illustrating example, let us the review the **spectrum_rabi** pulse sequence used both by Rabi flopping experiments and measurements of the $S_{1/2} - D_{5/2}$ spectra:

```
from common.okfpgaservers.pulser.pulse_sequences.pulse_sequence import pulse_sequence
from subsequences.RepumpDwithDoppler import doppler_cooling_after_repump_d
from subsequences.EmptySequence import empty_sequence
from subsequences.OpticalPumping import optical_pumping
from subsequences.RabiExcitation import rabi_excitation
from subsequences.Tomography import tomography_readout
from subsequences.TurnOffAll import turn_off_all
from subsequences.SidebandCooling import sideband_cooling
from labrad.units import WithUnit
from treedict import TreeDict

class spectrum_rabi(pulse_sequence):

    required_parameters = [
                           ('Heating', 'background_heating_time'),
                           ('OpticalPumping','optical_pumping_enable'),
                           ('SidebandCooling','sideband_cooling_enable'),
                           ]

    required_subsequences = [doppler_cooling_after_repump_d, empty_sequence, optical_pumping
        , rabi_excitation, tomography_readout, turn_off_all, sideband_cooling]

    replaced_parameters = {
                empty_sequence:[('EmptySequence','empty_sequence_duration')]
                       }

    def sequence(self):
        p = self.parameters
        self.end = WithUnit(10, 'us')
        self.addSequence(turn_off_all)
        self.addSequence(doppler_cooling_after_repump_d)
        if p.OpticalPumping.optical_pumping_enable:
            self.addSequence(optical_pumping)
        if p.SidebandCooling.sideband_cooling_enable:
            self.addSequence(sideband_cooling)
        self.addSequence(empty_sequence, TreeDict.fromdict({'EmptySequence.
            empty_sequence_duration':p.Heating.background_heating_time}))
        self.addSequence(rabi_excitation)
        self.addSequence(tomography_readout)
```

One can see that **spectrum_rabi** inherits from the **pulse_sequence** class. It specifies three required parameters that are directly used in the **sequence** method. For example, based on the parameter **p.OpticalPumping.optical_pumping_enable**, a decision is made whether or not to add the optical pumping subsequence. This pulse sequence does not explicitly specify any DDS or TTL pulses, the pulses are added via the appropriate subsequences. The subsequences are added using the **self.addSequence** command. Whenever a subsequence is added, it also has to be included **required_subsequences** list. All of the subsequences are themselves full-fledged pulse sequence as they also inherit from the same **pulse_sequence** base class.

Each pulse sequence has two variables **self.start** and **self.end** that keep track of the timing. The variable **self.start** is used to denote when the pulse sequence begins. All the pulses of that sequence are specified relative to this starting position. The variable **self.end** denotes the current end time of pulse sequence. The default behavior of the **self.addSequence** method is to append the added subsequence. This means that the start value of the added subsequence will be set equal to the end value of top level sequence. When pulses are added in the subsequence, its end value will advance. The end value of the top level sequence will then be updated to match the end value of the subsequence. It is also possible to add the subsequence at an arbitrary time point by using the **position** argument of the **self.addSequence** method. This behavior is illustrated using the simple **turn_off_all** subsequence.

```
from common.okfpgaservers.pulser.pulse_sequences.pulse_sequence import pulse_sequence
from labrad.units import WithUnit

class turn_off_all(pulse_sequence):

    def sequence(self):
        dur = WithUnit(50, 'us')
        for channel in ['729','397','854','866','radial']:
            self.addDDS(channel, self.start, dur, WithUnit(0, 'MHz'), WithUnit(0, 'dBm') )
        self.end = self.start + dur
```

All of the DDS pulses here simultaneously begin at the sequence's **self.start** position. When this subsequence is added to **spectrum_rabi** sequence, **self.start** = 10 μs because that was the end position of **spectrum_rabi** when the subsequence is added. All of the DDS pulses are 50 μs long, hence we increment the end position of **turn_off_all** by the same amount. This is then used to recalculate the new end position of **spectrum_rabi** to make sure other subsequences are added to the correct location.

The pulse sequence also demonstrates the syntax for adding DDS pulses. The arguments are the channel name, start time, duration time, frequency, amplitude, and the pulse phase. If the phase is not specified, it is set to 0 degrees. The special values of frequency of 0 MHz or amplitude of 0 dBm are used to denote the off settings of the channel. There is configuration file called **pulse_sequence_config.py** that contains optional conversions of the frequencies, amplitudes and phases of the added pulses. This is used, for example, to specify laser frequencies as seen by the ion, i.e double of the frequency when the DDS pulse drives an AOM in a double pass configuration. The syntax for the **addTTL** command only need the first three arguments: channel name, start, and duration. One can also apply TTL pulses on internal channels called **TimeResolvedCount** and **ReadoutCount** in order to record timetags of the pulses arriving from the PMT or the total number of the arriving photons, respectively.

The modular nature of the subsequences allows for another way of repurposing the structure of a subsequence. For example, the actual pulse sequences for Doppler cooling and state readouts are the same: both involve switching on 397 nm and 866 nm lasers at specified frequencies and for a specified duration. This is reflected in the way we compose the sequences.

```
from common.okfpgaservers.pulser.pulse_sequences.pulse_sequence import pulse_sequence
```

```
class doppler_cooling(pulse_sequence):


    required_parameters = [
                           ('DopplerCooling','doppler_cooling_frequency_397'),
                           ('DopplerCooling','doppler_cooling_amplitude_397'),
                           ('DopplerCooling','doppler_cooling_frequency_866'),
                           ('DopplerCooling','doppler_cooling_amplitude_866'),
                           ('DopplerCooling','doppler_cooling_duration'),
                           ('DopplerCooling','doppler_cooling_repump_additional')
                           ]

    def sequence(self):
        p = self.parameters.DopplerCooling
        repump_duration = p.doppler_cooling_duration + p.doppler_cooling_repump_additional
        self.addDDS ('397',self.start, p.doppler_cooling_duration, p.
            doppler_cooling_frequency_397, p.doppler_cooling_amplitude_397)
        self.addDDS ('866',self.start, repump_duration, p.doppler_cooling_frequency_866, p.
            doppler_cooling_amplitude_866)
        self.end = self.start + repump_duration

from common.okfpgaservers.pulser.pulse_sequences.pulse_sequence import pulse_sequence
from lattice.scripts.PulseSequences.subsequences.DopplerCooling import doppler_cooling
from treedict import TreeDict

class state_readout(pulse_sequence):
    '''
    Pulse sequence for reading out the state of the ion.
    '''
    required_parameters = [
                           ('StateReadout','state_readout_frequency_397'),
                           ('StateReadout','state_readout_amplitude_397'),
                           ('StateReadout','state_readout_frequency_866'),
                           ('StateReadout','state_readout_amplitude_866'),
                           ('StateReadout','state_readout_duration'),
                           ('StateReadout','use_camera_for_readout'),
                           ('StateReadout','camera_trigger_width'),
                           ('StateReadout','camera_transfer_additional')
                           ]

    required_subsequences = [doppler_cooling]
    replaced_parameters = {
        doppler_cooling:  [('DopplerCooling','doppler_cooling_frequency_397'),
                           ('DopplerCooling','doppler_cooling_amplitude_397'),
                           ('DopplerCooling','doppler_cooling_frequency_866'),
                           ('DopplerCooling','doppler_cooling_amplitude_866'),
                           ('DopplerCooling','doppler_cooling_duration'),
                          ]
                          }

    def sequence(self):
        st = self.parameters.StateReadout
        replace = {
        'DopplerCooling.doppler_cooling_frequency_397':st.state_readout_frequency_397,
        'DopplerCooling.doppler_cooling_amplitude_397':st.state_readout_amplitude_397,
        'DopplerCooling.doppler_cooling_frequency_866':st.state_readout_frequency_866,
        'DopplerCooling.doppler_cooling_amplitude_866':st.state_readout_amplitude_866,
        'DopplerCooling.doppler_cooling_duration':st.state_readout_duration + st.
            camera_transfer_additional,
                }
        self.addSequence(doppler_cooling, TreeDict.fromdict(replace))
        self.addTTL('ReadoutCount', self.start, st.state_readout_duration)
        if st.use_camera_for_readout:
```

```
self.addTTL('camera', self.start, st.camera_trigger_width)
```

Instead of specifying the same pulses in **state_readout** pulse sequences as in **doppler_cooling**, we add **doppler_cooling** as a subsequence while replacing all of the frequencies, amplitudes and durations with those of **state_readout**. We use the **TreeDict** hierarchical container class that allows for attribute-like access to the elements.

When the desired sequence is specified, we can use **all_required_parameters()** method to get a list of all the parameters required by the sequence. The list is used to automatically request the necessary parameters from a database, as described in the ScriptScanner Framework section. A dictionary containing all of the required parameters should be passed to the constructor when initializing the pulse sequence class. Then the pulse sequence can be programmed to Pulser with the **programSequence(self, pulser):** command where **pulser** is a reference to the Pulser server. One can also use the **SequencePlotter** class to make a plot of the programmed pulse sequence to confirm the pulses look as intended.

While we haven't described every pulse sequence used in the experiments, the presented features should be sufficient to comprehend their functionality. All of the pulse sequences used in the experiment can be found on the experimental GitHub Repository.

A.2 GUI Applications

In this section we illustrate how to create graphical user interfaces employing the PyQt Python library with LabRAD. We consider a simple application called **pmtWidget** that provides the user with a graphical access to the PMT functionality.

```
from PyQt4 import QtGui, uic
from twisted.internet.defer import inlineCallbacks
import os

SIGNALID = 874193

class pmtWidget(QtGui.QWidget):
    def __init__(self, reactor, parent=None):
        super(pmtWidget, self).__init__(parent)
        self.reactor = reactor
        basepath =  os.path.dirname(__file__)
        path = os.path.join(basepath,"qtui", "pmtfrontend.ui")
        uic.loadUi(path,self)
        self.connect()

    @inlineCallbacks
    def connect(self):
        from labrad.wrappers import connectAsync
        from labrad import types as T
        self.T = T
        cxn = yield connectAsync()
        self.server = cxn.normalpmtflow
        yield self.initializeContent()
        yield self.setupListeners()
        #connect functions
        self.pushButton.toggled.connect(self.on_toggled)
        self.newSet.clicked.connect(self.onNewSet)
        self.doubleSpinBox.valueChanged.connect(self.onNewDuration)
```

```
        self.comboBox.currentIndexChanged.connect(self.onNewMode)

    @inlineCallbacks
    def setupListeners(self):
        yield self.server.signal__new_count(SIGNALID)
        yield self.server.signal__new_setting(SIGNALID + 1)
        yield self.server.addListener(listener = self.followSignal, source = None, ID =
            SIGNALID)
        yield self.server.addListener(listener = self.followSetting, source = None, ID =
            SIGNALID + 1)

    @inlineCallbacks
    def initializeContent(self):
        dataset = yield self.server.currentdataset()
        self.lineEdit.setText(dataset)
        running = yield self.server.isrunning()
        self.pushButton.setChecked(running)
        self.setText(self.pushButton)
        duration = yield self.server.get_time_length()
        try:
            ran = yield self.server.get_time_length_range()
        except Exception:
            #not able to obtain
            pass
        else:
            self.doubleSpinBox.setRange(*ran)
        mode = yield self.server.getcurrentmode()
        index = self.comboBox.findText(mode)
        self.comboBox.setCurrentIndex(index)
        self.lcdNumber.display('OFF')

        self.doubleSpinBox.setValue(duration)

    def followSignal(self,signal,value):
        #print signal,value
        self.lcdNumber.display(value)

    def followSetting(self, signal, message):
        setting,val = message
        if setting == "mode":
            index = self.comboBox.findText(val)
            self.comboBox.blockSignals(True)
            self.comboBox.setCurrentIndex(index)
            self.comboBox.blockSignals(False)
        if setting == 'dataset':
            self.lineEdit.blockSignals(True)
            self.lineEdit.setText(val)
            self.lineEdit.blockSignals(False)
        if setting == 'state':
            self.pushButton.blockSignals(True)
            if val =='on':
                self.pushButton.setChecked(True)
            else:
                self.pushButton.setChecked(False)
                self.lcdNumber.display('OFF')
            self.pushButton.blockSignals(False)
            self.setText(self.pushButton)
        if setting == 'timelength':
            self.doubleSpinBox.blockSignals(True)
            self.doubleSpinBox.setValue(float(val))
            self.doubleSpinBox.blockSignals(False)

    @inlineCallbacks
```

```
    def on_toggled(self, state):
        if state:
            yield self.server.record_data()
            newset = yield self.server.currentdataset()
            self.lineEdit.setText(newset)
        else:
            yield self.server.stoprecording()
            self.lcdNumber.display('OFF')
        self.setText(self.pushButton)

    @inlineCallbacks
    def onNewSet(self, x):
        newset = yield self.server.start_new_dataset()
        self.lineEdit.setText(newset)

    @inlineCallbacks
    def onNewMode(self, mode):
        text = str(self.comboBox.itemText(mode))
        yield self.server.set_mode(text)

    def setText(self, obj):
        state = obj.isChecked()
        if state:
            obj.setText('ON')
        else:
            obj.setText('OFF')

    def onNewData(self,count):
        self.lcdNumber.display(count)

    @inlineCallbacks
    def onNewDuration(self, value):
        value = self.T.Value(value, 's')
        yield self.server.set_time_length(value)

    def closeEvent(self, x):
        self.reactor.stop()

if __name__=="__main__":
    a = QtGui.QApplication( [] )
    import qt4reactor
    qt4reactor.install()
    from twisted.internet import reactor
    pmtWidget = pmtWidget(reactor)
    pmtWidget.show()
    reactor.run()
```

The application is created below **if __name__ == ´__main__':**. The first step after creating a PyQt application is to install the qt4reactor. Typically, the event scheduling within programs written with the twisted asynchronous framework is controlled by the twisted reactor. Installing the qt4reactor combines the graphical Qt event loop with the twisted event loop, allowing the two asynchronous frameworks to function well together. When the widget is created, the user interface is loaded from an external **pmtfrontend.ui** file. The file is created with QtDesigner where it is possible to draw the needed buttons, displays and other elements. They can also created without the **.ui** file by creating the individual elements programmatically.

Once the user interface is loaded, the widget establishes a connection to LabRAD. This is an example of an asynchronous connection, because in addition to controlling LabRAD

servers the widget is also listening for signals that may be emitted from those servers. Inside the **def connect** method, we also 'wire up' the individual graphical elements to the tasks they are supposed to execute. For example, when the **newSet** button is clicked by the user, the **self.onNewSet** method will be executed. This particular method will communicate with the NormalPMTFlow server to create a new dataset.

The widget should always remain up to date of the server status. even when a change is made by a different client. To accomplish this, the widget subscribes to the **signal__new_setting** signal provided by the server. Whenever that signal is called, the widget will process the received the message in the **followSetting** method, updating the display.

The created graphical widgets may run independently or be combined into a single window. This is illustrated with the **LatticeGUI.py** file:

```
from PyQt4 import QtGui
from twisted.internet.defer import inlineCallbacks

class LATTICE_GUI(QtGui.QMainWindow):
    def __init__(self, reactor, clipboard, parent=None):
        super(LATTICE_GUI, self).__init__(parent)
        self.clipboard = clipboard
        self.reactor = reactor
        self.connect_labrad()

    @inlineCallbacks
    def connect_labrad(self):
        from common.clients.connection import connection
        cxn = connection()
        yield cxn.connect()
        self.create_layout(cxn)

    def create_layout(self, cxn):
        contrl_widget = self.makeControlWidget(reactor, cxn)
        histogram = self.make_histogram_widget(reactor, cxn)
        drift_tracker = self.make_drift_tracker_widget(reactor, cxn)
        centralWidget = QtGui.QWidget()
        layout = QtGui.QHBoxLayout()
        from common.clients.script_scanner_gui.script_scanner_gui import script_scanner_gui
        script_scanner = script_scanner_gui(reactor, cxn)
        script_scanner.show()
        self.tabWidget = QtGui.QTabWidget()
        self.tabWidget.addTab(contrl_widget,'&Control')
        self.tabWidget.addTab(histogram, '&Readout_Histogram')
        self.tabWidget.addTab(drift_tracker, '&SD_Drift_Tracker')
        layout.addWidget(self.tabWidget)
        centralWidget.setLayout(layout)
        self.setCentralWidget(centralWidget)

    def make_drift_tracker_widget(self, reactor, cxn):
        from common.clients.drift_tracker.drift_tracker import drift_tracker
        widget = drift_tracker(reactor, cxn = cxn, clipboard = self.clipboard)
        return widget

    def make_histogram_widget(self, reactor, cxn):
        histograms_tab = QtGui.QTabWidget()
        from common.clients.readout_histogram import readout_histogram
        pmt_readout = readout_histogram(reactor, cxn)
        histograms_tab.addTab(pmt_readout, "PMT")
        from lattice.clients.camera_histogram import camera_histogram
        camera_histogram_widget = camera_histogram(reactor, cxn)
```

```
        histograms_tab.addTab(camera_histogram_widget, "Camera")
        return histograms_tab

    def makeTranslationStageWidget(self, reactor):
        widget = QtGui.QWidget()
        gridLayout = QtGui.QGridLayout()
        widget.setLayout(gridLayout)
        return widget

    def makeControlWidget(self, reactor, cxn):
        widget = QtGui.QWidget()
        from electrode_client.electrode import electrode_widget
        from common.clients.CAVITY_CONTROL import cavityWidget
        from common.clients.multiplexer.MULTIPLEXER_CONTROL import multiplexerWidget
        from common.clients.PMT_CONTROL import pmtWidget
        from common.clients.SWITCH_CONTROL import switchWidget
        from common.clients.DDS_CONTROL import DDS_CONTROL
        from common.clients.LINETRIGGER_CONTROL import linetriggerWidget
        from quick_actions.quick_actions import actions_widget
        from indicator.indicator import indicator_widget
        from agilent_E3633A.agilent_E3633A import magnet_Control, oven_Control
        gridLayout = QtGui.QGridLayout()
        gridLayout.addWidget(electrode_widget(reactor, cxn),     0,0,1,2)
        gridLayout.addWidget(actions_widget(reactor, cxn),       1,0,1,2)
        gridLayout.addWidget(indicator_widget(reactor, cxn),     2,0,1,2)
        gridLayout.addWidget(magnet_Control(reactor, cxn),       3,0,1,1)
        gridLayout.addWidget(oven_Control(reactor, cxn),         3,1,1,1)
        gridLayout.addWidget(cavityWidget(reactor),              0,2,3,2)
        gridLayout.addWidget(multiplexerWidget(reactor),         0,4,3,1)
        gridLayout.addWidget(switchWidget(reactor, cxn),         4,0,1,2)
        gridLayout.addWidget(pmtWidget(reactor),                 3,2,1,1)
        gridLayout.addWidget(linetriggerWidget(reactor, cxn),    3,3,1,1)
        gridLayout.addWidget(DDS_CONTROL(reactor, cxn),          3,4,1,1)
        widget.setLayout(gridLayout)
        return widget

    def closeEvent(self, x):
        self.reactor.stop()

if __name__=="__main__":
    a = QtGui.QApplication( [] )
    clipboard = a.clipboard()
    import common.clients.qt4reactor as qt4reactor
    qt4reactor.install()
    from twisted.internet import reactor
    latticeGUI = LATTICE_GUI(reactor, clipboard)
    latticeGUI.setWindowTitle('Lattice GUI')
    latticeGUI.show()
    reactor.run()
```

We create a **MainWindow** that contains multiple tabs provided by the **QTabWidget**. Inside each tab, we programmatically place all of the required widgets. For example, the **pmtWidget** discussed earlier is placed inside a grid layout under Control tab.

Appendix B

Single Qubit Operations

This Chapter summarizes the conventions for single qubit operations and describes the procedures for state tomography and Ramsey drift tracking. The conventions used in the presented analysis were adopted from those used in Rainer Blatt's group in Innsbruck.

B.1 Basis

Any superposition $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ can be represented as a point on the Bloch sphere:

$$|\Psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi}\sin\left(\frac{\theta}{2}\right)|1\rangle \ . \tag{B.1}$$

We identify the atomic energy levels such that $|D\rangle$ is on top of the Bloch sphere:

$$|D\rangle \equiv |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \ , \tag{B.2}$$

$$|S\rangle \equiv |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \ . \tag{B.3}$$

Therefore, we refer to the elements in the density matrix in the following way:

$$\rho = \begin{pmatrix} \rho_{dd} & \rho_{ds} \\ \rho_{sd} & \rho_{ss} \end{pmatrix} \ , \tag{B.4}$$

where $\rho_{ij} = \langle I|\rho|J\rangle$.

B.2 Rotations

We first review the standard definitions of the Pauli matrices:

$$\sigma_+ = |D\rangle\langle S| = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} , \tag{B.5}$$

$$\sigma_- = |S\rangle\langle D| = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} , \tag{B.6}$$

$$\sigma_x = \sigma_+ + \sigma_- = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} , \tag{B.7}$$

$$\sigma_y = -i(\sigma_+ - \sigma_-) = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} , \tag{B.8}$$

$$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} . \tag{B.9}$$

The single qubit evolution under carrier transitions is described by the Hamiltonian:

$$H_\varphi = \frac{\hbar\Omega}{2}\left(\sigma_+ e^{i\varphi} + \sigma_- e^{-i\varphi}\right) . \tag{B.10}$$

For an extensive derivation, see the steps leading up to eq. (73) of Leibfried et al. [2]. Note that the sign of the laser phase ϕ was chosen in eq. (62) of the reference such that the traveling light wave has the form $e^{i(kx-wt+\phi)}$. Time evolution of this Hamiltonian results in qubit rotations:

$$R(\theta,\varphi) = e^{-iH_\varphi t/\hbar} = e^{i\frac{\theta}{2}(\sigma_+ e^{i\varphi}+\sigma_- e^{-i\varphi})} \tag{B.11}$$

$$= \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & ie^{i\varphi}\sin\left(\frac{\theta}{2}\right) \\ ie^{-i\varphi}\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} , \tag{B.12}$$

where the angle θ is defined as $\theta = -\Omega t$. When the laser phase $\phi = 0$, the state rotates about the x-axis:

$$R(\theta,0) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & i\sin\left(\frac{\theta}{2}\right) \\ i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} . \tag{B.13}$$

Specifically for a $\frac{\pi}{2}$ pulse applied to the ground state $S = |1\rangle$:

$$R(\frac{\pi}{2},0)|S\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} i \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}(|S\rangle + i|D\rangle) . \tag{B.14}$$

The result is an eigenstate of σ_y with eigenvalue of -1. Rotations about x *do not* follow the right-hand rule. When the laser phase $\phi = \frac{\pi}{2}$, the state rotates about the y-axis:

$$R(\theta,\frac{\pi}{2}) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} . \tag{B.15}$$

In the case of a $\frac{\pi}{2}$ pulse applied to the ground state:

$$R(\frac{\pi}{2},\frac{\pi}{2})|S\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} -1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}(|S\rangle - |D\rangle) \ . \tag{B.16}$$

The result is an eigenstate of σ_x with the eigenvalue of -1. So unlike rotation about σ_x, rotations about σ_y do rotate according to the right-hand rule. Which axis follows the right-hand rule is the consequence of the negative sign in the chosen definition for the angle θ.

B.3 Summary of Rotations

$\phi = 0$**:**

$$R(\theta,0) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & i\sin\left(\frac{\theta}{2}\right) \\ i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \ , \tag{B.17}$$

$$R(\frac{\pi}{2},0) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix} \ , \tag{B.18}$$

$$R(\frac{\pi}{2},0)|S\rangle = \frac{1}{\sqrt{2}}(|S\rangle + i|D\rangle) \ , \tag{B.19}$$

$$R(\frac{\pi}{2},0)|D\rangle = \frac{1}{\sqrt{2}}(|D\rangle + i|S\rangle) \ . \tag{B.20}$$

$\phi = \frac{\pi}{2}$**:**

$$R(\theta,\frac{\pi}{2}) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \ , \tag{B.21}$$

$$R(\frac{\pi}{2},\frac{\pi}{2}) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \ , \tag{B.22}$$

$$R(\frac{\pi}{2},\frac{\pi}{2})|S\rangle = \frac{1}{\sqrt{2}}(|S\rangle - |D\rangle) \ , \tag{B.23}$$

$$R(\frac{\pi}{2},\frac{\pi}{2})|D\rangle = \frac{1}{\sqrt{2}}(|S\rangle + |D\rangle) \ . \tag{B.24}$$

B.4 Tomography

An arbitrary density matrix for a mixed state qubit may be written as

$$\rho = \frac{I + \vec{r}\cdot\vec{\sigma}}{2} \equiv \begin{pmatrix} p_d & x+iy \\ x-iy & 1-p_d \end{pmatrix} \ , \tag{B.25}$$

where we have defined the entries of $\vec{r} = (r_x, r_y, r_z)$ to be:

$$r_x = 2x \ , \tag{B.26}$$
$$r_y = -2y \ , \tag{B.27}$$
$$r_z = 2p_d - 1 \ , \tag{B.28}$$

with the corresponding limits:

$$-1/2 \leq x \leq 1/2 \ , \tag{B.29}$$
$$-1/2 \leq y \leq 1/2 \ , \tag{B.30}$$
$$0 \leq p_d \leq 1 \ . \tag{B.31}$$

For a complete introduction see eq. (2.175) in Nielsen and Chuang [55]. The goal of state tomography is to measure $\vec{r}$ or, equivalently, measure every element of the density matrix ρ. This is done is three steps:

1. Measure the excited state population with the standard state readout method. This immediately yields the entry p_d.

2. Perform a rotation $R(\frac{\pi}{2}, 0)$. This transforms the state according to:

$$\rho_2 = R\rho R^\dagger = \begin{pmatrix} \frac{1}{2} + y & -ip + x + \frac{i}{2} \\ ip + x - \frac{i}{2} & \frac{1}{2} - y \end{pmatrix} \ . \tag{B.32}$$

   Then measurement excited state population yields $y + \frac{1}{2}$:

3. Perform a rotation $R(\frac{\pi}{2}, \frac{\pi}{2})$. This transforms the state according to:

$$\rho_3 = R\rho R^\dagger = \begin{pmatrix} \frac{1}{2} - x & p + iy - \frac{1}{2} \\ p - iy - \frac{1}{2} & x + \frac{1}{2} \end{pmatrix} \ . \tag{B.33}$$

   The excited state population is now $\frac{1}{2} - x$.

We see that by performing the three tomography steps we have measured p_d, x, and y, determining every entry of the density matrix ρ.

B.5 Ramsey Drift Tracker

The Ramsey drift tracker uses a Ramsey-type measurement to determine the drift of the laser frequency w_L with respect to the atomic resonance w_0 by measuring the detuning Δ:

$$\Delta = w_L - w_0 \ . \tag{B.34}$$

The detuning is defined according to experimental conventions as since blue-detuning corresponds to $\Delta > 0$. To measure Δ, we prepare a superposition state by applying $R(\frac{\pi}{2}, 0)$ to the ground state $|S\rangle$:

$$|\Psi_0\rangle = R(\frac{\pi}{2}, 0)|S\rangle = \frac{1}{\sqrt{2}}(|S\rangle + i|D\rangle) \ , \tag{B.35}$$

with the corresponding density matrix:

$$\rho_0 = \begin{pmatrix} \frac{1}{2} & \frac{i}{2} \\ -\frac{i}{2} & \frac{1}{2} \end{pmatrix} \ . \tag{B.36}$$

Then the state freely evolves for a duration τ. During this time, it will precess in the Bloch sphere. This evolution of coherences in the absence of the laser field given by eq. 7.42 in Ref. [43] with $\Omega = 0$ with $\tilde{\rho}_{12} \leftrightarrow \rho_{sd}$.

$$\frac{d\rho_{sd}}{dt} = -i\Delta\rho_{sd} \ . \tag{B.37}$$

The diagonal entries are unchanged and the coherence will evolve according to:

$$\rho_{sd}(t) = \rho_{sd}(0)e^{-i\Delta t} \ , \tag{B.38}$$

$$\rho_{ds}(t) = \rho_{sd}(t)^* = \rho_{sd}(0)^* e^{i\Delta t} = \rho_{ds}(0)e^{i\Delta t} \ . \tag{B.39}$$

Thus, during the laser-off time, the evolution of the density matrix is given by:

$$\rho(t) = \begin{pmatrix} \frac{1}{2} & \frac{i}{2}e^{i\Delta t} \\ -\frac{i}{2}e^{-i\Delta t} & \frac{1}{2} \end{pmatrix} \ , \tag{B.40}$$

and, hence, the state after time t is

$$\Psi(t) = \frac{1}{\sqrt{2}}(|S\rangle + ie^{i\Delta t}|D\rangle) \ . \tag{B.41}$$

We see that the Bloch vector rotates clockwise around the z-axis for blue detuning $\Delta > 0$. After the state evolution for a time τ, we preform another rotation: either $R_1 = R(\frac{\pi}{2}, \frac{\pi}{2})$ or $R_2 = R(\frac{\pi}{2}, -\frac{\pi}{2})$. The excited state population p_1 after the rotation R_1 yields the real apart of $\rho_{ds} = \frac{i}{2}e^{i\Delta t}$, see the third step of state tomography:

$$p_1 = \frac{1}{2} - \mathrm{Re}(\frac{i}{2}e^{i\Delta\tau}) = \frac{1}{2} + \frac{1}{2}\sin(\Delta\tau) \ . \tag{B.42}$$

Similarly, the excited state population p_2 after R_2 is found to be:

$$p_2 = \frac{1}{2} + \mathrm{Re}(\frac{i}{2}e^{i\Delta\tau}) = \frac{1}{2} - \frac{1}{2}\sin(\Delta\tau) \ . \tag{B.43}$$

The average of the two measurements $p_{\text{avg}} = \frac{1}{2}(p_1 + p_2)$ gives the excitation of the initial $\frac{\pi}{2}$ pulse. This may be used to track drift in the laser intensity. The detuning is extracted from the difference of the two measurements:

$$\sin(\Delta\tau) = p_1 - p_2 \ . \tag{B.44}$$

The detuning in Hz, $\Delta_f = \Delta/2\pi$ is easily found:

$$\Delta_f = \frac{\arcsin(p_1 - p_2)}{2\pi\tau} \ . \tag{B.45}$$

Appendix C

Numerical and Analytical Calculations

C.1 Molecular Dynamics

In this section we describe the principles behind the molecular dynamics simulations with trapped ion chains. The goal of the simulations is to determine the ion trajectories by numerically integrating the equations of motions. It is able to simulate the dynamics in the pseudopotential approximation or fully include the oscillating electric fields produced by a Paul trap. Additionally, one can include laser-ion interaction to simulate processes such as Doppler cooling and pulsed excitation.

This code has been developed and improved with helpful contributions from Thaned Pruttivarasin, Mark Kokish, and Manuel Gessner. The latest version utilizes Verlet integration with core components written in Cython. This way it harnesses the speed of C code while remaining easily accessible within a broader Python application that sets parameter values and launches the simulations. The code is publicly accessible on a GitHub Repository.

C.1.1 Verlet Integration

To integrate the equations of motions, we use Verlet Integration, chosen as it reduces errors compared to Euler's method. Please refer to reference [56] for a thorough description. Given the two previous two positions $\vec{x}_n$ and $\vec{x}_{n-1}$, we compute the acceleration $\vec{a}_n = A_n(\vec{x}_n)$ to determine the next position after a time step Δt:

$$\vec{x}_{n+1} = 2\vec{x}_n - \vec{x}_{n-1} + \vec{a}_n \Delta t^2 \ . \tag{C.1}$$

In the next section we derive the acceleration term $\vec{a}_n$ experienced by the ions in a Paul trap.

C.1.2 Motion in the Trap

We follow the treatment of Leibfried et. al [2] where the potential of the ion trap is assumed to be harmonic:

$$\Phi(x,y,z,t) = \frac{1}{2}U_{dc}(2z^2 - x^2 - y^2) + \frac{1}{2}U_{rf}(x^2 - y^2)\cos(\omega_{rf}t) \ . \tag{C.2}$$

In the expression above, U_{dc} is the potential applied to the DC electrodes (U in Leibfried) and U_{rf} refers to the potential on the RF electrodes ($\tilde{U}$ in Leibfried), applied with radial frequency ω_{rf}. We use the special choice of coefficients $\alpha = \beta = -1, \gamma = 2$ and $\alpha' = -\beta' = 1, \gamma' = 0$.

The secular motion of the ion can be described with a pseudopotential approximation. The potential energy yields

$$\Psi(x,y,z) = \frac{1}{2}eU_{dc}(2z^2 - x^2 - y^2) + \frac{e^2U_{rf}^2}{4m\omega_{rf}^2}(x^2 + y^2) \ . \tag{C.3}$$

From this, the trap frequencies are:

$$\omega_x = \omega_y = \sqrt{\frac{e}{m}\left(\frac{eU_{rf}^2}{2m\omega_{rf}^2} - U_{dc}\right)} \ , \tag{C.4}$$

$$\omega_z = \sqrt{\frac{2eU_{dc}}{m}} \ . \tag{C.5}$$

In order to lift the degeneracy of the radial modes, we apply an additional static bias U_{bias} on the RF electrodes as follows:

$$\Phi(x,y,z,t) = \frac{1}{2}U_{dc}(2z^2 - x^2 - y^2) + \frac{1}{2}U_{bias}(x^2 - y^2) + \frac{1}{2}U_{rf}(x^2 - y^2)\cos(\omega_{rf}t) \ . \tag{C.6}$$

The Laplace's equation $\triangle\Phi = 0$ still holds with the additional bias term. The trap frequencies are now non-degenerate and are given by:

$$\omega_x = \sqrt{\frac{e}{m}\left(\frac{eU_{rf}^2}{2m\omega_{rf}^2} - U_{dc} + U_{bias}\right)} \ , \tag{C.7}$$

$$\omega_y = \sqrt{\frac{e}{m}\left(\frac{eU_{rf}^2}{2m\omega_{rf}^2} - U_{dc} - U_{bias}\right)} \ , \tag{C.8}$$

$$\omega_z = \sqrt{\frac{2eU_{dc}}{m}} \ . \tag{C.9}$$

It is convenient to write the potential in terms of the trap frequencies instead of the geometrical dimensions:

$$\omega_z^2 = \frac{2eU_{dc}}{m} \ , \tag{C.10}$$

$$\omega_x^2 + \omega_y^2 + \omega_z^2 = \frac{e^2 U_{rf}^2}{m^2 \omega_{rf}^2} \ , \tag{C.11}$$

$$\omega_x^2 - \omega_y^2 = \frac{2eU_{bias}}{m} \ , \tag{C.12}$$

leading to the following replacements of the voltages in terms of the desired effective trap frequencies $(\omega_x, \omega_y, \omega_z)$:

$$U_{dc} = \frac{1}{2}\frac{m}{e}\omega_z^2 \ , \tag{C.13}$$

$$U_{rf} = \frac{m}{e}\omega_{rf}\sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2} \ , \tag{C.14}$$

$$U_{bias} = \frac{1}{2}\frac{m}{e}(\omega_x^2 - \omega_y^2) \ . \tag{C.15}$$

Therefore, the potential may be written as

$$\Phi(x,y,z,t) = \frac{m}{4e}\left[2\omega_{rf}\sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2}\left(x^2 - y^2\right)\cos(\omega_{rf}t) + \omega_z^2(2z^2 - x^2 - y^2) + \left(\omega_x^2 - \omega_y^2\right)\left(x^2 - y^2\right)\right] . \tag{C.16}$$

This leads to the following accelerations experienced by the ions:

$$\ddot{x} = \left[\frac{1}{2}(-\omega_x^2 + \omega_y^2 + \omega_z^2) - \omega_{rf}\sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2}\cos(\omega_{rf}t)\right] x \ , \tag{C.17}$$

$$\ddot{y} = \left[\frac{1}{2}(\omega_x^2 - \omega_y^2 + \omega_z^2) + \omega_{rf}\sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2}\cos(\omega_{rf}t)\right] y \ , \tag{C.18}$$

$$\ddot{z} = -\omega_z^2 z \ . \tag{C.19}$$

C.1.3 Coulomb Repulsion

The Coulomb potential for N ions is given by

$$\Phi_c = \sum_{i<j}\frac{e^2}{4\pi\epsilon_0}\frac{1}{|r_i - r_j|} = \sum_{i<j}\frac{e^2}{4\pi\epsilon_0}\frac{1}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}} \ . \tag{C.20}$$

This leads to the following accelerations experienced by the particle i:

$$\ddot{x}_i = \frac{1}{m}\sum_{j\neq i}\frac{e^2}{4\pi\epsilon_0}\frac{(x_i - x_j)}{((x_i-x_j)^2+(y_i-y_j)^2+(z_i-z_j)^2)^{\frac{3}{2}}} , \tag{C.21}$$

$$\ddot{y}_i = \frac{1}{m}\sum_{j\neq i}\frac{e^2}{4\pi\epsilon_0}\frac{(y_i - y_j)}{((x_i-x_j)^2+(y_i-y_j)^2+(z_i-z_j)^2)^{\frac{3}{2}}} , \tag{C.22}$$

$$\ddot{z}_i = \frac{1}{m}\sum_{j\neq i}\frac{e^2}{4\pi\epsilon_0}\frac{(z_i - z_j)}{((x_i-x_j)^2+(y_i-y_j)^2+(z_i-z_j)^2)^{\frac{3}{2}}} . \tag{C.23}$$

C.1.4 Laser-Ion Interaction

In this section we compute the probabilities for the ion to be excited or de-excited at every time step of the numerical simulation. We model the interaction of the laser with the two-level atom using a general form of the Einstein's equations [57]. In terms of the populations of the excited and ground states p_e and p_g (called σ_{bb} and σ_{aa} in the reference):

$$\frac{dp_e}{dt} = -\Gamma p_e + \Gamma_l(p_g - p_e) , \tag{C.24}$$

$$\frac{dp_g}{dt} = +\Gamma p_e + \Gamma_l(p_e - p_g) , \tag{C.25}$$

where Γ is the rate of the spontaneous emission and Γ_l is the rate of laser interaction, which is proportional to the laser intensity. In the steady state, $\frac{dp_g}{dt} = \frac{dp_e}{dt} = 0$. Using $p_g = 1 - p_e$, we have:

$$p_e = \frac{\Gamma_l}{2\Gamma_l + \Gamma} = \frac{1}{2}\frac{2\Gamma_l}{2\Gamma_l + \Gamma} . \tag{C.26}$$

Our goal is to find the laser coupling rate Γ_l in terms of saturation and detuning. To accomplish this we consider the optical Bloch equations [57]. The steady state solution is

$$p_e = \frac{1}{2}\frac{s_{\text{eff}}}{s_{\text{eff}} + 1} , \tag{C.27}$$

allowing us to identify

$$\Gamma_l = \Gamma\frac{s_{\text{eff}}}{2} , \tag{C.28}$$

where the saturation parameter s_{eff} is defined in terms of the Rabi frequency Ω and the detuning Δ:

$$s_{\text{eff}} = \frac{\frac{\Omega^2}{2}}{\Delta^2 + \frac{\Gamma^2}{4}} . \tag{C.29}$$

It is common to instead define the saturation parameter s_0 as

$$s_0 = \frac{2\Omega^2}{\Gamma^2} = s_{\text{eff}}\,(\Delta = 0) . \tag{C.30}$$

In this case,

$$s_{\text{eff}} = \frac{s_0}{\left(\frac{2\Delta}{\Gamma}\right)^2 + 1} \ , \tag{C.31}$$

and the laser coupling rate is given by:

$$\Gamma_l = \frac{\Gamma}{2} \frac{s_0}{\left(\frac{2\Delta}{\Gamma}\right)^2 + 1} \ . \tag{C.32}$$

If we start in the ground state, then per unit time Δt, the probability for the ion to get excited is $p_{\text{exc}} = \Gamma_l \Delta t$. Similarly, starting from the excited state, the total rate to be de-excited is $(\Gamma_l + \Gamma)\,\Delta t$, which is the sum of the rates of stimulated emission and spontaneous emission. These probabilities are computed for every time step, and the stochastic nature of the process requires us to average over multiple simulations to obtain accurate predictions.

C.2 Non-Linearity in the Ion Potential

In this section we calculate the amount of non-linearity seen by a single ion moving radially in an ion chain. The calculation is relevant for the discussion of the normal modes of ion motion presented in Section 4.2.1. We consider the motion of ion i, while all of the other ions in the chain are assumed to be fixed at their equilibrium positions. We consider the Taylor expansion of equation 4.2 and calculate the higher order term:

$$\frac{\partial^4 V}{\partial q_i^4}|_{\text{equil}} = \frac{9e^2}{4\pi\epsilon_0} \sum_{\substack{j=1 \\ j\neq i}}^{N} \frac{1}{|z_i^0 - z_j^0|^5} \ . \tag{C.33}$$

The equation of motion for the particle in the radial direction is a non-linear oscillator known as the Duffing oscillator:

$$\ddot{q}_i + \omega^2 q_i + \epsilon q_i^3 = 0 \ , \tag{C.34}$$

where

$$\omega^2 = \omega_x^2 - \sum_{i\neq j} \frac{e^2}{4\pi\epsilon_0} \frac{1}{m} \frac{1}{|z_i - z_j|^3} \ , \tag{C.35}$$

$$\epsilon = \frac{3e^2}{8\pi\epsilon_0} \frac{1}{m} \sum_{i\neq j} \frac{1}{|z_i - z_j|^5} \ . \tag{C.36}$$

For a Duffing oscillator, the resonance frequency depends on the amplitude of the oscillations [58].

$$\Delta = \frac{3\epsilon A^2}{8\omega} \ . \tag{C.37}$$

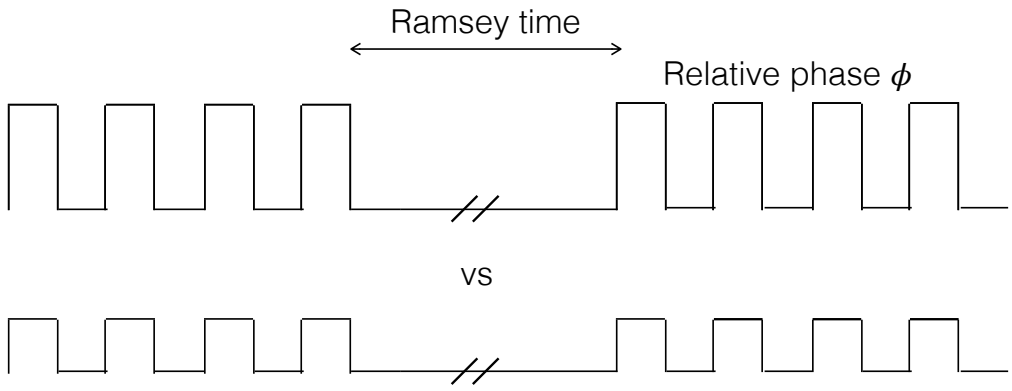

Figure C.1: The schematic of Ramsey-type sequence that may be used to detect a small non-linearity in the trapping potential. The ion is excited to a significant energy used pulsed excitation with a saturated laser (top). The system evolves for a Ramsey time, after which another series of pulsed excitations with a relative phase ϕ is applied. The result is compared to the sequence with the exact same timing but less intense pulsed laser (bottom). Any difference in the phase ϕ that minimizes the final energy corresponds to an additional phase shift accumulated due the non-linearity present for large oscillation amplitudes.

Here, Δ is the detuning of a driving field from the natural frequency ω that maximizes oscillation amplitude A. For the derivation of this result, see equation 1.9 in [58] with $\Delta = \sigma\epsilon$ and $\alpha\epsilon \to \epsilon$.

We make the approximation that only the adjacent ion contributes and that the inter-ion distance is given by the natural length-scale l defined in equation 4.11. We also assume $\omega = \omega_y$, then

$$\epsilon = \frac{3e^2}{8\pi\epsilon_0}\frac{1}{ml^5} = \frac{3}{2}\left(\frac{e^2}{4\pi\epsilon_0 m}\right)^{-\frac{2}{3}}\omega_z^{\frac{10}{3}}\ , \tag{C.38}$$

and, hence,

$$\Delta = \frac{9}{16}\frac{A^2}{\omega_y}\left(\frac{e^2}{4\pi\epsilon_0 m}\right)^{-\frac{2}{3}}\omega_z^{\frac{10}{3}}\ . \tag{C.39}$$

We compute that for radial frequency $\omega_y = 2\pi \times 2\text{MHz}$, the shift in frequency due to the non-linearity is rather small:

$$\Delta = 2\pi \times 67\ \text{Hz}\left(\frac{A}{1\mu m}\right)^2\left(\frac{\omega_z}{2\pi \times 200\text{KHz}}\right)^{\frac{10}{3}} \tag{C.40}$$

. If the ion is excited to significant energy with pulsed excitation, the non-linearity in the potential may potentially be detected using a Ramsey technique presented in Figure C.1.