Classical Optimization Techniques for a Trapped Ion Quantum Simulator

by

Hin Pok Cyrus Fung

A report
presented to the University of Waterloo
in fulfillment of the
report requirement for
PHYS 437A Research Project

Waterloo, Ontario, Canada, 22 August 2022

Author's Declaration

I hereby declare that I am the sole author of this report. This is a true copy of the report, including any required final revisions, as accepted by my examiners.

I understand that my report may be made electronically available to the public.

Abstract

As part of the design of ion traps, we must determine the appropriate voltages that should be applied to an ion trap during operation, to manipulate ions in the desired fashion. We approach this problem using various numerical optimization schemes in this research. However, since we have insufficient control over the system in our ion trap, tradeoffs must be made to obtain an acceptable solution.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Ion traps

Ion traps have emerged as a promising platform for both quantum computing[1] and quantum simulation [2]. For example, "[T]he use of [such] quantum simulators for studying spin models may inform our understanding of exotic quantum materials and shed light on the behavior of interacting quantum systems that cannot be modeled with conventional computers."

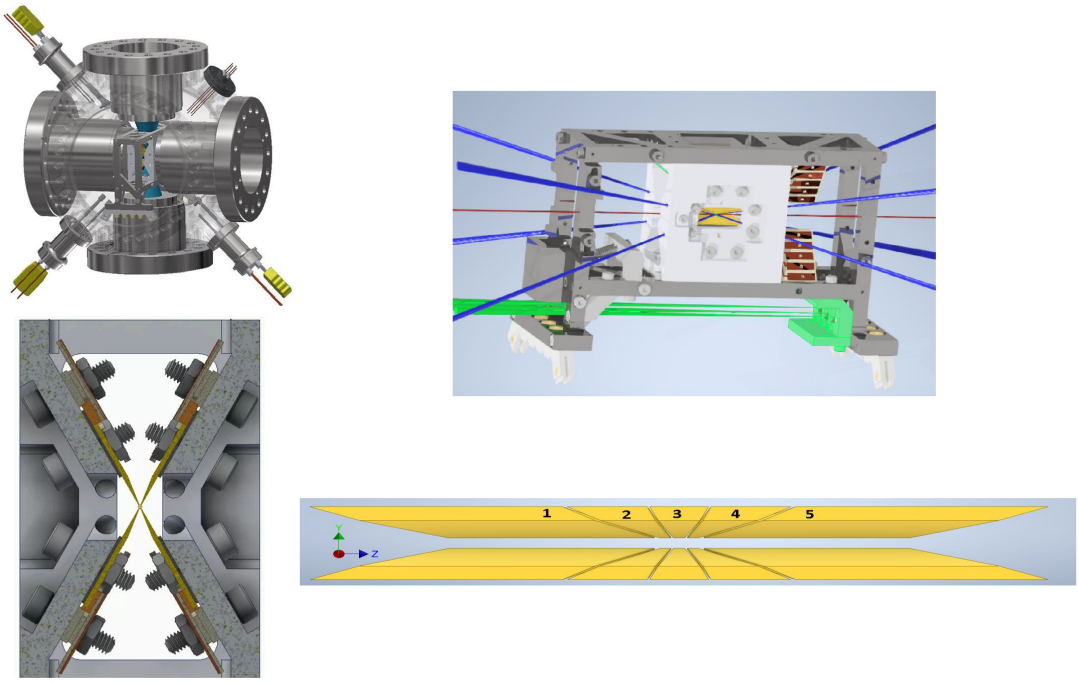
In particular, our lab uses a segmented blade trap, a variation of the commonly used Paul trap, which has a number of distinct advantages, such as the ability to be made small. Moreover, it is optically open, ie it allows "flexible laser access from multiple directions, allowing for the variety of qubit manipulations necessary in modular architectures."[3]

1.2 Problem statement: the need for optimization

To operate the blade trap, we must apply appropriate voltages to the blade segments. In particular, choosing suitable voltages in order to manipulate ions in the desired fashion is a non-trivial problem, since we have control over (as we will see) 21 voltage parameters in the trap.

Thus the dimension of the search space is large. Moreover, this number is anticipated to increase, as the architecture is scaled up to capture higher number of ions (eg if more blade segments are used). This implies that a brute-force search or choosing voltages manually

Figure 1.1: Example of a blade trap.



are not viable approaches, since the search space cannot be explored exhaustively.

In addition, in different experiments, we may wish to configure the ions differently (eg rotated principal axis, translated ion chain). This further suggests the need for an automated search program.

In this research project, this problem is approached as a computational optimization problem. The goal is to write a computer program, which uses optimization techniques, such that when given experimental parameters of interests as input (to be elaborated in the following), the program outputs the voltages that create such a setup. Moreover, the program should be able to characterize the resulting field, such that a human operator can decide whether it is a good solution (whether it satisfies the experimental requirements).

1.3 Previous work

In engineering, we often encounter the scenario where we have control over some parameter(s) A, and we want to adjust A such that a different parameter(s) B falls within a certain acceptable range. This is known as a control problem.

It should be noted that optimization techniques have also been applied to other control problems arising from the design of trapped-ion systems.[4]

Chapter 2

Experimental or Theoretical Techniques

We first go through the setup of the problem (the physics involved) before covering our optimization scheme/approach.

2.1 Overview: physics

In general, we have n ions ($^{171}\mathrm{Yb}^{+}$) in our blade trap. These ions are subject to the electric field generated by the blade trap, and also the Coulomb interaction among themselves. We index the ions by $\mu = 1, 2, \ldots, n$. The μth ion has position $\vec{r}_\mu = (x_\mu, y_\mu, z_\mu)$. The total potential energy of the system is

$$V = \left(\sum_\mu q\phi_{\text{blade}}(\vec{r}_\mu) \right) + V_{\text{Coulomb}} = \sum_\mu V_{\text{blade}}(\vec{r}_\mu) + \sum_{\mu<\nu} \frac{1}{4\pi\epsilon_0} \frac{q^2}{\|\vec{r}_\mu - \vec{r}_\nu\|}$$

where ϕ_{blade} is the electric potential and V_{blade} is the potential energy associated with the field generated by the blade segments.

Our problem is to find voltages so that the ions are in the desired configuration, specifically

- They have the right equilibrium positions.
- They have the right center of mass (COM) modes and x, y eigenfrequncies (to be explained in Section 2.1.3).

2.1.1 Forward problem: segment voltages to V_{blade} field

We first consider how the voltages generate the V_{blade} field.

AC

Note that by Earnshaw's theorem, an electrostatic field cannot trap ions. The solution adopted by ion traps is to use a time-varying field $E(r,t) = E_0(r)\cos(\Omega t)$, essentially a rotating hyperbola.[5] The time varying field is implemented by applying an AC current. The effective potential energy (whose gradient is the average force) experienced by a charge q with mass m is $V_{\text{pseudo}} = \frac{q^2 E_0^2(r)}{4m\Omega^2}$.

DC

In addition, we also apply a DC offset to the blade segments. By the superposition principle, the DC offset generates an electric potential that is added to V_{21}, which provides additional control over the shape of V_{blade}.

The superposition principle also implies that the field generated by each blade segment can be considered independently from each other. Let V_l be the electric potential generated when 1 Volt is applied to the lth segment and all other segments have 0 Volts. The V_l field can then be found by solving the boundary value problem $\nabla^2 V_l = 0$. This was done numerically in the multiphysics simulation software COMSOL before this project.

AC revisited

The time-varying field $E(r,t) = E_0(r)\cos(\Omega t)$ is generated by applying a voltage of $v_{\text{AC}}\cos(\Omega t)$ to the diagonal blades, and $-v_{\text{AC}}\cos(\Omega t)$ to the anti-diagonal blades. Again the electric potential ϕ_{AC} (such that $E_0(r) = -\nabla\phi_{\text{AC}}$) satisfies the boundary value problem.

Consider the case where the we apply 1 Volt to the diagonal blades and -1 Volt to the anti-diagonal blades. This is a superposition of the V_l fields we found in the DC calculation. Taking the negative of the gradient, we find $E_1(r)$. In general $E_0(r) = E_1(r)v_{\text{AC}}$ by the superposition principle, so $V_{\text{pseudo}} = \frac{q^2 E_0^2(r)}{4m\Omega^2} = \frac{q^2 E_1^2(r)}{4m\Omega^2} v_{\text{AC}}^2$. Let $v_{21} = v_{\text{AC}}^2$ and $V_{21} = \frac{q^2 E_1^2(r)}{4m\Omega^2}$, then $V_{\text{AC}} = V_{\text{pseudo}} = v_{21}V_{21}$.

In general, let v_l be the voltage of the lth blade segment. Each v_l can assume any arbitrary

value, subject to constraints imposed by the equipment. If V_l is the V field generated by the lth blade segment, by the superposition principle, the field generated by any combination of voltages on the blade segments is

$$V_{\text{blade}} = V_{\text{DC}} + V_{\text{AC}} = \left(\sum_{l=1}^{20} v_l V_l \right) + v_{21} V_{21} = \sum_{l=1}^{21} v_l V_l$$

Next, since we are only interested in a small region around the origin (the ions are within dozens of microns from the origin), we can expand each V_l as a Taylor series around the origin

$$V_l = \sum_{i,j,k} \alpha_{i,j,k,l} x^i y^j z^k$$

We choose to only consider $i, j = 0, 1, 2$ and $k = 0, 1, 2, 3, 4$ and neglect higher order terms (which are small around the origin). For clarity, we write $m = (i, j, k)$ (ie we flatten the $(3, 3, 5)$ array into a 1D array), hence

$$V_l = \sum_{m} \alpha_{ml} x^i y^j z^k$$

So

$$\begin{aligned} V_{\text{blade}} &= \sum_{l=1}^{21} v_l V_l = \sum_{l=1}^{21} v_l (\sum_{m} \alpha_{ml} x^i y^j z^k) = \sum_{l=1}^{21} \sum_{m} v_l \alpha_{ml} x^i y^j z^k \\ &= \sum_{m} \sum_{l=1}^{21} \alpha_{ml} v_l x^i y^j z^k = \sum_{m} (\sum_{l=1}^{21} \alpha_{ml} v_l) x^i y^j z^k = \sum_{m} b_m x^i y^j z^k \end{aligned}$$

Therefore the overall coefficients b_m are obtained by multiplying the matrix $[\mathbf{A}]_{ml} = \alpha_{ml}$ by the vector $[\mathbf{v}]_l = v_l$, which are the segment voltages (except for $v_{21} = v_{\text{AC}}^2$). This is expected, since the map from the boundary voltages to the V field is linear, and the map from V to its Taylor series coefficients is also linear, so the overall transformation is also linear.

2.1.2 Equilibrium positions

After the ions are loaded into the trap, they eventually settle in their equilibrium positions $\mathbf{q}_{\text{eqm}} = \text{argmin}_{\mathbf{q}} V(\mathbf{q})$. ie the total potential energy V is minimized (reaches its global

minimum), and that the net force acting on each ion is zero ($\mathbf{F}_{\text{blade}}$ completely cancels out $\mathbf{F}_{\text{Coulomb}}$).

The ions are essentially stationary except for micromotion, which comes from the fact that V_{pseudo} is in fact a rapidly varying field.[5] Hence the ions oscillate rapidly around their equilibrium positions; the amplitude of this oscillation is small and thus can be neglected.

2.1.3 Eigenmodes and eigenfrequencies

As stated in [6], '[I]n a crystallized ion chain configuration, the Coulomb interactions act as "springs" connecting ions and permit phonons of different collaborative vibrational modes. By introducing spin-dependent forces to the system, we can have phonon mediated spin-spin interactions.' This spin-spin interaction is then used to engineer different Hamiltonians for quantum simulation.

Therefore, we are interested in the vibrational modes of the ions. As shown in[7], they are calculated as follows. Consider the generalized coordinates
$\mathbf{q} = (x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n, z_1, z_2, \ldots, z_n)$ and the matrix $\mathbf{C} = \frac{1}{m}\text{Hess}(\phi)$, where m is the mass of each ion and $[\text{Hess}(\phi)]_{ij} = \frac{\partial^2 \phi}{\partial q_i \partial q_j}$ is the Hessian of ϕ.

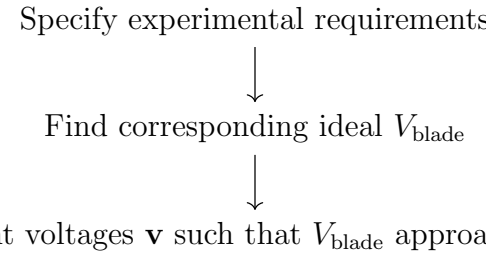
By solving the Euler-Lagrange equations and considering $\mathbf{q} = \mathbf{q}_{\text{eqm}}$, we obtain $\ddot{q}_k = -\sum_j C_{kj} q_j$,[1] where C_{kj} is evaluated at $\mathbf{q}_{\text{eqm}}$. By comparing this equation with the simple harmonic equation $\ddot{q}_k = -\omega^2 q_k$, we see that the eigenmodes are the eigenvectors of $\mathbf{C}$, and the eigenfrequencies[2] are the square root $\sqrt{\lambda}$ of the eigenvalues λ of $\mathbf{C}$.

In particular, center of mass modes are desirable, because they enable coupling between any pair of ions. In a COM mode, all ions oscillate in phase, and ideally with the same amplitude.

2.2 Optimization

We approach the control problem as follows: first, we find the ideal V_{blade} field that matches the requirements of a specific experimental setup; we then find the voltages that best creates this ideal field.

[1] To obtain this expression, we also need to redefine $\mathbf{q}' = \mathbf{q} - \mathbf{q}_{\text{eqm}}$. We write $\mathbf{q}'$ as $\mathbf{q}$ for simplicity.
[2] Angular eigenfrequencies, ie with a factor of 2π.



This optimization process has been fully implemented in a computer program.

2.2.1 Target specification

We express the target field in the form $V_{\text{target}} = \sum_{i,j,k} b_{i,j,k} x^i y^j z^k$ (recall $i, j = 0, 1, 2$ and $k = 0, 1, 2, 3, 4$). Ideally, if $V_{\text{blade}} = V_{\text{target}}$, then experimental requirements are fulfilled.

Note that this step is highly customizable. For example, if we want to carry out an experiment with some arbitrary $b_{i,j,k}$, then these coefficients can be manually inputted into the program, which optimizes for this target field.

However in general, we often restrict ourselves to target fields of the form $V_{\text{target}} = b_{2,0,0}x^2 + b_{0,2,0}y^2 + b_{0,0,2}z^2 + b_{0,0,4}z^4$. This is because of the following considerations.

By default, we assume the ion chain lies on the z-axis. To achieve this, we want much higher trapping strength in the x and y directions compared to the z direction (so that any deviation from the z-axis will be extremely energetically unfavorable, which implies that the equilibrium positions are very close to the z-axis). This also implies that often, we can treat this as a 1D problem, since the x and y trapping can be made strong enough to counteract any other effects,[3] such that the ions stay as a 1D chain.

We use harmonic trapping (ie x^2 and y^2), such that $b_{2,0,0} = \frac{1}{2}m\omega_x^2$ for the ideal field, where ω_x is the eigenfrequency of the x COM mode; similarly $b_{0,2,0} = \frac{1}{2}m\omega_y^2$. This is related to the fact that when deriving the vibrational modes, we assumed that V can be approximated by a harmonic oscillator locally.

Additional considerations are as follows:

Equilibrium positions

For instance, we may want the ions to be in specific positions to facilitate optical addressing. This may imply that we want the ions to be equidistant from each other. In addition,

[3] The x and y trapping, x^2 and y^2, is mostly provided by V_{pseudo}. Increasing v_{AC} increases V_{pseudo}.

we want the ions to maintain a minimum separation d_m from each other to minimize crosstalk.[8] Typically, $d_m \approx 4$ μm.

In general, we also want to pack as many ions as possible in a given region. If a minimum separation is to be maintained, this also implies that the ions should be roughly equidistant from each other, such that the separation distance is roughly equal to d_m (as close to d_m as possible, but not smaller than d_m).

Note that $V_{\text{target}}(x, y, z) = b_{2,0,0}x^2 + b_{0,2,0}y^2 + b_{0,0,2}z^2 + b_{0,0,4}z^4$ can be easily modified for different scenarios. For example, if we want to translate the ion chain in the z direction, we can consider the translated field $V'_{\text{target}}(x, y, z) = V_{\text{target}}(x, y, z - \Delta z)$. Similarly, if we want to rotate the ion chain,[4] we can use $V'_{\text{target}}(x, y, z) = V_{\text{target}}(x', y', z')$, where (x', y', z') are the rotated coordinates.[5] Such V'_{target} will still be of the form $\sum_{i,j,k} b_{i,j,k}x^i y^j z^k$.

Eigenmodes and eigenfrequencies

Since $V_{\text{target}} = b_{2,0,0}x^2 + b_{0,2,0}y^2 + b_{0,0,2}z^2 + b_{0,0,4}z^4$ has no "cross-terms" (eg terms such as $x^i y^j$ or $y^j z^k$), Hess(ϕ) is block diagonal, such that each normal mode is in either the x, y, or z direction. This is desirable experimentally.

Similarly, we want $\omega_x \neq \omega_y$ (asymmetric x and y trapping), so that exciting the x COM mode does not excite the y COM mode (and vice versa). Experimentally, we typically use $\omega_x = 5$ MHz $\times 2\pi$ and $\omega_y = 4.8$ MHz $\times 2\pi$; we want V_{target} to have such COM eigenfrequencies (hence x and y trapping strength).

Under $V_{\text{target}} = b_{2,0,0}x^2 + b_{0,2,0}y^2 + b_{0,0,2}z^2 + b_{0,0,4}z^4$, all ions have the same oscillation amplitude in the x (and y) COM modes. This can be verified in a direct calculation of the eigenmodes from the Hessian. Again this is because the only x and y terms in V_{target} are x^2 and y^2, such that the x (and y) trapping is uniform for all z along the z-axis.

For the actual V_{blade}, the oscillation amplitude of the COM mode may not be completely uniform. However it should still be roughly uniform, so that all pairs of ions can be coupled through phonon mediated spin-spin interactions.[6]

[4] In addition to rotating the ion chain, such that it is no longer along the z-axis, we may also want to rotate the eigenmodes; eg instead of pointing in $\hat{x}$, $\hat{y}$, $\hat{z}$, the eigenmodes now point in $\frac{1}{2}(\hat{x}+\hat{y})$, $\frac{1}{2}(-\hat{x}+\hat{y})$, $\hat{z}$. This is achieved by a rotation around the z-axis.

[5] eg if we want to rotate the ion chain using the rotation matrix R, then (x', y', z') is found by applying R^{-1} to (x, y, z).

[6] If the amplitude of some ions is too small, then they are effectively decoupled from the other ions; the spin-spin is too weak. In extreme cases, the amplitude of some ions may be zero; then the system does not have a true COM mode.

Computing $b_{0,0,2}$ and $b_{0,0,4}$

Note that at this point, the coefficients $b_{0,0,2}$ and $b_{0,0,4}$ have not been determined. In principle, they can be determined through a nested optimization: we specify an initial pair of $b_{0,0,2}$ and $b_{0,0,4}$ coefficients. In each iteration, we compute the equilibrium positions corresponding to this V_{target}. We then adjust $b_{0,0,2}$ and $b_{0,0,4}$ so that the equilibrium positions become closer to the target equilibrium positions $\mathbf{q}_{\text{target}}$. This can be written succinctly as

$$\begin{aligned}&\text{argmin}_{(b_{0,0,2},b_{0,0,4})}\|\mathbf{q}_{\text{eqm}} - \mathbf{q}_{\text{target}}\| \\ &= \text{argmin}_{(b_{0,0,2},b_{0,0,4})}\| \left(\text{argmin}_{\mathbf{q}} V(\mathbf{q})\right) - \mathbf{q}_{\text{target}}\|\end{aligned}$$

In practice, a nested optimization is too computationally expensive (it takes too long). A heuristic algorithm is used in the computer program instead: if $\mathbf{q}_{\text{target}}$ is indeed the equilibrium position, then the net force at $\mathbf{q}_{\text{target}}$ should be zero. Thus the heuristic algorithm minimizes the gradient of V at $\mathbf{q}_{\text{target}}$, ie

$$\text{argmin}_{(b_{0,0,2},b_{0,0,4})} \left\| \nabla V\big|_{\mathbf{q}_{\text{target}}} \right\|$$

So there is only one loop of optimization. This is only a heuristic, because even if $\nabla V\big|_{\mathbf{q}_{\text{target}}} \approx 0$, this does not imply $\mathbf{q}_{\text{target}}$ is a local minimum.[7] And even if $\mathbf{q}_{\text{target}}$ is a local minimum, this does not imply it is the global minimum.

Finally, it should be noted that instead of $V_{\text{target}} = b_{2,0,0}x^2 + b_{0,2,0}y^2 + b_{0,0,2}z^2 + b_{0,0,4}z^4$, we may also consider the simpler target $V_{\text{target}} = \frac{1}{2}m\omega_x^2 x^2 + \frac{1}{2}m\omega_y^2 y^2 + \frac{1}{2}m\omega_z^2 z^2$. This has the advantage that it is not necessary to do an optimization to determine the b coefficients. In practice the heuristic algorithm for the quartic potential $V_{\text{target}} = b_{2,0,0}x^2 + b_{0,2,0}y^2 + b_{0,0,2}z^2 + b_{0,0,4}z^4$ works well enough, so we typically use the quartic potential instead of the quadratic potential to achieve better ion equilibrium positions.

2.2.2 Voltage optimization

As shown above, given the segment voltages $\mathbf{v}$, we can find the resulting $V_{\text{blade}} = \sum_m b_m x^i y^j z^k$ through the matrix equation $\mathbf{A}\mathbf{v} = \mathbf{b}$, where $[\mathbf{b}]_m = b_m$ are the coefficients in the polynomial. This is the forward problem.

During voltage optimization, we are interested in the inverse problem. We already have a

[7] eg consider $f(x, y) = 0.001x$.

target $\mathbf{b}$ from V_{target}. We want to find $\mathbf{v}$ such that $\mathbf{A}\mathbf{v} = \mathbf{b}$.[8]

However in practice, since $\mathbf{A}$ is a 45×21 matrix, it is not invertible. In fact by the rank-nullity theorem, the range of $\mathbf{A}$ has a dimension of 21 at most; whereas the codomain has dimension 45, ie $\mathbf{b} \in \mathbb{R}^{45}$. Thus the matrix equation is overdetermined, ie for an arbitrary $\mathbf{b}$, we may not always find a solution $\mathbf{v}$. We do not have sufficient control: there are 45 controlled variables,[9] but only 21 manipulated variables.[10] This contributes greatly to the difficulty of the voltage control problem.

As such, we only demand $\mathbf{A}\mathbf{v} \approx \mathbf{b}$, ie we look for an approximate solution such that the resulting V_{blade} approaches V_{target}. This can be formulated as an optimization problem: given a loss function $L(\mathbf{b}_{\text{blade}}, \mathbf{b}_{\text{target}})$ which quantifies the "distance" between V_{blade} and V_{target}, we want to find $\operatorname{argmin}_{\mathbf{v}} L(\mathbf{b}_{\text{blade}}, \mathbf{b}_{\text{target}}) = \operatorname{argmin}_{\mathbf{v}} L(\mathbf{A}\mathbf{v}, \mathbf{b}_{\text{target}})$. In any optimization problem, specifying the loss function is a key step.[11]

In this work, we use the 2-norm function[12] for L, because the linear least squares problem has been studied extensively, such that its properties are well-known, and that its solution has already been implemented in existing numerical packages in optimized, high performance code. It is also less computationally expensive than other choices of L;[13] plus the 2-norm agrees with the intuitive notion of distance.

In particular, the solution to the linear least squares problem always exists and is unique, and has a simple closed-form expression, ie the Moore–Penrose pseudoinverse. We explore this in more detail in the implementation section.

In addition, due to physical constraints of the hardware, not all voltages can actually be attained. For example, the voltages on the end segments can only be within $[0, 1000]$ Volts. ie we vary the voltages subject to these constraints, and the constrained (or bounded) optimization problem can be expressed as $\operatorname{argmin}_{\mathbf{v} \in S_{\text{physical}}} L(\mathbf{A}\mathbf{v}, \mathbf{b}_{\text{target}})$, where S_{physical} is the

[8] $\mathbf{A}$ depends on the hardware, ie the geometry of the blade trap, so $\mathbf{A}$ stays fixed as we adjust the voltages.

[9] In control theory, controlled variables are the variables whose values should be close to some specified value.[9]

[10] Variables that we can directly adjust.

[11] If the loss function is specified incorrectly, the optimizer will not behave as expected. An extreme example is the hypothetical "paperclip maximizer".

Some loss functions are computationally intractable, such that they cannot be computed on any existing computer within a reasonable time.

[12] ie the Euclidean distance $\|u - v\| = \sqrt{\sum_{i=1}^{n} (u_i - v_i)^2}$.

[13] For instance, we can define L as $\|\mathbf{q}_{\text{eqm}} - \mathbf{q}_{\text{target}}\|$. But again this would be a nested optimization, since $\mathbf{q}_{\text{eqm}}$ must be computed through an inner optimization loop. Moreover, there is no guarantee that the resulting eigenmodes and eigenfrequencies are close to that of V_{target}.

set of physical voltages, attainable in the blade trap.

In this research, we rewrite our problem as a linear least squares problem in two different ways.

Coefficient-level optimization

We compare the coefficients $\mathbf{b}_{\text{blade}}$ and $\mathbf{b}_{\text{target}}$ directly, ie the loss function is $L = \|\mathbf{Av} - \mathbf{b}_{\text{target}}\|$. However in practice, the solution to this has poor performance, namely because error in different coefficients $[\mathbf{b}]_m = b_m = b_{i,j,k}$ have different effects, and our system has different sensitivity to different coefficients.[14]

To reflect this, we weight each coefficient differently. For instance, if the system is very sensitive to changes to a particular coefficient, we assign a higher weight to this coefficient. The loss function of the resulting weighted least squares problem (which is still linear) is $L = \|\mathbf{WAv} - \mathbf{Wb}_{\text{target}}\|$, where $\mathbf{W}$ is a diagonal matrix, whose diagonal components is the weight for that dimension.[15]

The optimizer thus needs to prioritize errors in different coefficients according to their weights. Since in general, an exact solution $\mathbf{Av} = \mathbf{b}_{\text{target}}$ does not exist, tradeoffs must be made to obtain an acceptable solution; we need to sacrifice accuracy in certain terms if they have less physical significance. A weighted optimization reflects this tradeoff.

However, in the current implementation, these weights are found manually through trial and error. It can be time-consuming to try different combinations of weights before an acceptable solution is found; there is also no guarantee that a particular set of weights gives the best possible solution, so fine-tuning is always necessary.

[14] The 2-norm is symmetric with respect to each component. As a simplified example, suppose $\mathbf{b}_{\text{target}} = (0, 0)$. Then $\mathbf{b}_{\text{blade}} = (1, 0)$ and $\mathbf{b}_{\text{blade}} = (0, 1)$ both have the same loss value, so they are both equally good from the point of view of the optimizer.

However, the same amount of absolute error in different components leads to different level of physical effects. As an extreme example, any changes to the constant coefficient $b_{0,0,0}$ have no physical effects whatsoever, so we do not care about the error in this coefficient.

More generally, even after non-dimensionalization, typically the $b_{i,j,k}$ coefficients have very different orders of magnitude. eg for $\mathbf{b}_{\text{target}}$, we typically have $b_{2,0,0} \sim 0.5$, while $b_{0,0,4} \sim 10^{-7}$ or less. Thus the same amount of absolute error leads to very different relative error hence physical effects in different coefficients.

[15] Essentially, instead of $L(u, v) = \|u - v\| = \sqrt{\sum_{i=1}^{n}(u_i - v_i)^2}$, we now have $L'(u, v) = \sqrt{\sum_{i=1}^{n}(w_i(u_i - v_i))^2}$. The error in the ith dimension is now weighted by w_i, so if w_i is large, even a small $\|u_i - v_i\|$ leads to a large contribution to L. The optimizer is thus incentivized to place higher priority to errors in the ith dimension.

Grid-level optimization

To bypass this issue, another approach is to evaluate $\mathbf{V}_{\text{blade}}$ and $\mathbf{V}_{\text{target}}$ over a grid and compare their values.

Note that evaluating $\mathbf{V}_{\text{blade}}$ on a grid is achieved by performing a linear transformation on $\mathbf{b}_{\text{blade}}$. ie suppose $V_{\text{blade}} = \sum_m b_m x^i y^j z^k$. Evaluating V_{blade} at (x_1, y_1, z_1) gives $V_{\text{blade}}(x_1, y_1, z_1) = \sum_m b_m x_1^i y_1^j z_1^k$. In general, if we want to evaluate $\mathbf{V}_{\text{blade}}$ at points (x_p, y_p, z_p) from $p = 1$ to N and arrange the results into a vector, the resulting vector is $\mathbf{E}\mathbf{b}_{\text{blade}}$ where $[\mathbf{E}]_{pm} = x_p^i y_p^j z_p^k$ and $[\mathbf{E}\mathbf{b}_{\text{blade}}]_p = V_{\text{blade}}(x_p, y_p, z_p)$. Therefore $L = \|\mathbf{E}\mathbf{A}\mathbf{v} - \mathbf{E}\mathbf{b}_{\text{target}}\|$.

For simplicity, we sample the points (x_p, y_p, z_p) from a grid. The grid points can be evenly or unevenly spaced.[16] We can vary the range of the grid as well as the separation between grid points. Note that different grids leads to different loss functions hence different solutions. Thus the weights in coefficient-level optimization and the grid in grid-level optimization are both examples of "hyperparameters",[17] parameters that are specified manually before the optimization, and are not adjusted during the optimization.

Then it appears that grid-level optimization, like coefficient-level optimization, also suffers from "hyperparameter-tuning" (the need to fine-tune hyperparameters manually). However in practice, it is usually easier to fine-tune the grid than to fine-tune the weights, because there are fewer parameters to specify,[18] and because the grid is a "higher-level description" compared to weights.[19]

[16] An example of an evenly spaced (1D) grid is $[0, 1, 2, 3, 4, 5]$. An unevenly spaced grid is $[0, 2, 2.333, 2.666, 3, 5]$; we may use an unevenly spaced grid when we want to include more points in a region of interest, ie when error in that region matters more.

[17] Technically speaking, hyperparameters show up in machine learning rather than optimization. However, the "hyperparameters" in our optimization problem is completely analogous to the hyperparameters in machine learning.

[18] Consider an evenly spaced grid. Assuming the ion chain is along the z-axis and is centered at the origin, by symmetry, we want the grid to have the same extent in the $-x$ and $+x$ direction (similarly for y and z). So we only need to specify the extent of the grid in the $+x$, $+y$, and $+z$ directions, as well as the grid spacing in the x, y, and z directions, so there are 6 parameters in total. In contrast, there are $3 \cdot 3 \cdot 5 = 45$ weights.

[19] Instead of working with individual weights/coefficients, grid-level optimization focuses on the overall shape of $\mathbf{V}_{\text{blade}}$. Therefore, the user does not have to manually determine the sensitivity in each coefficient (which involves running the optimization many times to gain intuition), then adjust the weights accordingly.

In particular, working with weights means that the user has to determine the relative significance of each coefficient, and decide how to balance the different physical effects by choosing a particular set of

2.3 Characterization

Since we are working with approximate solutions, we must characterize the solution obtained from the optimizer to decide if it actually satisfies the experimental requirements.[20] As a simple check, we can check if $\mathbf{v}$ given by the optimizer is actually realizable in the blade trap. We should also check if the equilibrium positions actually lie on the z-axis; if the x and y trapping is too weak, this assumption can be violated.

Determining whether a solution is acceptable depends on the details of a particular experiment, so this step cannot be automated. For example, in some optical addressing schemes, the equilibrium positions[21] must be at the target positions exactly (if the position of the optical equipment is fixed), so this condition cannot be relaxed.

However in other schemes, the optical equipment can be adjusted to accommodate any ion positions (provided that the ion spacing is greater than d_m). In that case, large deviations between the actual and target equilibrium positions can be tolerated — especially if this leads to higher accuracy in the eigenmodes and eigenfrequencies. Examples of such tradeoffs in practice will be given in Results and Discussion.

2.4 Implementation

The voltage optimization and characterization program is implemented in python, in a Jupyter notebook. Numerical packages such as NumPy and SciPy are used. We now revisit the entire optimization and characterization workflow, focusing on the most important implementation details.

2.4.1 Nondimensionalization

The characteristic length scale in this problem is $L_0 = 10^{-6}$ m, since $d_m = 4 \cdot 10^{-6}$ m. Similarly, the natural energy scale is $V_0 = \frac{q^2}{4\pi\epsilon_0}\frac{1}{L_0}$, the potential energy due to the Coulomb

weights. On the other hand, in grid-level optimization, the optimizer automatically balances all effects, by demanding that the overall shape of $\mathbf{V}_{\text{blade}}$ should approach that of $\mathbf{V}_{\text{target}}$. This is a more "global" approach.

[20] Even though $\mathbf{V}_{\text{target}}$ does satisfy the experimental requirements, $\mathbf{V}_{\text{blade}}$ may differ from $\mathbf{V}_{\text{target}}$ sufficiently significantly, such that they exhibit qualitatively different behavior.

[21] When we computed $\mathbf{b}_{\text{target}}$, ie when we optimized $b_{0,0,2}$ and $b_{0,0,4}$, we assumed that the total number of ions is n. Here, during characterization, we also assume the total number of ions is n. Note that the number of ions must be specified when we calculate the equilibrium positions and the eigenmodes/eigenfrequencies.

interaction between two ions with L_0 separation. Thus if we use SI units, the numbers will be extremely small.[22] In particular, the coefficients $\alpha_{i,j,k,l}$ has dimensions $\frac{\text{energy}}{\text{length}^{i+j+k}}$, so the entries of the $\mathbf{A}$ matrix vary greatly in their orders of magnitude. Thus the matrix vector multiplication $\mathbf{Av}$ is numerical unstable.

To solve this, we nondimensionalize position and energy[23] through $x \to \tilde{x} = \frac{x}{L_0}$ and $V \to \tilde{V} = \frac{V}{V_0}$. For example, upon nondimensionalization,

$$V_{\text{blade}} = \sum_m b_m x^i y^j z^k = \sum_m b_m L_0^{i+j+k} (\frac{x}{L_0})^i (\frac{y}{L_0})^j (\frac{z}{L_0})^k$$
$$\tilde{V} = \frac{V_{\text{blade}}}{V_0} = \sum_m \frac{b_m L_0^{i+j+k}}{V_0} (\frac{x}{L_0})^i (\frac{y}{L_0})^j (\frac{z}{L_0})^k = \sum_m \tilde{b}_m \tilde{x}^i \tilde{y}^j \tilde{z}^k$$

ie $b_m \to \tilde{b}_m = \frac{b_m L_0^{i+j+k}}{V_0}$.

We typically nondimensionalize before computations (eg optimization).

2.4.2 Linear least squares

The solution to the linear least squares problem $\text{argmin}_{\mathbf{v}} \|\mathbf{Av} - \mathbf{b}\|$ always exists, and has a unique solution if we further impose $\text{argmin}_{\mathbf{v}} \|\mathbf{v}\|$. In fact, this solution has a closed-form expression, $\mathbf{v} = \mathbf{A}^+ \mathbf{b}$, where $\mathbf{A}^+$ is the Moore–Penrose pseudoinverse.

While there is a closed form expression $\mathbf{A}^+ = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T$ provided that $\mathbf{A}^T \mathbf{A}$ is invertible, this is numerically unstable.[24] To solve this, numerical algorithms compute the rank of the matrix first in a separate step, by calculating the singular values. If a singular value is smaller than some cut-off, it is taken to be zero. Therefore we use existing code implementations in NumPy and SciPy.

[22] Especially for terms like $x^2 y^2 z^4$; its size will be around $L_0^{2+2+4} = 10^{-48}$ m.

[23] The fundamental dimensions are M (mass), L (length), and T (time). The dimension of energy is $\frac{ML^2}{T^2}$, so we are free to fix the energy scale after fixing the length scale.

The voltage is independent from both length and energy. Since the voltage we use is typically within 0.01 to 1000 volts, we do not nondimensionalize the voltage.

[24] If we evaluate the determinant of $\mathbf{A}^T \mathbf{A}$ numerically, even if $\mathbf{A}^T \mathbf{A} = 0$, we may get a nonzero number due to numerical error. eg the matrix $\begin{pmatrix} 1 & 0 \\ 0 & 10^{-20} \end{pmatrix}$ is technically invertible, but in practice the 10^{-20} entry should probably be zero; it is nonzero due to numerical error.

2.4.3 Existing code

TrICal

First, note that we used an existing python package, TrICal, written by another member of the research group. This package implements a number of calculations that are used in ion traps, eg calculating the ion equilibrium positions, eigenmodes, and eigenfrequencies.

The equilibrium positions are calculated as follows. First, the polynomial potential V_{blade} and the Coulomb potential V_{Coulomb} are nondimensionalized. Then the optimization $\tilde{\mathbf{q}}_{\text{eqm}} = \text{argmin}_{\tilde{\mathbf{q}}} \tilde{V}(\tilde{\mathbf{q}})$ is performed using `scipy.optimize.minimize` with `method='SLSQP'`, ie Sequential Least Squares Programming.[25] See [10] for details of this algorithm. Finally we dimensionalize $\tilde{\mathbf{q}}_{\text{eqm}}$ to obtain $\mathbf{q}_{\text{eqm}}$.

When computing the eigenmodes and eigenfrequencies, again V_{blade} and V_{Coulomb} are first nondimensionalized. $\tilde{V}_{\text{blade}}$ and $\tilde{V}_{\text{Coulomb}}$ are elementary functions, such that their second derivatives are also elementary functions – we can write those as functions in python. From this we find $\tilde{\text{Hess}}(\tilde{\phi})$, which we evaluate at $\tilde{\mathbf{q}}_{\text{eqm}}$, hence we find $\tilde{\mathbf{C}} = \frac{1}{m}\tilde{\text{Hess}}(\tilde{\phi})$. The eigenvalues and eigenvectors of $\tilde{\mathbf{C}}$ are computed using `numpy.linalg.eigh`. The eigenvectors are then normalized, and the eigenvalues are dimensionalized; the square root of the eigenvalues is the eigenfrequencies.

Other existing code

There is also existing code written by group members for calculating the $\alpha_{i,j,k,l}$ coefficients. There is an existing COMSOL simulation from -4 to 4 microns in the x and y directions (with 0.1 micron spacing between grid points), and from -50 to 50 microns in the z direction (1 micron spacing). The COMSOL data is nondimensionalized; note that the Vandermonde matrix[26] maps the polynomial coefficients (written as a vector) to values of the polynomial evaluated at specific points. Hence the polynomial interpolation is a linear least squares problem, which is solved using `scipy.optimize.lsq_linear`.[27]

Thus we found $\tilde{\alpha}_{i,j,k,l}$ for $l = 1$ to 20.[28] For $l = 21$ (AC), recall the α coefficients are

[25] Without imposing any bounds or constraints. Also, we pass the function to be minimized as an argument to `scipy.optimize.minimize`; in this case, this function is $\tilde{V}(\tilde{\mathbf{q}})$.

The tolerance for termination is `1e-15`, and the maximum number of iterations to perform is `1000`.

[26] Implemented using `numpy.polynomial.polynomial.polyvander`, `numpy.polynomial.polynomial.polyvander2d`, and `numpy.polynomial.polynomial.polyvander3d`.

[27] With `1e-10` tolerance and no maximum number of iterations before termination.

[28] We work with the nondimensionalized $\tilde{\alpha}_{i,j,k,l}$ in the rest of this voltage control problem.

obtained by doing a polynomial interpolation for $V_{21} = \frac{q^2 E_1^2(r)}{4m\Omega^2}$, where $E_1(r)$ is obtained by taking the negative of the gradient of the DC field we obtain upon setting the voltage of the (anti)-diagonal blades to (-1) 1 volt. Since we already have the $\tilde{\alpha}$ coefficients for DC, we can express E_1 as a polynomial directly instead of doing an interpolation afterwards. The gradient is taken using `numpy.polynomial.polynomial.polyder`; E_1^2 is simply the inner product of the gradient.[29]

There is also existing code that optimizes $b_{0,0,2}$ and $b_{0,0,4}$ when we compute V_{target}. We find the z-polynomial[30] such that $\nabla V_{\text{blade}} \approx -\nabla V_{\text{Coulomb}}$ at the target equilibrium positions. $-\nabla V_{\text{Coulomb}}\big|_{\mathbf{q}_{\text{target}}}$ is fixed during the optimization, and $\nabla V_{\text{blade}}\big|_{\mathbf{q}_{\text{target}}}$ is computed from the z-polynomial coefficients by performing a matrix multiplication. Thus the polynomial coefficients are found by `numpy.linalg.lstsq` (linear least squares).[31]

2.4.4 Voltage optimization

Both coefficient-level optimization and grid-level optimization is of the form $\text{argmin}_{\mathbf{v}} \|\mathbf{MAv} - \mathbf{Mb}_{\text{target}}\|$ where $\mathbf{M}$ is some matrix.[32] These matrices are nondimensionalized (except for $\mathbf{v}$). The constrained linear least squares optimization is performed using `scipy.optimize.lsq_linear`.

2.4.5 Characterization

The voltages are inspected to see if they are reasonable. $\mathbf{b}_{\text{blade}} = \mathbf{Av}$ is compared to $\mathbf{b}_{\text{target}}$ coefficient by coefficient.

The equilibrium positions of $\mathbf{b}_{\text{blade}}$ and $\mathbf{b}_{\text{target}}$ are plotted and compared, along with the cross-section of $\mathbf{V}_{\text{blade}}$ and $\mathbf{V}_{\text{target}}$ along the z-axis. The eigenfrequencies[33] are plotted as a "spectrum" graph. The eigenmodes are also plotted, along with the variation of the x and y trapping strength along the z-axis.[34]

The trapping strength is calculated as follows. Consider the x trapping. Ideally, it is of the form $\frac{1}{2}m\omega_x^2 x^2$, which is uniform for different z. Note that ω_x can be recovered from $\frac{1}{2}m\omega_x^2 x^2$ by differentiating with respect to x twice, divide by m, then take the square root.

[29] Here we use $\Omega =$ `3.5e7`.
[30] ie terms of the form 1, z, z^2, z^3, z^4, z^5.
[31] With `rcond=None`
[32] Matrix multiplication is implemented using `numpy.matmul`.
[33] After dividing by 2π.
[34] The x and y eigenmodes is related to the x and y trapping strength.

In actuality, due to terms such as x^2z^2, this trapping depends on z. We define the x trapping strength as $\sqrt{\frac{1}{m}\frac{\partial^2 \mathbf{V}_{\text{blade}}}{\partial x^2}}$ as evaluated along the z-axis; a similar result follows for y.

Note that all plots are generated using Matplotlib.

Chapter 3

Results and Discussion

The voltage optimization program is a general tool, so it can be customized for different experimental setups. Here we go through a few scenarios that are of experimental interest.

In the following, all V_{target} uses the quartic potential.

3.1 Example: 10 ions, minimum spacing 4 microns

First, we give an example where all plots generated by the program are shown. We use coefficent-level optimization with all weights set to `1`, except for z^2 (`1e3`) and z^4 (`5e6`).

In this case, mode 1 is the x COM and y COM modes for both V_{blade} and V_{target}. Notice that the x and y COM modes have equal amplitude for V_{target}, whereas V_{blade} has uneven amplitude. The z COM mode is the last one (mode 10).

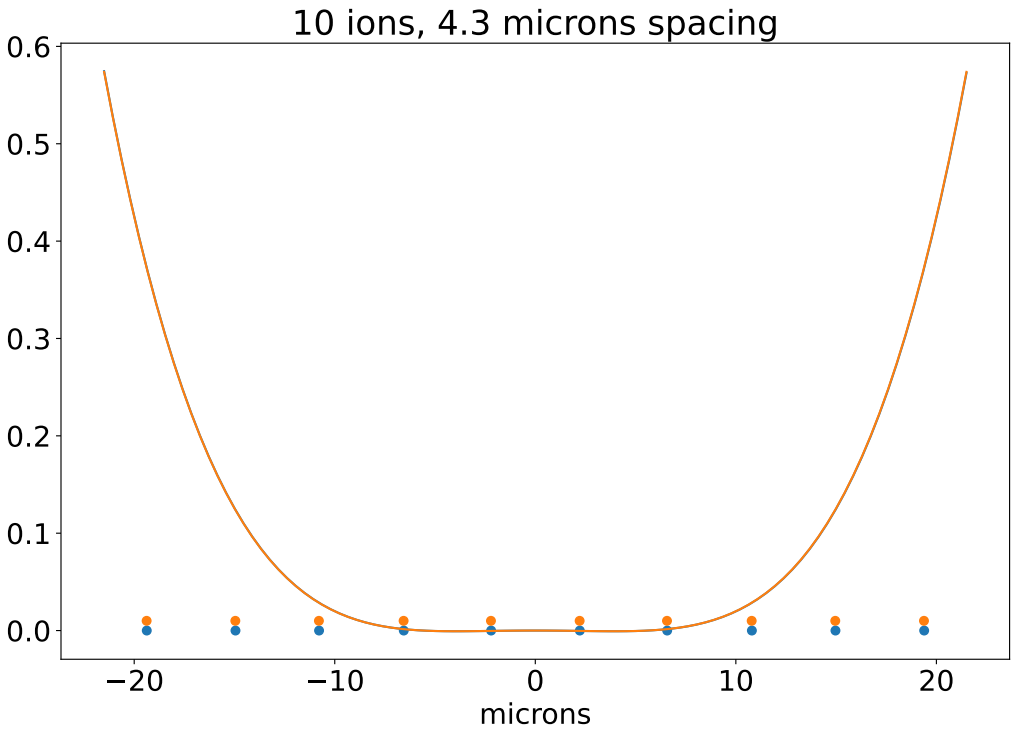
The x and y COM modes of V_{blade} have uneven amplitude because the x and y trapping strength is not uniform across z: the larger the trapping, the smaller the amplitude.

Finally, note that the x and y eigenmodes of V_{target} are essentially cosine waves, as expected from a system of coupled oscillator with a uniform spring constant (trapping strength).

3.2 30 ions, minimum spacing 4 microns

Ideally, we want as many ions as possible in the ion trap. Again we use coefficient-level optimization with all weights set to `1`, except for z^2 (`1e3`) and z^4 (`5e6`).

Figure 3.1: Cross-section of V and the corresponding equilibrium positions (10 ions).



Blue stands for V_{blade} and orange stands for V_{target}. Notice the overlap between $\boldsymbol{q}_{\text{blade}}$ (blue points) and $\boldsymbol{q}_{\text{target}}$ (orange points). Moreover, V_{target} (blue curve) and V_{target} (orange curve) overlap completely.

Figure 3.2: The eigenfrequencies of V_{blade} (10 ions).

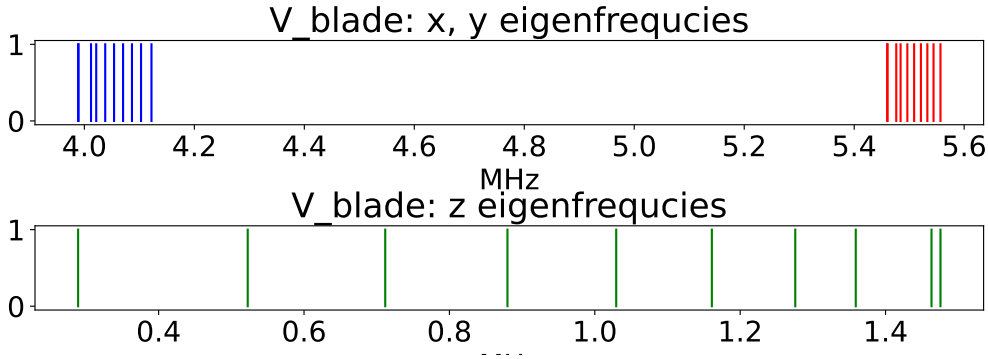
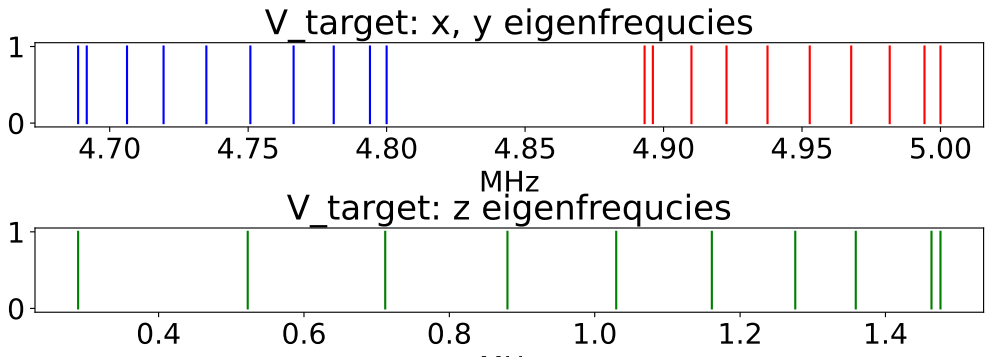


Figure 3.3: The eigenfrequencies of V_{target} (10 ions).



However, a straightforward scaling to higher number of ions does not work. This is because with the same ion spacing, the ion chain now occupies a larger region, such that the variation of the x and y trapping strength becomes greater across this region. In fact, because the ions at the edge experience much stronger trapping, their amplitude is practically zero in the COM modes. ie we do not have good COM modes, which is a significant problem in experiments.

For instance, for the x modes, mode 1 to 6 represent the modes of the edge ions; they are decoupled from the COM mode (mode 7). In Figure 3.9, we see that this is because the x trapping near the edge is substantially greater than the x trapping near the center.

3.3 30 ions, minimum spacing 1.7 microns

One way to bypass the bad COM mode issue is to reduce the spacing between ions, so that the ion chain occupies a smaller region in space, so as to reduce the amount of variation of the x and y throughout the ion chain.

We use grid-level optimization where the grid goes from -4 microns to 4 microns in increments of 1 micron in the x and y directions, and -60 to 60 microns in increments of 1 micron in the z direction. Notice that $\mathbf{q}_{\text{blade}}$ do not overlap with $\mathbf{q}_{\text{target}}$ (Figure 3.10), because we are sacrificing $\mathbf{q}_{\text{eqm}}$ to see if we can obtain good COM modes for 30 ions. This is an example of the tradeoffs we make as we do not have enough control over the system.

Indeed, in Figure 3.11, we see that the x and y COM modes are much better than before (the amplitude is more even, and all ions oscillate in phase). Unfortunately since the minimum spacing is less than d_m, this is not a viable solution (not suitable for experiments).

Figure 3.4: The eigenmodes of V_{blade} (10 ions).

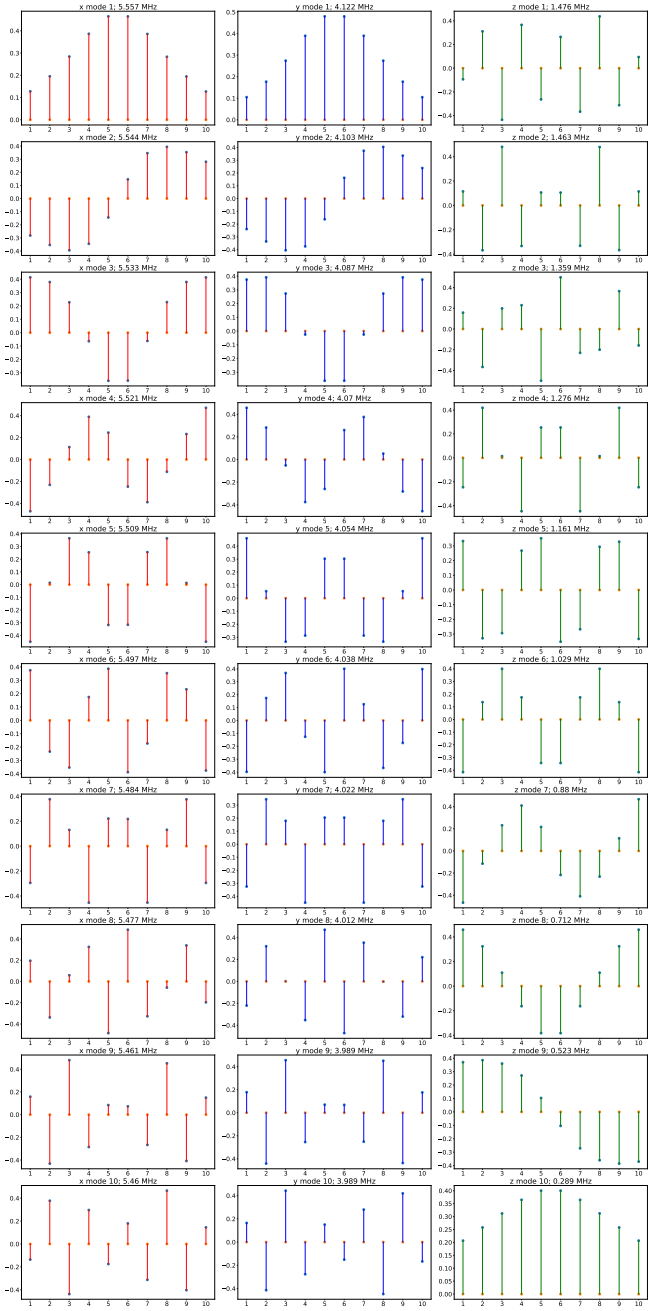


Figure 3.5: The eigenmodes of V_{target} (10 ions).

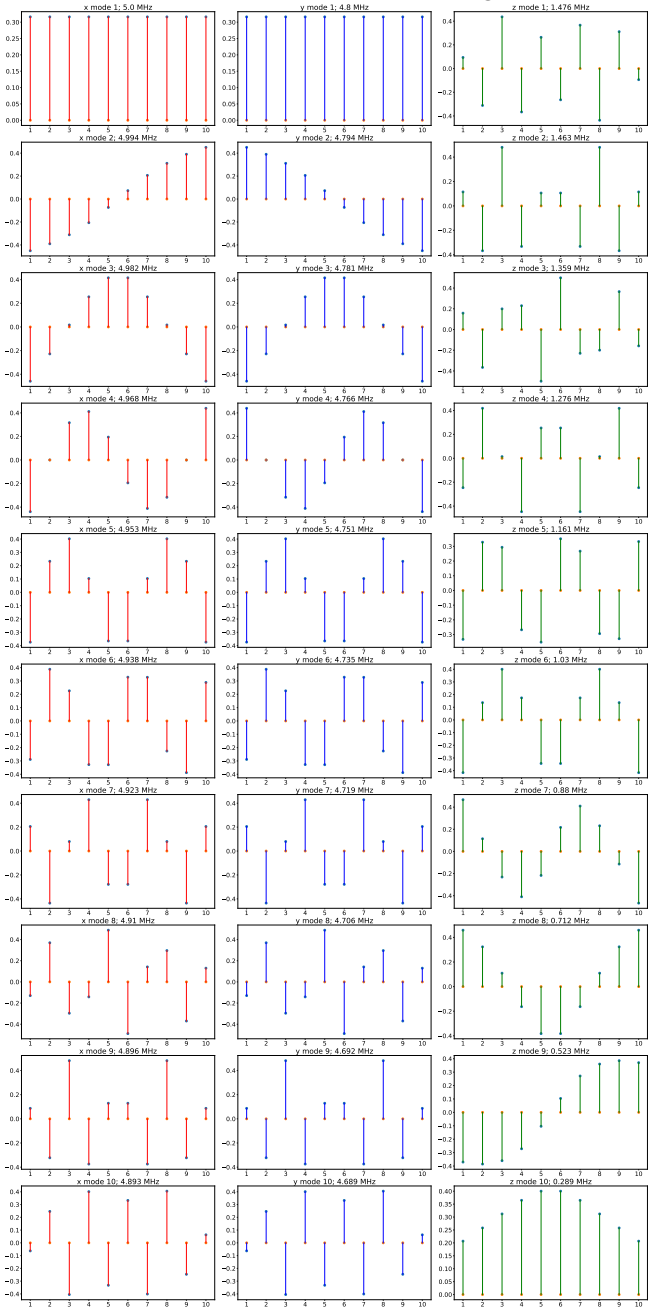
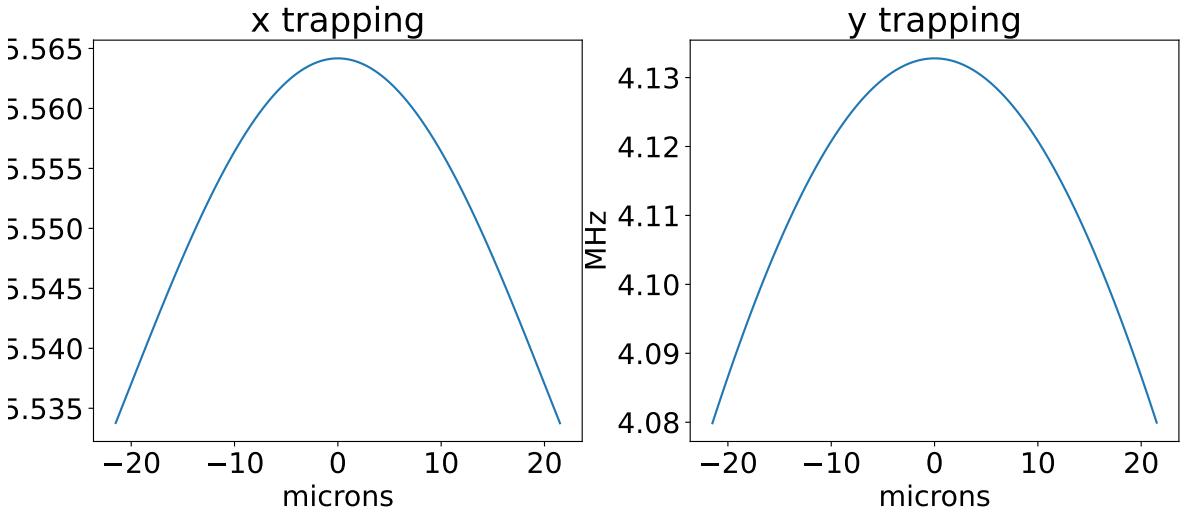


Figure 3.6: The x and y trapping strength of V_{blade} (10 ions).

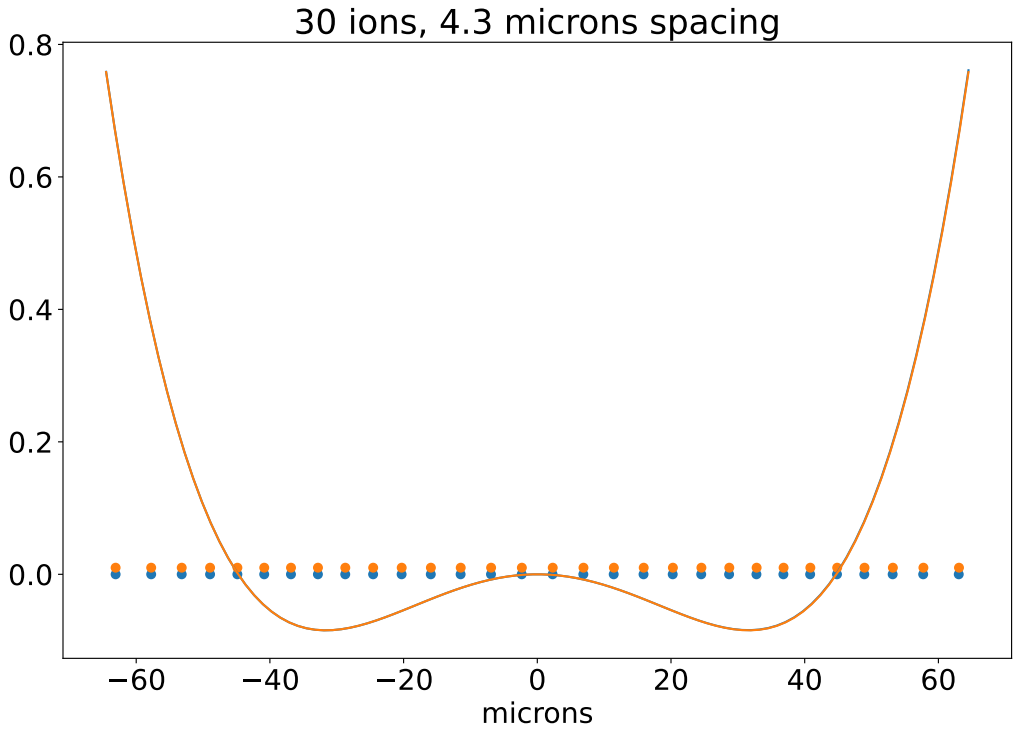


3.4 30 ions, x^2z^2 suppression

Another possible solution is found after much fine-tuning of the weights. This time, we use coefficient-level optimization with all weights set to `1`, except for z^2 (`1e3`), z^4 (`5e6`), x^2z^2 (`1e5`), and y^2z^2 (`1e5`).

As seen in Figure 3.13, $\mathbf{q}_{\text{blade}}$ overlap with $\mathbf{q}_{\text{target}}$ reasonably well. More significantly, the minimum separation is greater than d_m. In Figure 3.14, we see that the x COM mode is mode 3, which has a roughly uniform amplitude and the same phase except for the two ions right at the edge. In an experiment, this may be acceptable, provided that we do not use the first and last ions.

Figure 3.7: Cross-section of V and the corresponding equilibrium positions (30 ions).



Again we also have good overlap between $\boldsymbol{q}_{\text{blade}}$ *(blue points) and* $\boldsymbol{q}_{\text{target}}$ *(orange points).*

Figure 3.8: The eigenmodes of V_{blade} (30 ions).

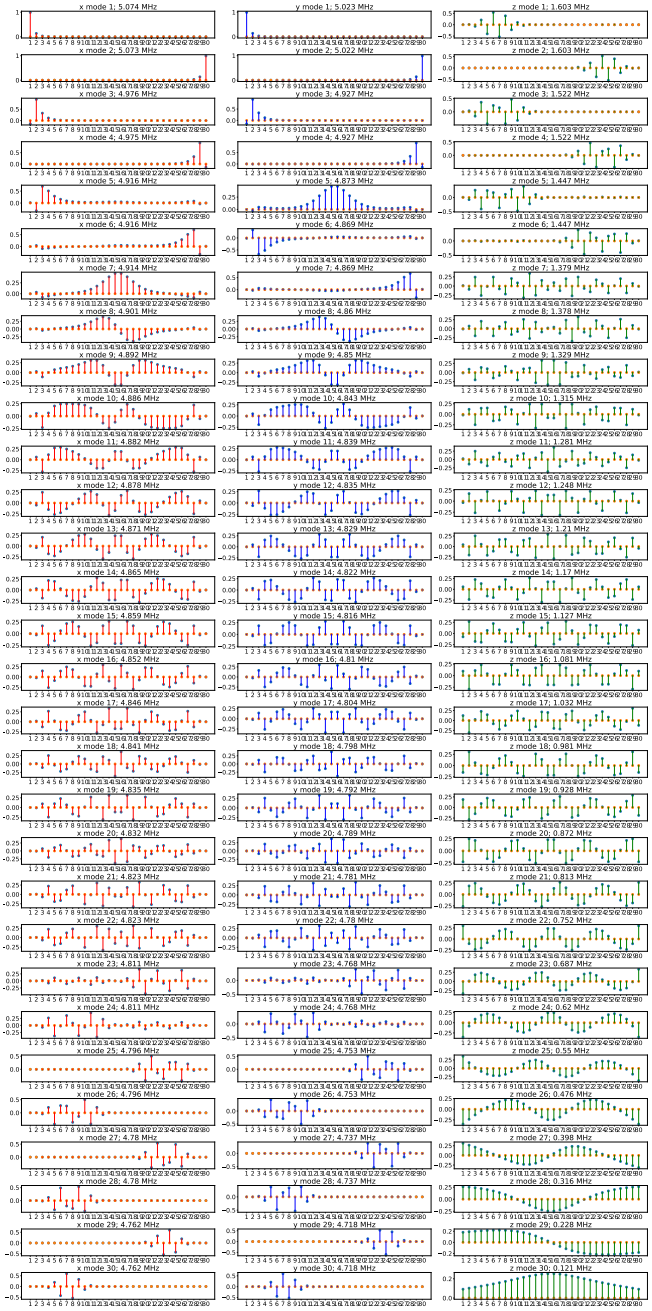


Figure 3.9: The x and y trapping strength of V_{blade} (30 ions).

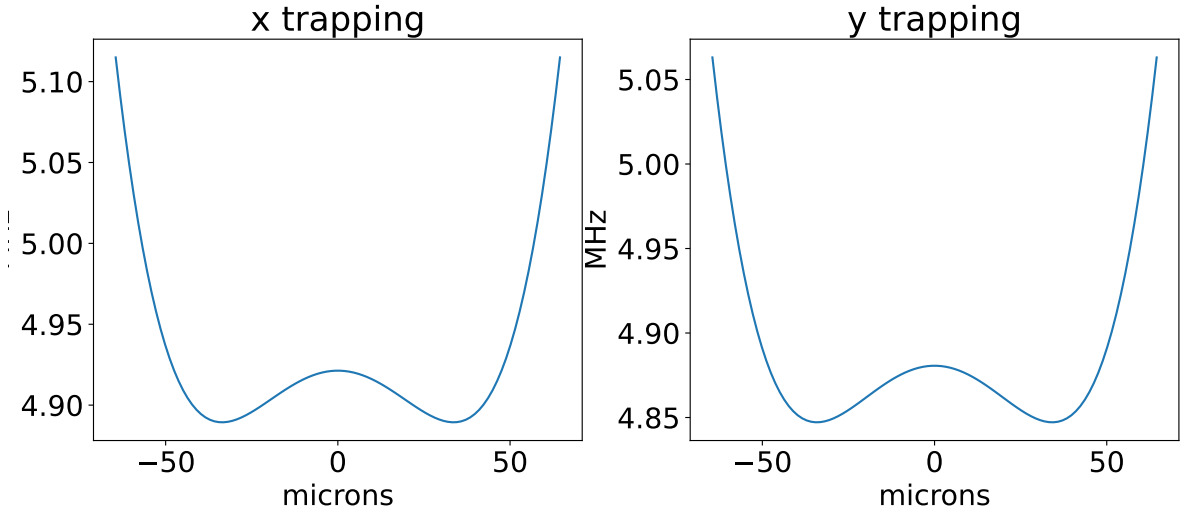


Figure 3.10: Cross-section of V and the corresponding equilibrium positions (30 ions, minimum spacing 1.7 microns).

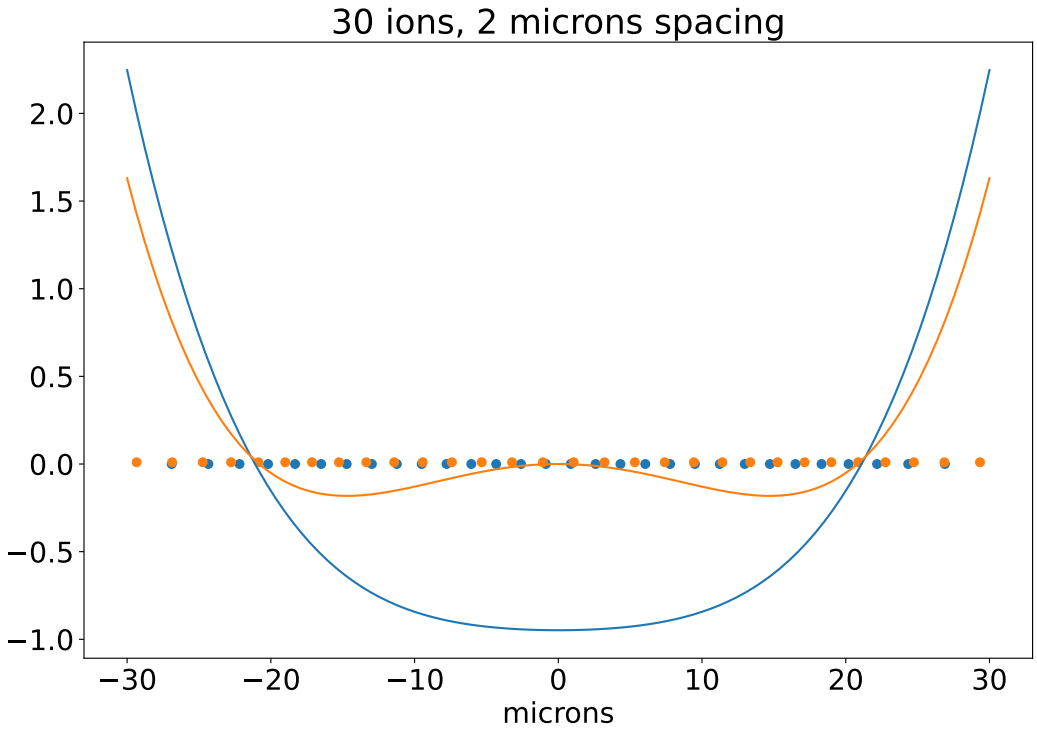


Figure 3.11: The eigenmodes of V_{blade} (30 ions, minimum spacing 1.7 microns).

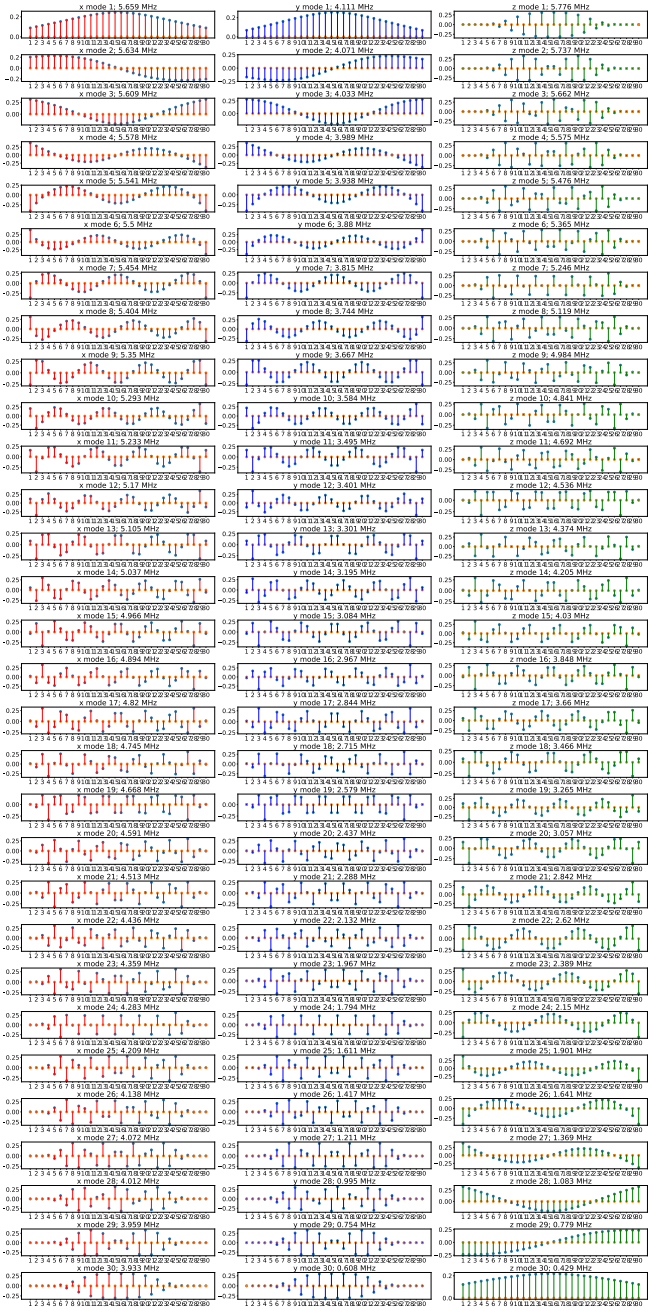


Figure 3.12: The x and y trapping strength of V_{blade} (30 ions, minimum spacing 1.7 microns).

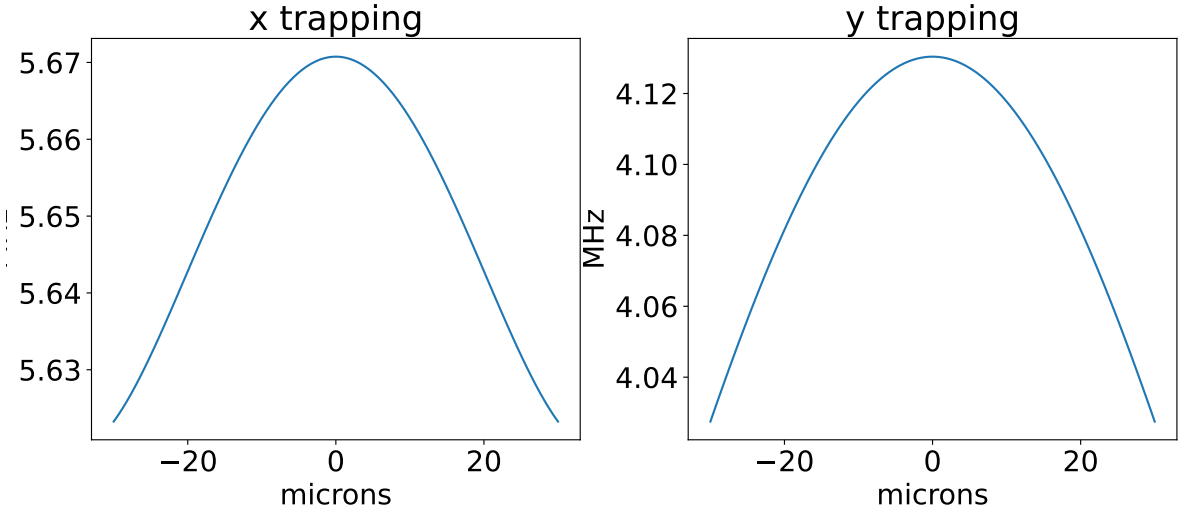


Figure 3.13: Cross-section of V and the corresponding equilibrium positions (30 ions, x^2z^2 suppression).

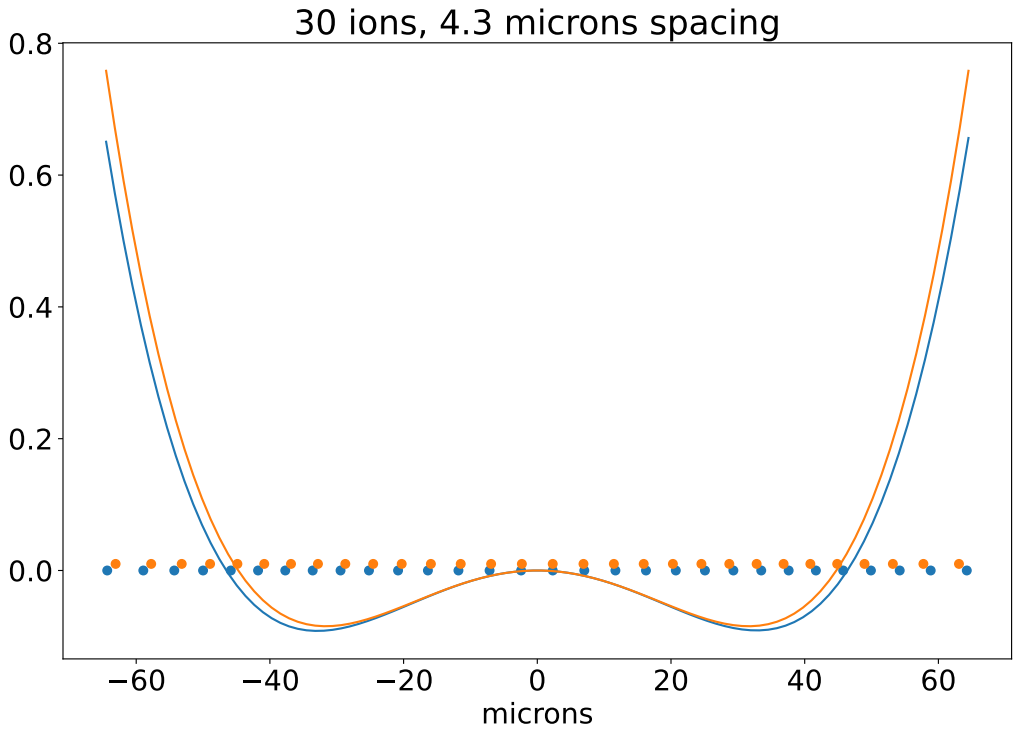


Figure 3.14: The eigenmodes of V_{blade} (30 ions, x^2z^2 suppression).

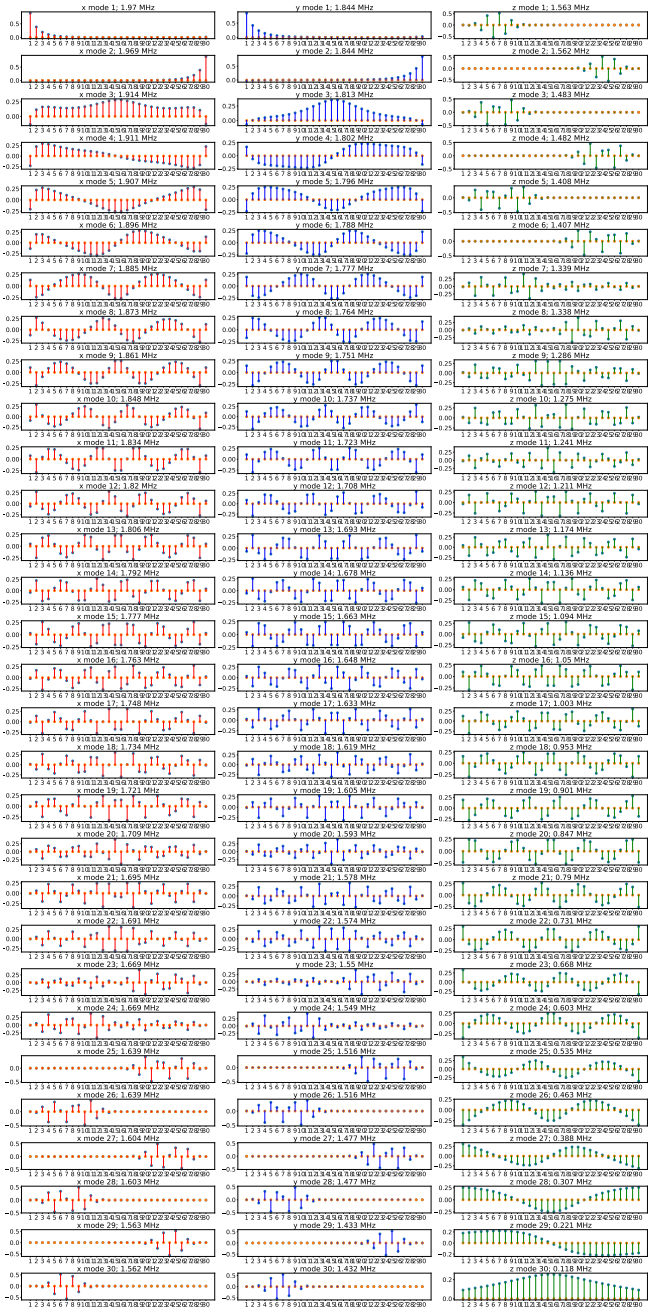
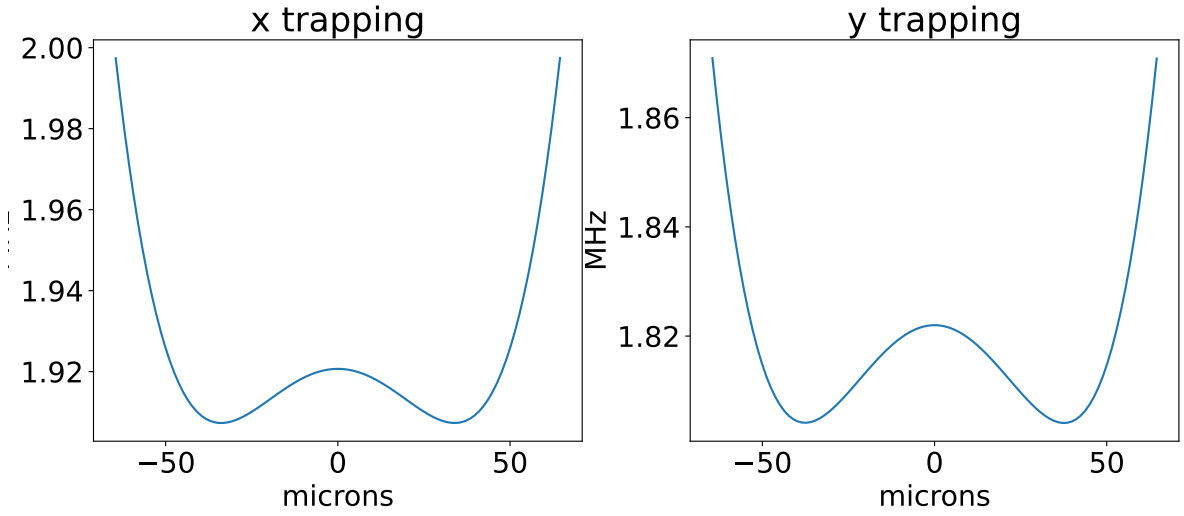


Figure 3.15: The x and y trapping strength of V_{blade} (30 ions, x^2z^2 suppression).



Chapter 4

Concluding Remarks

The initial objective of solving the voltage control problem has been achieved. However, as a next step, more sophisticated optimization schemes should be explored to produce better tradeoffs.

In particular, up to this point, we have assumed that the blades in the ion trap are aligned perfectly. This is not true in the actual experimental apparatus. In account for this, it may be more effective to perform the optimization physically: ie, adjusting the voltages and measure the changes to the ion positions/eigenmodes in real time.

Moreover for each experimental setup, a new optimization must be performed. Optimizations becomes time-consuming for large number of ions. Therefore, it may be desirable to use machine learning to do the optimization: ie, optimized voltages are learnt during training, such that voltages can be outputted quickly during inference.

There is a recent work by DeepMind that implements such an optimization using reinforcement learning, a form of machine learning.[11] There, they tackle a similar problem, where they use magnetic actuator coils to shape and maintain a high-temperature plasma within the tokamak vessel. It may be worthwhile to explore a similar optimization scheme for this problem.

Acknowledgements

Special thanks must be given to my supervisor Dr. Kazi Rajibul Islam, Yi Hong Teoh, Nikhil Kotibhaskar, and Sainath Motlakunta for their valuable mentorship and insights during this project. Yi Hong Teoh is the author of the TrICal package, which I used in this project extensively. I have benefited greatly from the existing code written by our research group members.

References

[1] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, "Trapped-ion quantum computing: Progress and challenges," *Applied Physics Reviews*, vol. 6, p. 021314, jun 2019.

[2] C. Monroe, W. Campbell, L.-M. Duan, Z.-X. Gong, A. Gorshkov, P. Hess, R. Islam, K. Kim, N. Linke, G. Pagano, P. Richerme, C. Senko, and N. Yao, "Programmable quantum simulations of spin systems with trapped ions," *Reviews of Modern Physics*, vol. 93, apr 2021.

[3] D. A. Hucul, *A Modular Quantum System of Trapped Atomic Ions*. PhD thesis, University of Maryland, 2015.

[4] Y. H. Teoh, M. Sajjan, Z. Sun, F. Rajabi, and R. Islam, "Manipulating phonons of a trapped-ion system using optical tweezers," *Physical Review A*, vol. 104, aug 2021.

[5] N. Kotibhaskar, "Design and construction of an ion trapping apparatus for quantum simulation experiments," Master's thesis, University of Waterloo, 2019.

[6] C.-Y. Shih, "Holographic optical manipulation of trapped ions for quantum simulation," Master's thesis, University of Waterloo, 2019.

[7] Y. H. Teoh, "Machine learning and optimization techniques for trapped-ion quantum simulators," Master's thesis, University of Waterloo, 2021.

[8] C.-Y. Shih, S. Motlakunta, N. Kotibhaskar, M. Sajjan, R. Hablützel, and R. Islam, "Reprogrammable and high-precision holographic optical addressing of trapped ions for scalable quantum control," *npj Quantum Information*, vol. 7, p. 57, Apr 2021.

[9] D. E. Seborg and N. J. Hoboken, *Process dynamics and control*. John Wiley & Sons, 4th ed., 2011.

[10] D. Kraft, "A software package for sequential quadratic programming," *Tech. Rep.*, 1988.

[11] J. Degrave, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de las Casas, C. Donner, L. Fritz, C. Galperti, A. Huber, J. Keeling, M. Tsimpoukelli, J. Kay, A. Merle, J.-M. Moret, S. Noury, F. Pesamosca, D. Pfau, O. Sauter, C. Sommariva, S. Coda, B. Duval, A. Fasoli, P. Kohli, K. Kavukcuoglu, D. Hassabis, and M. Riedmiller, "Magnetic control of tokamak plasmas through deep reinforcement learning," *Nature*, vol. 602, pp. 414–419, Feb 2022.