A Software Package for the Simulation and Optimization of Trapped Ion Experiments

Michael R. Mitchell

Advisor: Professor Hanneke
December 12, 2024

Submitted to the
Department of Physics and Astronomy of Amherst College
in partial fulfillment of the
requirements for the degree of
Bachelors of Arts with honors

Abstract

The trapped ion platform has found significant application in fields as diverse as quantum computing and sensing, reaction chemistry spectroscopy, atomic timekeeping, metrology, and precision measurements for tests of fundamental physics. This platform excels because of the superlative storage times, high level of coherence, and intrinsic reproducibility of trapped ions. For these reasons, there may be wide application for a software package that allows for the general simulation and optimization of trapped ion experiments. One specific use of the platform is the probing of the dissociation of O_2^+ from a prepared rovibrational state, via the technique of Laser-Cooled Fluorescence Mass Spectrometry (LCF-MS). Although LCF-MS is a widely used and conceptually simple technique, the details of the cross-species excitation in moderately sized, highly 3-dimensional Coulomb crystals yield rich dynamics. Due to observed difficulties in conducting reproducible LCF-MS on these crystals, we have turned to numerical simulation methods to guide our understanding of these dynamics, with the goal of optimizing the resolution at which we can describe the number of O_2^+ in our trap. To this end, we have developed an application (dubbed the Quantum Logic Ion Control Simulator, QLICS) to conduct simulations of our experiment dynamics as well as the expected fluorescence data output. In this thesis, I provide a brief review of our experimental system, emphasizing the parameters controlling LCF-MS execution. Then, I describe how we integrate the relevant physics into our simulation environment while also conducting small 'benchmarking' experiments to verify the validity of our simulations. Next, I conduct an exploration of our system using QLICS to both improve our understanding of the experiment dynamics and, more importantly, demonstrate how the program can be generally used to study similar experiments in the future. Through these explorations, we study the implications of using different coolant species, exemplifying how the software can be used to inform tangible, strategic decisions about the lab's future directions. Finally, I showcase the most powerful feature of QLICS, its ability to incorporate machine learning for the online optimization of trapped ion experiments. I will present a new technique that appears to substantially improve our experiment's resolution in certain regimes, based on results from our machine learning-guided studies of the system. We conclude with a discussion of potential future applications of QLICS's machine learning capabilities to enhance experimental insight and performance.

Acknowledgments

First and foremost, I would like to thank Professor Hanneke. After 3 classes and more than 2 yrs of research advising, it would be difficult to overstate the role you have played in my physics education. The passion and kindness with which you approach every activity in the lab has played a major role in nurturing my own interest in research. Often, it is not until well after some discussion that I realize how misguided, redundant, or unfocused some of my questions or thoughts might have been. In these moments, I am consistently amazed by the open-mindedness and fairness with which you manage to approach all of our interactions. This patience has greatly fostered my confidence in approaching problems, and without this support, I would never have been able to begin to pursue this work. I am deeply grateful to have been able to work in your lab over these past few years and for the exceptional undergraduate research experience you have facilitated.

I also want to thank Ambesh for being an invaluable resource during his time at Amherst. Whether we were studying elastic collisions in the ion trap or on the pool table, you provided a constant source of mentorship and fun. Thank you very much for all the many different contributions and perspectives, as well as the expertise, you brought to the group. Thank you Lukas for the many lively discussions both in the lab and, especially, at DAMOP. While - or perhaps because - our thesis projects had such little conceptual overlap, it was always extremely refreshing and fun for me to check in on what you were working on or thinking about. I especially appreciate your receptiveness to my many random and naïve optics questions. Thank you to Paul as well. It has been a pleasure working with you this semester, and I can't wait to see how you are able to further develop the machine learning side of QLICS.

In addition, I want to thank Doug Hall and the whole Amherst College HPC team. Without your assistance and guidance, several of the results reported in this work would not have been computationally possible. I am very excited to see where further integration between QLICS and FrostByte will take us. I also want to thank the many previous thesis students, whose work built the system we have today. Clearly, without their contributions, there would be no experiment to simulate, and the work reported here would live only in the computer (not to mention how much their thesis documents themselves have helped my own understanding of the experiment). That being said, I want to specifically thank Will Henshon, whose circuit board played a particularly relevant role to my work this past year. Thank you to all the various SURF and other undergrad students with whom I have shared the lab. The approach to trapped-ion dynamics simulation reported here only came after a few other approaches were first investigated, and so I want to thank all of them for that initial pioneering.

I would not be here at the end of my thesis and undergrad degree if it wasn't for the constant support of my parents. While this is certainly true in general, I also mean it in a very literal sense, since my mother was the one who first encouraged me to apply to the SURF program when I was unsure if I would be interested in doing research. Thank you both for your constant encouragement.

Thank you also to Nora, for your unwavering patience and support throughout the many late nights I spent both coding and writing this document. Without you, these past few years would not have been nearly as great.

Finally, I want to thank the football coaching staff and all my friends and teammates. I know Coach Ballard, in particular, has been waiting for the release of this document with great anticipation.

This thesis is based upon work supported by the Amherst College Provost and

Dean of the Faculty and by the National Science Foundation under grants PHY-2207623 (RUI PM). It was performed in part using high-performance computing equipment at Amherst College obtained under National Science Foundation grant CNS-2117377. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Contents

List of Figures

Chapter 1

Introduction

"Gallia est omnis divisa in partes tres..."

– Julius Caesar, Opening Line of *Commentarii de Bello Gallico*

Like Caesar, I have conceptually divided the introductory material into *tres partes*: motivation, system, and the simulation software — the subject of this thesis.

While the motivation for creating a simulation environment was originally a simple desire to understand and resolve issues in our own experiment, we came to realize that the potential applications of an integrated simulation and experiment optimization program extend well beyond our specific use-case. As such, the motivation of this work (that is, the motivation to shift our focus from our own use-case towards more general use by packaging our workflow as a CLI[1] app) draws from the many diverse uses for trapped ions in modern research and industrial settings, outlined in Sec. 1.1.

Before we can discuss our simulations, we must first introduce the system we hope to represent. I have abbreviated the system discussion to emphasize not just the aspects of our system relevant to the trap, but the aspects key to how we structure our simulation environment. One might expect the relative narrowness of this approach to ultimately restrict the general application of the software package, but

[1]Command Line Interface. Unlike the real-world experiment control software, we have not yet integrated any GUI.

the abstraction of much of our experiment actually generalizes the discussion away from our specific use-case. This carries over into the nature of the program itself, and although the details of the Hanneke lab trapped ion system play a major role in the results presented in this work, we have worked to minimize the extent to which these details restrict the applicability of the software itself.

We will then conclude our introduction with an explanation for why we found the need to study our system in a simulation environment, instead of studying the real system itself. While this reasoning is specific to our group, it is probable that the reader interested in exploring the possibility of simulating will relate to some of our needs. To garner inspiration, and see the scope of studies that can be conducted using the software in its current state, I refer the reader to the list of 'suggested projects' in Sec. 5.1.

1.1 Motivation

While we do introduce the motivation of the Hanneke lab here, it is only a single example among many possible use-cases of trapped ions. We therefore give the applications of trapped ions primacy in our discussion. In other words, there is ample motivation in studying trapped ion dynamics for the sake of better understanding trapped ion dynamics.

1.1.1 Ion Trapping

With major applications in fields as disparate as quantum computing, reaction chemistry, and precision measurement spectrometry, ion traps (and the closely related quadrupole mass filters) have emerged as one of the main technological devices developed and used for precision control of atoms and molecules in modern research settings. In general, ion traps are devices that use electric fields to confine charged particles in 3-dimensional space. Their effectiveness stems from their versatility, their long storage times and deep trapping potentials, and the relative ease[2] of production [1].

Quantum Computing

Trapped ions are recognized as one of the leading platforms for quantum computing. Sympathetic arguments point to the gate fidelity[3] superiority (in both 1 and 2-qubit operations) of trapped ion systems as well as the aforementioned coherence advantage [1]. Despite these advantages, ion traps are not wholly superior to other platforms (compare specifically with superconducting qubits) due to their low gate speeds and difficulties in scaling the number of qubits. In application, this latter restriction may seems to be more of an engineering issue than an intrinsic one, the former is mitigated in part due to the high gate fidelity and long coherence times [1].

Chemistry and Spectrometry

A primary advantage ion traps have relevant to the field of chemistry is that they can facilitate controlled observation of reactions, allowing for detailed study of reaction mechanisms [2]. Ion traps are especially helpful at studying the reactions of highly reactive chemicals, since it allows the experimentalists to almost perfectly isolate the reactants from the environment. This has aided in the study of hard to isolate reactions, such as ion/ion reactions [3]. Often, the traps allow for chemical reactions to be probed using various mass spectrometry techniques [4]. In addition, a common type of mass spectrometer, the quadropole mass filter, uses the same dynamics of a linear Paul trap[4] to filter a sample by species mass in analytical chemistry contexts

[2]That is not to say that there are not very delicate and complicated ion trapping systems. Rather, it is relatively (compared to other trapping systems) straightforward to 'get off the ground' and trap charged particles at all - a feature that goes a long way for some applications.

[3]Loosely, the accuracy with which the gate operations are implemented.

[4]A quadropole mass filter is, in fact, just a linear Paul trap with the endcaps removed.

[5, 6].

Precision Measurements for Fundamental Physics

Closely related to the use of RF traps to study chemical reactions in charged particles are the applications this technology finds in the context of precision measurement for tests of fundamental physics. By allowing us to hold a reproducible (all atoms / molecules of the same species are the same) system, in which we can control the internal states, in space, we can optically drive state transitions. This allows us to follow the scheme of preparing our ions into a known quantum state by some means, and then driving them into another state. If we do this, we can take precise measurements of the energy levels between such states, allowing us to probe the internal structure of atoms and molecules [7]. This feature of ion trapping is, of course, the original motivation for our use of them in the Hanneke Lab.

1.1.2 Our Precision Measurement Goal

Previous work has shown how the energy of certain transitions in O_2^+ are especially sensitive to the proton-to-electron mass ratio, μ [8, 9]. Since our goal is to measure an energy difference, control of the initial rovibrational state is essential, which is why we conduct this measurement in an ion trap.

$\dot{\mu}/\mu$ and physics beyond the standard model

The standard model, for all its triumphs, is known to be flawed in that it fails to explain reliably observed facts of the universe (gravity, matter/ anti-matter asymmetry, etc). Various theories predict small time variation in fundamental constants (including μ) and by making such measurements, we hope to either constrain these theories, or make a direct measurement contrary to predictions by the standard model (which predicts no time-variation in μ), see Ch. 2 of Annika Lunstad's thesis for much more

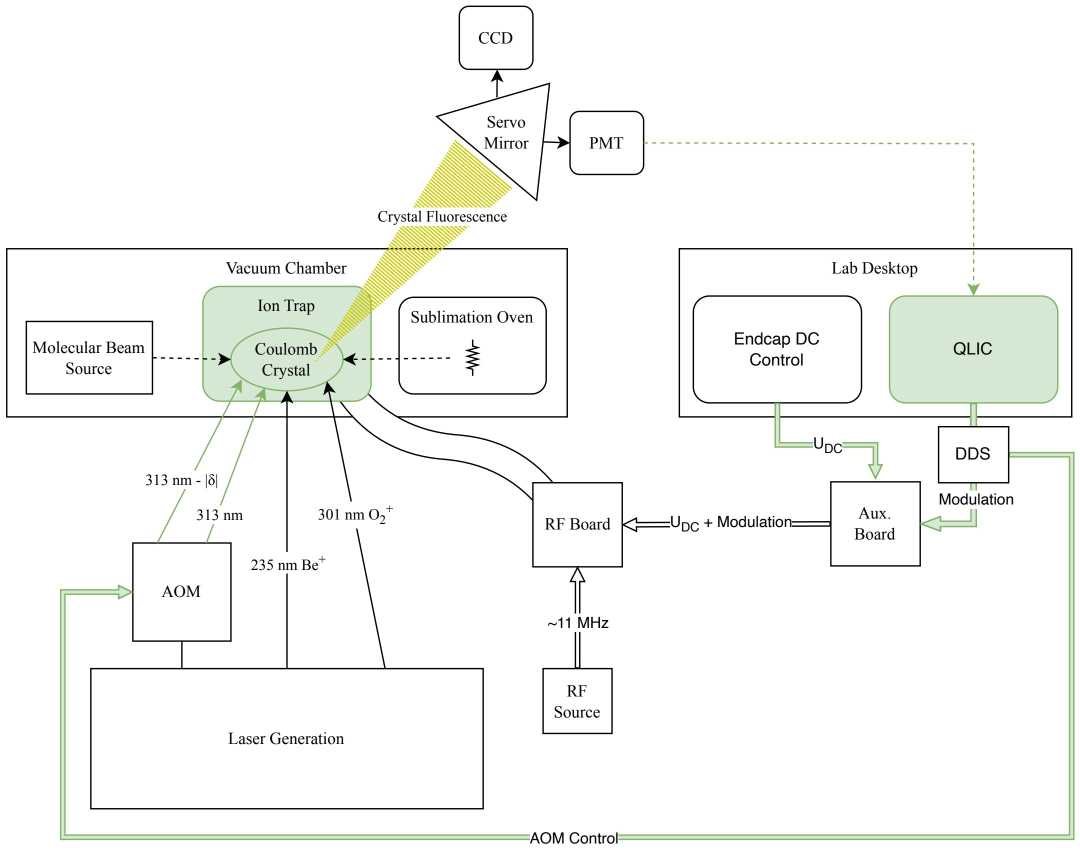


Figure 1.1: Block diagram of the relevant aspects of the trapped ion system. The items most relevant to this thesis have been colored colored green. Outlined arrows indicate electronic inputs controlling the experiment.

information, [9]. Obviously, either of these results would represent a great advance in our understanding of the nature of reality, as even a more precise zero measurement of $\dot{\mu}$ would help to constrain the list of proposed theories.

1.2 System

In very brief overview, our system (Fig. 1.1) consists of a linear Paul trap located in a vacuum chamber, accessible to lasers, and controlled by a computer system which reads information out of the trap with a PMT (photo-multiplier tube) for photon counting. We currently (intentionally) trap beryllium and molecular oxygen, introduced to the trap via a sublimation oven and a molecular beam respectively. Many of the blocks in Fig. 1.1 required full theses worth of development. I will try to refer to the appropriate sections of these works as they come up.

1.2.1 General Structure and Vacuum Chamber

Obviously, a delicate system needs to be isolated from the environment. Because the assumed primary cause of ion loss is statistical collision with neutral background gas, it makes sense to isolate the system in a low pressure environment. In Ch. 2 of his thesis, Will Henshon explains why we needed an 'ultrahigh vaccuum' and how such a vacuum is achieved [10]. For our purposes, it is enough to know that a typical pressure at the ion trap[5] is about 1×10^{-10} Torr. In addition to the trap itself, both a beryllium oven and an electron gun are situated in the trapping chamber. The beryllium oven consists of a beryllium wire wrapped around a tungsten core through which we pass a current, heating the wire and sublimating beryllium. The electron gun serves as a general purpose ionizer, ionizing atoms and molecules without regard for their internal state or species. This device acts as only a quick and dirty method of getting ions in the trap but does not play a role in our final experiments. Finally, to allow our lasers into this chamber we need to use windows of some sort. These windows are made of material specifically selected to be transparent in the frequencies of the lasers needed for our final quantum control efforts [10]. There are 6 (3 pairs) of such window ports that allow optical access to the heart of the trap chamber and an additional, unpaired port that allows our detection system to 'view' the trapped ions. We release the oxygen in pulses which travel through the two prior chambers (separated by skimmers) before entering the trap. The total path is on the order of 50 cm long. The source for the gas-pulse beam can easily be switched so that other

[5]Note that the vacuum chamber pressure is not constant across the sub-chambers, see [10] for more details.

gases, or oxygen sources of various purity[6] could easily be used.

1.2.2 Lasers

At the time of writing, there are three (final) lasers integrated into our working system. These are the 313 nm cooling laser (and its associated detuned component), the 235 nm beryllium ionization laser, and the 301 nm molecular oxygen ionization laser [8, 11–13].

Most relevant to this work is the 313 nm cooling laser. This laser is made from the third harmonic of an external cavity diode laser (ECDL) lasing at 939 nm [12]. Before entering our vacuum chamber and trapping region, the 313 nm light passes through an acousto-optic modulator [7] which splits it into a detuned component and a (lab-frame) resonant component. Typical detuning is around 400 MHz and the detuned component receives the majority ($\approx 90\%$) of the optical power. Both of these beams are controlled by our experiment software via a DDS board and so can be switched on and off with sub-microsecond timing [11], a feature that is key to the execution of our experiments[8]. The detunned and resonant components of the 313 nm laser co-propagate with a k-vector of

$$\hat{k} = \frac{1}{2}\hat{x} + \frac{1}{2}\hat{y} + \frac{\sqrt{2}}{2}\hat{z}, \tag{1.1}$$

so there is a slightly greater projection along the trap axis (the z-direction throughout this thesis).

The 235 nm beryllium ionization laser is made from from frequency doubling a 470 nm ECDL, the subject of Christian Pluchar's thesis, [13], and the oxygen ion-

[6]See [11] for the effect of source purity on beam temperature and thus state preparation.

[7]Actually it passes through two - and both components pass through each twice - but the upshot is that we generate a detuned component and a resonant component.

[8]In addition to the convenience of efficient execution, our crystals are only stable for a limited amount of time before statistical processes, namely collisions with background gas, cause ion loss.

ization laser (301 nm) is a Sirah dye laser pumped by an Nd:YAG (532 nm), further described, along with the 2 + 1 REMPI scheme in Ch. 2 of Addison Hartman's thesis, [8]. By switching a few mirrors, we can send the O_2 ionization laser into our vacuum chamber at either the TOF-MS position [14], or into the trap itself (the latter configuration is the only one of concern for this work). When it is sent in to the trapping region, it is perpendicular (in the benchtop plane) to the cooling laser. The Be ionization laser is anti-parallel to the O_2 ionization laser, meaning that we must aim to avoid having both on at the same time, and we must be careful to use physical barriers to prevent each from traveling up the other's optical paths for the protection of our optical devices.[9] Neither of these lasers are controlled electronically by our experiment manager software, and so we must manage them manually, which prevents us from automating any crystal-generation experiments for the time being.

1.2.3 Trap

Our RF trap consists of 2 segmented and 2 unsegmented electrodes. This creates a trapping potential we approximate as harmonic in all 3 dimensions (see Sec. 2.2.1). The key takeaway is that the strength of the trapping potential (the coeffecient of the dominant, second order term in the potential series expansion), is related to the charge-to-mass ratio of a particular particle. This species dependence causes the secular frequencies of lighter particles to both be higher, and the geometry of multi-species crystals to behave in certain ways, see Fig. 1.2.

The high voltage RF potential of the trap is controlled by an electronic system, detailed throughout Ch. 3 of Will Henshon's thesis [10]. The electronic system is broken into two boards, one is the 'main board' placed immediately outside of the vacuum chamber, and the other is an auxiliary board connecting our computer-controlled signals to the main RF board. While the main board controls the high

[9] In practice, it is much more important to protect the Be laser generation area from the powerful O_2 laser. Nevertheless, the point is that we can't use both at once.

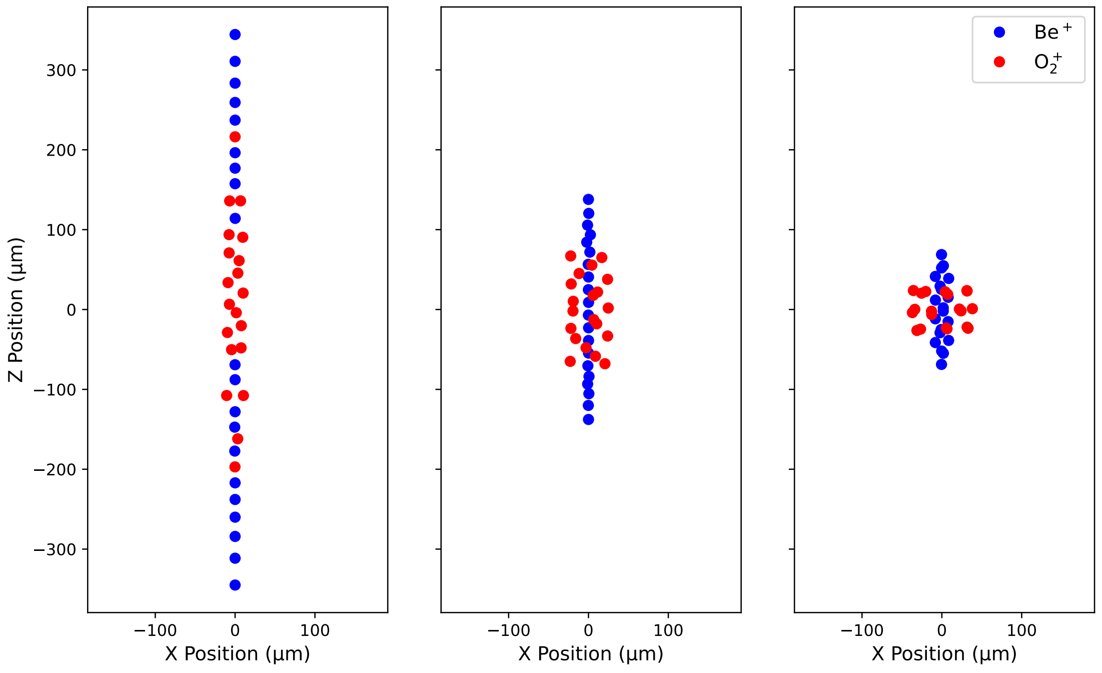
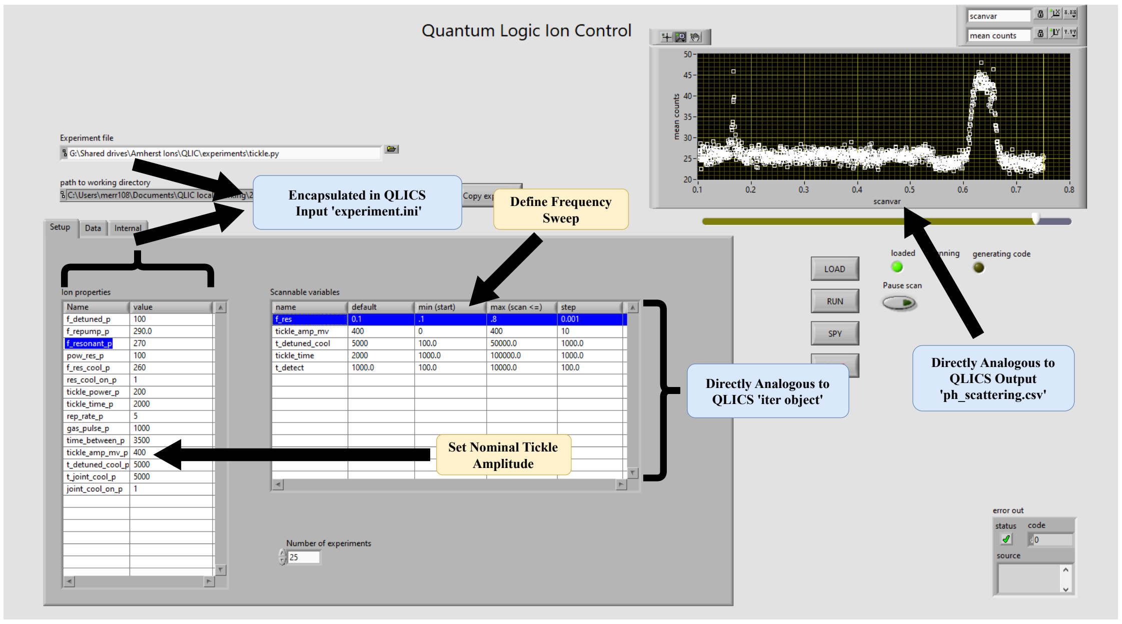


Figure 1.2: Large, mixed species crystals with progressively larger axial confinement (radial confinement is held constant). Note how for significant axial confinement, the ions arrange themselves such that the lower charge-to-mass ratio ions are on the outside. The relationship between different trap parameters and the crystal geometry will be a key concept throughout this works. The figures were made by numerical simulations done in qlicS.

voltage, high frequency electronics creating the trapping potential, the auxiliary board delivers the small, 'modulating' signals to particular segments of the trap's electrodes. This process of disturbing the crystal with small time-dependent signals is often referred to as 'tickling', and this technique allows us probe the crystals' internal composition, as we will discuss shortly.

Central to this thesis is the technique of Laser-Cooled Fluorescence Mass Spectrometry (LCF-MS), a technique for the detection of ions that cannot be detected directly (i.e. if they are not fluorescing). Broadly, because the secular frequency is species-dependent, we can biasedly excite certain species' motion by modulating the trap potential at their secular frequency, creating a driven harmonic oscillator. In principle, this excited motion couples to a visible species through Coulomb interactions as visualized by the videos in Appx. A. Then, the Doppler shift from the detection species' motion causes a change in its recorded fluorescence.

This process is managed by an experimental controller, a LabVIEW program called 'Quantum Logic Ion Controller' (QLIC). The program takes Python files that define experimental sequences as input. While nominally written in Python, these scripts actually contain instructions sent to our direct digital synthesis (DDS) board, which is able to take the digital signal from our computer and turn it into an analog waveform. As mentioned in 1.2.2, this is essential as we require pulses at frequencies greater than that possible with our computer software interface to efficiently run our experiments. Once these scripts are loaded into the QLIC program, we are able to set the variables we wish to change over the course of the experiment, the values of variables we wish to hold constant, and the range and step sizes with which we wish to conduct our variable sweep. The most common and important experiment we run with QLIC are the frequency sweep experiments, where we linearly sweep the frequency of a signal modulating the trap potential, recording the fluorescence response of the crystal at each modulating frequency. The series of pulses QLIC


Quantum Logic Ion Control
Encapsulated in QLICS Input 'experiment.ini'
Define Frequency Sweep
Set Nominal Tickle Amplitude
Directly Analogous to QLICS 'iter object'
Directly Analogous to QLICS Output 'ph_scattering.csv'
Setup
Data
Internal
LOAD
RUN
SPY
loaded
generating code
Pause scan
scanvar
mean counts
error out
status
code
source

Experiment file
G:\Shared drives\Amherst Ions\QLIC\experiments\tickle.py

path to working directory
C:\Users\merr108\Documents\QLIC local

Ion properties

Name	value
f_detuned_p	100
f_repump_p	290.0
f_resonant_p	270
pow_res_p	100
f_res_cool_p	260
res_cool_on_p	1
tickle_power_p	200
tickle_time_p	2000
rep_rate_p	5
gas_pulse_p	1000
time_between_p	3500
tickle_amp_mv_p	400
t_detuned_cool_p	5000
t_joint_cool_p	5000
joint_cool_on_p	1

Scannable variables

name	default	min (start)	max (scan <=)	step
f_res	0.1	.1	.8	0.001
tickle_amp_mv	400	0	400	10
t_detuned_cool	5000	100.0	50000.0	1000.0
tickle_time	2000	1000.0	100000.0	1000.0
t_detect	1000.0	100.0	10000.0	100.0

Number of experiments
25

Figure 1.3: QLIC front panel screenshot with key controlled parameters of the real world tickle experiment highlighted, and analogues to important QLICS objects noted. As will be discussed in Ch. 3, the capabilities of QLIC are exactly the QLICS 'Run Experiment from File' subroutine. The data displayed is photoshopped in from the mixed-species crystal discussed in [11]. Variable values are roughly true to the system when the original experiment was run, but should not be taken as exact.

creates via the DDS board defines the ‘experimental sequence’: the duration and times we leave on the cooling laser, probing resonant laser, trap modulation, count photons for, etc. An example of the program front panel while running such an experiment is shown in Fig. 1.3.

1.2.4 Imaging System

To see the crystal and take data from our experiments we have installed an imaging system above the trap (as seen in Fig. 1.1). A microscope with a total magnification of 30 magnifies the photons that are re-emitted by the ions in the crystal, if we shine a laser resonant with a cycling transition on it. This light is then sent to either a PMT or EMCCD camera, which we can switch back and forth between with a servomotor-controlled flipper mirror. Images of the crystal are sent to a computer in the lab, where we can often easily resolve the individual ions within the crystal (typically spaced by several microns). The PMT photon counter is controlled by the QLIC system and so we can receive photon counts over time frames needed per our experiment. These results are logged live as the experiment is conducted in the LabVIEW program’s front panel, as shown in Fig. 1.3.

1.3 Simulation

Because we are so greatly restricted in the ways we can actually probe our Coulomb crystals[10], there is a great potential for better understanding the dynamics of our system through direct numerical simulation. However, numerical simulations are inherently approximations of the actual physics, and so the tool must be used with care so as to ensure our results are both physically accurate and true to our system.

[10]Trapped ions that are cooled into an apparently rigid structure are often referred to as ‘Coulomb crystals’.

1.3.1 Simulation and Optimization Need for Our Lab

The initial motivation for simulating dynamics at all came from the unpredictable and confusing results we were getting from our real-world experiments. Specifically, we would see resonances at unexpected frequencies, an absence of resonance response behavior that we expect at times, and resonances that resulted in both peaks and dips in the signal. In addition, we observed the unusual phenomena of resonances that, we assumed, represent the disturbance of the Be^+ in the crystal as peaks. This behavior would appear inconsistently and is not in agreement with our theoretical understanding of our system. Also, because there are many parameters over which we could conduct our experiment, we want to find a systematic way of identifying the best locations in this parameter space to improve our various experiment's signals. In other words, we want to treat our simulation environment as a virtual environment in which we can conduct optimizations, as well as explorations to help us understand the actual dynamics of the crystals.

1.3.2 Breadth of Usecases for a Trapped Ion Simulator and Optimizer

While the objectives in the above subsection could have been accomplished by simply creating and understanding our simulations, after some progress was made in the project it became clear that the machinery we developed to get (and maintain) working simulations was significant enough that it would be in the long-term benefit of the lab to create a simulation app that inherits all this machinery and makes life easier for the ion-trapper. We called this program 'QLICS' (Quantum Logic Ion Control Simulator) since, when all is going well, it should be simply simulating the happenings of our experiments managed by QLIC. The program follows the general operational structure of taking a configuration file (or batch of configuration files) as input, and returning photon-count-style data as output, a process which will be described in greater detail in Ch. 3. The program is written almost exclusively in Python and contains a full suite of examples, outlined in Appx. B, and that will be discussed in the associated documentation [15]. The internal structure of the program will be covered in Ch. 3. Finally, because optimization of our signal is such a significant goal, we have incorporated a workflow which allows the app to directly integrate with an online[11] machine learning optimization package called M-LOOP [16]. This allows us to tap into a powerful suite of machine learning tools, allowing neural networks to take hold of our simulated experiment, changing the values of various parameters as it optimizes our signal for us.

[11]Not 'on the internet', but 'as our experiments are running'.

Chapter 2

Background and Simulation Benchmarking

This chapter will cover the experimental techniques relevant to our simulation work. I will not attempt to provide a complete theory of any of these techniques. Rather, I will only discuss these tools at the level of detail needed to understand the context within which the simulation and crystal analysis work was completed.

2.1 Overview

The core of the experiment is the RF ion trap (the linear Paul trap). I will start by discussing the basic principles of ion trapping in this device, including the parameters relevant for the rest of the thesis, before turning to our specific trap design. We will then move on to the technique used to indirectly and non-destructively analyze our trapped crystals' composition. Next, I will discuss the necessary principles of sympathetic laser cooling before laying out our methods for simulating this system. Finally, I will discuss several benchmarking experiments we completed to ensure agreement between our simulations and expected results (obtained through both analytical and numerical methods) for some simple and relevant systems. This final step is essential for giving us the confidence that our simulations are ‘correct’ - that is, that they reflect both real physics and our specific experimental setup.

2.2 Relevant Techniques

2.2.1 RF Trap Principles

General Potential

While the stable confinement of ions in 3-dimensions with static electric fields is disallowed by Earnshaw’s Theorem, traps constructed with time-oscillating electric fields can provide stable confinement over multiple oscillations of the trapped particles. A common subset of traps based on this principle are called ‘linear’ when oscillating fields are used in only 2 dimensions, and a static electric field is used in the third [17].

I will now reproduce the portions of the discussions found in Refs. [10, 17, 18] relevant to the simulation work. First, consider that the total electric potential, $\Phi(x, y, z, t)$, can be separated into time-independent and time-dependent parts, which we then expand (with the quadratic terms being the lowest-order term of significance for a potential with a minimum), such that:

$$\Phi(x, y, z, t) = \frac{1}{2}V(\alpha x^2 + \beta y^2 + \gamma z^2) + \frac{1}{2}U\cos(\Omega t)(\alpha' x^2 + \beta' y^2 + \gamma' z^2) \tag{2.1}$$

where Ω is the oscillation frequency on the RF electrodes, U is the amplitude of the RF potential, and V is static potential[1]. Note that U and V are both single effective values that map to slightly more complicated configuration of potentials on the physical electrodes.

In the region within the trap, there are no additional charges defining the trapping potential, so Laplace’s equation holds true ($\nabla^2\Phi = 0$). By applying this constraint,

[1]Despite the capital lettering, U and V are simply scalar constants

we can restrict the values of the expansion coefficients. The specific solution to this constraint for linear Paul traps is that

$$-(\alpha + \beta) = \gamma > 0 \tag{2.2}$$

$$\alpha' = -\beta'. \tag{2.3}$$

Where we have let $\gamma' = 0$ since the the time-oscillating portion of the potential will be confined to the x and y dimensions. To clarify, we can justify the separate analysis of the primed and unprimed spatial coefficients by superposition or by the fact that Laplace's equation must be satisfied for all times.

Motion in the axial dimension

Since $\gamma' = 0$, the equation of motion in the z-direction, which we will now refer to as the 'axial' dimension, is particularly straightforward to derive:

$$\ddot{z} = -\frac{Q}{m}\frac{\partial \Phi}{\partial z} = -\frac{QV\gamma}{m} z \tag{2.4}$$

where Q is the charge of a particle in the trap. This yields the axial secular frequency, of

$$\omega_z = \sqrt{\frac{QV\gamma}{m}} = \sqrt{\frac{2\kappa|e|V}{mz_0^2}} \tag{2.5}$$

where in the final equality we let $\gamma = \frac{2\kappa}{z_0^2}$, and $Q = |e|$ since we only need to consider singly-ionized particles. Here e is the charge of the electron, z_0 is a characteristic length scale in the trap, customarily chosen to be half the length of the central electrode, and κ is a dimensionless factor defining the geometry of the trap. κ has been simulated to be around 0.17, and has been found experimentally to be about 0.11, although this value seems to change for different trap parameters.[2] The key

[2]For the experimental basis of these values, see Sec. 4.2.4 of Will Henshon's thesis, [10].

feature to note is that ω_z scales as $\sqrt{1/m}$.

Motion in the radial dimension

For the first pass we will assume trap symmetry in the x and y dimensions (which we will now refer to as the 'radial' dimensions), and will operate under this assumption for the derivation of ω_r, the secular radial frequency of a trap with no endcap voltage ($V = 0 \implies \omega_z = 0$). Similarly to 2.4 we get,

$$\ddot{x} = -\frac{Q}{m}\frac{\partial \Phi}{\partial x} = \frac{-Q}{m}(U\alpha' \cos(\Omega t) + \alpha V)x \tag{2.6}$$

and by letting $\alpha' = \frac{1}{r_0^2}$ and $\alpha = -\frac{\kappa}{z_0^2}$ and applying the substitutions:

$$\xi = \frac{\Omega t}{2}, a_x = -\frac{4QV\kappa}{m\Omega^2 z_0^2}, q_x = -\frac{2QU}{m\Omega^2 r_0^2} \tag{2.7}$$

we can recognize 2.6 as a Mathieu differential equation with a_x and q_x being the Mathieu parameters.[3] The details of the approximate solution for this equation can be found in Refs. [17, 18], but for our purposes it is sufficient to say that to first order and for q_x well below 1 (which allows us to ignore the oscillations at Ω, as will be discussed shortly), we find that

$$\omega_r = \frac{q\Omega}{2\sqrt{2}} = \frac{|e|U}{\sqrt{2}m\Omega r_0^2} \tag{2.8}$$

where we have again let $Q = |e|$ by the same reasoning of the preceding section. Note that because $\omega_r \propto 1/m$, the secular frequency of motion is unique to each species[4]

[3]Note that some derivations introduce the sign difference on the a and q Mathieu parameters, but here we introduce it on α and α'. In the end, this doesn't ultimately matter since we square q to get the physically meaningful frequencies anyways, Eq. 2.9

[4]We, of course, assume no rare isotope or chemical compound related conspiracy to disguise an unwanted species as an expected one. This assumption is especially valid when we are loading via photoionization. When loading with the electron gun, there has been evidenced for the ionization and trapping of unexpected species, though we still generally assume we are safe from a mass-uniqueness standpoint. In addition, if we consider doubly-ionized species, their are further means of

(we are concerned, essentially exclusively, with single ionization). In addition, since we assumed radial symmetry, $q_x = |q_y|$ (and so $\omega_x = \omega_y = \omega_r$). As previously stated, this is the ω_r for a trap that is unbounded in the axial direction.

Of course, we are seeking 3-dimensional confinement. From the zero enclosed charge case of Gauss's law it is clear that the counter-propagating electrical field lines due to the static potential must split out into the x and y dimensions at the trap's center. These field lines serve to decrease the field strength in the x and y dimensions. Under the assumption that they split out symmetrically:

$$\omega_x = \omega_y = \sqrt{\omega_r^2 - \omega_z^2/2} \tag{2.9}$$

a slight reduction to the strength of the radial trapping potential. This is the approximation of the radial frequency that is used in our molecular dynamics simulations [15, 19]. It should be noted that the assumption of a cylindrically symmetric endcap voltage is actually not accurate for our physical trap, since only 2 of our 4 electrodes are segmented (and hold the endcap potentials), Fig. 2.1. This has the effect of breaking the $\omega_{x,y}$ symmetry.[5]

While it is important to understand how the trap parameters effect the secular motion in each dimension, it is worth stating explicitly that, by far, the most important point of this discussion is the charge-to-mass ratio dependence of both ω_x, ω_y, and ω_z as seen in Eqs. 2.5 and 2.8. This means that each singly-ionized species (of differing masses) has a unique secular frequency. This forms the core of the spectroscopic techniques introduced in Sec. 2.2.2 and referred to throughout this thesis.

disguising one species as another. While we don't realistically expect such a conspiracy to occur, it is good to keep the possibility of these types of errors in mind.

[5]By appropriately modifying the a parameters for x and y we get $\omega_x = \sqrt{\omega_r^2 - \sigma_0\omega_z^2}$ and $\omega_y = \sqrt{\omega_r^2 - (1-\sigma_0)\omega_z^2}$, for which Eq. 2.9 is clearly the special case that $\sigma_0 = 0.5$. David Lane investigates the value of σ_0 and reports that this value has a V_{DC} dependence. See Sec. 3.5 of [20] for more details.

Pseudopotential Approximation

Since Eqs. 2.5 and 2.9, define the frequencies of simple harmonic oscillators, they can be used to approximate Eq. 2.1 as a static, 3-dimensional harmonic potential:

$$\Phi_{\mathrm{pseudo}}(x, y, z) = \frac{1}{2}[\omega_{x,y}^2 m(x^2 + y^2) + \omega_z^2 mz^2] \tag{2.10}$$

This 'pseudopotential approximation' is extremely useful in simulating the dynamics within the trap. In addition to decreasing the runtimes for simulations, it allows us to solve many simple systems analytically, providing a basis for the benchmarking experiments conducted at the end of this chapter. Note that while this approximation is normally very good near the trap null position in certain traps, it loses[6] all of the 'micro-motion', the motion of the ions at the RF frequency, Ω. Naturally, this characteristic defines the range at which this approximation is valid, with the basic assumption being that the motion at the secular frequency dominates the motion at the RF frequency, or $|q_{\mathrm{x,y}}| << 1$. This restriction on $q_{\mathrm{x,y}}$, also restricts $|a_{\mathrm{x,y}}| << 1$ as is required to maintain a stable trap, see Fig. 1 of [21]. Typically, we try and keep $q_{\mathrm{x,y}} < 0.3$, with the maximum of the corresponding allowed range of $|a_{\mathrm{x,y}}|$, of course, lower.

2.2.2 The 'Tickle'

In broad terms, the main goal of the simulation work is to accurately analyze the composition of the Coulomb crystals in the trap. The primary issue we face is that, while we can directly image Be^+ ions with a resonant laser beam, we do not have a light source to cause O_2^+ or O^+ to fluoresce. We therefore use the indirect technique of superimposing an additional time-oscillating electric field, resonant with the secular

[6]Glossing over the micro-motion is a major source of computational acceleration since it allows us to set our timestep size based on the highest secular frequency (several hundreds of kHz) instead of the trap RF frequency (about 10 MHz).

frequencies of motion for these ‘dark’ species in the hopes that their increased motion will disturb the coolant species we are imaging enough to cause a change in the fluorescence signal. We often call this process ‘tickling’ the ions, because the term evokes the image of a small (in magnitude) but precise (in frequency) disturbance causing a large scale excitation in a system. This term also captures the periodic nature of the disturbance.

The Physical Trap and Electrode Segments

The linear Paul trap is constructed out of 4 long electrodes with semi-cylindrical tips [18]. We number the electrodes 1-4, progressing either clockwise or counterclockwise from a viewpoint looking down the trap axis, Fig. 2.1. The trap RF is created by using a counter-wound transformer to create two π out-of-phase voltage signals. Each of these signals are sent to two opposing electrodes (see Fig. 2.1) such that the phases of each electrode satisfy $\phi_1 = \phi_3 = \phi_2 + \pi = \phi_4 + \pi$, both the transformer and the phase difference are discussed in Sec. 3.4 of [10]. Since we drive our RF voltage on all 4 of our electrodes, the effective U, introduced in Eq. 2.1, is actually double the RF amplitude received by any single trap electrode (in other words, the peak-to-peak amplitude). Two of these electrodes are solid and two are segmented into 5 segments. The electrodes are orientated such that the segmented electrodes are directly across from each other (and likewise for the unsegmented electrodes). The electrodes are numbered such that electrodes 2 and 4 are the segmented ones. The lengths of these segments vary, with the middle three segments being much shorter than the long segments on the outsides. The segments themselves are also numbered from 1 to 5 going down the length of an electrode. Therefore, we can refer to a specific segment, say the middle segment on electrode 2 as ‘segment 2-3’. We can also send additional static and time-dependent voltages to any of the electrodes or segments independently of each other.[7] This is how we create the 'endcap' voltage of Eq. 2.1. It should be noted that we typically add different voltages (often about 10 V) to the outer segments (segments 1 and 5) to our segmented electrodes then to our inner ones. While these 'outer endcaps' (segments 1 and 5, which are the large, blue, teal, yellow, and green segments in Fig. 2.1) have been previously shown to have some effect on the harmonicity of the axial potential, the 'inner endcaps' (segments 2 and 4) dominate the axial potential in the actual trapping region, simply from their proximity (the trapping region is much smaller than the 3 mm width of the small segments). Therefore, for the remainder of this work, we treat V as simply being the potential on the inner endcaps. We have also demonstrated control of the position of a trapped crystal by varying these static charges so as to push the crystal around [10].

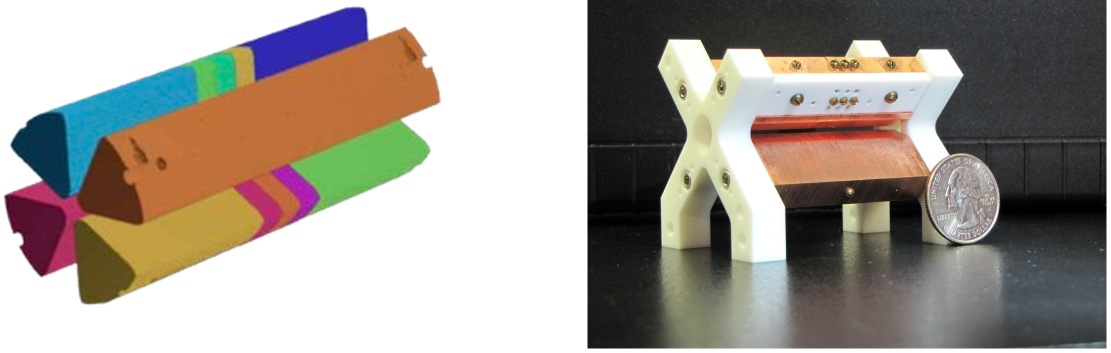


Figure 2.1: A cartoon of trap electrodes and photograph of the full trap assembly with a quarter for scale. Note that 2 of the electrodes are segmented into 5 segments, indicated by the varying colors in the cartoon. Also note that the cartoon includes only the tips of the electrodes, since we typically ignore the portions of the electrode far away from the trap center in simulations of the potential. The three central segments are 3 mm in length.

There are several types of 'tickle' electric fields we can create by sending time-dependent signals to combinations of these electrodes. While there are many specific

[7]While we have set it up so that this wouldn't be an issue, we don't actually have *complete* independent control of each segment with our current auxiliary board set up. Nevertheless, we have control of all the segments needed to create the different types of tickles discussed here.

fields we can create, the most important broadly fall into four categories, two of which we regularly conduct experiments using, and two of which have not conducted significant experiments using yet. The first two categories include ‘symmetric’ and ‘asymmetric’ tickles. A ‘symmetric tickle’ typically sends a sinusoidal signal to the middle segment of an electrode (say, segment 2-3). A tickle of this sort would be expected to predominately push a crystal symmetrically in the radial direction. An ‘asymmetric tickle’ typically sends a sinusoidal signal to one of the inner, off center electrodes (say, segment 2-2 or 2-4). This field is expected to predominately drive the crystal in the axial direction, although there is clearly some component in the radial direction as well (especially if conducted on only one electrode, as is normally done). The other tickling schemes include ‘squeeze tickles’ which would involve sending signals to multiple outside electrodes in order to drive a sinusoidal axial compression of the crystal, and an ‘unsegmented tickle’, a tickle using only an unsegmented electrode which would be expected to behave like a very symmetric tickle.

Using the computer software SIMION, we are able to simulate the electric fields that result from a combination of potentials placed on any electrode segment(s). The simulation works by iteratively calculating the electric field (by solving Laplace’s equation) at a user-defined resolution at points within a region of interest and outputting all three components of the electric field for various positions. The typical method is to apply static voltages to some combination of electrode segments in the SIMION environment. As seen in Fig. 2.2, by then fitting a series expansion through second order to the output of these simulations we can get an approximate multivariate vector function of the electric field for every position. This function is then used, along with the pseudopotential, in our dynamics simulations.

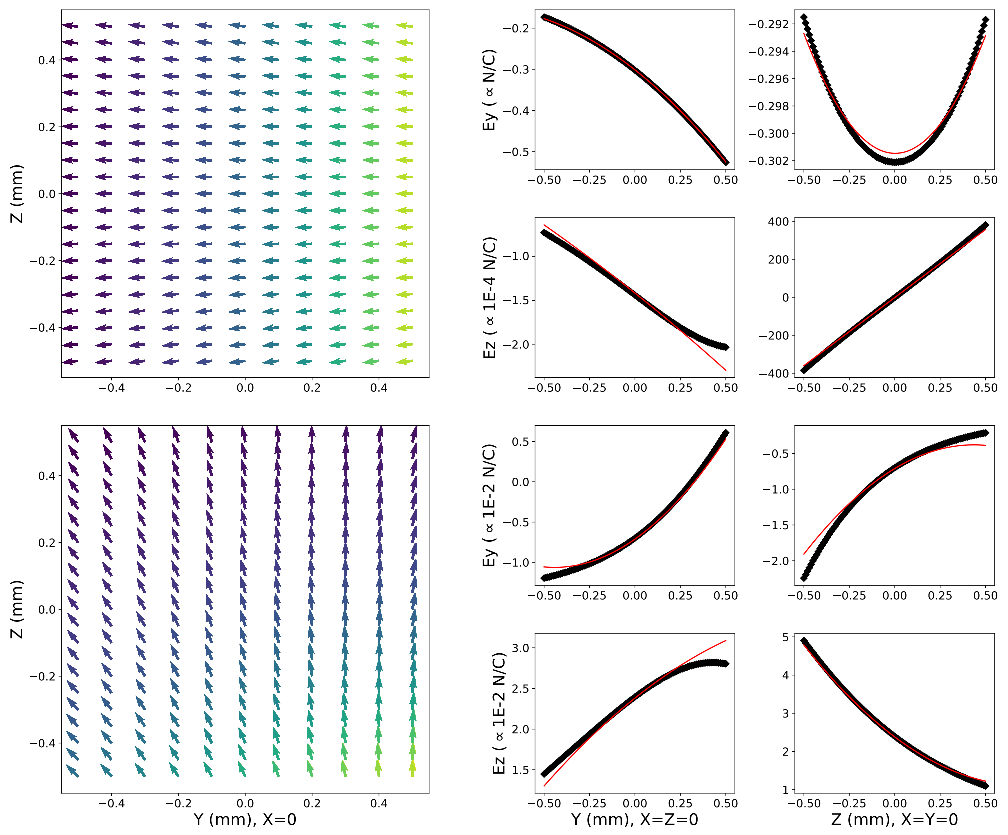


Figure 2.2: The two plots to the left show projections of the modulating $\vec{E}$ field in the y-z plane ($x = 0$), with the top plot showing a 'symmetric' tickle and the bottom plot an 'asymmetric' tickle. Notice how for a point at the trap center ($x = y = z = 0$), the symmetric tickle is almost uniformly in the $-\hat{y}$ direction, while the asymmetric tickle has a significant component in the axial ($\hat{z}$) direction. Additional slices compare our second order fit to selected components of the simulated electric field in the right hand plots (the 4 on top corresponding to the symmetric tickle, and the bottom 4 the asymmetric). Of course, we also fit the E_x components as well as the x dependence of E_y and E_z, but chose these slices because they highlight the different characteristics of the two tickle types. Since the ions are typically close the trap center, the quality of the fits are really only important in the central range of these plots.

2.2.3 Laser Cooling

In order to actually form a 'crystal', we must of course cool the system. By sending a laser with frequency detuned below the resonance of a cycling transition of our coolant ions, we can cool the system by driving the transition (and thus imparting a momentum opposite to that of the motion) of a certain velocity class of ions traveling *towards* the laser beam. Since the photon is reemitted in a random direction, the net effect over many cycles is that the system cools, this is the technique known as ‘doppler cooling’ or simply ‘laser cooling’.

Linear Approximation of Net Cooling Force

The following highly abbreviated analysis is based off of the discussion found in Ch. 3 of the book by Metcalf and van der Straten on the subject [22], and will only consider one dimension. We can easily adapt the laser cooling force defined below to three dimensions by dotting it with the laser's k-vector.

Because the reemission of the photon is in a random direction, this step in the process imparts no net change in momentum over many cycles, and so the force caused by the entire process is only the force caused by the initial absorption:

$$F_{\text{abs}} = (\hbar k)\gamma_p \tag{2.11}$$

where $\hbar k$ is just the usual photon momentum and γ_p is the total scattering rate. For a saturated system, the population is split equally between the excited and ground states (with population share in the excited state defined as ρ_{ee}). This allows us to say that [22]

$$\gamma_p = \gamma\rho_{\text{ee}} = \frac{s_0\gamma/2}{1 + s_0 + (\frac{2\delta}{\gamma})^2} \tag{2.12}$$

where we have used the on-resonance saturation parameter $s_0 \equiv \frac{I}{I_s}$ and the spontaneous decay rate γ. Therefore, an atom at rest experiences a force in the direction of $\vec{k}$ of magnitude:

$$F_{\text{abs}} = \frac{\hbar k s_0\gamma/2}{1 + s_0 + (\frac{2\delta}{\gamma})^2} \tag{2.13}$$

while it doesn't matter yet since the atom is at rest, δ is the detuning in the atom's reference frame.

To find the cooling force on an atom in motion, Metcalf et al. solves the optical Bloch equations, incorporating the velocity of the atoms as a first order perturbation [22]. The result is that we get a cooling force as a function of velocity:

$$F(v) = \frac{\hbar k s \gamma/2}{1+s} + \frac{\hbar k^2 s \gamma \delta}{(1+s)^2(\delta^2 + \frac{\gamma^2}{4})} v \tag{2.14}$$

where we have substituted an off-resonance saturation parameter, s

$$s = \frac{s_0}{1 + (2\delta/\gamma)^2}. \tag{2.15}$$

It should be noted that we have used the non-relativistic Doppler shift for light. The advantage of this approach is that since Eq. 2.14 is linear, we can try to treat the system (in the pseudopotential approximation) as a simple harmonic oscillator undergoing velocity-linear damping with a damping coefficient from eq. 2.14:

$$\beta = \frac{\hbar k^2 s \gamma \delta}{(1+s)^2(\delta^2 + \frac{\gamma^2}{4})} \tag{2.16}$$

with an additional uniform radiation pressure force F_0:

$$F_0 = \frac{\hbar k s \gamma/2}{1+s}. \tag{2.17}$$

Note that in Eq. 2.16, for red detuned light $\delta < 0$ and so $\beta < 0$.

Scattering Rate and Simulating Photon Detection

An early attempt at paramterizing a laser cooling force was done by adapting the scattering rate of the atoms. While we no longer use this method in our cooling models, it will demonstrate how we calculate the scattering rate from the resonant beam to simulate the data we actually get in the lab. It is worth emphasizing that in the real world experiment, our only route to probing any characteristic of the coolant ions goes through their photon scattering rate, so good control over this feature is key in any attempt to establish congruence between the real experiment and simulation.

We start with the photon scattering rate for a cycling transition [23]:

$$w(\Delta) = \frac{s\Gamma/2}{1 + s + (\frac{\Delta}{\Gamma/2})^2} \tag{2.18}$$

where s is defined in the same way as eq. 2.15, Γ is the natural linewidth of the transition, and Δ is the detuning frequency *in the lab frame.* We can immediately see that by plugging in $\Delta = f_\mathrm{r} - f_\mathrm{o}$, (where f_r frequency of light resonant with the transition and f_o is the frequency of light observed by the atom) we can find the scattering rate of an atom in motion as a function of its velocity and the frequency of our cooling laser in the lab frame:

$$w(v) = \left(\frac{s\Gamma}{2}\right) / \left\{ 1 + s + \left[\frac{f_\mathrm{r} - f_\mathrm{l}\sqrt{\frac{1+v/c}{1-v/c}}}{\Gamma/2} \right]^2 \right\} \tag{2.19}$$

where we have, of course, said $fo = f_\mathrm{l}\sqrt{\frac{1+v/c}{1-v/c}}$ to account for the full relativistic Doppler shift. By integrating Eq. 2.19 we can convert the output of our integration engine (which provides per-body position and velocity data over time) into simulated photon count data.

2.3 Benchmarking Experiments

Since it is so easy to mis-parameterize aspects of a physical system, bench-marking a simulation environment is essential to demonstrate the validity of the simulations we are conducting. Here we explore three bench-marking experiments that were compared to either analytical or semi-analytical methods in order to provide a basis for the physical 'realness' of our simulations.

2.3.1 3-ion Crystal

First, we will discuss the frozen (that is, the minimum energy state) of a pure, 3 Be$^+$ crystal. While the system only consists of 3 bodies, I will continue to use the word 'crystal' to emphasize its rigidity. For this discussion we will use the pseudopotential approximation in both the analytical solution and the simulation. Since, for the homogeneous 3-body case there are not multiple local potential energy minima in which the ions could get 'stuck' in, we can find the global potential energy minima by a simple force balancing technique. This straightforward problem is solved below.

Analytic Solution to 3-ion Crystal

For the analytical solution we will assume that the zero-velocity cooling laser pressure is 0, that is, referring to Eq. 2.14, $F_0 = 0$. In addition, we will assume that $k_z << k_r$ (ks are the spring constants for the analogous harmonic oscillator), so that the 3-ion chain aligns along the z axis. Because of these two assumptions and the symmetry of the problem, we can say that the minimum energy state of the small crystal consists of a central ion at the trap center flanked by two ions in the axial direction at distances of equal magnitude $+z$ and $-z$ (see Fig. 2.3).

Considering the atom at the $+z$ position, by balancing the sum of the Coulomb repulsion forces due to both the central and $-z$ ion with the force due to the trap potential, we can find the equilibrium position of this ion, $+z$. The magnitude of the force due to the pseudopotential is:

$$F_p = \omega_z^2 m z \tag{2.20}$$

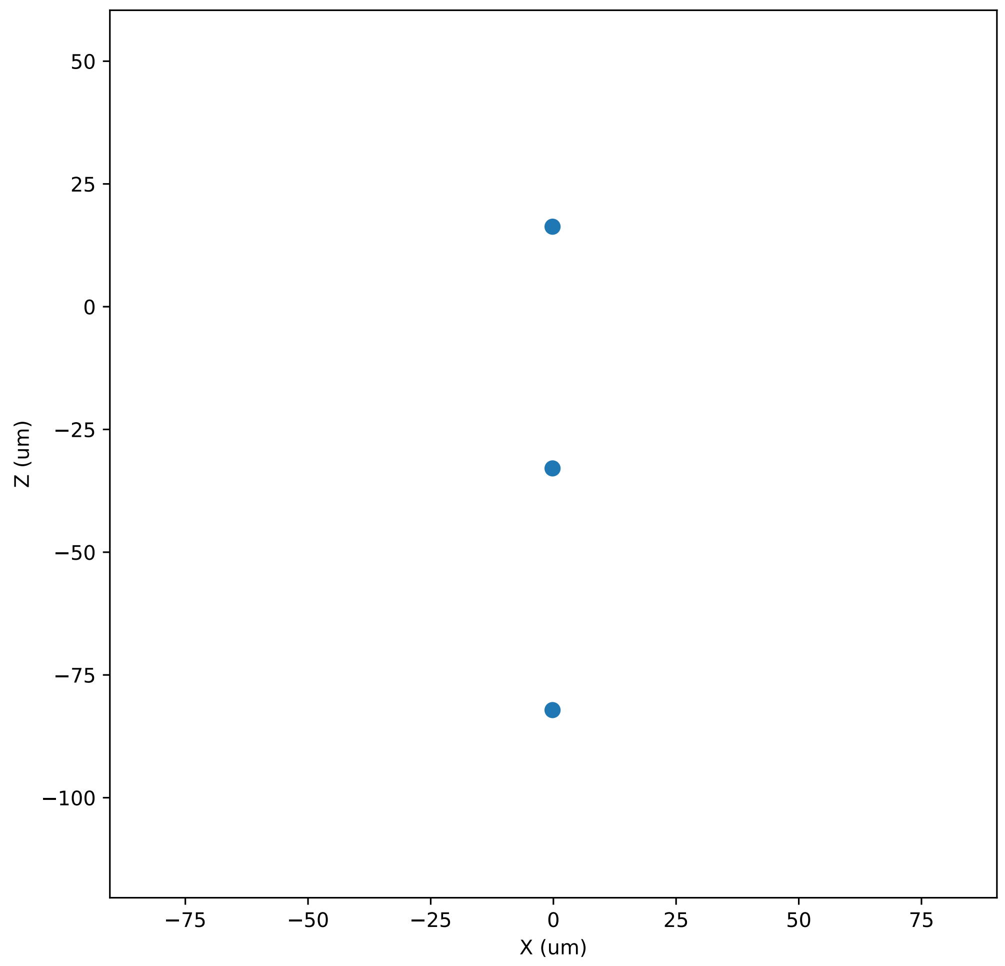


Figure 2.3: The 3-ion crystal in question. Image made from (py)LIon simulation discussed in Sec. 2.3.1. Note the significant axial displacement due to the F_0 term in our cooling laser model.

and the magnitude of the force due to the two Coulomb repulsions is

$$F_c = F_{c1} + F_{c2} = k_e e^2 (\frac{1}{z^2} + \frac{1}{4z^2}) \tag{2.21}$$

where we have again assumed single-ionization. Equating 2.20 and 2.21 and solving for z yields:

$$z = (\frac{5e^2k_e}{4\omega_z^2 m})^{1/3} \tag{2.22}$$

where k_e is Coulomb's constant. For an $\omega_z = (2\pi)64\text{kHz}$, $z \approx 4.922 \cdot 10^{-5}\text{m}$.

Simulation Solution for 3-ion Crystal

We start by parameterizing a pseudopotential for parameters equivalent to that used in the analytic solution using (py)LIon's 'linearpaultrap' function. We then drop the 3 ions in a 3 ion cloud (defined by the (py)LIon 'createioncloud' function) with a radius of $1 \cdot 10^{-4}$m. Next, we add a laser cooling force to the system defined by Eq. 2.14, and allow the simulation to progress for 6 ms. By the end of the simulation, all three ions have final velocities in the axial direction on the order of 10^{-11} m/s (the radial direction velocities are on the order of 10^{-7} m/s, so it is safe to say that the crystal is pretty frozen. The final z-positions of the ions are: $1.62912 \cdot 10^{-5}$m, $-3.29338 \cdot 10^{-5}$m, $-8.21588 \cdot 10^{-5}$m. The two Δz's in this crystal are exactly $4.9225 \cdot 10^{-5}$m, demonstrating excellent alignment with our analytic results. It should be noted that the entire crystal is significantly displaced from the trap center, due to the non-negligible velocity-independent radiation pressure term, F_0 as described in Eq. 2.17.

2.3.2 Laser Cooling and Driving

We will divide our benchmarking of our two laser cooling models into two tests. First we will confirm that the force defined in Eq. 2.14 is being properly integrated by LAMMPS. For this we will study the simple case of a flying free ion. Next, we will test both the damping rate and the steady-state amplitude of a single particle in a psuedopotential, to ensure that our damping parameters are being properly included in the simulation.

Free (Untrapped) Ion

First we will find velocity as a function of time for a free ion traveling anti-parallel to our cooling beam. We will start with Newton's second law: $F(v) = m\frac{dv}{dt}$. Writing Eq. 2.14 in terms of Eqs. 2.17 and 2.16 and we get:

$$\int_0^{t_b} \frac{1}{m} dt = \int_{v_0}^{v_f} \frac{1}{F_0 + \beta v} dv \tag{2.23}$$

evaluating these integrals means that

$$t_b = \frac{m}{\beta} \ln \left| \frac{F_0 + \beta v_0}{F_0 + \beta v_f} \right| \tag{2.24}$$

where t_b is the amount of time the cooling laser has been on, v_0 is the velocity at the moment the laser is turned on, and v_f is the velocity at time t_b. Recalling that β is negative as defined in Eq. 2.16, we can see that $t_b > 0$ for $v_0 > v_f$. Since it is a bit programmatically inconvenient to define an initial velocity for our ion, we simply accelerate it with a uniform force for a time, t_a. Thus,

$$v_0 = \frac{F_d}{m} t_a \tag{2.25}$$

and substituting Eq. 2.25 into Eq. 2.24 and solving for $v_f(t_b)$

$$v_f(t_b) = \frac{F_0}{\beta}(e^{\frac{-|\beta| t_b}{m}} - 1) + e^{\frac{-|\beta| t_b}{m}} \frac{F_d t_a}{m} \tag{2.26}$$

where we have used $\beta = -|\beta|$ strategically to emphasize that these exponentials decay. As expected, we see in Fig. 2.4 that there is perfect alignment between the Eq. 2.26 and the linear cooling method. This comes as no surprise since we are literally applying this force, but provides the peace of mind that our laser cooling force is properly integrated into our simulations.

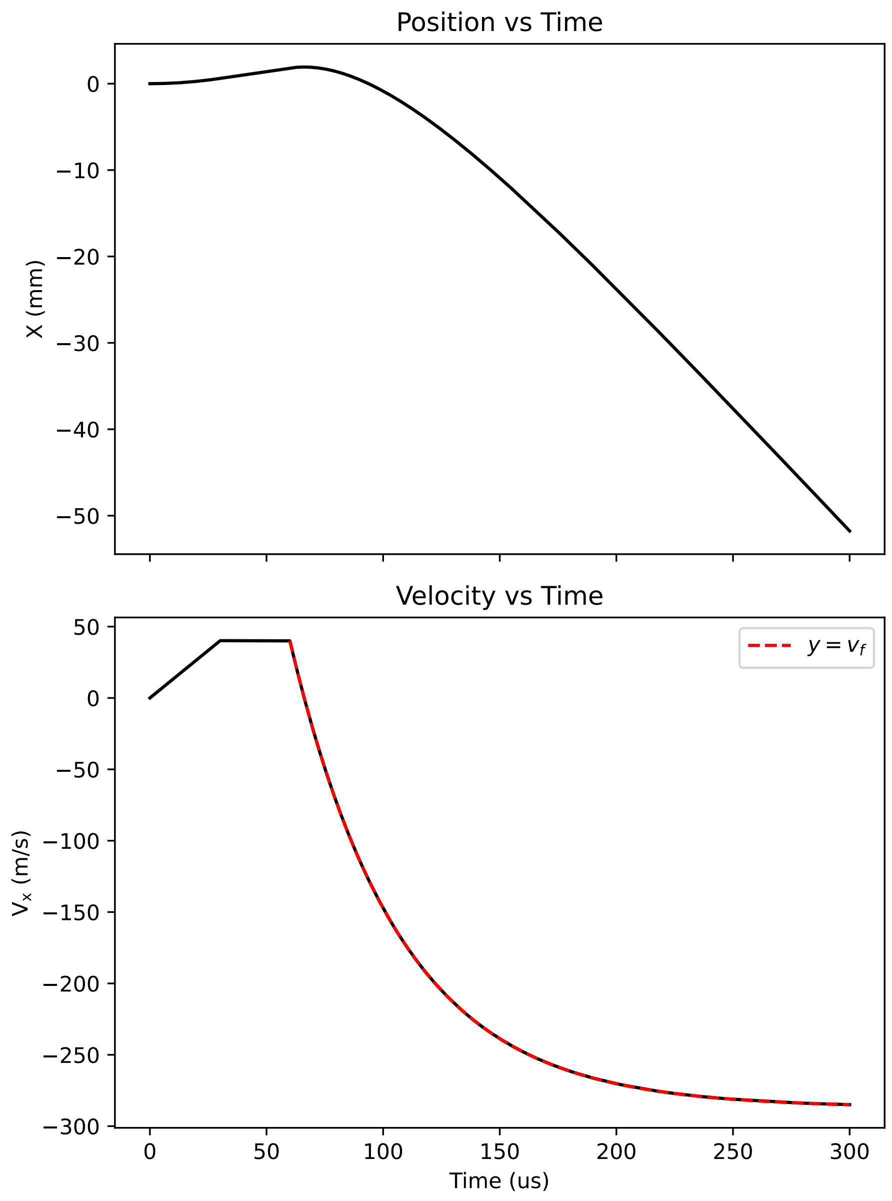


Figure 2.4: Position and velocity plots over time for a free ion experiencing the laser cooling force defined by Eq. 2.14. The overlayed velocity curve was created directly from Eq. 2.26, demonstrating a successful benchmark of our laser cooling model.

Trapped Ion

We will now compare the predicted velocity-damping to a simulation of single ion in a pseudopotential with some arbitrary initial displacement. The results of this simulation are shown in Fig. 2.5, where we plot the position and velocity of an ion oscillating in the trap as its motion is damped by the laser cooling force. The analytical damping curve for such an ion is overlayed for benchmarking.

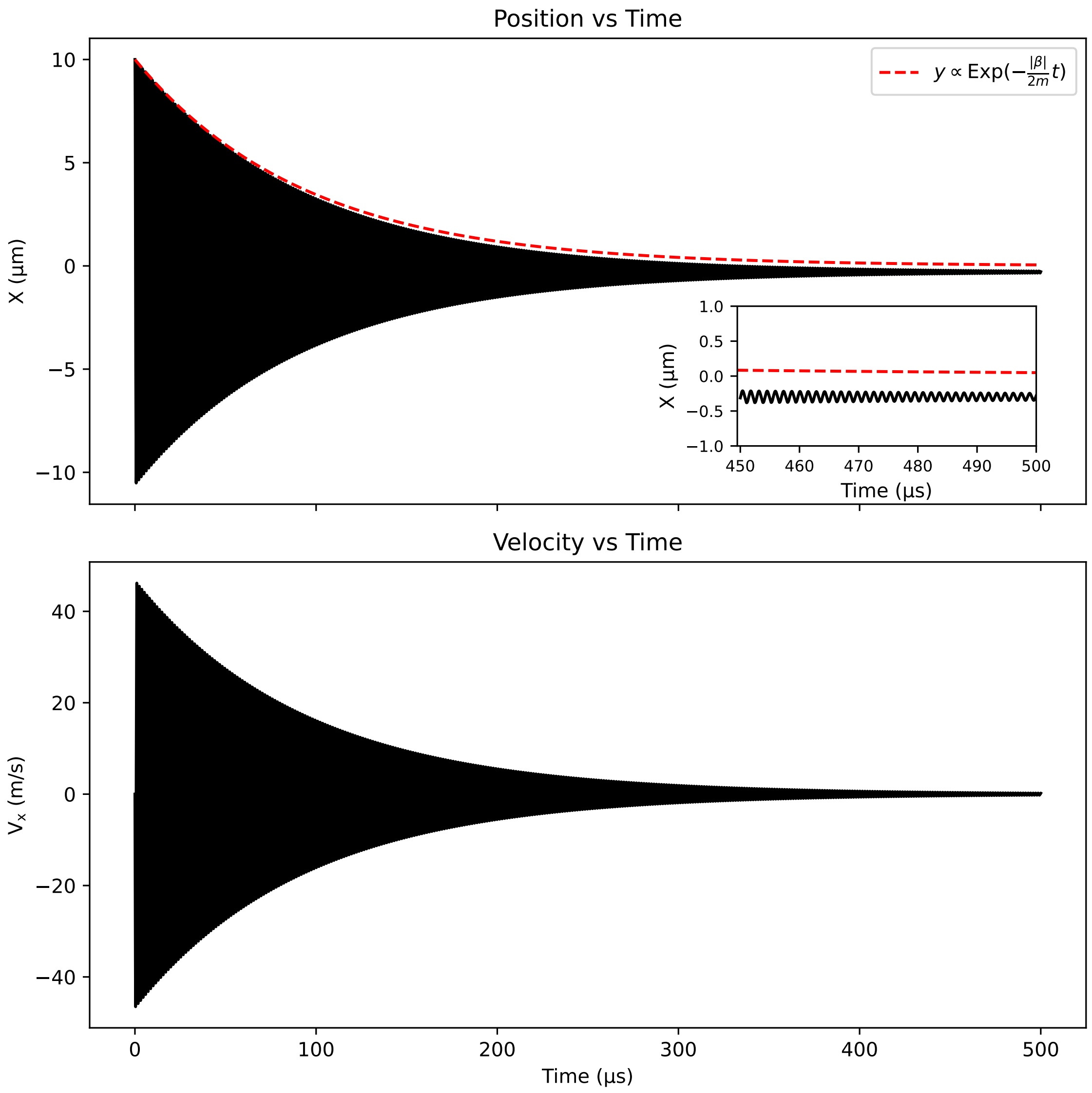


Figure 2.5: The analytic curve is generated using the same cooling force as in Fig. 2.4. The inset plot shows that the discrepancy between the curve and the resultant data is simply due to the F_0 term and not indicative of an integration error.

The slight discrepancy seen in Fig. 2.5 in curve and simulation at high t_b is likely due to the F_0 term, which was not included in the curve generation. While at this point it seems to be safe to say that we have properly encoded the cooling force into our simulations, we will include the damped-driven oscillator as well.

By applying a uniform electric field that is oscillating on-resonance with the ion's secular frequency our system becomes a textbook damped, driven harmonic oscillator. The steady-state amplitude for such a system is given by

$$A = \frac{F_d}{|\beta|\omega_r}. \tag{2.27}$$

We again see pretty close alignment between our expectations and the simulation output, with the slight downward shift again due to the F_0 cooling force term. We can now move on, confident in our understanding of our cooling schemes.

2.3.3 Multi-species Crystals

Benchmarking the geometry of frozen, multi-species ('mixed') crystals is very similar to what we did in 2.3.1, except that it is often simpler to take an energy minimization approach. Professor Hanneke has a pre-existing Mathematica notebook that generates images of minimum energy positions for a given crystal size and composition. This notebook has been qualitatively benchmarked against real world crystal images[8] and functions by minimizing the system's energy through numerical calculations. There are some considerations to be made for which system parameters we want to make this comparison on. Interesting and useful crystal structures often involve a moderate or large number of O_2^+ ions, so we can get a complete view of the Coulomb 'shell' they create around the Be^+ core. We also want enough Be^+s so that the crystal has a sizeable core, but it is beneficial to not have so many that the Be^+s lose their chain form, which is easier to analyze. In addition, tighter axial confinement (and slightly looser radial confinement) exaggerates this 'shell' structure. Given these considerations we decided to study a 10 Be^+, 10 O_2^+ crystal with trap paramaters of:

[8]A student working in the Hanneke lab may refer to Professor Hanneke's March 8, 2023 Lablog post to see this comparison.

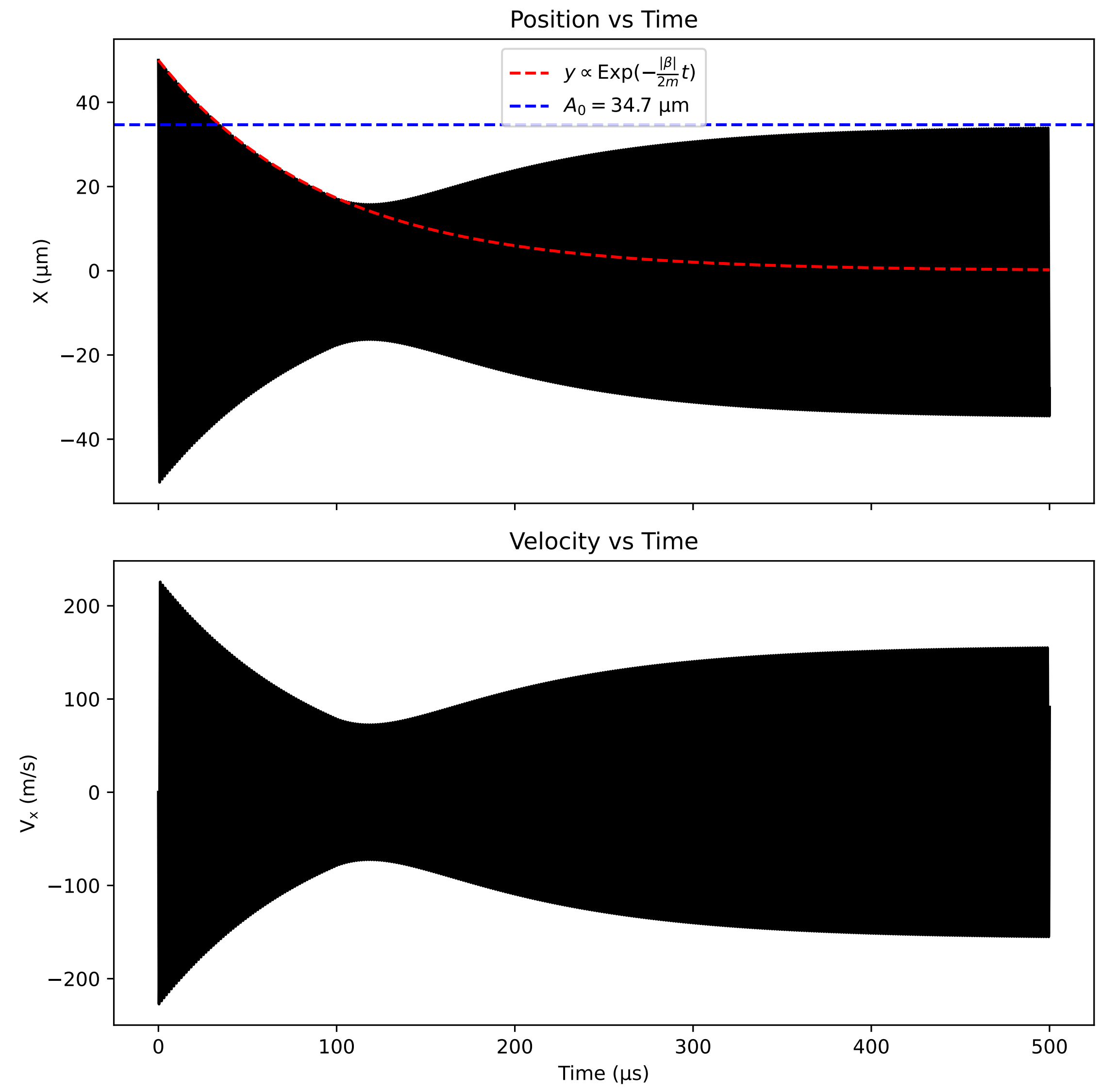


Figure 2.6: The horizontal blue line indicates the steady-state amplitude calculated in Eq. 2.27. While hardly visible in this figure, the discrepancy between the calculated steady state amplitude and the final oscillation amplitude due to the F_0 term in our laser cooling function persists.

$r_0 = 1.25 \cdot 10^{-3}$ m, $z_0 = 1.5 \cdot 10^{-3}$ m, $\kappa = 0.17$, $\Omega/2\pi = 7.3 \cdot 10^6$ Hz, $U = 31$ V, and $V = .7$ V. These yielded the 4 secular frequencies of $\omega_{x,y\ \mathrm{Be}^+} = 507$ kHz, $\omega_{z\ \mathrm{Be}^+} = 169$ kHz, $\omega_{x,y\mathrm{O}_2^+} = 132$ kHz, and $\omega_{z\mathrm{O}_2^+} = 90$ kHz.

The outputs of these two methods are shown in Fig. 2.7. Note first the general similarities in structure - both methods produce a nearly linear chain of central Be$^+$

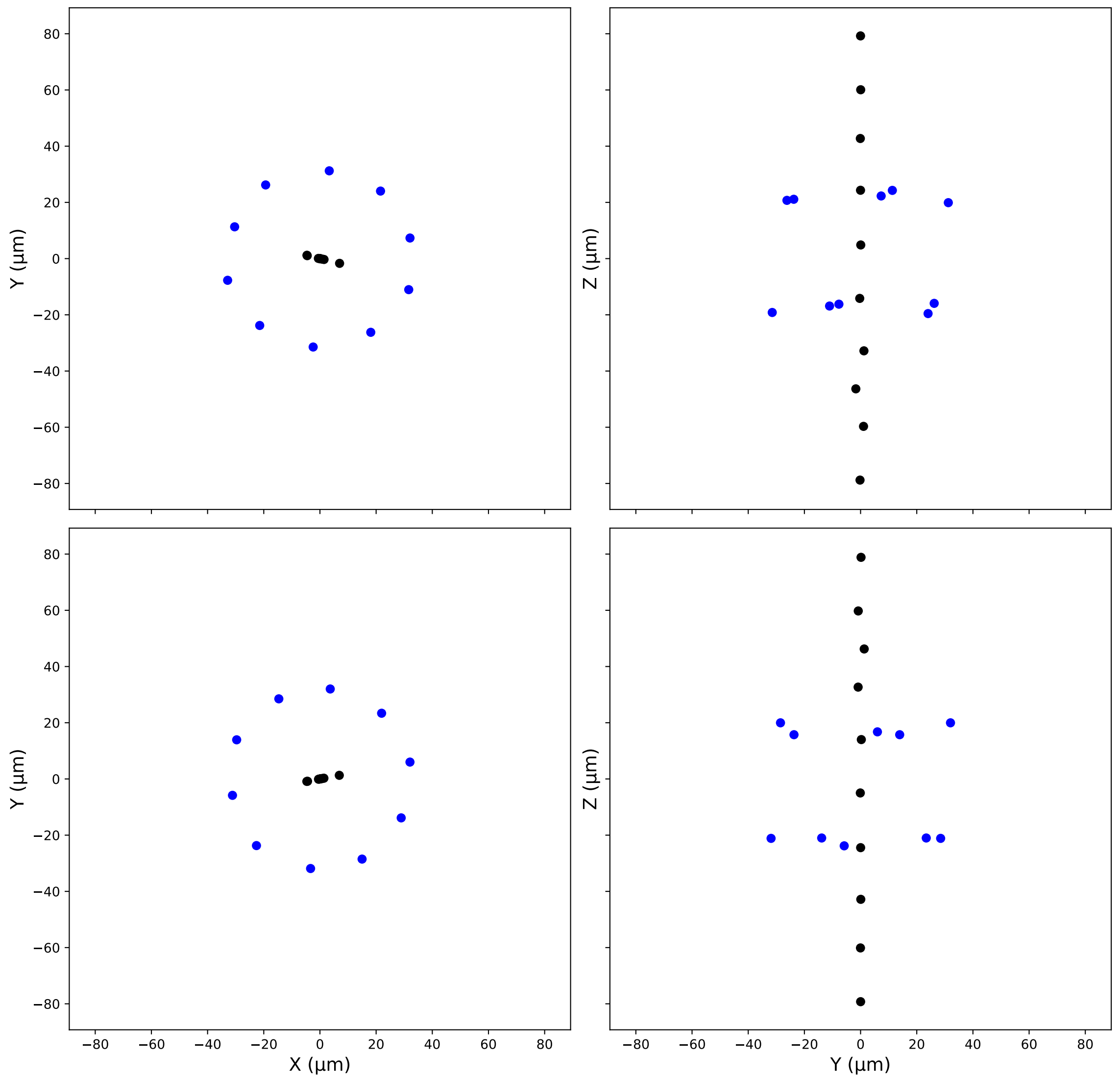


Figure 2.7: The top two plots show the results of the numerical minimization technique, and the bottom two come from our (py)LIon simulations. The species are indicated by color in both, with the outer, blue points indicating O_2^+ ions and the inner, black points indication Be^+. Note the general congruence in crystal spacing, radius, and length as well as the slight deviation in structure details, which originates from the many similar low-energy configurations that are extremely close in structure to the true low-energy state.

surrounded by two rings of O_2^+. The radii of these rings are both about 40 μm and the axial distance between the rings are about 40 μm as well. There are some minor differences in the relative positions of some of the ions. This is likely due to the fact that even though both methods find a state near the minimum energy state, there are many crystal states with energy very close to the true global minimum. Because the (py)LIon simulations start with a high-energy ion cloud that it then cools down, it is very likely that this system got ‘stuck’ in a local minimum energy state that is just slightly different from the true energy minimum.

Chapter 3

Quantum Logic Ion Control Simulator

In this chapter we will discuss, in more detail, how we transplant the physics of Ch 2. into the simulation environment. This will include both the low level computational tools we use and the overall structure of the QLICS program. This chapter is meant to provide an understanding of the internal structure of the program and highlight its capabilities and limits. It is not meant to be a user manual, that information will be found in the repository documentation, [15].

3.1 Simulation Philosophy

Conceptually, molecular[1] dynamics (MD) simulation (the framework of numerical simulation we used) is very simple: it is the repeated process of calculating the net force felt on bodies, updating their momentum using $\vec{F} = m\frac{d\vec{v}}{dt}$, and evolving their positions over a set timestep duration. In other words, we are using numerical

[1]While the term "molecular" might suggest a chemical component to this simulation approach, it actually has more to do with the context MD simulations are commonly used in (biophysics, chemistry, etc). MD simulation is nothing more than the iterative application of Newton's 2nd Law and our simulation results are therefore confined to representing phenomena that arise from classical dynamics.

methods to find solutions to differential equations by evolving step-by-step. Naturally, this discretization process will incur some error, and there are different ways of doing this integration process which produce different types of error (for different systems). In the following section, I will introduce a couple of the core principals of this subject and explain the pros and cons of various integration methods.

3.1.1 Integration Methods

There are several methods different methods one can use for applying this prescription, with different benefits and drawbacks, as well as different positions in the (general, but as we will see, not always consistent) trade-off between computational complexity and error accumulation rate, [24]. In other words, different approaches have different error scaling with the timestep size, Δt, and decreasing Δt, of course, has an associated computational cost. For our system, we have exclusively used the Verlet integration[2] methods in the simulations reported in this work (besides Fig. 3.1).

Euler Method

To explain why we believe this algorithm is best suited for our system, we will contrast it with two other common integration methods: the basic Euler method, and the fourth-order Runge-Kutta method (the primary alternative we would realistically use). Our basic problem is that of evolving the function defining the positions of each ion forward in time. Since the spatial dimensions are clearly independent in a general 3D harmonic oscillator, for this analysis we will take the differential equations defining each ions' position to be separable.[3] Therefore, we can consider just one dimension for simplicity. Let $x(t)$ be the position function of a body over time, with time-evolution

[2]More specifically, we use the velocity-Verlet algorithm.

[3]In our actual simulations, we can have cross dimensional terms in the modulating electric field. However, that coupling won't change the conclusions reported here so we ignore this complication.

defined by $t = t_0 + \Delta t$. Then by simply applying the Taylor series:

$$x(t) = x(t_0) + \dot{x}(t_0)\Delta t + O(\Delta t^2) \tag{3.1}$$

and by discounting the second-order terms we have the Euler method for evolving functions with a single calculation.[4] Because $x(t_0)$ is either an initial condition or a previously calculated value, the single calculation in the second term means this method provides excellent time scaling. Unfortunately, it has been shown that this method of integration has an unbounded error in general for periodic systems, causing the energy of the system to not be conserved [25]. Often, this phenomenon happens surprisingly quickly, Fig. 3.1. This is a function of it not being a symplectic integrator (an integrator that preserves the phase space volume in conservative systems) [24, 25]. This makes it essentially useless for problems that involve oscillations about an energy minimum over long periods of time (such as an ion oscillating about a minimum in the potential caused by a trap and its neighbors) despite the efficient timestep scaling.

Runge-Kutta Method

Looking at Eq. 3.1, it is clear that the primary candidate for improvement of the method is going to be in the second term (the discounting of the higher-order terms can be justified with reasonably small Δt). The Runge-Kutta integration method attempts to solve this issue by expanding the second term into multiple samples. While the simplest of these solutions would be the “midpoint method”, a method where $\dot{x}$ is evaluated at $t_0 + \Delta t/2$ instead of after the full timestep duration, the most common method of this class used in MD simulation is the 4th order Runge-Kutta

[4]Note that here we only provide the position equation. However, since we can’t query $\dot{x}$ directly (we are trying to solve a second-order differential equation), we must also step through the velocities by the same technique. While we don’t give this simple algorithm explicitly, in Appx. C, you will see this concept in the provided RK4 and VV algorithms, and it can be easily transferred to the Euler method.

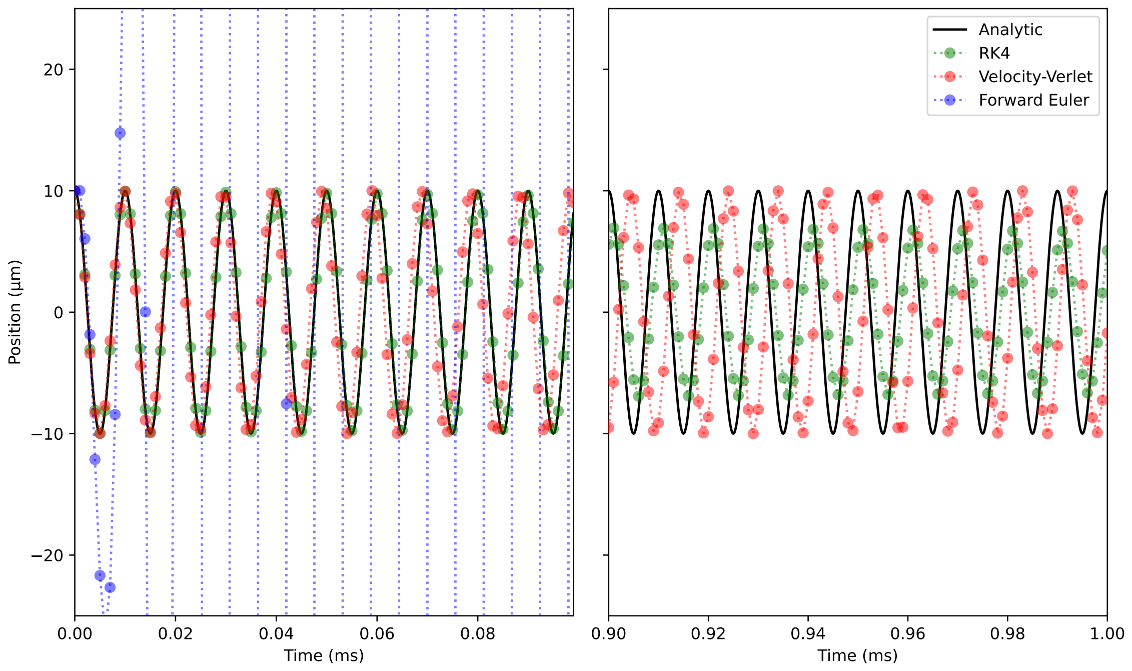


Figure 3.1: Comparison of different integration methods estimating a harmonic oscillator with a frequency of about 100 kHz over 1 ms. We zoom in on the first and last tenth of the time range to show the short-term and long-term behavior of the integrators (leaving the Euler method out of the final tenth since it has gotten absurdly bad at this point). Note that while the fourth order Runge-Kutta method is truer to the initial conditions of the system in the short run, the Velocity-Verlet algorithm preserves the system's total energy in the long run. Timestep size was significantly increased to accentuate the difference between the two methods.

method, RK4 [26]. RK4 simply samples the slope of the function to be simulated at 4 different positions, see Appx. C for a general definition and its application to a physical system. RK4 has been shown to be capable of handling both SHM and more complicated oscillators [25]. However, it is not symplectic and so does not provide energy conservation, even with adaptive step sizing. Therefore, over long periods of time, it will diverge from the analytic solution for a system defined by a time-independent Hamiltonian [24]. In addition, and perhaps more significantly, because it requires 4 calculations per timestep, it provides worse timescaling.

Velocity-Verlet Method

Like the Euler method, the equations defining the velocity-Verlet algorithm will be immediately recognizable:

$$x(t) = x(t_0) + v(t_0)\Delta t + a(t_0)\frac{\Delta t^2}{2} \tag{3.2}$$

$$v(t) = v(t_0) + [a(t_0) + a(t)]\frac{\Delta t}{2}. \tag{3.3}$$

By simply using Eq. 3.2 to find $x(t)$, the summed forces at $x(t)$ to find $a(t)$, and plugging into Eq. 3.3, we can evolve our system forward [27]. For an explicit algorithm of this process, see Appx. C. It has been shown (and is plausible, given the direct treatment of the second derivative) that this integration method is symplectic [25]. Note that this only means that energy is conserved on average, not at every timestep (in truth, the total energy calculated by the VV method oscillates over time). In addition, the 2 calculations keep our time scaling better than RK4. However, the velocity-Verlet actually has worse error than RK4 in the short term, as seen in Fig. 3.1. In short, the choice between velocity-Verlet and Runge-Kutta boils down to our assessment that this is an energy conservation problem, not an initial value problem. For us, energy conservation (and time scaling) is more important than short-term accuracy to the initial conditions, since they are unknown due to the thermal distribution of the ions in the actual experiment.

3.1.2 QLICS as a Curated Subset of LAMMPS' Capabilities

The program we use to conduct our integrations is LAMMPS (Large-scale Atomic / Molecular Massively Parallel Simulator), [26]. While the LAMMPS software is often used in material science and biochemical settings, QLICS really only uses LAMMPS for its time integration, Coulomb and electric field force calculation, and kinematic output (as opposed to the more advanced features necessary when using LAMMPS to model more complicated systems). We have already discussed the different integration options LAMMPS offers, but at the time of this writing QLICS only supports the use of the velocity-Verlet algorithm. One important point of note is that LAMMPS does allow for dynamically changing the timestep duration, and this is a feature perserved in QLICS. Each timestep, QLICS has to ability to use an explicitly defined force to apply laser cooling to the system. In addition, when we are using the pseudopotential approximation, QLICS, via (py)LIon, defines the trap as an explicit linear restoring force (i.e., a Hooke's law force) with a characteristic frequency that is the secular frequency.[5] Finally, we can provide LAMMPS with electric fields of arbitrary space and time dependence (although QLICS limits this to second-order space dependence and sinusoidal time dependence). LAMMPS superimposes the electric fields internally and finds the force on a per-ion basis using $\vec{F} = q\vec{E}$. Finally, although LAMMPS' *thermo output* has capabilities much beyond what is accessed with QLICS, we only use its per-atom position and velocity output. LAMMPS has the capabilities to mitigate inter-ion force calculations scaling quadratically with the number of bodies

[5]It should be noted that since this frequency is dependent on the mass of the ion, we require the (py)LIon method to execute the pseudopotential calculation for each species. This is because we have observed that, when setting the `pseudo` parameter of the `pylion.functions.linearpaultrap` function to `True`, even when `ions='all'`, the trap will only use the secular frequencies of one species, an issue worth warning the user about.

in the simulation by limiting the pairwise interactions it includes, [26], however we avoid making this simplification because we typically study relatively small crystals ($\approx 10-30$ bodies), and because it doesn't seem to be common practice in these types of simulations [19].

3.1.3 High-Performance Computing

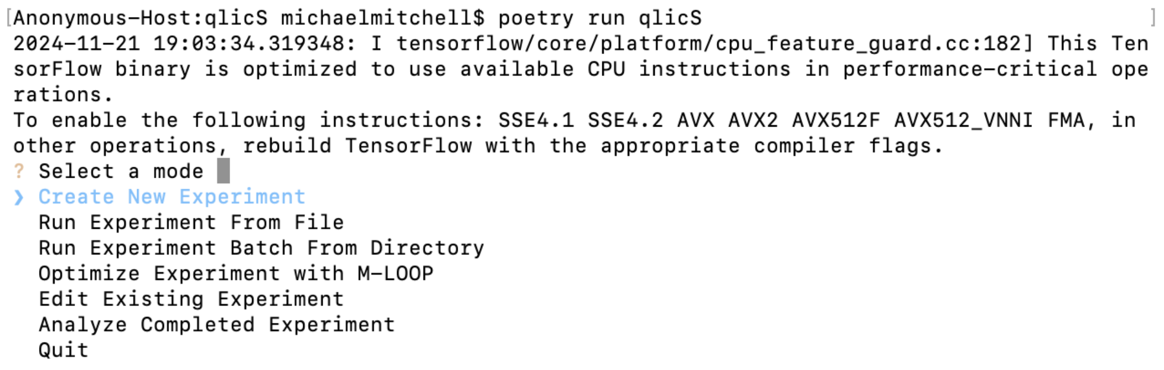
Throughout this project we have been the beneficiaries of the College's new high-performance computing (HPC) system, FrostByte [28]. Before we started running simulations in the QLICS environment, we were able to submit simulations to the system in parallel, massively[6] increasing our simulation capacity. However, once we started focusing on the online optimization of our systems, we realized that we could not simply use the SLURM task manager to distribute tasks across nodes, since the input parameters for different simulations now required the output of the previous one. A major future development goal for QLICS will be figuring out how to reincorporate parallel processing into the workflow, for both optimization and batch simulation routines. This may be done at the LAMMPS level, the 'batch'[7] (that is, the per variable step in a variable sweep) level, or the optimization level. While this has limited the parallelization of the machine-learning driven optimizations done in QLICS, we have conducted large scale simulations using pure (py)LIon scripts where we have been able to see significant (order of magnitude) reduction in total simulation time. An example of data generated by this process is that of Fig. 4.2.

[6]At times we were able to run more than 10 simulations in parallel. This brought experiments that we would project to takenearly a month on a single computer down to a few days on the HPC system.

[7]I am using 'batch' in a similar but different sense in the following section.

3.2 Interlude: Context and Key Definitions

I want to briefly pause to highlight how QLICS itself fits into the bigger picture of our numerical integration discussion. Throughout this work, we refer to 'experiments' and the fact that QLICS takes experiment configuration files as input. While we discuss the details of such files' content in the following section, it may be helpful to first formally define 'experiment' as a list of instructions defining the system we are integrating, how we integrate ('evolve') it, and how we process the simulation results (integrating Eq. 2.19 over time) to create a mock experimental output. This three-step process can thought of in relation to the LAMMPS integrator directly: LAMMPS code generation, LAMMPS simulation execution, and LAMMPS per-atom position and velocity vector output at each experiment timestep. Note that since LAMMPS *thermo output* provides only atom position and velocity vectors over timesteps, we can either convert these values into a scattering rate with Eq. 2.19, or a mean RMS velocity for the system. Of course, we can visualize the per-atom position and momentum data directly as well, but we try to keep our reliance on this approach to a minimum in our results, since we cannot duplicate this measurement in the real system. Importantly, the times at which we perform the first type of integration are defined by the detection sequence and are typically time frames much shorter than the full dynamics simulation. We then output these photon counts with timeframe tags as well as sweep variable identifiers (for instance, the frequency of the modulating field after which we performed the detection), Appx. B. The purpose of this interjection has been to just orient the reader and emphasize the *central*, most basic functionality of QLICS. There are many important features in the program that exist 'around' this core function, as discussed below.



```
[Anonymous-Host:qlicS michaelmitchell$ poetry run qlicS                          ]
2024-11-21 19:03:34.319348: I tensorflow/core/platform/cpu_feature_guard.cc:182] This Ten
sorFlow binary is optimized to use available CPU instructions in performance-critical ope
rations.
To enable the following instructions: SSE4.1 SSE4.2 AVX AVX2 AVX512F AVX512_VNNI FMA, in
other operations, rebuild TensorFlow with the appropriate compiler flags.
? Select a mode
❯ Create New Experiment
  Run Experiment From File
  Run Experiment Batch From Directory
  Optimize Experiment with M-LOOP
  Edit Existing Experiment
  Analyze Completed Experiment
  Quit
```

Figure 3.2: QLICS launch and main menu. We will only discuss the ‘Run Experiment From File’, ‘Run Experiment Batch From Directory’, ‘Optimize Experiment with M-LOOP’, and ‘Analyze Completed Experiment’ here as they are the most important and the only ones needed to understand the program’s functionality.

3.3 QLICS

While we have expanded beyond direct use of (py)LIon, the period of using that package as our simulation launchpad set the groundwork for the Python-LAMMPS work flow that we carry over into QLICS, namely, the process of turning our Python-defined system into LAMMPS code via a Jinja template [19]. The primary need that (py)LIon was unable to simply fulfill for us was the ability to easily and automatically reproduce experiments with slightly shifted parameters, to allow us to explore the experimental parameter space efficiently. In this chapter, I will start the discussion of the program at the highest level of abstraction before descending down and discussing the innards of the subroutines we introduce. I will stop a level of abstraction above explicit discussion of the classes and functions used in the code, information on this level will be available in our docs, [15].

3.3.1 Modes of Use

At the highest level, QLICS can be used in 1 of 4 modes, offered in the command-line interface (CLI) upon the initial start of the program, Fig. 3.2. These 4 modes and their functionality are outlined in Fig. 3.3. The most fundamental of these is the 'Run from File' mode.

Running Individual Experiments

As seen in Fig. 3.3 and as one might guess, this 'Run Experiment From File' mode itself can be treated as a subroutine for both the 'Run from Batch' mode and the 'M-LOOP Optimization' mode. At the heart of the mode is the 'Execute Experiment' subroutine, which we will deconstruct in Sec. 3.3.2. This mode takes an INI[8] file that defines the experiment and simulation specifications as its input. Then, if a detection laser is defined, returns a CSV file that replicates the QLIC output (photon count vs modulating frequency, see Appx. B). In all simulations, a 'dump' TXT file is generated from the LAMMPS 'thermo' output (via (py)LIon), see Ref. [19, 26] and their associated documentation. This simple workflow (shown in Fig. 3.3) comprises the 'core functionality' of QLICS.

Running Sets of Experiments

The 'Run Experiment Batch From Directory' mode is a very simple application of the 'Run from File' subroutine. This mode takes a directory of INI files as input and iterates through them[9] running 'Run from File' for each. It should be noted that the output from this mode is not aggregated in anyway and so a unique output directory is created for each INI in the input directory. This mode is most useful for manually investigating the changes in the results of experiments with changes to the initial characteristics of those experiments. For example, this mode could be used to submit a series of experiments (say a frequency sweep tickle experiment) for many different

[8]INI files are plain text configuration files developed for some of the earliest MS-DOS systems. Although more familiar formats like YAML or JSON offer more modern features, INI's single level of nesting (the often-cited drawback) is sufficient for our needs, and its syntax provides, in my opinion, a visually straightforward style for defining experimental sequences.

[9]Serially, a natural improvement for this program will be the optionality for parallel execution in HPC systems.

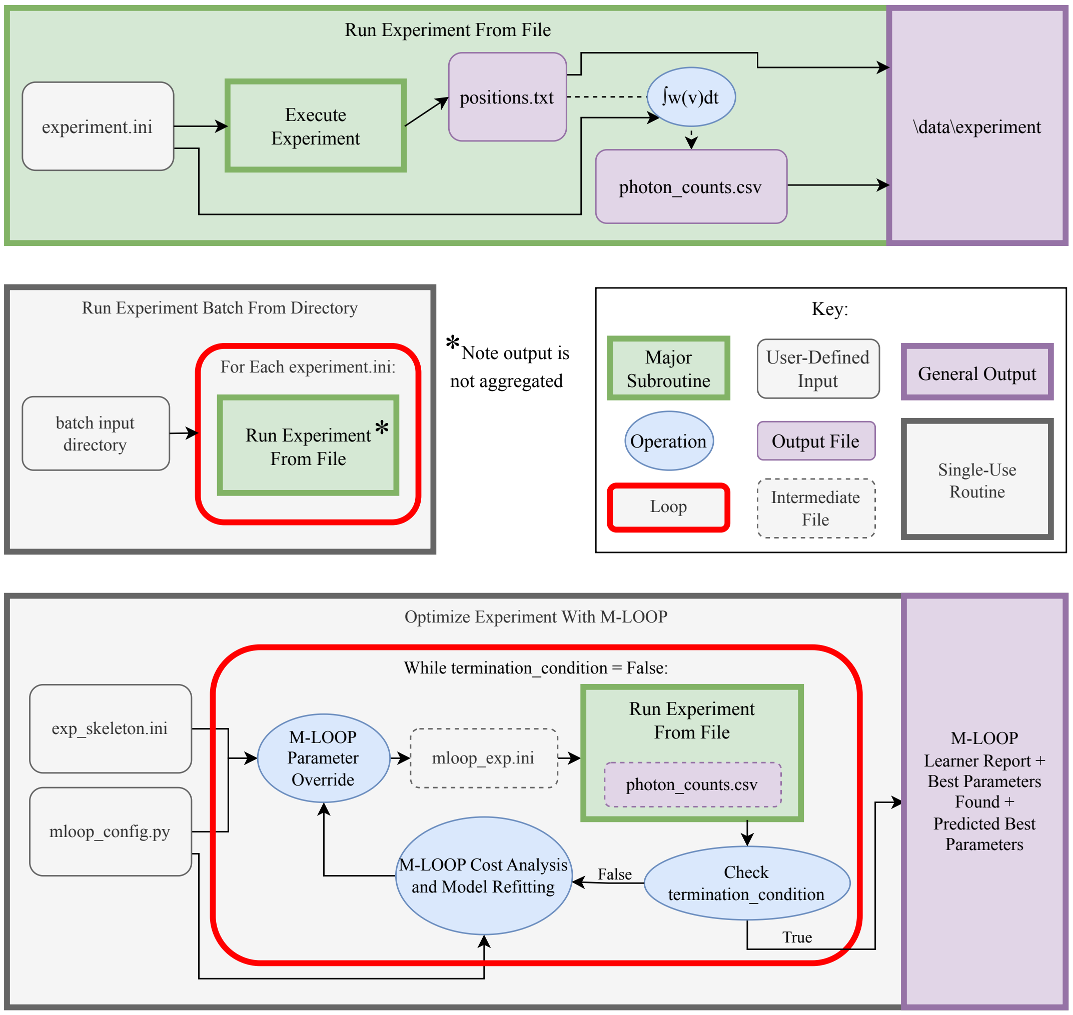


Figure 3.3: Block diagrams of the three major simulation modes of QLICS. $\int w(v)\,dt$ refers to the scattering rate defined by Eq. 2.19 (which, of course, has a timestep-wise time dependence). The purpose of this diagram is to highlight the logical relationship between major aspects to aid the high-level discussion of this chapter. To this end I elected to sacrifice strict accuracy for clarity by approximating and omitting several programmatically important relationships that would obscure this discussion. A reader interested in getting these details exactly will find them in the QLICS documentation [15].

crystal compositions, as was done in Fig. 4.3. This mode is used throughout the results of Ch. 4.

Optimizing Experiments

In many ways the 'Optimize Experiment With M-LOOP' mode can just be thought of as an automated version of 'Run from Batch'. Here instead of manually defining each experiment with explicit INI files, we provide a 'template' INI file. This sets the general structure of the experiment we wish to optimize. In addition, we supply another configuration file, specifically for M-LOOP. This python (.py) file serves two purposes. First, it defines the cost function that the machine learner program aims to optimize, as well as how we measure the uncertainty from each experiment. The definition of the cost function must be carefully considered so that the learner actually optimizes a physically significant aspect of the experiment (as opposed to trivially optimizing the cost function via some loophole) an example of which can be found in Appx. B and more detailed discussion on setting up these optimizations will be available in the QLICS documentation [15]. Second, the .py file configures the learner itself, identifying the parameters it is allowed to control, the range of the parameter space we allow it to explore, the type[10] of learning we want to use, and the termination conditions. In addition to the typical CSV and TXT outputs that accompany each experiment, a successfully complete optimization run will generate 'controller and learner archives' as explained in the M-LOOP docs [29]. These archives can easily be visualized using `mloop.visualizations.show_all_default_visualizations_from_archive()` with the archive files as arguments described in the M-LOOP documentation, see Fig. 4.11 or [29] for examples.

[10]See sec. 3.4 and [16] for more discussion on the pros and cons of various learners.

Analyzing Completed Experiments

Disconnected from the previous three modes in that it doesn't execute a simulation, the 'Analyze completed Experiment' mode simply serves as a general purpose tool for plotting the outputs of an experiment. While it is likely that the user will want to customize their own data visualization, this mode provides a quick and dirty way of checking that *individual* simulations are behaving as expected. This mode can process either the CSV or TXT output files discussed earlier. If a CSV file is taken as input, it will simply plot a photon count vs. frequency plot, akin to the QLIC output. If a TXT file is taken as input, the user has the option to select either 'Whole', 'Individual', or 'Crystal Image' via the CLI dialogue. If 'Whole' is selected, the RMS velocity of the entire crystal structure[11] is plotted as a function of time. If 'Individual' is selected, per-ion plots of either position vs time and velocity vs time (or both) are generated. Finally, if 'Crystal Image' is selected, axial and radial flat projections of the crystal (at a defined time) are generated (as well as a 3D matplotlib visualization should the user request). The user is given the ability to define a separation index for a dual-species crystal, so that ions of different species are color-coded. While there are obviously many features that could be easily added to this mode, I have kept it very bare-bones in the interest of development time. This is an area ripe for improvement in future iterations of the program. As will be discussed in Sec. 3.3.1 all of the visualization output is stored in the 'data' folder. The sub-folders contain the prefixes 'ph_count' and 'analysis' as appropriate (the former being used for analysis of CSVs and the latter for TXTs).

More Information on Input and Output

While we have explained the use of INI files in the context of the program, we have not yet discussed their internal structure. Exact syntax documentation can be found in

[11]An easy area for improvement for this visualization would be the ability to plot RMS by species.

Appx. B and a more detailed discussion will be available in the QLICS documentation [15]. Here we will just talk about their broad principles in preparation for Sec. 3.3.2.

While the predominate purpose of the INI files is to define the simulation itself, it is also used as a sort of memory storage for QLICS in the LAMMPS simulation creation process,[12] a feature worth mentioning, but that does not affect the user.

In the INI file, we define the 'universe' as two sections, one simply listing the universal constants, and another initializing our ion types, [15]. Ion-type initialization includes defining the species name, charge, mass, and transition of interest (including frequency, natural linewidth, and saturation). We next initialize a 'sim_parameters' section, which is concerned with broad characteristics of our simulation (evolution sequence and step size, boundary style and how we handle lost ions etc.) [15]. Very similar to the 'sim_parameters' section is the 'detection' section (related to, but not to be confused with, the 'scattering_laser' object), which defines our detector size and location, as well as the detection sequence.

After the mandatory initializations, the user may introduce 'object' initializations. These are INI sections that represent 'devices' in our experiment, namely, cooling lasers, scattering lasers, traps, ion clouds, modulating electric fields, or the important iteration object ('iter'). With each type of object, a relevant set of key-value pairs must be defined [15]. At the end of the INI file is the command list. While the object declarations are just that, declarations, the command list is where these predefined objects are inserted into the simulation program. This process is highlighted in Fig. 3.4. The QLICS program is different than perhaps a more typical and flexible program in that, while objects can be defined and not used, under typical conditions, each object may be referenced only once in the main command list. Because of this, objects may be referenced only by 'type' in the command list; see Fig. 3.4. While objects may not be reused in the overall command list, they can be reused by using the 'iter'

[12]While this doesn't disrupt the function of the program or hurt its usability, it may be better form to split these two tasks into seperate INI files in the future.

object's command list (although again they can't be reused in this sub command list either). The 'iter' object can be thought of basically as a loop, allowing for concise programming of repetitive experiments without compromising the emphasis on 'explicit-ness' we are striving for, see Fig. 3.4.

It should be noted that in addition to appending objects, the command list also takes the `evolve` and `remove_` commands. These define the sequence in which we step forward through our simulation and allow for the explicit removal of objects previously appended. You may notice that the detection sequence is handled differently than the evolve sequence (there is no analogues `detect` command). This is because detection is treated as a purely *passive* activity (in other words, we do not currently incorporate recoil due to resonant beams in the ion dynamics) and therefore the entire detection calculation is done *after* the simulation is completed. Often our detection calculations take a significant amount of time to complete (on par with or more than the time of the simulation itself). Improving the efficiency of our photon count calculation algorithm is another potential area for vast improvement in QLICS' performance.

Finally, it should be noted that QLICS has a hard-coded output system (similar to QLIC). Upon execution of the 'Run from File' subroutine, a directory with the current timestamps as its name is created in the 'data' directory. This directory serves as a kind of working dump location, where the simulation INI, CSV (if detection is used), and TXT files, along with the generated LAMMPS code and LAMMPS .log file are saved for future reference. As mentioned earlier, the analysis directories are also automatically stored in the data folder. While all potentially useful information is saved in 'data', the folder frequently because unwieldy and so it is advised to remove 'data' from the QLICS folder before running high numbers of experiments by either 'Run Experiment from Batch' or 'Optimize Experiment with M-LOOP'. The additional 'Controller' and 'Learner' archives from M-LOOP are stored at the same level as 'data'.

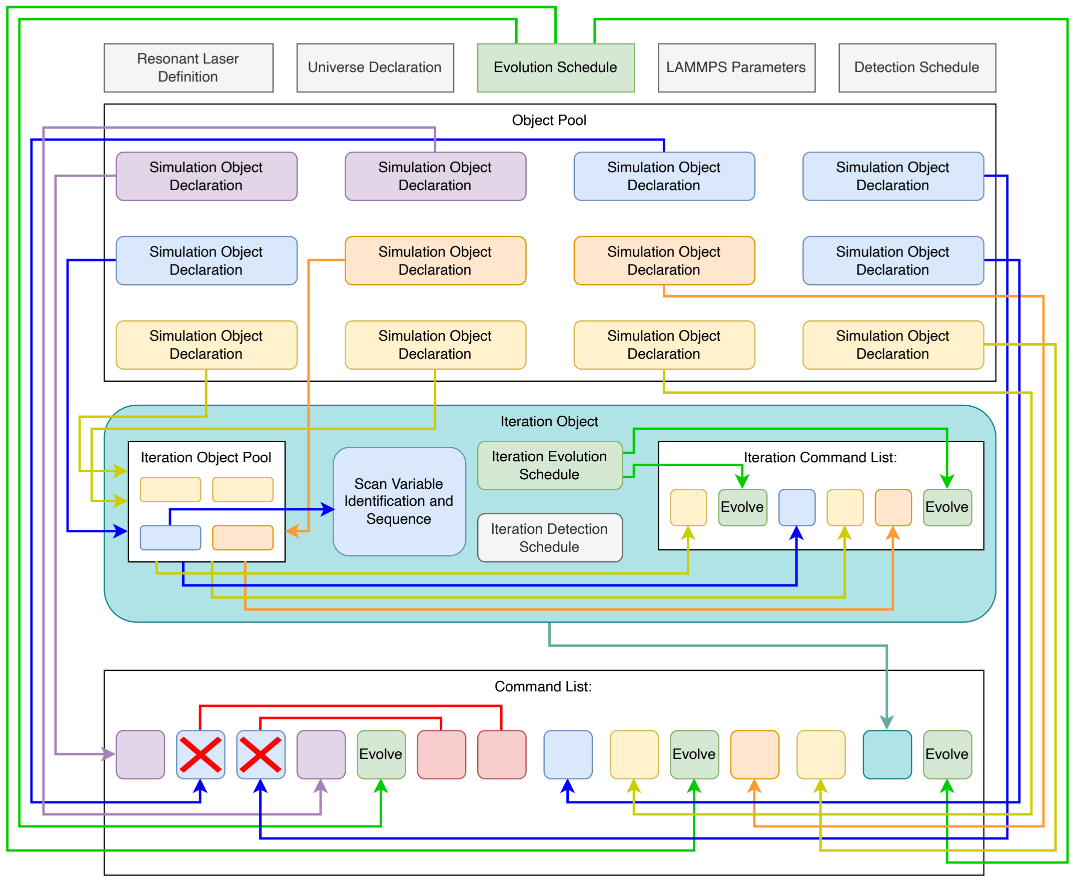


Figure 3.4: Block Diagram depicting how a typical INI file, with iteration object, is structured. Starting from the top row: square rectangles indicate mandatory simulation-wide configurations, color-coded based on accessibility. The main object pool indicates the initial definition of 'ordinary' simulation objects (i.e. traps, ions clouds, modulation fields, cooling lasers, etc). Note that each ordinary simulation object may be explicitly accessed only once. The iteration object acts as a 'mini' experiment in that it access objects from its own pool, an *explicitly-defined* subset of the primary pool and contains its own evolution schedule, detection schedule, and sub-command list. The iteration loop is defined by the iteration variable, a variable of an ordinary object picked out from its own pool. Finally the main command list appends ordinary objects by referencing their *type*, with order defined within the objects themselves, indicated by color coded commands (type-order is represented here by position in the main pool going from top left to bottom right). The red commands represent the explicit removal command.

3.3.2 The 'Execute Experiment' Subroutine

So far we have discussed at the level of the 'Run Experiment by File' subroutine. However, at the heart of this routine (and by extension every aspect of QLICS besides the 'Analyze Completed Experiment' mode) is the 'execute experiment' subroutine as seen in Fig. 3.3. We will not discuss this in as great of detail, but will just go over a few points of interest. This section should be thought of as a bridge between the discussion of the preceding section and the function-by-function breakdown in the documentation [15].

Processing and Rewriting INI

After receiving the input INI, QLICS processes its information into memory and rewrites the INI in the now initialized dump directory. This seemingly pointless operation is done for several reasons. First, in the processing stage, QLICS starts with the experimental sequence and grabs the previously defined objects as needed. This means that objects that are defined, but are never accessed are dropped from memory and not stored in the 'dumped' INI, preventing input file bloat since we commonly copy and modify INI files instead of writing new ones from scratch. Second, this allows QLICS to use the new INI file as a modifiable document for the storage of what essentially act as 'public' variables. While there may be drawbacks to this atypical flow of information, it significantly helps keeping one very important public variable, our timestep duration, uniform across the program. Since so many of the equations relating to our objects depend on accessing a consistent value, this technique seems to be acceptable. Finally, when in our 'Optimize Experiment with M-LOOP' mode, we override the appropriate values in the 'copied' INI file with the values the M-LOOP learner wants to test at this point. To make it very clear, under no circumstances is the original input INI file modified because of this 'copying and editing' scheme.

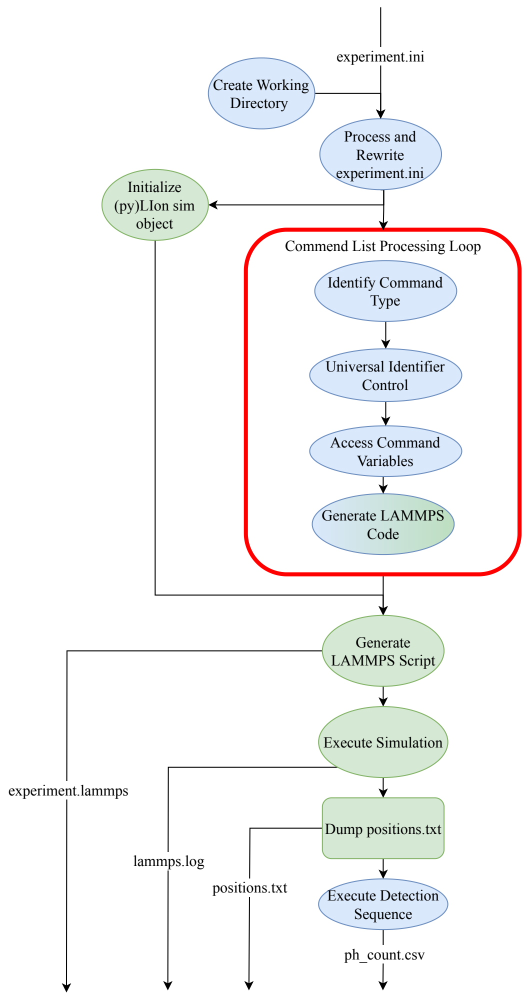


Figure 3.5: Inside of the ‘Execute Experiment’ subroutine introduced in Fig. 3.3. Operations managed by (py)LIon are demarcated green.

Experiment Creation

After the INI is copied and we complete a couple of simulation preparation tasks, we enter the 'Command List Processing Loop', Fig. 3.5. This routine is the actual process of turning our INI file objects into LAMMPS code. QLICS goes down the experimental sequence, accessing each command and conducting the appropriate operation to append it to the simulation. For example, if the command indicates the appending of an ion cloud, the program will access the appropriate object information (in this case, ion type, cloud radius etc.), and generate the appropriate LAMMPS code. When the (py)LIon-defined functions are suitable for generating LAMMPS code, those methods are utilized, otherwise QLICS generates LAMMPS code by its own means. This is also where the 'iter object' is expanded, with this appending sequence being called across their command lists with the appropriate variable overrides for each iteration. Similarly to experiment objects, removal and evolve commands are added in by appending simple ((py)LIon-defined) LAMMPS commands to the (py)LIon sim class [19]. After this process is complete, (py)LIon submits the simulation and LAMMPS runs it.

Scattering Data CSV

An important aspect of our output already touched upon is that the CSV output is actually generated *off of* the TXT output. In other words, the simulation is complete and the TXT file is created before QLICS, accesses the scattering laser object, and calculates the number of scattered photons detected. This output is then what M-LOOP uses, not the full TXT output.[13]

[13]Though we have discussed loosening this restriction in future iterations of QLICS.

3.4 Machine Learning for Experiment Optimization

While we use the machine learning package out of the box, without any additional edits, it is still worthwhile briefly discussing the how the program works, and especially how our approach differs from other supervised learning techniques. This understanding informs how we set our termination conditions, as well as how we decide which of the provided learner models to use.

3.4.1 Brief Overview of Machine Learning for Online Optimization

We can cast our discussion of QLICS' structure into an ML format by simply treating each execution of the 'Run Experiment from File' subroutine as a stochastic process taking a set of input parameters and returning a cost that is a function of those parameters (as well as some uncertainty).[14] The model is then simply fitting to a set of correlation lengths connecting each parameter to the cost function [16]. Unlike other forms of supervised learning, where one is trying to fit a model to preexisting data, in online optimization, the model is actively interacting with the source of the data, and is deciding, at each step, which parameters to query the cost at. While the intelligent selection of input parameters is a benefit, the fact that experiments must be completed before the next selection is made greatly hampers the total number of data points we can use to train the model (often less than a hundred is necessary). Fortunately, the potential for extremely fast learning has been shown in the online optimization of some complex physical systems using Gaussian Process (GP) learners [16]. While M-LOOP ships with several learning 'controllers', we generally stick to the neural network or the GP learner controllers, with conventional wisdom and the

[14]Just like the original M-LOOP experiment [16].

M-LOOP documentation suggesting that neural networks are more efficient for high-dimensional parameter spaces due having better scaling with the number of data points to be fitted [29]. While it would be ideal to set our termination condition off of some kind of physically important cost threshold, at this point, we have predominately just been setting time or run number limits to ‘get a feel’ for the parameter space.

Chapter 4

Results from Preliminary Applications of QLICS

While the primary 'outcome' of this thesis is the existence of the QLICS program itself, it is worthwhile to look at the results from some initial applications of the program, relevant to our lab. This will demonstrate the flexibility of the program, show some realistic use-cases, and serve as a starting point for our understanding of this system. Conceptually, I've divided the sections of this chapter into two categories.

In the first category (Secs. 4.1, 4.2), we study relationships that we already had a relatively strong physical intuition for. Therefore, the 'results' of the simulation serve to both confirm our understanding and illuminate some details of the relationships that we need numerical methods to fill in. Because these are not open-ended explorations of the parameter space, we tend not to use the M-LOOP features in these types of studies, and just explicitly define experiments to test the relationships we want to.[1] While this more 'traditional' approach is perhaps less exciting than having a neural network distill unexpected relationships from the system, it demonstrates a

[1]Because it can be a bit confusing, I want to emphasize that the exploration of systems we have some physical intuition for is *not* the same as benchmarking a system that can be solved analytically. Experiments in this first category are still 'explorations' not 'confirmations'.

very practical use of the program for our lab. As we will see, these types of experiments can provide confirmation of and help us to understand the degree to which we can improve our number-resolution in detecting dark ions by using a heavier coolant species. It is worthwhile to confirm this assumed relationship and estimate the expected improvement prior to introducing the new species in the experiment since adding this new species is a non-trivial project (in general, requiring us to open the chamber, design and manufacture a new oven, and introduce appropriate ionization lasers).

The second category (Sec. 4.3) introduces some examples of more open-ended 'experiments' driven by our machine-learning based optimizations. A technique of interest and promise that was distilled by this exploration-focused research is the heat preparation sequence, Sec. 4.3. In Ch. 5, I will introduce other areas where we have conducted initial optimization experiments. Although the results of many of these experiments were not extremely conclusive, there is a lot of room for improvement in their execution, and it is very possible that a more detailed exploration of these corners of the parameter space will yield useful results in the future.

4.1 Number-resolution Study

High number-resolution for the detection of both O_2^+ and O^+ is key to showing that we are driving the dissociative transition we wish to demonstrate. The general goal is to develop a scheme where we can see the signal of O_2^+ decrease step-by-step in congruence with the rise in O^+, thereby providing strong evidence that we are dissociating our state-prepared O_2^+ generated per [11]. Throughout this chapter, we consider the 'signal' of such an experiment to be the difference in photons counted when tickling on-resonant with the ions' secular frequency and the number of photons counted when tickling off-resonantly, often normalized by the off-resonant count value. In terms of experimental pulses, this is often done by modulating off-resonance, detecting, re-cooling, modulating on-resonance, detecting, re-cooling, modulating off-resonance, and detecting again. Here, I call this the ‘one-look detection sequence’, and it is really just an abbreviation of the linear sweep across the frequency range. It has been previously shown that, in some systems, this relationship scales linearly with dark ion number for some regimes [23]. As such, the problem of increased number-resolution is expected to look something like Fig. 4.1. However, we have seen many cases where this scaling has been poor, see Fig. 4.2. In Fig. 4.3, a simulation of the photo-dissociation experiment mentioned above, we see both poor and good number-resolution (in O_2^+ and O^+, respectively). This implies that the masses of our dark and coolant ions are very important to the LCF-MS signal, as we will be discussed below.

4.1.1 Crystal Geometry

It is clear that the geometry of the crystal plays a significant role in the degree of inter-species coupling we observe in our system since the Coulomb interaction (for fixed charges) is only dependent on the distance between bodies. While we will use this concept of average inter-species coupling throughout the chapter, it is most obviously seen in the geometry of the frozen, pre-tickled state of a crystal. Letting n be the number of coolant ions and m being the number of dark ions, we can define a ‘mean inter-species Coulomb interaction parameter’, d as:

$$d = \frac{1}{nm} \sum_{i=n}^{m+n} \sum_{j=0}^{n} \|\vec{r}_{ij}\|^{-2} \tag{4.1}$$

here, of course,

$$\|\vec{r}_{ij}\| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}, \tag{4.2}$$

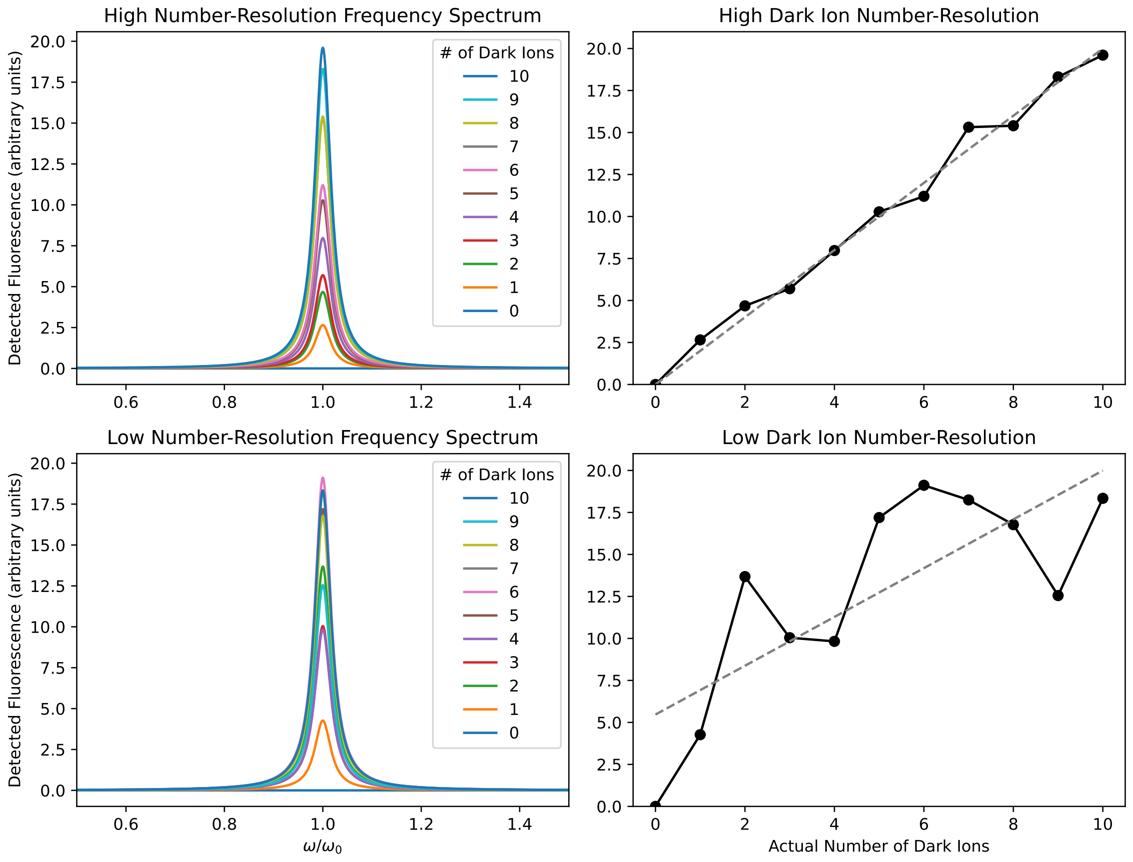


Figure 4.1: A model of what we might expect the fluorescence data of an experiment with high-number-resolution (top plots) vs. an experiment with low-number-resolution (bottom plots). The number-resolution is therefore related to the signal-to-noise ratio of the plots in the right-hand column. Note that while this idealized data looks essentially Lorentzian, our actual data represents a combination of both Lorentzian and Gaussian processes.

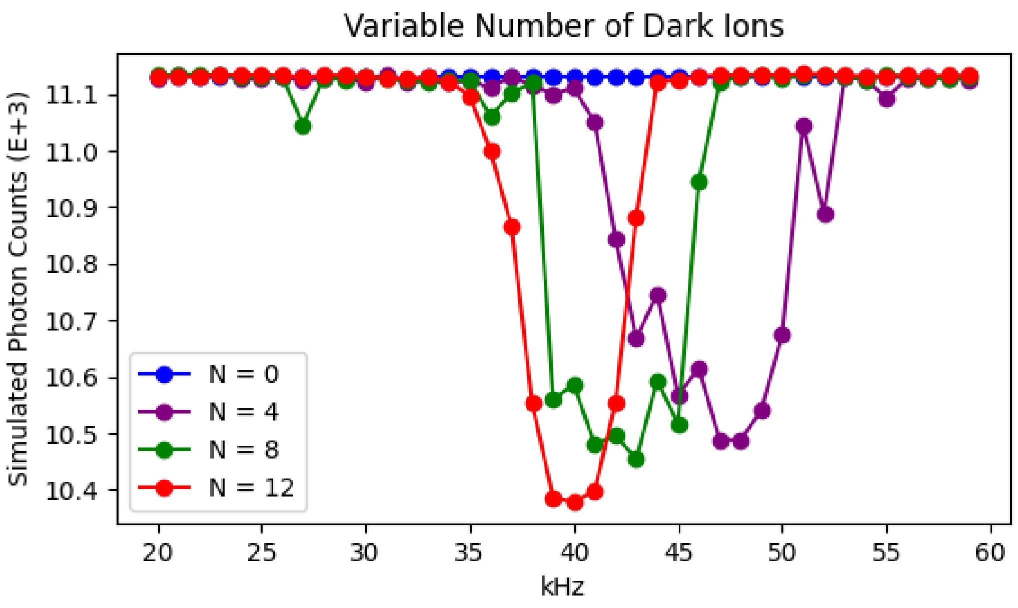


Figure 4.2: An early simulation of the 'tickle' experiment, with clearly poor number-resolution. Unlike the other simulations in this chapter, this experiment was done with an axial tickle. The number-dependent resonance shift is an additional issue limiting number-resolution capabilities, but it seems to be limited to axial modulations. We (and others, [30]) have seen resonance responses create both positive and negative differences from the baseline, although the decrease is more reproducible in simulation and is better understood.

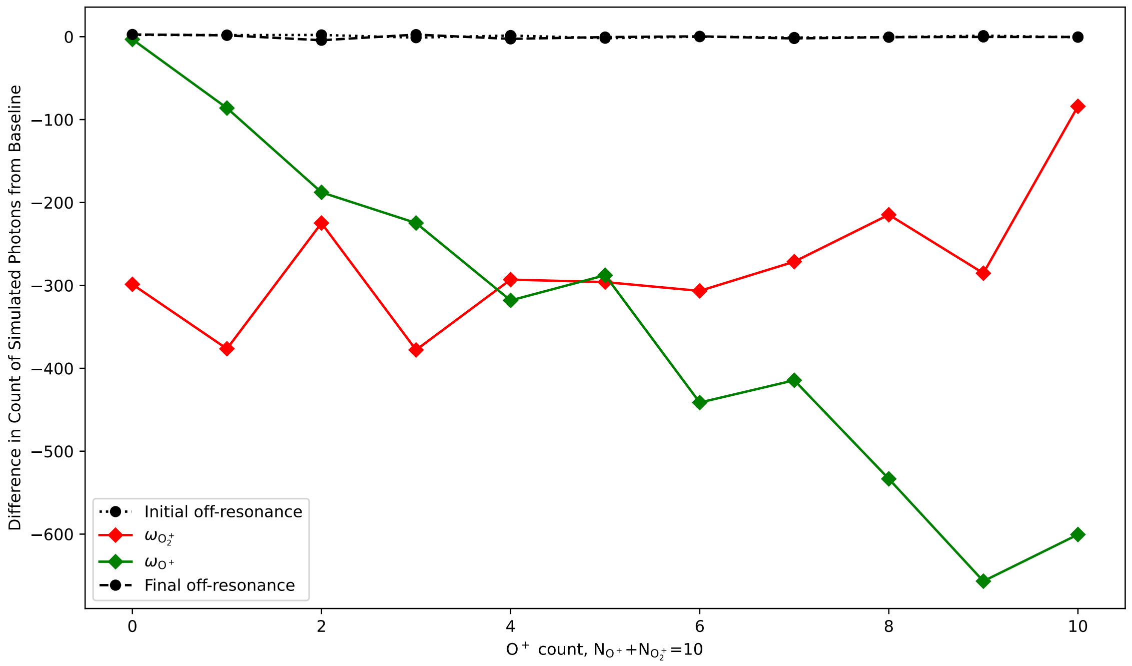


Figure 4.3: Simulations conducted for a crystal of 20 Be^+ and 10 dark ions, with the amount of amount of O^+ and O_2^+ trading off, to model probing the molecular oxygen photo-dissociation. The experimental sequence included a probe at a high frequency off-resonance value, a probe at the O_2^+ secular frequency, a probe at the O^+ secular frequency, and a final probe at the high off-resonance value (with cooling between each). Because their is good agreement between the initial and final probes, we assume little hysteresis impacting our results. Note how O^+ offers much better number-resolution than O_2^+, presumably due to its mass being more similar to Be^+.

the distance of each dark-visible ion pair. We have also imagined the ions' positions to be stored in a list accessed by index where the coolant ions come first, hence the summation indices (this reflects how QLICS handles the ion kinematic data). As we will see, Eq. 4.1 captures some, but not all, of the trans-species coupling of a given crystal in a single value. While many parameters (RF voltage, RF frequency, κ, axial endcap voltage etc.) contribute to the geometry of the crystal, their effects can all be contained in the ratio of $\omega_{\mathrm{x,y}}/\omega_{\mathrm{z}}$: the ratio of the strength of the (time-averaged) radial confinement to the axial confinement. Since the RF frequency is essentially fixed, κ is clearly fixed (and not easily controlled, for both see [10]), and we don't typically tune the RF amplitude, the easiest way of controlling the crystal geometry is by changing the endcap voltage, thereby controlling the entire crystal geometry with a single variable. As seen in Fig. 4.4, there is a happy medium between the relaxed, linear crystal, and the squeezed, 3D, crystal where the inter-species Coulomb interaction parameter is maximized. This demonstrates the tradeoff between the two sources of inter-species distance. In the first case, the lighter ions 'run off' in the axial directions more quickly than the heavier ones, creating poor coupling between the coolant ions at the ends of the chain and our centrally located dark ions. In the latter case, the heavier species is squeezed out in a ring whose radius increases more rapidly than the thickness of the central 'cigar' of coolant ions. This relationship suggests that it may be the case that axial tickles are more effective in 'skinny' crystals, and radial tickles are more effective in 'fat' crystal, though more study is needed to confirm this hypothesis.

4.1.2 Power Broadening

As discussed in back in Ch. 2, Sec. 2.2.3, the scattering rate of an ion as a function of its velocity is a Lorentzian curve. Of course, this curve is a function of the saturation parameter, s_0 as defined in Ch. 2 and [22]. For large s_0 ($s_0 >> 1$), the width of the

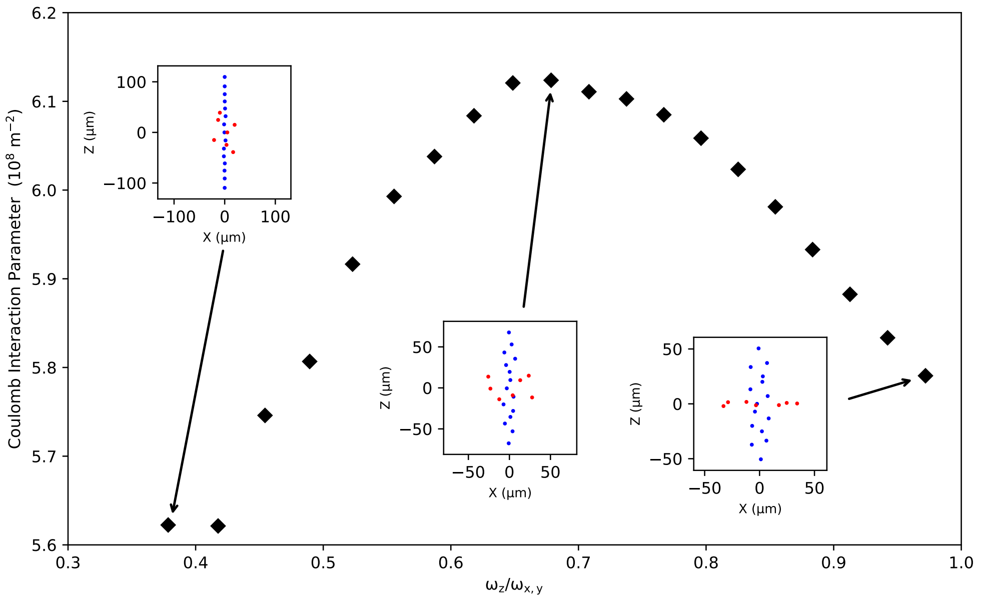


Figure 4.4: Inter-species Coulomb interaction parameter as a function of crystal geometry. As axial compression increases both species pass through two regimes: the lighter species goes from a 1-D 'chain' to a 3-D 'cigar', while the heavier species goes from a cylindrical shell to a ring.

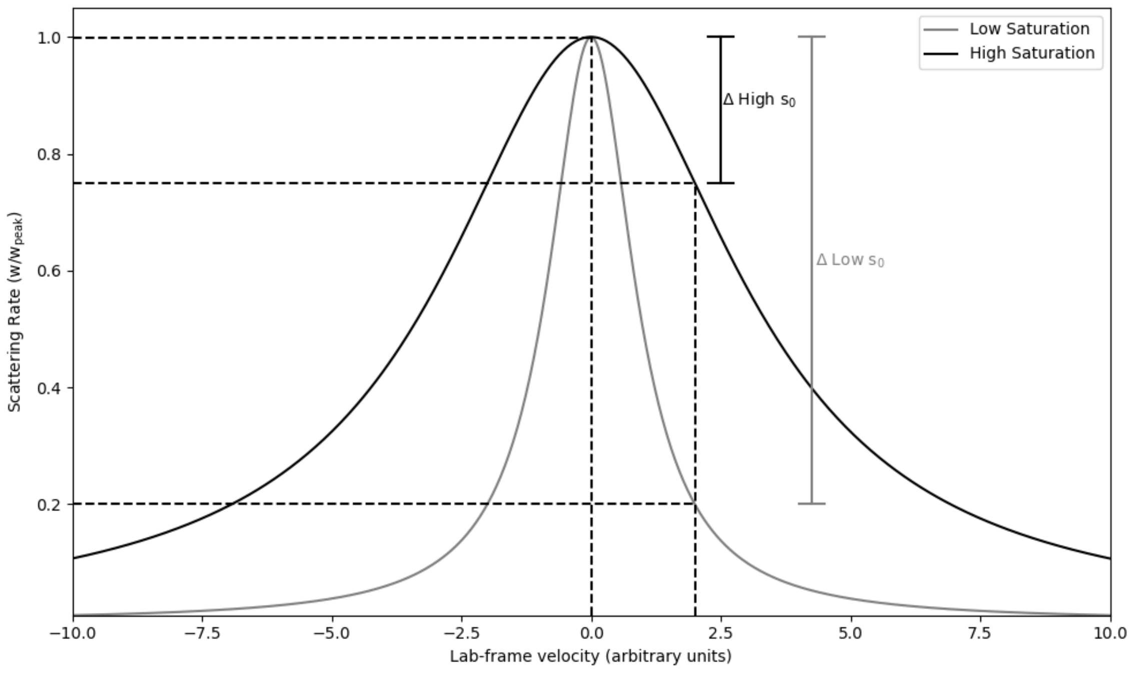


Figure 4.5: For the same velocity, measured in the lab / laser table frame, there will be a greater change in the scattering rate if our s_0 is not very large. Note that we consider 'low' saturation to still be well above what is consider 'saturation'.

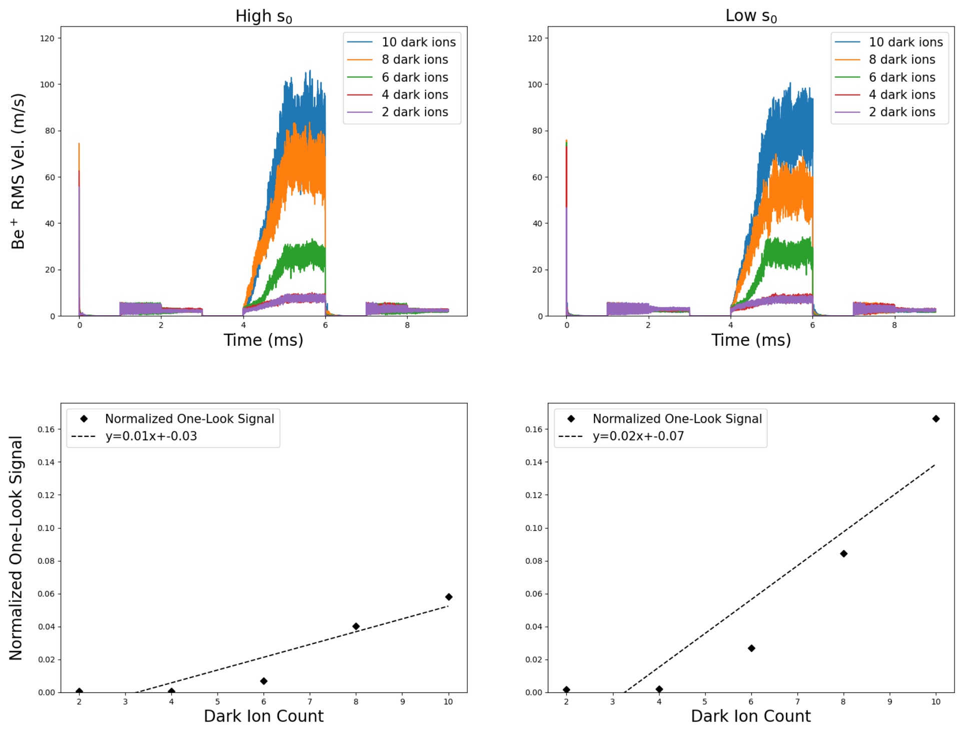


Figure 4.6: The principle discussed in Fig. 4.5 demonstrated in a full simulation. Notice how between the high and low saturation runs, the dynamics don't systematically change. Despite this, the fact we are not power broadening in the right hand column allows for finer velocity probing and, by extension, a stronger ion number to one-look detection signal correlation.

Lorentzian increases with s_0, while its peak does not, a common phenomena known as 'power broadening'. As shown in Fig. 4.5, as the detection laser saturates, the same change in velocity results in a smaller change in scattering rate, making our experiments less sensitive to the disturbance of the beryllium ions. This sensitivity difference can, of course, be seen in full simulations, where we run the experiment for identical systems, changing only our detection laser saturation as seen in Fig. 4.6. The fact that this shows up in our simulations is not surprising at all, since we are really just changing the way QLICS processes the velocity data it generates.

4.1.3 Coolant Species Mass

The results of [23] were naturally of interest to us, seeing as it is an example of a system for which other researchers were able to achieve number-resolution while we were struggling to achieve the same results (even in simulation, as seen in Fig. 4.2). A glaring difference between their system and ours were the species used: H_2^+ and H_3^+ sympathetically cooled within a Be^+ crystal. Note that their coolant species is much *heavier* than their dark ion species. This has a significant effect on the crystal geometry in that it places the dark ions in the crystal center, Fig. 4.7. While it doesn't clearly improve the static mean inter-species coupling, the expected source of benefit from the changed geometry, causing the motion of the excited ions colliding with the dark ions more often in their oscillations, has been observed. Simulations conducted in a set system[2] over various masses of coolant ions were conducted, Fig. 4.8, demonstrating an improvement of about 3x in normalized one-look detection signal size when using calcium as our coolant species over beryllium. In the future, we expect to conduct similar experiments to study the expected benefits from using Ca^+ as a coolant species to better inform our decision to move in that direction.

4.2 Decoupling of pure Be crystal size and LCF-MS signal

Since our main goal is to study the excitation of dark ions via our beryllium ions, getting a handle on the dynamics of a pure beryllium crystal is a natural stepping stone. If we assume that the detection laser hits all the ions in the crystal equally for the

[2]It is a subtlety in how we hold the trap parameter constant as we very coolant mass. If we simply hold the trap RF voltage constant, as in the dotted series of Fig. 4.8, changing the ion mass changes the whole crystal geometry, confounding our results. We therefore varied the RF voltage as we varied the coolant mass so as to hold the trap q parameter constant for the lightest species. This preserves the shape of the crystal 'core'. Note that this means the secular frequency of the dark ions is also changing, so we have to change the tickle frequency to match as well.

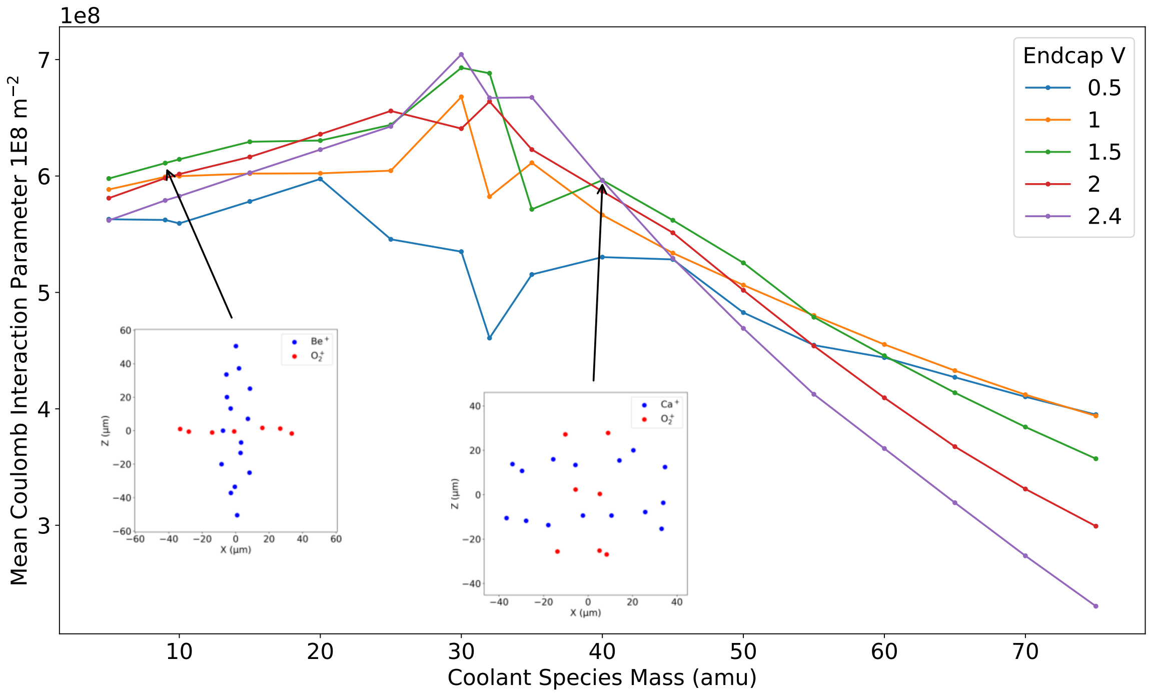


Figure 4.7: Mean inter-species Coulomb interaction parameter as a function of coolant species and endcap V. Crystal images taken for both beryllium and calcium coolant ions. Notice that calcium does not appear to have an obviously better interaction parameter then beryllium.

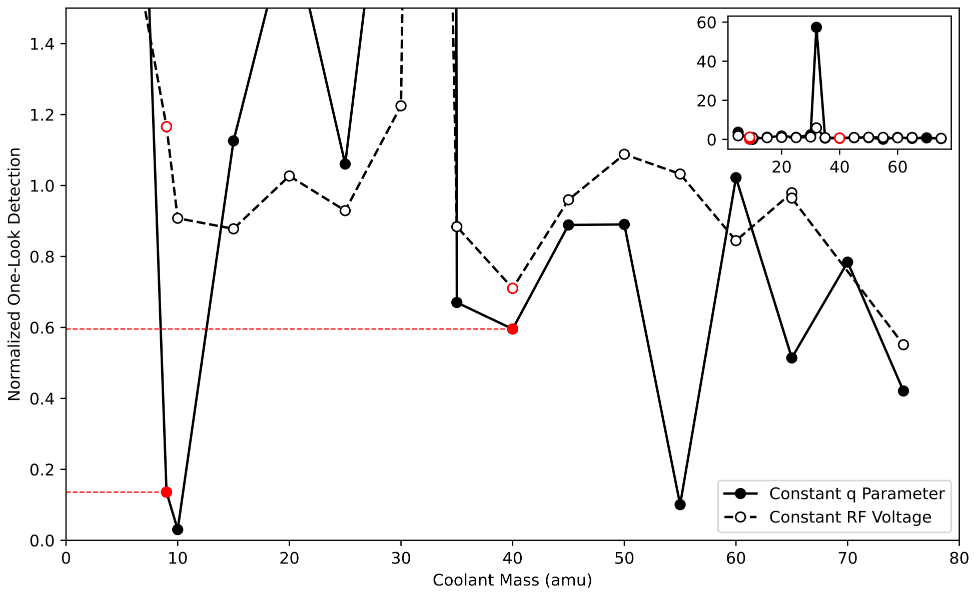


Figure 4.8: For reasons discussed in Sec. 4.1.3, it is very possible we did not completely isolate the mass-dependence of the normalized one-look detection signal size here. To confirm the more than 3x improvement in relative signal size in moving from beryllium to calcium, we would have to find optimal operating conditions for both and compare. Also note that the size of the one-look detection signal is not necessarily the same as number-resolution. The inset plot in the top right corner shows all datapoints, including the massive response when the coolant species is equal in mass to the dark ion species.

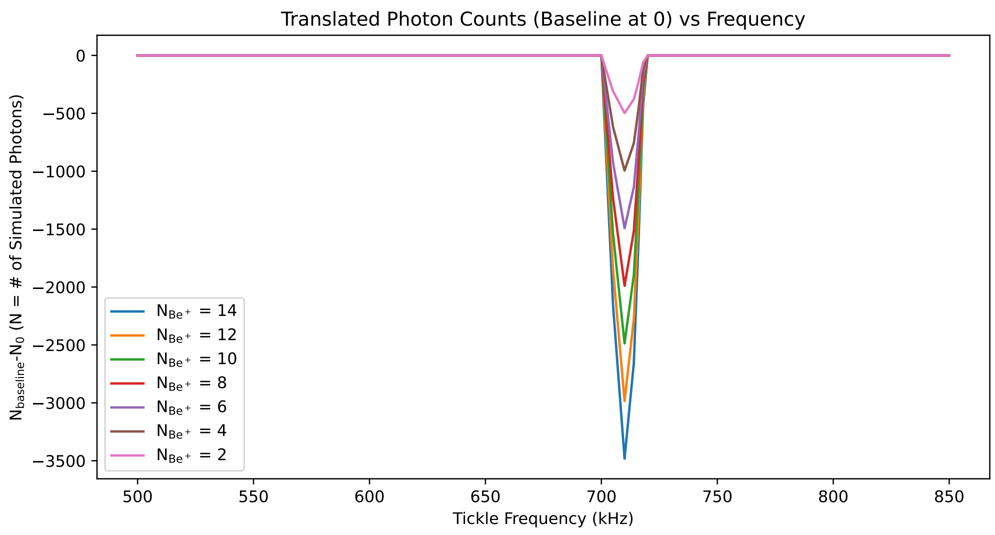
duration of the experiment (this simplification is what we simulate), we would expect a purely radial modulation to excite all ions in a pure beryllium crystal uniformly. Therefore, while the baseline scattering rate of the crystal will change as a function of the crystal size, the LCF-MS signal size (dip magnitude over baseline value) will be crystal size-independent. Obviously, this independence will be true regardless of tickle amplitude (a value that is hard to measure precisely in our current circuit). Since this experiment isolates the detection laser power from the crystal features / uncertainty of our tickle amplitude so well (variables that are hard to control and hard to measure, respectively), it is a useful way of checking the congruence of our simulations and our real-world experiment.

4.2.1 Simulation Expectations

In our simulations of this system we used a 100 mV tickle, with a trap RF of 66.4 V amplitude, endcap voltage of 2 V, and typical geometric values. The scaling with crystal size can be seen in Fig. 4.9. In Fig. 4.9 we calculate the ratio of one-look detection signal magnitude to the baseline, and find that there is no relationship between crystal size and this normalized signal size.

4.2.2 Experimental Results

We conducted this experiment in the real-world trap with a 400 mV tickle (nominal, the electronics cut this down by a few orders of magnitude), $58 - 60$ V RF amplitude, and 2 V on the endcaps. While we can't directly control the number of ions in the crystal, we attempted to get multiple data points by reloading beryllium (using the electron gun) multiple times, using our CCD imaging system to count the number of ions visually. For large crystals, we found it sometimes difficult to count the number of ions directly because of the 3-dimensional nature of the crystal. We combated this by relaxing the endcap voltage so the crystal arranged itself linearly, counting



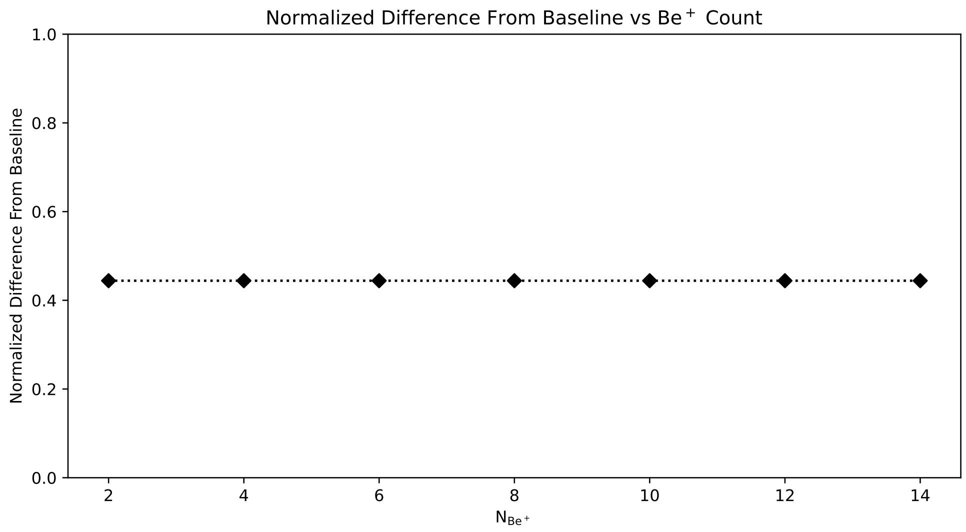


Figure 4.9: The baselines in the top plot have been translated so the series can be directly compared. For a spatially uniform tickle and detection beam profile, the total scattering rate scales perfectly linearly, as expected.

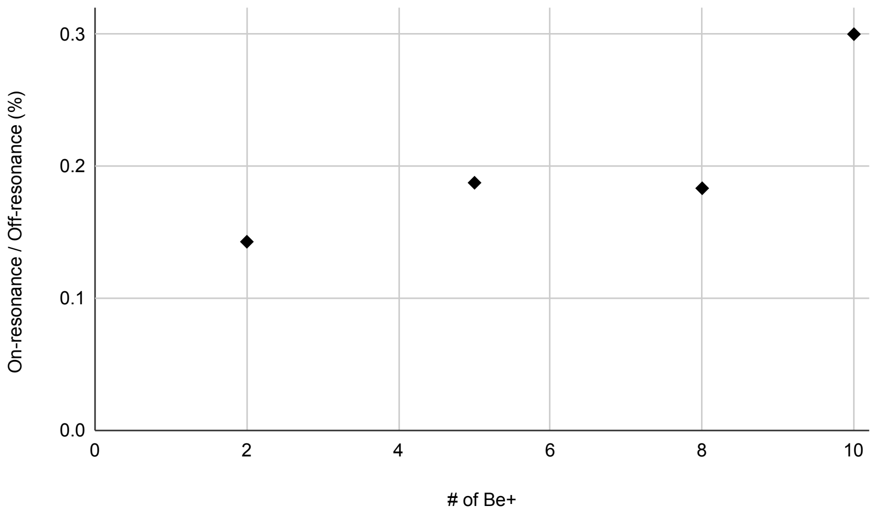


Figure 4.10: Real world data of the single species crystal size experiment, plotting the ratio of photons detected when tickling on-resonance to photons detected when tickling off-resonance. Due to loading difficulties we were only able to take measurements on 4 clearly unique crystals. While it is hard to draw conclusions from such few points, it does seem that their may be non-linear scaling effects, possibly due to the non-uniformity of the resonant beam profile.

the ions in this relaxed state, and then recompressing the crystal back into its usual formation. With this method, we were now limited by the width of our camera's field of view, as well as the beam waist size on the crystal (it was clear that the ions would sometimes be out of the detection beam for these large crystals). This limited our experiments to the range of about 10 ions. A plot of the change in LCF-MS signal size can be found in Fig. 4.10, and should be compared to Fig. 4.9. While we were only able to make these measurements on crystals of a few different sizes, it seems like we do not see the same invariance in relative LCF-MS signal size. We suspect that the partial illumination of the crystal may be a significant source of non-linearity in the photon count growth. Further experiments could explore this hypothesis by conducting a similar experiment at different levels of axial compression (changing the length of the crystal size and the ions' exposure to the resonant beam).

4.3 Heat Preparation

While our machine learning-led explorations have not literally given us a clear set of parameters for which our number-resolution in the beryllium / molecular oxygen system would be optimized, it has helped us devise a technique to help improve the inter-species coupling of the system. While, in retrospect, we didn't strictly require an artificial neural network to uncover this 'heat prep' technique, our path to its discovery demonstrates one of the benefits of having this feature in QLICS. This is a way in which exploration can be automated[3] and has helped us to find creative ways of exploiting the features of our system for our benefit.

4.3.1 ML Origins and Physical Intuition

While running ML optimizations of one-look LCF-MS signal size, we noticed an interesting trend - a decrease in the number of coolant ions and a decrease in DC endcap voltage aided our signal size (over some parameters). While there is some reasoning behind a lower DC voltage helping with the Coulomb interaction parameter (to a point, as shown in Fig. 4.4), upon closer study of these simulations we came to realize that the crystals were actually not cooling completely (again, as implied by the left-tail of Fig. 4.4). This suggests that we may achieve improvement in our number-resolution by maintaing heat within our system. This makes some physical sense - by melting our rigid crystal, we may, decrease *some* of the inter-species distances as the ions are mixed together, though we are also fighting against the increased volume of gas phase cloud, see Fig. 4.12.

[3]I should say one small step of the exploration process can be automated, and it still requires a great amount of oversight as well as someone to interpret the results - we are far from the point of replacing the job of the (simulated) ion trapper!

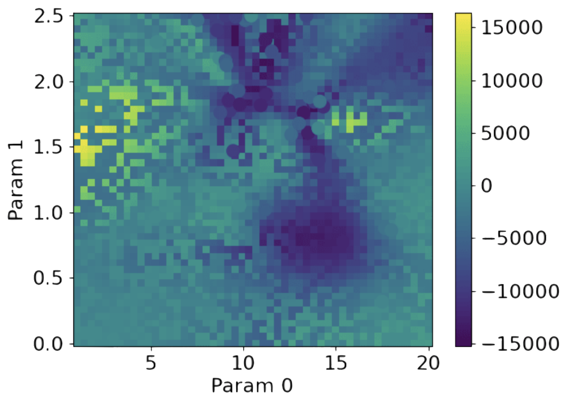


Figure 4.11: Unedited output from an M-LOOP optimization of a cost function defining peak size. The learner was given control over the endcap voltage and number of coolant ions. While it is often hard to provide definitive interpretations of these landscapes, we have just begun using them to guide manual explorations of promising regions - a process which lead to discovering the heat prep sequence.

4.3.2 Heat Prep Sequence

In the heat preparation sequence the only additional step we are adding to the typical experimental sequence is a tickle at the beryllium resonance to heat the crystal. We have tried tickling at about 100 mV for approximately 2 ms. An important feature to study is the trade-off of the added heat's benefit to the one-look signal and the added heat's increase in the noise level. In the limit of no added heat (traditional detection sequence), our number-resolution, while poor in terms of slope, exhibits no 'low-end cutoff', as will be seen in the higher temperature preparations, Fig. 4.13.

As we move into a 'moderate' amount of heat preparation, we begin to see significant improvement (almost 4x) in the slope defining our number-resolution over the high dark ion number regime, Fig. 4.14. This comes at the cost of our number-resolution at the low end in the number of dark ions as the increase in noise in the one-look LCF-MS signal due to the added heat makes it harder to resolve small signals (the left plot of Fig. 4.14). While this may seem to be a significant draw back, a realistic real-world detection sequence will likely use these two techniques (no heat prep and heat prep) in congruence to more precisely identify the number of dark ions

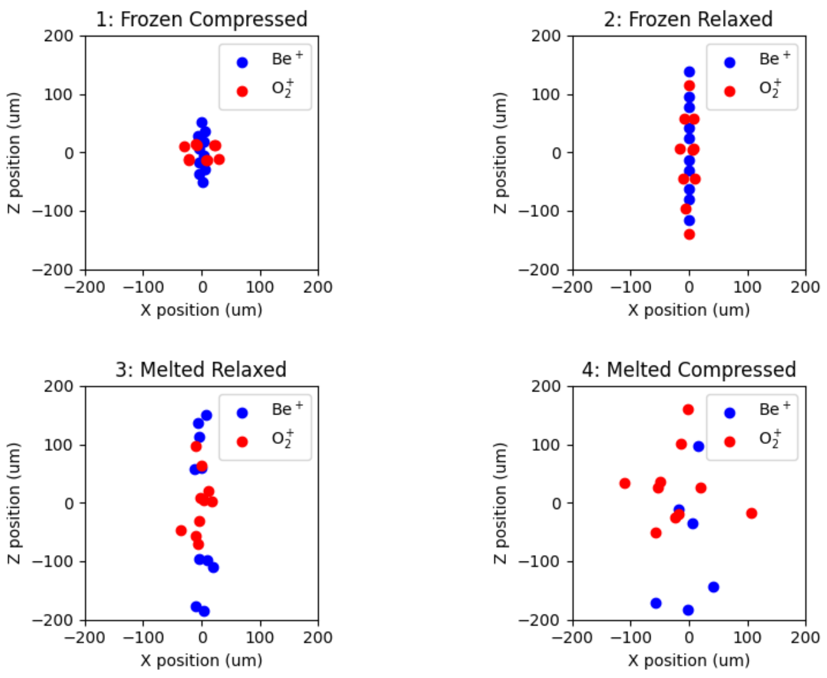


Figure 4.12: The first melting scheme we tried (depicted above) involved freezing the crystal (1), relaxing the DC voltage (2), tickling on the Be$^+$ resonance before stopping and letting it thermalize (3), and recompressing to the original endcap voltage (4). Note that this process does not necessarily improve our inter-species Coulomb interaction parameter due to the high energy of the molten crystal (now cloud), but it does destroy the crystal geometry.

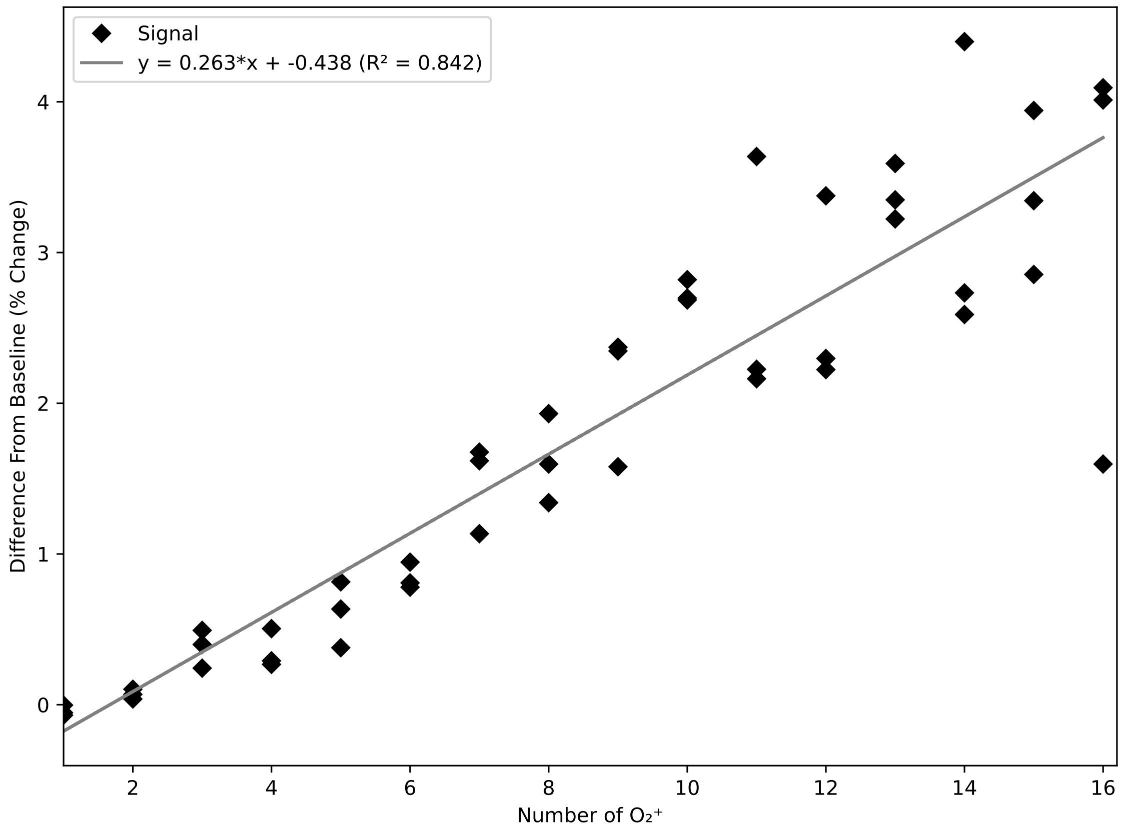


Figure 4.13: Number-resolution for a 500 mV tickle of crystals ($V_{\mathrm{DC}} = 2\,\mathrm{V}$) with 10 Be^{+} ions with no heat preparation, repeated 3 times for each configuration. Notice how the number-resolution/deviation is at its best when the number of dark ions is less than the number of coolant ions. This is because our crystals are imperfectly cooled so the uncertainty in our one-look LCF-MS signals increases.

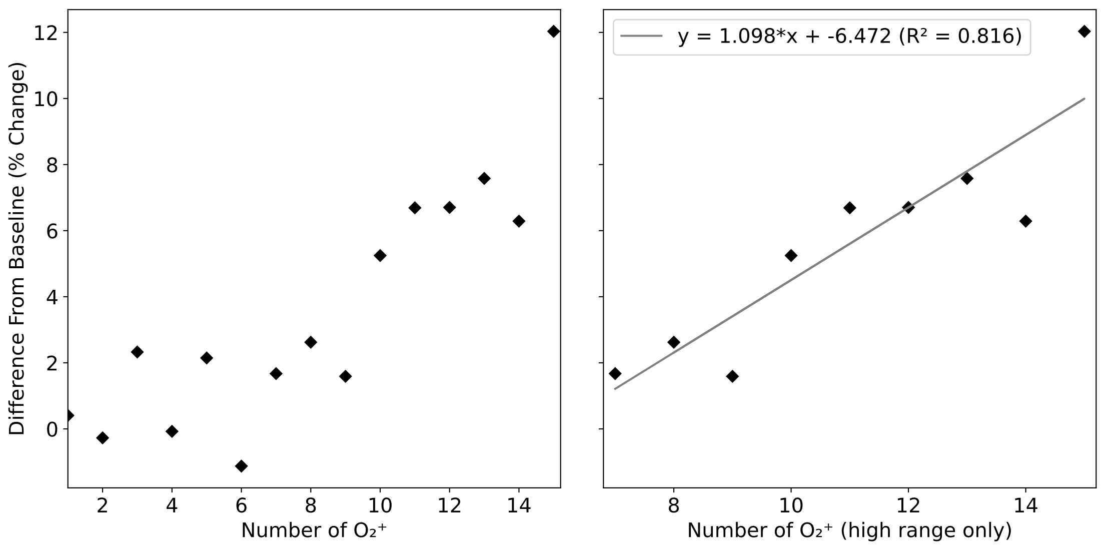


Figure 4.14: A repeat of the experiment done in Fig. 4.13, this time with a 'moderate' (controlled by amplitude of the heat tickle) amount of heat added. Notice how instead of one linear relationship there is a region with no predictive power and one with a higher predictive slope than was achieved with no heat prep. While these simulations were only conducted once for each crystal, it is expected that added heat will cause a general increase in noise.

in the Coulomb crystal.

Finally, as we might expect, for a high temperature heat prep our one-look LCF-MS signal is essentially obscured by noise, making any predictions unreliable despite the reasonably strong slope, Fig. 4.15. Thus, the effect of adding heat on our number-resolution can be summarized as a different position in the trade-off between LCF-MS signal to dark-ion-number correlation and noise.

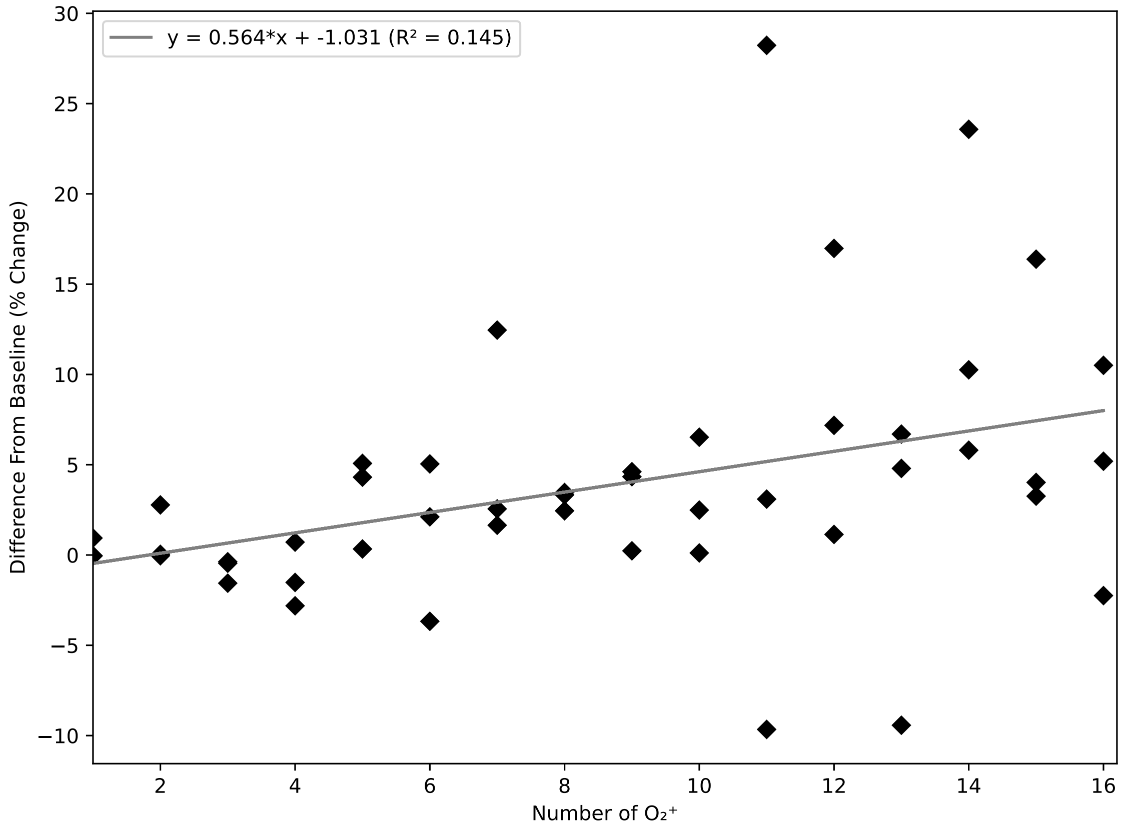


Figure 4.15: For high amounts of heating, the entirety of our dark ion number range falls in the regime dominated by noise. Because we have been keeping the number of coolant ions constant throughout this section, we cannot explore the crystals with more than 16 O_2^+, since, in those crystals, the dark ions can not be cooled enough to remain in the trap.

Chapter 5

Next Steps

The next steps of this project can be split into two main categories. The first is a catalog of potential studies, relevant to our lab's goals, that can be conducted *right now* with QLICS. I provide these examples to demonstrate the many areas of exploration the Hanneke Lab can now pursue, demonstrate the versatility of QLICS, and as inspiration to future students. It is worth emphasizing that the results of Ch. 4 are are really just the smallest 'tip of the iceberg' in terms of the ML-driven crystal dynamics research now available to us. The second category includes work that can be done to update and improve QLICS, as well as features that may improve its usability or applicability.

5.1 Suggested Future Explorations

In no particular order, here are some possible ML optimization projects that I thought would be interesting to pursue, may provide significant insight into our system, and can be implemented with QLICS in its current state of development.

5.1.1 Direct Number-Resolution Optimization of the Current System

The reader may have noticed that in the discussions of Ch. 4, we often use LCF-MS signal size as the metric of comparison when measuring different experimental systems against each other. This fact was also true for the many inconclusive ML explorations not depicted in this thesis. While this is a reasonable signal to try optimize in general, the metric of experimental importance for us is the number-resolution. A study of this sort will likely require the optimization of a quite high-dimensional parameter space, which may mean its best to use the neural network learner.

To set the skeleton (the 'experiment.ini') of such an experiment, one would want to come up with some relatively efficient baseline setting and probing scheme (efficient because the domain of the search will be relatively large, due to the high dimensionality if nothing else). While certainly different types of skeletons will satisfy, I naïvely imagine the simplest to be an iteration through different numbers of dark ions, conducting a typical, abbreviated detection scheme of tickling off resonance, on the secular resonance for the dark ions, and off resonance again. The purpose of this pulse sequence is to establish a baseline against which to measure the resonance signal, and the final baseline is to ensure the integreity / uncertainty of the experiment (the excited species' are able to be re-cooled). The change in these signal-to-baseline values would then be compared to the true change in numbers of dark ions, with some defined relationship between them likely the same type of a linear fit compared to noise we discuss in Ch. 4. This will likely define the cost function that one wants to optimize.

The optimizer will have to have control over many parameters, some of which may be: the number of coolant ions, the trapping parameters (both $\omega_{x,y}/\omega_z$, but also absolute values of both, as a higher- or lower-energy driven oscillator may have a positive effect on the inter-species collisions), the saturation of the resonant detection laser, the amplitude and duration of the tickle, possible heat prepping and the duration / intensity of it, and the shape (Taylor expansion coefficients) of the modulating electric field to name just a few.

Needless to say, an optimization of this scale is not a trivial task, and would require a significant amount of work both in setting up and in the interpretation of the ML results. However, this project would have great impact, for if a good parameter set was found, we would be able to greatly inform the eventual O_2^+ dissociation experiment.

5.1.2 Modulating Electric Field Shape Optimization

What might be described as a subset of the project suggested above, the optimization of the electric field shape would be a relatively straight forward (relative to the full system optimization) but very interesting study. The learner (again, likely a neural network) would have to be given some, or all of the coefficients of the electric field's Taylor series expansion which yields 21 total coefficient 'knobs' to control (a 3D second-order expansion yields 6 spatially dependent and 1 spatially independent coefficient for each component the electric field). One could imagine optimizing either the LCF-MS signal (simpler) or the number-resolution value (harder). In addition, it is possible that a study of how the crystal shape and modulation amplitude interact with the shape optimization could yield interesting results. While we have 4 general tickling schemes discussed in Sec. 2.2.2, we obviously don't actually have arbitrary control over the Taylor coefficients of the tickling field, and so a next step of this line of study would be to restrict the model to the use of electric fields that can be practically created with our current trap. An exciting property of this project would be that it would be relatively easy to test its findings against the real-world experiment with our current electronics.

5.1.3 Cooling and Loading Rate Optimization

A subject of practical concern is the rate at which we can trap ions, a limit that we ran into when conducting the study of Fig. 4.10. It may be that different trap parameters make it easier to 'capture' different velocity classes of ions. It is also clear that the configuration of our cooling laser will have a major effect on the loading rate. An optimization of this sort would explore the effect of different saturation parameters and different detuning amounts on this rate. Furthermore, it may be that we get a geometric advantage by either changing the k vector of the cooling laser, or splitting the detuned component into multiple beams (maybe counter-propagating cooling lasers would be advantageous, or maybe increasing the projection in the high-frequency radial direction would increase our trapping ability). One could also imagine that dynamically stepping the detuning amount, or introducing near-detuned and far-detuned components would allow us to capture a broader velocity class of ions. Since there is a logistical cost in changing the optics on the real-world table, it may be necessary to introduce some artificial 'penalty' to the cost function for the complexity of the solution it finds. It may also be that the optimal parameters change as a function of the ions already within the trap (we do see this, as sometimes ions are lost while we are actively loading and sometimes it seems as though we hit limits where we can't effectively load any more ions). Finally, it is possible that the initial capture of Be^+ and the sympathetic cooling of O_2^+ have different sets of optimal parameters. A better grasp of these concepts may significantly improve the day-to-day life of the ion trapper working with our system.

5.1.4 'Best vs. Best' Comparison of Be^+ and Ca^+'s Effectiveness in the LCF-MS Scheme

The study shown in Fig. 4.8 is fundamentally flawed in that neither holding the trap RF voltage or the 'q parameter of the lightest ion' constant fairly compares crystals of different coolant species. While we discuss the issues with holding the trap RF voltage constant in the footnote of Sec. 4.1.3, the approach of holding q constant (by varying RF voltage amplitude) changes the secular frequencies of the dark ions. This means that for a fixed tickle duration, we are driving over a different number of oscillations. A more fair comparison would be to compare the optimized LCF-MS or number-resolution signal that is practically achievable in both Be^+ and Ca^+, the essence of the 'Best' vs. 'Best' comparison. While this is similar in principle to the project of Sec. 5.1.1, it would probably be acceptable to consider the optimization over a smaller, more realistically achievable parameter space, since this optimization project is highly sensitive to practical considerations (after all, we expect Ca^+ to provide a geneal benefit over Be^+, the question is rather if the expected benefit is worth the difficulty in introducing a new coolant species into the system).

5.1.5 Reverse 'Optimization' to Establish Congruence with Real-World Experiment

My final suggestion is a 'congruence test' to check the possibility of establishing agreement between simulation results and, not analytic results, but experimental observations. The essence of online optimization is that the generation of the data is controlled by the optimizer, not preexisting. However, we could imagine taking a broad frequency sweep (akin to that reported in our paper demonstrating the detection of dark ions, [11]), defining the same experiment (structurally, some of the scaling coefficients such as tickle amplitude are not exactly known) in QLICS, and running an optimization to try and recover the same spectrum. The cost function would therefore be defined as the difference between the simulated and experimental spectra. The purpose of such a study would be to understand the relationship between our experimental and simulated system, highlighting the accuracy with which our simulation models the actual experiment.

5.2 Future Features and Improvements

Clearly, there are many exciting and important projects that can be done with QLICS in its current state. However, there are also many areas in which the program can be improved, including both possible new features and changes to the way the program is structured. An immediate example of such a task is the transition to a more standard, object-oriented organization of the QLICS source code.[1] Although I expect many more to appear as we continue to extend the uses of the program, I will highlight some possible areas of improvement that I have come across during this work.

5.2.1 Machine Learning Improvements

As a function of our treatment of M-LOOP as an optimizing black box, the integration of the simulation and ML optimization sides of QLICS is likely imperfect. It may be that direct control over the details of the ML mechanisms will help us to achieve faster and more reliable convergence on optimal parameter sets. In doing so, we could revise the workflow for improving our intuition of the interplay of different paramaters on a given cost function by further investigating the means of visualizing the intermediary steps in an online optimization. While M-LOOP neural network learners are able to produce cost landscapes for 2-dimensional optimizations (such as Fig. 4.11), it would be of great benefit to be able to use the entire course of a high-dimensionality

[1]Many thanks to Paul Grajzl for significant work on this topic.

online optimization run to help improve our own intuition of the system. It has been suggested[2] that full 'maps' of high-dimensional cost landscapes can be used to minimize the need to repeat redundant optimizations in similar systems. While certainly an ambitious goal over large domains, the creation of a catalog of such 'maps' would demonstrate great progress in our understanding of this complex system. In short, it is an open question of how we can improve our use of machine learning to optimize our experiments, now that QLICS has provided us with a mechanism of easily returning a cost for a set of input paramaters.

5.2.2 More Realistic Laser Models

Our laser-cooling model is primitive for two main reasons. First, we use a linear approximation of a Lorentzian process. It is unclear how dramatically this causes our simulation dynamics to deviate from the true dynamics, but if it is a major issue, we could re-parameterize the cooling force using a more precise model. More importantly, we currently treat both our cooling and detection lasers as infinite in beam waist radius and spatially uniform in intensity. This is actually a poor approximation, as we often see ions in large crystals to fluoresce to different degrees based on their position. This could possibly be a major source of issues we see in our real-world experiments. A fix to this would be to re-parameterize both as having a true Gaussian profile, defined in an experiments' configuration file along with the other laser parameters. Finally, we treat the resonant beam as completely passive, and do not simulate the recoil the ions feel due to absorption or re-emission of light due to this beam. Changing this will make our simulations more true to life, and may have implications, specifically, for the optimal execution of the heat preparation sequence.

[2]By Paul Grajzl, in an informal discussion on November 11, 2024

5.2.3 More Realistic Trap Model

Because our endcap electrodes are not radially symmetric, we sometimes observe splitting between the actual secular frequencies, ω_x and ω_y, in our real trap [20]. This is a characteristic that can be molded with (py)LIons `linearpualtrap` method using the 'anistropy' parameter [19]. However, this functionality was not carried over into QLICS. Furthermore, we have observed anharmonic behavior in our trap [10]. There is currently no way to include such a detail in a QLICS simulations. Repairing these two simplifications may aid in the congruence between our simulations and real experiments.

5.2.4 Improved Performance

The two most obvious areas that we can improve QLICS runtime performance are in the scattering rate integration during the detection routine and in the ability to run experiments in parallel during the ML optimizations.

We have often seen that the detection step of a single experiment often has a runtime on the order of the LAMMPS simulation runtime itself. While it would require more study to confirm, this may be due to the use of the full relativistic doppler shift, $f_{\text{O}} = f_{\text{I}}\sqrt{\frac{1+v/c}{1-v/c}}$ in our scattering rate equation, Eq. 2.19, with the repeated square root operation causing the slow execution. Improvement to this algorithm (perhaps by using the low-velocity doppler shift approximation, certainly valid for typical ion speeds on the order of 100 m/s or less), would be greatly beneficial.

While it is unclear how the details of this execution would work, finding a way to allow the ML optimizer to run simulations in parallel on the HPC system would greatly improve the time it takes to fit neural networks and GP models. We have already seen an order of magnitude improvement in execution times for general sets of simulations, and so would expect similar acceleration in this context. The primary issue with this is that, in general for the online optimization scheme, the model uses the cost function calculation at the previous step to inform its next test point in parameter space, a process that doesn't naturally fit with parallel execution [16, 29]. The reason for this restriction in M-LOOP is that it was made with the optimization of real-world experiments, not simulated experiments, in mind [16]. Once a solution to this issue is found, the integration of system-conscious parallel execution can be easily incorporated into the 'Run Experiment Batch From Directory' mode.

5.2.5 Full Simulation Access for ML Optimizer

Finally, the artificial restriction[3] of the ML optimizer having only access to the simulation output via the scattering data (not giving it access to position or velocity data) may have been too limiting to the scope of QLICS' application. While the original thought was that this was necessary to keep the program more true to the real world experiment (which we only access via scattered photon counts), it limits the detailed crystal-structure-type experiments we may want to study. For instance, we can not run an optimization with the cost function related to the mean inter-species Coulomb coupling parameter introduced in Eq. 4.1. Furthermore, users with trapped ion applications that do not depend on the fluorescence of the ions in their experiments will be unnecessarily restricted by this requirement. Therefore, removing the restriction should help the program's general applicability.

5.3 Final Thoughts

In this work, we have introduced a software package for the general-purpose simulation of ions trapped in linear Paul traps and provided examples of its use. Although QLICS heavily relies on several other programs (namely (py)LIon, LAMMPS, and M-LOOP [16, 19, 26]), it brings new functionality in its experiment-focused approach and the

[3]In the sense there is no program-logic based need for this restriction.

integration of dynamics simulation with online ML-driven optimization. Because the trapped-ion platform is used in such a wide variety of contexts, it is our hope that the program and the discoveries it may facilitate will find application beyond our use-case. Nevertheless, our use-case is far from exhausted. While we have begun the journey of better understanding the interaction between the many parameters defining our Coulomb crystal experiments, the development of QLICS is only the launch point for the many interesting and important studies of Coulomb crystal dynamics yet to come.

Appendix A

Videos of Sympathetic Excitation

A.1 Tickling GIFs

The two supplementary GIFs, 'offres.gif' and 'onres.gif' show simulations of the excitation of a mixed species crystal with a modulating electric field both on-resonance with the O_2^+ secular frequency and off-resonance. Note that when tickled on-resonance, the Be^+ ions are dramatically disturbed, but when tickled off-resonance the Be^+ core is not. The tickle amplitude was increased in both videos to emphasis the effect. Note that the initial cooling period is not done with a simulated laser cooling force, but rather with a genaric, 3D, linear damping force, $F = -|b|v$ for video brevity. This is a (0 K) (py)LIon-defined Langevin bath [19].

The GIFs can be accessed on Prof. Hanneke's website, `https://dhanneke.people.amherst.edu/`, or in the Amherst College Library Archives. Amherst College Physics and Astronomy Dept. faculty readers will also find the GIFs in the attached supplemental files.

Appendix B

QLICS Examples

Here I will provide a couple of the main example experiments QLICS will ship with. Then I will provide one completely annotated configuration file of an experiment of the form depicted by Fig. 3.4. Next, I will break down the structure of the fluorescence output CSV file. Finally, I will break down the syntax of the M-LOOP configuration file with an example of such input.

B.1 Examples

The examples introduced below follow a progression that builds up to a full LCF-MS experiment. We start with the sympathetic cooling of a 'dark ion' species. Then we show an experiment demonstrating the excitation of the Be^+ ions' secular frequency. Finally, we present a full LCF-MS experiment for the detection of dark ions in a mixed crystal. I have included the full configuration files for completeness, but only completely annotated the final configuration file. Throughout this thesis, we have 'cleaned' up the figures that were output by the QLICS program itself. Here, I do not do this, to show the raw output images using the QLICS 'Analyize Completed Experiment Mode'.

B.1.1 Sympathetic Cooling of Co-trapped Ions

This simulation merely creates a cloud of 20 Be^+ ions and 5 O_2^+ ions in a pseudo-potential trap. It then Doppler cools the Be^+, which, in turn, cool the O_2^+ sympathetically. The configuration code is below, but first note that although we used newlines to make the code readable (denoted by the ↪ symbol), the actual program INI file should keep keys and values on one line.

```
[directory]
dump_dir = /Users/michaelmitchell/qlicS/data/2024-11-24_16-24-53/

[live_vars]
current_timesequence_pos = 1

[constants]
h = 6.626e-34
c = 299792458
amu = 1.6605402e-27
ele_charge = 1.60217663e-19
boltzmann = 1.380649e-23

[ions]
be+ = [{'mass': 9,'charge': 1}, {'natural linewidth':
   ↪ 113097335.52923255,'absorption center': 957800000000000.0,'
   ↪ saturation': 765}]
o2+ = [{'mass': 32, 'charge': 1}, {'natural linewidth': None,
   ↪ 'absorption center': None,'saturation': None}]

[sim_parameters]
log_steps = 10
timesequence = [[1e-08, 5e5]]
lammps_boundary_style = ['f', 'f', 'f']
lammps_boundary_locations = [[-.001, .001],[-.001, .001],[-.001,
    ↪ .001]]
lammps_allow_lost = False
gpu = False

[ion_cloud_0]
uid = 1
species = be+
radius = 1e-4
count = 20

[ion_cloud_1]
uid = 2
species = o2+
radius = 1e-4
count = 5

[trap_0]
uid = 100000001
target_ion_pos = 0
radius = 1.25e-3
length = 1.5e-3
kappa = 0.17
frequency = 11.04e6
voltage = 66.4
endcapvoltage = 1.5
pseudo = True

[trap_1]
uid = 200000001
target_ion_pos = 1
radius = 1.25e-3
length = 1.5e-3
kappa = 0.17
```

```
frequency = 11.04e6
voltage = 66.4
endcapvoltage = 1.5
pseudo = True

[cooling_laser_0]
uid = 569202603907002
target_ion_pos = 0
target_ion_type = be+
beam_radius = 0.0001
saturation_paramater = 100
detunning = -300000000.0
laser_direction = [0.5, 0.5, 0.7071067811865475]
laser_origin_position = [0, 0, 0]

[exp_seq]
com_list = dumping,cloud,cloud,trap,trap,cooling_laser,evolve
```

Fig. B.1 is an unedited 3D image, generated in QLICS of the crystal at the end of the cooling simulation. Fig. B.2 shows the RMS-velocity of the entire crystal over the course of the simulation.

B.1.2 Pure Be$^+$ Crystal 'Chirp' Detection Sequence

The following experiment conducts a 'chirp' detection sequence - a LCF-MS experiment where the modulating frequency is not swept linearly, but rather with increased resolution near the expected resonance. This allows us to cover a broad frequency range quickly, while also seeing the details of our resonance. This experiment showcases the use of the 'iter object', the simulated detected photon count calculation, and the approximation of our true modulating electric field (here we use the coefficients for an electric field created by the middle segment of an electrode). Here is the

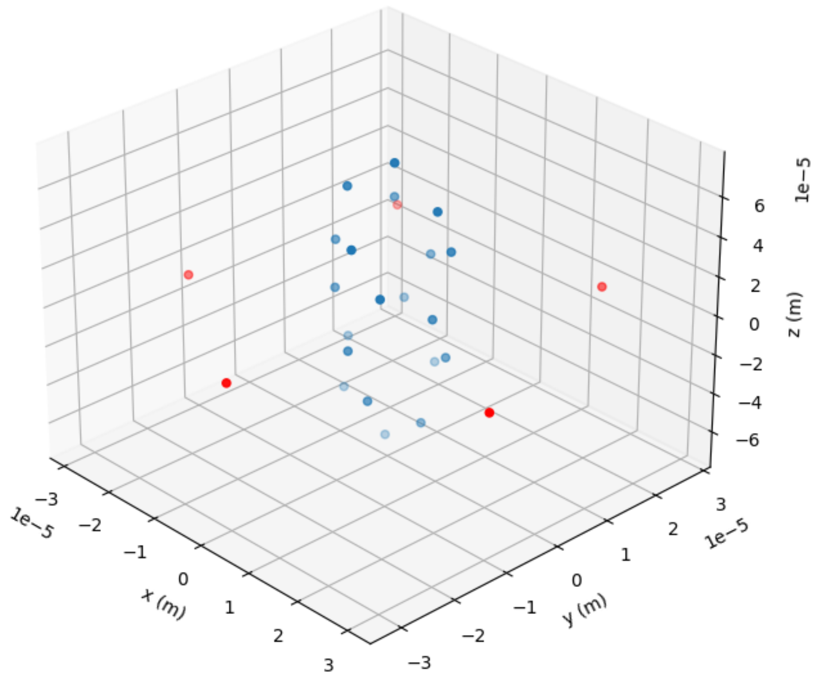


Figure B.1: An unedited image of the crystal, taken at the final timestep of the simulation, made using the ‘Analyze Completed Experiment’ mode. Note that QLICS does not make the aspect ratio of such images square by default, so the crystal appears ‘shorter’ then it actually is.

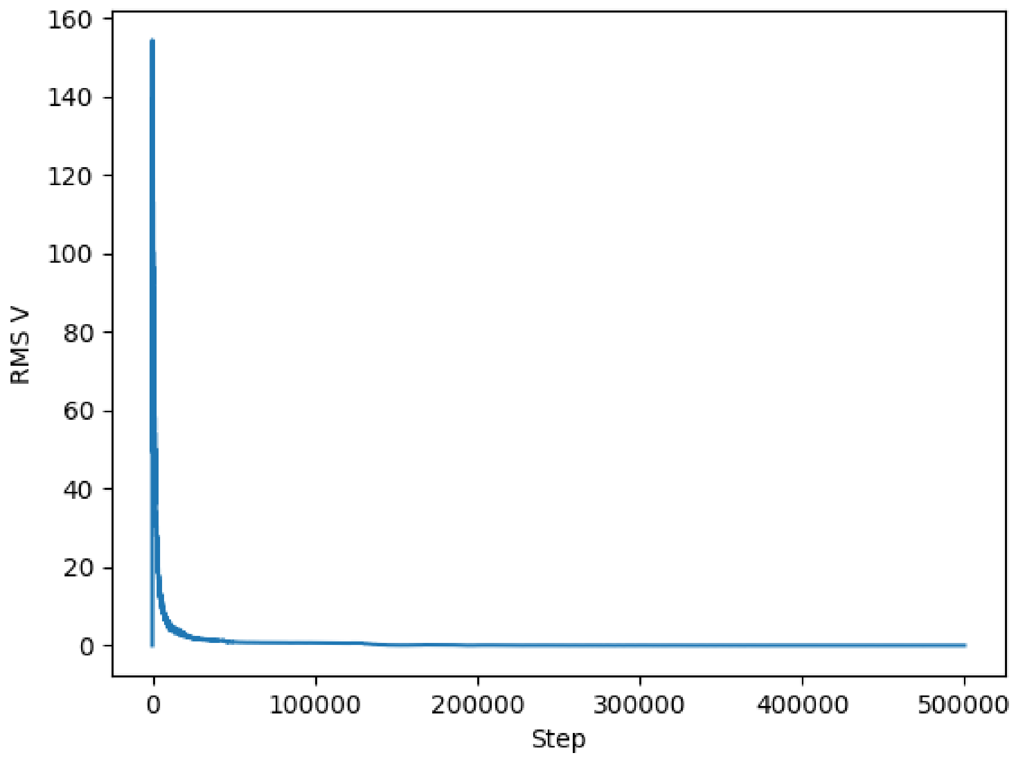


Figure B.2: Unedited QLICS image, made using the ‘Whole’ options of the ‘Analyze Completed Experiment’ mode. Note that RMS-velocity is calculated over all ions in the simulation, regardless of species.

configuration code:

```
[directory]
dump_dir = /Users/michaelmitchell/qlicS/data/2024-11-24_16-49-16/

[live_vars]
current_timesequence_pos = 34

[constants]
h = 6.626e-34
c = 299792458
amu = 1.6605402e-27
ele_charge = 1.60217663e-19
boltzmann = 1.380649e-23

[ions]
be+ = [{'mass': 9,'charge': 1}, {'natural linewidth':
    ↪ 113097335.52923255,'absorption center': 957800000000000.0,'
    ↪ saturation': 765}]
o2+ = [{'mass': 32,'charge': 1}, {'natural linewidth': None,'
    ↪ absorption center': None,'saturation': None}]

[sim_parameters]
log_steps = 10
timesequence = [[1e-07, 50000.0]]
lammps_boundary_style = ['f', 'f', 'f']
lammps_boundary_locations = [[-.001, .001],[-.001, .001],[-.001,
    ↪ .001]]
lammps_allow_lost = True
gpu = False

[detection]
detection_timestep_seq = [[]]
detector_area = 0.0001
```

```
detector_effeciency = 0.01

detector_distance = 0.2


[modulation_0]

uid = 469202603907006

amp = 1e-3

frequency = 850000

ex0 = -0.000454696e3

exx1 = 0.276281e6

exx2 = -0.000161148e9

exy1 = -0.000262039e6

exy2 = -0.000152101e9

exz1 = -3.26987e-1

exz2 = 0.0000886927e9

ey0 = -0.301462e3

eyx1 = -0.000257338e6

eyx2 = 0.153281e9

eyy1 = -0.347874e6

eyy2 = -0.198196e9

eyz1 = -0.000142671e6

eyz2 = 0.0347355e9

ez0 = -0.00014145e3

ezx1 = -3.27456e-1

ezx2 = 0.0000990478e9

ezy1 = -0.000164334e6

ezy2 = -0.0000207108e9

ezz1 = 0.0716623e6

ezz2 = -0.000120806e9

x_shift = 0

y_shift = 0

z_shift = 0

static = [0, 0, 0]
```

```
[ion_cloud_0]
uid = 1
species = be+
radius = 0.001
count = 10

[trap_0]
uid = 1000001
target_ion_pos = 0
radius = 1.25e-3
length = 1.5e-3
kappa = 0.17
frequency = 11040000
voltage = 66.4
endcapvoltage = 2
pseudo = True

[cooling_laser_0]
uid = 569202603907002
target_ion_pos = 0
target_ion_type = be+
beam_radius = 0.0001
saturation_paramater = 100
detunning = -300000000.0
laser_direction = [0.5, 0.5, 0.7071067811865475]
laser_origin_position = [0, 0, 0]

[cooling_laser_1]
uid = 369202603907003
target_ion_pos = 0
target_ion_type = be+
beam_radius = 0.0001
saturation_paramater = 100
```

```
detunning = -300000000.0
laser_direction = [0.5, 0.5, 0.7071067811865475]
laser_origin_position = [0, 0, 0]

[scattering_laser]
scattered_ion_indices = [0, 10]
target_species = be+
laser_direction = [-0.5, -0.5, -0.7071067811865475]
saturation_paramater = 100
frequency = 957800000000000.0

[iter]
scan_objects = ["modulation_0","scattering_laser","cooling_laser_1"]
scan_var = ["modulation_0", "frequency"]
scan_var_seq = [500000, 600000, 700000, 705000, 710000, 714000,
   ↪ 716000, 718000, 720000, 750000, 850000]
iter_timesequence = [[1e-08, 1e4],[1e-07, 1e3],[1e-07, 10000]]
iter_detection_seq = [[1e4, 1.1e4]]
com_list =
   ↪ tickle,evolve,r_469202603907006,evolve,cooling_laser,evolve

[exp_seq]
com_list =
   ↪ dumping,cloud,trap,cooling_laser,evolve,r_569202603907002,iter
```

The QLICS, 'Analyze Completed Experiment' function can be used to plot the acquired frequency spectrum, as seen in Fig. B.3. In addition, by looking at the RMS-velocity output, we can corroborate this spectrum: Fig. B.4.

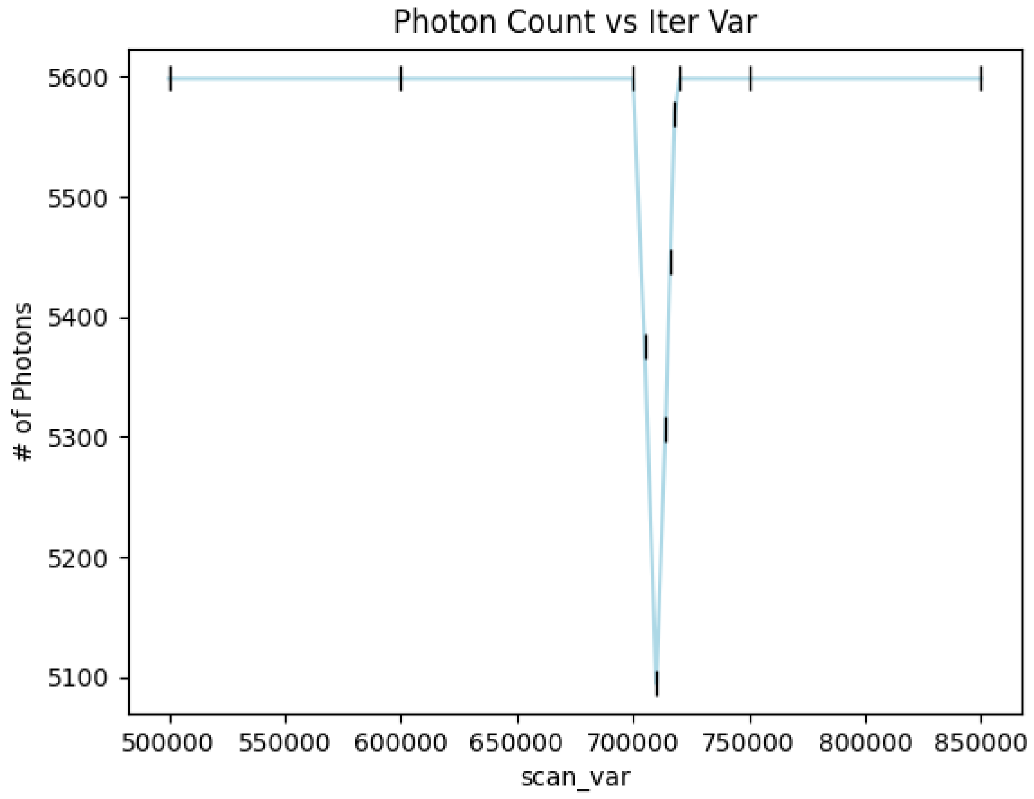


Figure B.3: Again, this figure is the unedited output from the QLICS 'Analyze Completed Experiment' mode. The 'chirp' nature of the iteration is clear by the X-positions of the data-points. Here, the scan_var is, of course, the modulating electric field frequency.

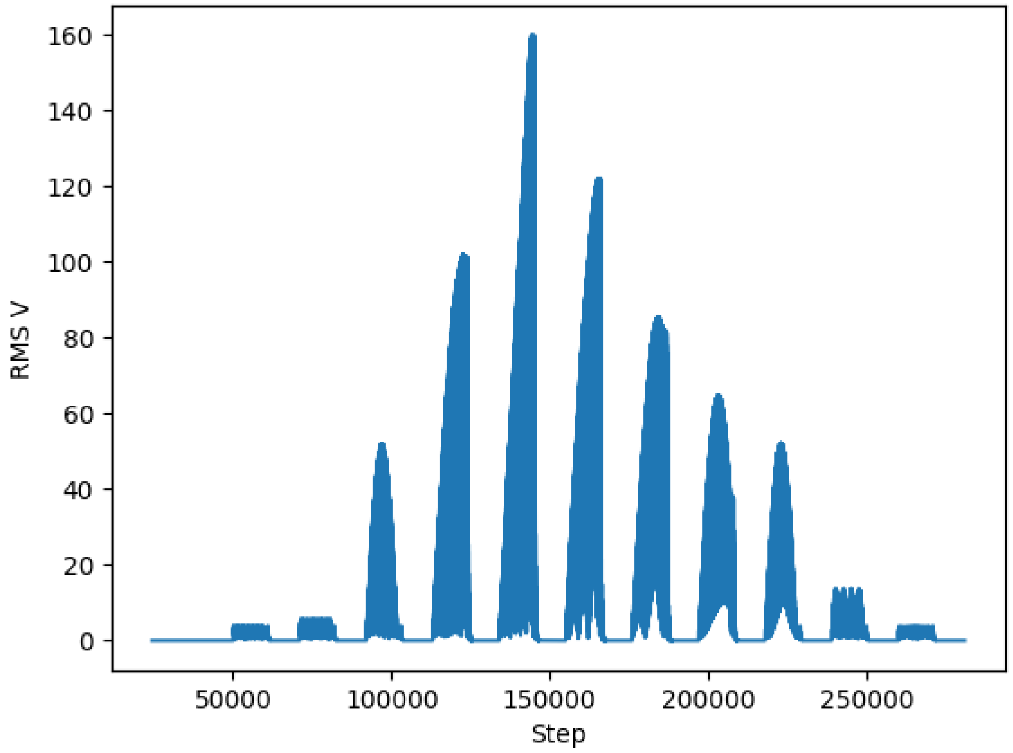


Figure B.4: While this is an unedited image made by QLICS, the initial data-points were removed, so that high RMS velocity in the initial cooling period didn't washout the signal at the various modulation pulses. Notice how the photon count dips generally agree with the RMS-velocity at each detection point. However there are some exceptions, such as the third data-point, which, despite a high peak RMS-velocity, does not translate into a significant change in fluorescence.

B.1.3 LCF-MS Example

Finally we will look at the LCF-MS example. While the results of this experiment don't perfectly isolate the signal, the structure of the INI file will help provide a skeleton for how these types of experiments may be conducted. Furthermore, the issues with the generated spectrum highlight the difficulty in achieving reproducible LCF-MS in our current system.

Annotated Experiment Configuration INI File Syntax

Here we provide a configuration INI file that was used in the LCF-MS example above with annotations indicated by a '#' symbol.

```
# The directory section is not user-edited (user input in this
   ↪ section will be overwritten automatically).  Much like the the
   ↪  live_vars section, it is used only for internal reference.
[directory]
dump_dir = /Users/michaelmitchell/qlicS/data/2024-11-24_12-11-36/

# The live_vars section is also not user-edited.  This manages
   ↪ timestep sequence position across QLICS, so that calculations
   ↪ dependent on timestep size are in agreement.
[live_vars]
current_timesequence_pos = 33

# Declaration of universal constants, all fields are mandatory.
   ↪ QLICS generally expects values in the INI file to be in SI
   ↪ units.
[constants]
h = 6.626e-34
c = 299792458
amu = 1.6605402e-27
ele_charge = 1.60217663e-19
```

```
boltzmann = 1.380649e-23


# Definition of ion 'types'.  The first item of the list defines the
   ↪  ion's mass and charge as required by (py)LIon.  The second
   ↪ paramaterizes the transition used for both cooling and
   ↪ resonant detection.  There is not currently any way to
   ↪ simulate a species with multiple relevant transitions.  Note
   ↪ that for species that are not addressed optically, None
   ↪ objects can be passed without error.
[ions]
be+ = [{'mass': 9, 'charge': 1},{'natural linewidth':
   ↪ 113097335.52923255, 'absorption center': 957800000000000.0,
   ↪ 'saturation': 765}]
o2+ = [{'mass': 32, 'charge': 1},{'natural linewidth': None,
   ↪ 'absorption center': None,'saturation': None}]
# Parameters that set the LAMMPS environment.
#   - log_steps: how many timesteps between LAMMPS thermo output
   ↪ dumping events.  Used to reduce the size of positions.txt
   ↪ files.  Note that log_steps is independent of timestep size,
   ↪ so it doesn't change the dynamics, only how often we sample
   ↪ dynamics.
#   - timesequence: a list of lists defining each evolution command.
   ↪   Each sublist is of the form [timestep_size,
   ↪ number_of_timesteps], allowing the timestep size to be
   ↪ dynamically adjusted.  In this example, all evolution is done
   ↪ in the iter object, so timesequence is left empty.
#   - lammps_boundary_style: the arguments of the LAMMPS boundary
   ↪ command.  See LAMMPS documentation, docs.lammps.org/boundary.
   ↪ html, for more information.
#   - lammps_boundary_locations: set the size of the simulation
   ↪ boundary in SI units and Cartesian coordinates.
#   - lammps_allow_lost: boolean allowing LAMMPS to complete
   ↪ simulation if lost (the definition of lost is controlled by
    ↪ boundary style and boundary location)
#   - gpu: (py)LIon gpu acceleration feature (see (py)LIon
    ↪ documentation)
[sim_parameters]
log_steps = 10
timesequence = []
lammps_boundary_style = ['f', 'f', 'f']
lammps_boundary_locations = [[-.001, .001],[-.001, .001],[-.001,
    ↪ .001]]
lammps_allow_lost = False
gpu = False

# This section should be thought of as only the 'detector' and is
    ↪ not related to the 'scattering_laser'.
#   - detection_timestep_seq: A list of lists, with each sublist in
    ↪ the form [timestep_start, timestep_stop], defining what parts
    ↪ of the simulation the detector is counting photons at.  The
    ↪ value is left blank here since we only want to detect at times
    ↪  within the iter sequence in this example.
The remaining variables define a constant by which to reduce the
    ↪ photon count by:
    detector_effeciency(detector_area/(4*pi*detector_distance**2))
[detection]
detection_timestep_seq = [[]]
detector_area = 0.0001
detector_effeciency = 0.01
detector_distance = 0.2

# The `tickle' electric field.  Of the form: E = A * R * Cos(wt),
    ↪ where R(x,y,z) is the position dependence vector.
#   - uid: explicit control of the (py)LIon per-object
    ↪ identification variable.  This value can not be repeated with
    ↪ any other object and allows for explicit removal if desired.
```

```
#   - amp: The amplitude multiplier, A in the equation above.
   ↪ Generally, its value is set a bit arbitrarily.
#   - frequency: The frequency of oscillation, in Hz.  Note that in
   ↪ this example, the value is overwritten in the iter object, so
   ↪ the value in the initial declaration here does not matter.
#   - position expansion coefficients: coefficients of the second-
   ↪ order Taylor series expansion in 3 dimensions for some field.
   ↪  Note that here, we are using a uniform field for simplicity.
   ↪  Notationally, the first letter refers to the component of the
   ↪  E-field, the second letter refers to the dimension the
   ↪ componant varies with respect to, and the final number
   ↪ indicates the order of the term.
#   - shift variables: These translate the electric field
   ↪ definition.
#   - static: this vector defines a time-independent, and spatially-
   ↪ independent component, if desired.
[modulation_0]
uid = 469202603907006
amp = 1e-3
frequency = 195683.7
ex0 = -0.000454696e3
exx1 = 0.276281e6
exx2 = -0.000161148e9
exy1 = -0.000262039e6
exy2 = -0.000152101e9
exz1 = -3.26987e-1
exz2 = 0.0000886927e9
ey0 = -0.301462e3
eyx1 = -0.000257338e6
eyx2 = 0.153281e9
eyy1 = -0.347874e6
eyy2 = -0.198196e9
eyz1 = -0.000142671e6
```

```
eyz2 = 0.0347355e9
ez0 = -0.00014145e3
ezx1 = -3.27456e-1
ezx2 = 0.0000990478e9
ezy1 = -0.000164334e6
ezy2 = -0.0000207108e9
ezz1 = 0.0716623e6
ezz2 = -0.000120806e9
x_shift = 0
y_shift = 0
z_shift = 0
static = [0, 0, 0]

# This object directly accesses the (py)LIon ion cloud function,
   ↪ creating a spherical cloud of ions.  Again, we incorporate
   ↪ mandatory, explicit UID control so the cloud may be removed
   ↪ in the command list.
[ion_cloud_0]
uid = 1
species = be+
radius = 1e-4
count = 20

# A second ion cloud, this time for o2+.  Notice how the UID must be
   ↪  different.  Each object declaration requires a different '
   ↪ index', the value after the type declaration.  These must
   ↪ incrementally increase from 0, and define the order with which
   ↪  the objects are appended to the simulation.
[ion_cloud_1]
uid = 2
species = o2+
radius = 1e-4
count = 5
```

```
# The first object of type 'trap'.  The UID is an arbitrary integer.
    ↪   Because we expect the traps to generally be pseudopotential
    ↪ approximations, a 'target ion' must be defined.  Note that ion
    ↪  target is identified not by type, but by appending group.
    ↪ The remaining arguments are the same as defined in the (py)
    ↪ LIon docs.
[trap_0]
uid = 11012277363
target_ion_pos = 0
radius = 1.25e-3
length = 1.5e-3
kappa = 0.17
frequency = 11040000
voltage = 66.4
endcapvoltage = 1.5
pseudo = True

# Once again, we have to increment the object number.  While the
    ↪ value of the UID looks very specific, it really is just an
    ↪ arbitrary unique number.  Note that we make the same trap
    ↪ twice, with different targets to co-trap different species.
    ↪ The reason for this is explained in Ch. 3.
[trap_1]
uid = 12012277363
target_ion_pos = 1
radius = 1.25e-3
length = 1.5e-3
kappa = 0.17
frequency = 11040000
voltage = 66.4
endcapvoltage = 1.5
pseudo = True
```

```
# The cooling laser object.  While target_ion_pos and
   ↪ target_ion_type may look redundant, this makes it not
   ↪ mandatory to apply a cooling force to all ions of the same
   ↪ species.  beam_radius and laser_origin_position, are relics of
   ↪  an early attempt to incorporate a finite-sized laser beam,
   ↪ and currently have no effect on the simulation.  The remaining
   ↪  variables are self-explanatory, and correspond to the
   ↪ discussion in Ch. 2.
[cooling_laser_0]
uid = 569202603907002
target_ion_pos = 0
target_ion_type = be+
beam_radius = 0.0001
saturation_paramater = 100
detunning = -300000000.0
laser_direction = [0.5, 0.5, 0.7071067811865475]
laser_origin_position = [0, 0, 0]

# This object defines the resonant beam.
#   - scattered_ion_indices:  The list of ions (in their order of
   ↪ creation) over which the ion scattering rate calculation is
   ↪ conducted.  This explicit control is often redundant, and
   ↪ likely will be updated at some point to just consider for
   ↪ calculation by species.
# The remaining variables are as defined in Ch. 2.
[scattering_laser]
scattered_ion_indices = [0, 20]
target_species = be+
laser_direction = [-0.5, -0.5, -0.7071067811865475]
saturation_paramater = 5
frequency = 957800000000000.0
```

```
# The iteration object basically acts as a 'for loop'.
#   - scan_objects: The subset of the main object declaration pool
    ↪ to be used in the iter object's sub-command list
#   - scan_var: The object variable to be scanned from the iter
    ↪ object pool, for which the loop varies.
#   - scan_var_seq: A list of values that the scan_var is set to in
    ↪ the loop.  The length of this loop defines the number of loops
    ↪  in the iteration.
#   - iter_timesequence: Analogous to the primary timesequence, this
    ↪  list of lists defines the evolution commands in the iter
    ↪ command list
#   - iter_detection_seq: Analogous to the primary detection
    ↪ sequence.  It should be noted that these timestep values are
    ↪ relative to the iteration loop.  So here, we conduct detection
    ↪  from 2.9e5-3e5 timesteps measured from the start of each loop
    ↪  in the iteration.
#   - com_list: The sub-command list for the iteration object.  This
    ↪  is the experimental sequence that is conducted for each
    ↪ iteration of the iter object.  Note that while the explicit
    ↪ removal commands (r_UID) are defined for particular UIDs, the
    ↪ UIDs of objects introduced into the iteration command list are
    ↪  actually incremented over the loops.  Because of this it is
    ↪ important to space the initial UID values of objects that will
    ↪  be used inside iterations out, to avoid UID conflicts.
[iter]
scan_objects = ["cooling_laser_0","modulation_0"]
scan_var = ["modulation_0", "frequency"]
scan_var_seq = [175683.7, 177683.7, 179683.7, 181683.7, 183683.7,
    ↪ 185683.7, 187683.7, 189683.7, 191683.7, 193683.7, 195683.7]
iter_timesequence = [[1e-08, 1e5],[1e-08, 1e5],[1e-08, 1e5]
    ]
iter_detection_seq = [[2.9e5, 3.0e5]]
com_list = cooling_laser,evolve,r_569202603907002,tickle,
    evolve,r_469202603907006,evolve


# The primary command list.  Note how objects are appended, not by
   ↪ explicit reference, but by type.  This is why the number
   ↪ ordering is so important in the initial object declaration.  A
   ↪  full list of allowed commands will be available in the QLICS
   ↪ documentation.
[exp_seq]
com_list = dumping,cloud,cloud,trap,trap,iter
```

B.2 Photon Count Over Experiment Iterations

Here, I provide the output of the LCF-MS experiment defined above. Note that this spectrum does not produce the exact frequency data that we would expect, and the results did seem to be moderately unstable over multiple runs of the simulation. The data is reported in Tbl. B.1, and plotted in Fig. B.5.

det_start	det_stop	modulation_0 - frequency	photon count
290000.0	300000.0	175683.7	9362.7827
590000.0	600000.0	177683.7	9332.1938
890000.0	900000.0	179683.7	9348.9394
1190000.0	1200000.0	181683.7	9330.6339
1490000.0	1500000.0	183683.7	9327.7157
1790000.0	1800000.0	185683.7	9280.5361
2090000.0	2100000.0	187683.7	9303.5036
2390000.0	2400000.0	189683.7	9249.6448
2690000.0	2700000.0	191683.7	9314.6967
2990000.0	3000000.0	193683.7	9305.1626
3290000.0	3300000.0	195683.7	9313.2308

Table B.1: Raw CSV output from QLICS (obviously, transferred into a LaTeX table, but otherwise unedited) for the LCF-MS experiment described in Sec. B.1.3. QLICS provides the value of the 'scan_var' in the loop as well as the absolute time-steps over which the detection was conducted.

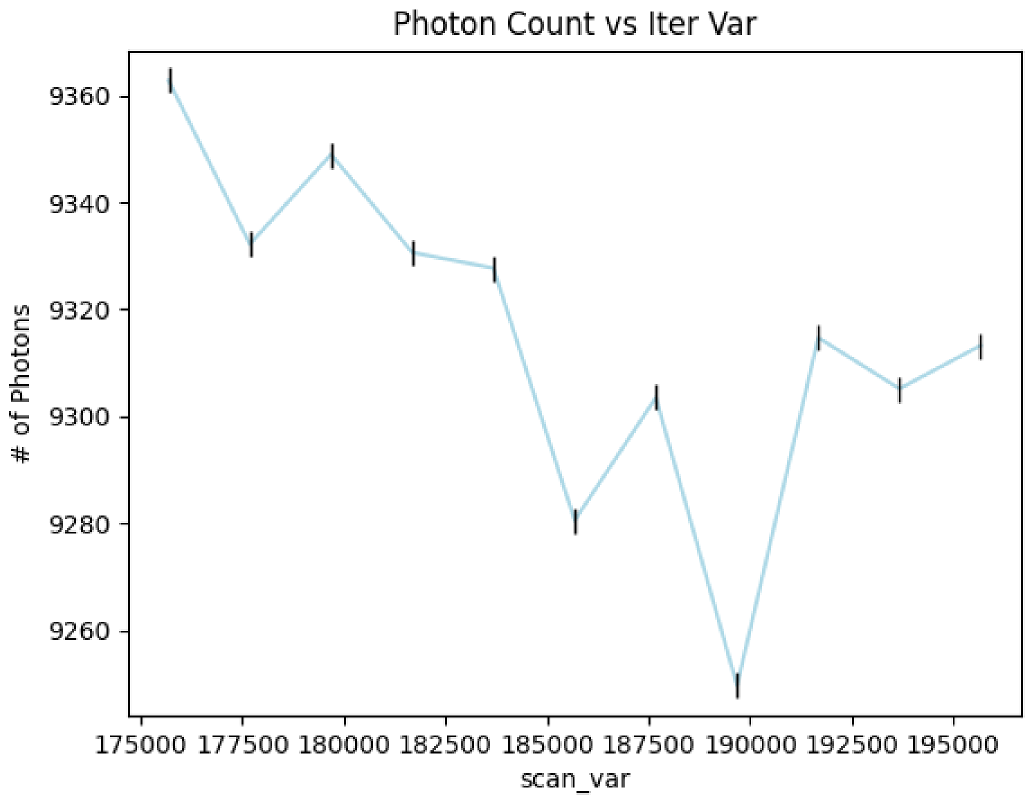


Figure B.5: Unedited QLICS output, plotting the results reported in Tbl. B.1. Again, scan_var here is the modulation frequency. The radial secular frequency for oxygen trapped in the trap defined by this experiment is about 185 kHz, yet this is not the minimum point in our data. The lack of reproducibility in this experiment may suggest some issues with the experiment integrity. One may try to tweak the parameters of the experiment in the annotated INI file above to try and improve these results.

B.3 Optimization Configuration File Example

Finally, I will provide an example of the M-LOOP-controlling input Python file. For brevity, I will not include the necessary experiment skeleton INI file, but the full example will be available in the QLICS documentation. While I won't go over every line of the example file I will highlight a few points below.

The file defines two functions that QLICS expects to locate, 'get_run_info' and 'return_controller'. The latter is extremely basic, and just returns the M-LOOP 'create_controller' output. This is where we delcare the type of learner we will be using, the parameter space we will be searching (dimensions and range), and the termination conditions of the optimization. The former actually controls the process of running the simulation. In this example, we are trying to optimize the LCF-MS signal in a mixed-species crystal by changing the number of beryllium and molecular oxygen ions, as well as by changing the endcap voltage. Since changing the endcap voltage changes the radial secular frequency (Eq. 2.9), we need to recalculate the expected secular frequency based on the neural network learner's endcap potential value. This parameter (really the full list of frequencies to modulate at) is inserted into the iter object's `scan_var_seq` as an override of what was originally submitted in the experiment INI file. Similarly, we override the relevant values of the ion numbers of both species to apply the parameter set the neural network wants us to.

Once we receive the photon count results from the simulation, we pass this list through a simple algorithm defining the quality of our LCF-MS signal (here we are looking at optimizing peak size). This is our cost function. The cost function used here was rather hastily defined, and a more advanced cost function may improve our optimization results significantly. In addition, we may define an uncertainty value based on these results. A very primitive uncertainty algorithm (just the standard deviation of the so-called baseline, the photon count at off-resonance modulation) was used here. While this optimization was done for one QLICS experiment per parameter step, we could easily conduct multiple experiments for each given parameter set. This would come in handy if, for example, we wanted to define our cost function on number resolution instead of LCF-MS signal size. Here is the code:

```
from qlicS.cl_console import run_from_file
import mloop.controllers as mlc
import numpy as np


def get_run_info(experiment_dir, params) -> dict:
    def f_r(mass, ev, charge=1.60217663e-19, kappa=0.17,
       ↪  length=1.5e-3, freq=11.04e6, voltage=66.4,
       ↪  radius=1.25e-3):
        ar = -4 * charge * kappa * ev / (mass * length**2 * (2 *
           ↪  np.pi * freq) ** 2)
        qr = 2 * charge * voltage / (mass * radius**2 * (2 * np.pi *
           ↪  freq) ** 2)
        wr = 2 * np.pi * freq / 2 *np.sqrt(ar + qr**2 / 2)
        f_xy = wr/2/np.pi
        return f_xy
    amu = 1.6605402e-27
    o2_res = f_r(32*amu, params[2])
    scan_seq = [78000, 100000, 125000, o2_res, 225000, 250000,
       ↪  278000]
    try:
        scat = run_from_file(optimize_mode=True, exp=experiment_dir,
            ↪  cloud_0_count=int(round(params[0])),
            ↪  cloud_1_count=int(round(params[1])),
            ↪  iter_scan_var_seq=scan_seq,
            ↪  trap_0_endcapvoltage=params[2],
            ↪  trap_1_endcapvoltage=params[2],
            ↪  scattering_laser_scattered_ion_indices=[0,
            ↪  int(round(params[0]))])
        non_res_list = []
        res_count = scat[4][3]
        for s in scat:
            if s[3] != res_count:
                non_res_list.append(s[3])
        # Optimizing Peaks
        res_diffs = [(res_count-i) for i in non_res_list]
        avg_diff = sum(res_diffs)/len(res_diffs)
        cost = -(avg_diff)
        bad = False
        uncer = np.std(non_res_list)
    except Exception as e:
        bad = True
        cost = 0
        uncer = 0
        print(e)
    return {"cost": cost, "uncer": uncer, "bad": bad}
def return_controller(interface):
    return mlc.create_controller(
        interface,
        "neural_net",
        max_num_runs=100,
        param_names=['Num_Be', 'Num_O2', 'V_DC'],
        num_params=3,
        min_boundary=[1, 1, 0,],
        max_boundary=[20, 20, 2.5],
        no_delay=False,
    )
```

Appendix C

Integration Algorithms

Here, we provide the 4th order Runge-Kutta algorithm, reproduced from [25]. We then explain how both RK4 and the Velocity-Verlet algorithms were used on a physical system to create Fig. 3.1.

C.1 RK4 Algorithm Applied to a Physical System

The general form of the fourth order Runge-Kutta method, given by [25] is:

$$k_1 = (\Delta t) f(t_n, y_n) \tag{C.1}$$

$$k_2 = (\Delta t) f\left(t_n + \frac{\Delta t}{2}, y_n + \frac{k_1}{2}\right) \tag{C.2}$$

$$k_3 = (\Delta t) f\left(t_n + \frac{\Delta t}{2}, y_n + \frac{k_2}{2}\right) \tag{C.3}$$

$$k_4 = (\Delta t) f(t_n + \Delta t, y_n + k_3) \tag{C.4}$$

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{C.5}$$

where Δt is our timestep size, t_n and y_n are the time and estimated value of some function y at step number n, and $f(t, y) = \frac{dy}{dt}$, a known function for which we are trying to find $y(t)$ by numerical methods. A derivation of the coefficients used in the final equation can be found in [31].

For a simple harmonic potential, U, $F(x) = -\nabla U = -m\omega^2 x$ and so $\frac{d^2x}{dt^2} = -\omega^2 x$. We write this as two first-order differential equations:

$$\dot{v} = -\omega^2 x \tag{C.6}$$

and

$$\dot{x} = v \tag{C.7}$$

that we then step forward, applying the Runge-Kutta algorithm on these equations together. Thus, the integration algorithm for this system would be:

$$k_{1,a} = -\omega^2 x_n, \qquad k_{1,v} = v_n \tag{C.8}$$

$$k_{2,a} = -\omega^2 (x_n + \frac{\Delta t}{2} k_{1,v}), \qquad k_{2,v} = v_n + \frac{\Delta t}{2} k_{1,a} \tag{C.9}$$

$$k_{3,a} = -\omega^2 (x_n + \frac{\Delta t}{2} k_{2,v}), \qquad k_{3,v} = v_n + \frac{\Delta t}{2} k_{2,a} \tag{C.10}$$

$$k_{4,a} = -\omega^2 (x_n + (\Delta t) k_{3,v}), \qquad k_{4,v} = v_n + (\Delta t) k_{3,a} \tag{C.11}$$

with

$$v_{n+1} = v_n + \frac{1}{6}(k_{1,a} + 2k_{2,a} + 2k_{3,a} + k_{4,a}) \tag{C.12}$$

$$x_{n+1} = x_n + \frac{1}{6}(k_{1,v} + 2k_{2,v} + 2k_{3,v} + k_{4,v}) \tag{C.13}$$

.

C.2 VV Algorithm Applied to a Physical System

The general Velocity-Verlet algorithm (from Eqs. 3.2 and 3.3) is:

$$x(t_n + \Delta t) = x(t_n) + v(t_n)\Delta t + a(t_n)\frac{\Delta t^2}{2} \tag{C.14}$$

$$v(t_n + \Delta t) = v(t_n) + [a(t_n) + a(t_n + \Delta t)]\,\frac{\Delta t}{2} \tag{C.15}$$

where we have replaced $t_0 = t_n$ for consistency with Sec. C.1. For the same system as discussed above ($\frac{d^2x}{dt^2} = -\omega^2 x$), we can apply the Velocity-Verlet algorithm as follows (using $a = \frac{d^2x}{dt^2}$ for clarity):

$$a_n = -\omega^2 x_n \tag{C.16}$$

$$x_{n+1} = x_n + (\Delta t)v_n + \frac{\Delta t^2}{2}a_n \tag{C.17}$$

$$a_{n+1} = -\omega^2 x_{n+1} \tag{C.18}$$

$$v_{n+1} = v_n + \frac{\Delta t}{2}(a_n + a_{n+1}). \tag{C.19}$$

Note that as we step this algorithm across n the two queries of the total force to calculate a_n and $a_n + 1$ are redundant (we can save a_{n+1} in memory and use it as a_n in the next timestep), so we actually only have to calculate the acceleration once per timestep (technically, once per timestep plus 1 since we calculate the acceleration twice during the very first timestep, and we generally do want the very final velocity).

Appendix D

Supplemental Figure

While getting good number resolution on the LCF-MS response is generally hard, sometimes it is illuminating and encouraging to look at the RMS response of the full crystal, since this is a much easier problem. Here is an old plot showing this.

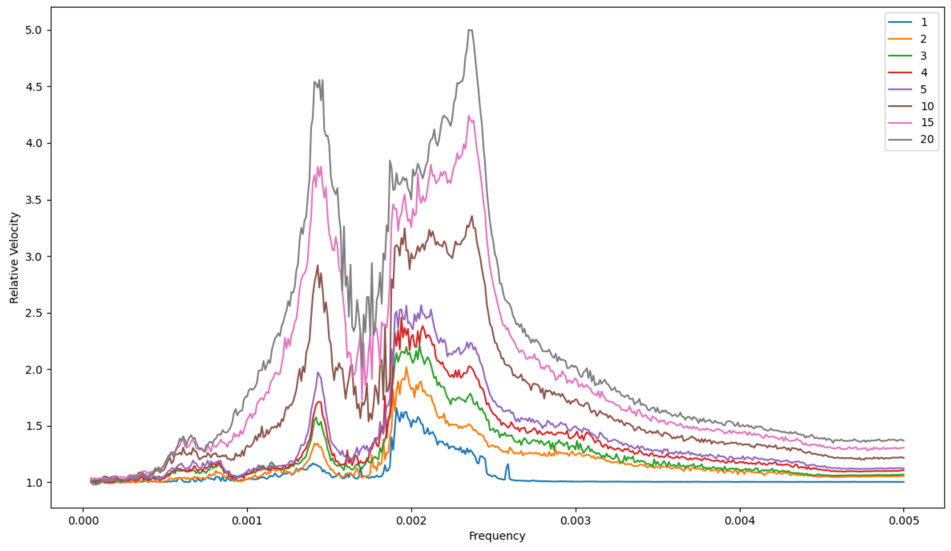


Figure D.1: RMS-velocity change in a very large, mixed crystal under a linear frequency sweep. The growth of these peaks, is very encouraging, showing that it may be possible to isolate a strong between the number of dark ions and crystal disturbance.

Bibliography

[1] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, “Trapped-ion quantum computing: Progress and challenges,” Applied Physics Reviews **6**, 021314 (2019).

[2] B. Nourse and R. Cooks, “Aspects of recent developments in ion-trap mass spectrometry,” Analytica Chimica Acta **228**, 1 (1990), ISSN 0003-2670.

[3] S. A. McLuckey and J. L. Stephenson, “Ion/ion chemistry of high-mass multiply charged ions,” Mass Spectrometry Reviews **17**, 369 (1998), ISSN 0277-7037.

[4] W.-P. Peng, Y. Cai, Y. Lee, and H.-C. Chang, “Laser-induced fluorescence/ion trap as a detector for mass spectrometric analysis of nanoparticles,” International Journal of Mass Spectrometry **229**, 67 (2003), ISSN 1387-3806.

[5] P. E. Miller and M. B. Denton, “The quadrupole mass filter: basic operating concepts,” Journal of chemical education **63**, 617 (1986).

[6] P. H. Dawson, *Quadrupole mass spectrometry and its applications* (Elsevier, 2013).

[7] D. Hanneke, B. Kuzhan, and A. Lunstad, “Optical clocks based on molecular vibrations as probes of variation of the proton-to-electron mass ratio,” Quantum Science and Technology **6**, 014005 (2020).

[8] A. Hartman, "Two-Photon Vibrational Transitions in O_2^+," (2022), Amherst College Undergraduate Thesis.

[9] A. Lunstad, "Driving Forbidden Vibrational Transitions in Molecular Oxygen," (2021), Amherst College Undergraduate Thesis.

[10] W. Henshon, "Radiofrequency circuit design for ion trapping of O_2^+ molecules," (2023), Amherst College Undergraduate Thesis.

[11] A. P. Singh, M. Mitchell, W. Henshon, A. Hartman, A. Lunstad, B. Kuzhan, and D. Hanneke, "State Selective Preparation and Nondestructive Detection of Trapped (O_2^+)," (2024), *Submitted for publication* `http://arxiv.org/abs/2410.14832`.

[12] R. A. Carollo, D. A. Lane, E. K. Kleiner, P. A. Kyaw, C. C. Teng, C. Y. Ou, S. Qiao, and D. Hanneke, "Third-harmonic-generation of a diode laser for quantum control of beryllium ions," Optics Express **25**, 7220 (2017).

[13] C. Pluchar, "An Ultraviolet Laser for Beryllium Photoionization," (2018).

[14] B. Kuzhan, "A Molecular Beam Apparatus to Search for Time-Variation of Fundamental Constants," (2021), Amherst College Undergraduate Thesis.

[15] M. Mitchell, "qlicS," (2024), URL `https://github.com/mmitch12202/qlicS`.

[16] P. B. Wigley, P. J. Everitt, A. van den Hengel, J. W. Bastian, M. A. Sooriyabandara, G. D. McDonald, K. S. Hardman, C. D. Quinlivan, P. Manju, C. C. N. Kuhn, I. R. Petersen, A. N. Luiten, *et al.*, "Fast machine-learning online optimization of ultra-cold-atom experiments," Scientific Reports **6**, 25890 (2016).

[17] D. Leibfried, R. Blatt, C. Monroe, and D. Wineland, "Quantum dynamics of single trapped ions," Rev. Mod. Phys. **75**, 281 (2003), URL `https://link.aps.org/doi/10.1103/RevModPhys.75.281`.

[18] S. Qiao, "Constructing a Linear Paul Trap System for Measuring Time-variation of the Electron-Proton Mass Ratio," (2013), Amherst College Undergraduate Thesis.

[19] E. Bentine, C. Foot, and D. Trypogeorgos, "(py)LIon: A package for simulating trapped ion trajectories," Computer Physics Communications **253**, 107187 (2020).

[20] D. Lane, "Developing a Quantum Toolbox: Experiments with a Single-Atom Harmonic Oscillator and Prospects for Probing Molecular Ions," (2017), Amherst College Undergraduate Thesis.

[21] D. Leibfried, R. Blatt, C. Monroe, and D. Wineland, "Quantum dynamics of single trapped ions," Rev. Mod. Phys. **75**, 281 (2003).

[22] H. J. Metcalf and P. van der Straten, *Laser Cooling and Trapping* (Springer, 1999).

[23] F. Schmid, J. Weitenberg, J. Moreno, T. W. Hänsch, T. Udem, and A. Ozawa, "Number-resolved detection of dark ions in Coulomb crystals," Phys. Rev. A **106**, L041101 (2022).

[24] D. Donnelly and E. Rogers, "Symplectic integrators: An introduction," American Journal of Physics **73**, 938 (2005).

[25] S. Brorson, "Numerically Solving Ordinary Differential Equations," (2022), (electronic), URL `https://math.libretexts.org/@go/page/108067`.

[26] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in 't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, *et al.*, "LAMMPS - a flexible simulation tool for particle-based

materials modeling at the atomic, meso, and continuum scales," Comp. Phys. Comm. **271**, 108171 (2022).

[27] W. C. Swope, H. C. Andersen, P. H. Berens, and K. R. Wilson, "A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters," The Journal of Chemical Physics **76**, 637 (1982), ISSN 0021-9606.

[28] A. Wagaman and L. Spector, "Acquisition of High Performance Computing System for Interdisciplinary Research and Teaching," National Science Foundation (2021), grant number: 2117377.

[29] M. R. Hush, "M-LOOP Documentation," (2024), accessed: 2024-11-22, URL `https://m-loop.readthedocs.io/en/stable/`.

[30] U. Fröhlich, *Coulomb Crystal Studies, Sympathetic Cooling, and Mass Spectrometry Using Laser-Cooled Be+ Ions*, Ph.D. thesis, Heinrich-Heine-Universität Düsseldorf, Düsseldorf (2008).

[31] L.-H. Lyu, "Appendix C of NSSP Notes," (2024), URL `http://www.ss.ncu.edu.tw/~lyu/lecture_files_en/lyu_NSSP_Notes/Lyu_NSSP_AppendixC.pdf`.