Comprehensive Documentation: Modular Ion Simulation Codebase

Jack Beda
jack.beda.ca

August, 2025

Contents

1 Introduction

Dear reader! This document was written in the summer of 2025 as part of a 10 week REU (Research Experience for Undergraduates) with the University of Washington in Boris Blinov's ion-trapping group. While participating in the REU, I worked with Boris Blinov[1], Boris Pashinsky[2], and various other students including Carl Thomas[3], Hunter Parker[4], Rebecca[5], Jane Gunn[6], and Simon. This project was based on the paper [1] where we wrote code to simulate the motion of trapped ions in two dimensions. Anyone intending to use this code is strongly recommended to work through the three example Jupyter Notebooks provided with the code and described in the Use Examples section. These are intended as tutorials for someone new to the code, but familiar with it's purpose and python programming. It is also worth familiarizing oneself with the units conventions used in the code, as described in the Units section.

1.1 Github

The code for the project is hosted on github here: github.com/jfbeda/trapped_ion_simulation.

2 Units

We do not follow the conventions of [2] with respect to units. In our simulation, we take as unit of length the micrometer (µm), our unit of time the microsecond (µs), and our unit of mass the atomic mass unit (amu). We also fix the value of the electron charge (e) and the Boltzmann constant (k_B). This allows us to define a unit of energy, κ, which is defined as

$$\kappa \stackrel{\text{def}}{=} (\text{amu})(\mu\text{m}^2)(\mu\text{s}^{-2}) \approx 1.660\,54 \times 10^{-27}\,\text{J} \approx 10.364\,253\,704\,3\,\text{neV}. \tag{1}$$

In these natural units the vacuum permittivity is given by:

$$\epsilon_0 = 8.854\,18 \times 10^{-12}\,\text{C}^2\,\text{kg}^{-1}\,\text{m}^{-3}\,\text{s}^2 \tag{2}$$

$$= 8.854\,18 \times 10^{-12}\left(\frac{1}{1.602\,18 \times 10^{-19}\,\text{e}}\right)^2\left(\frac{1}{1.660\,54 \times 10^{-27}\,\text{amu}}\right)^{-1}\left(10^6\,\mu\text{m}\right)^{-3}\left(10^6\,\mu\text{s}\right)^2 \tag{3}$$

$$\epsilon_0 = 5.727\,660\,742\,3 \times 10^{-7}\,\text{e}^{-2}\mu\text{m}^{-3}\,\mu\text{s}^2\text{amu} \tag{4}$$

We shall call our natural units of temperature θ defined as

$$\theta \stackrel{\text{def}}{=} \frac{\kappa}{k_B} \approx 0.120\,272\,422\,607\,\text{mK} \tag{5}$$

In summary, below we present all the units. Those in bold are fixed by assumption. The others are calculated

- **length**: µm
- **time**: µs

[1]Boris Blinov: https://sites.google.com/view/iontrap/, blinov@uw.edu, +1 (206) 221-3780.
[2]Boris Pashinsky: pashinsk@uw.edu, boris.pashinsky@gmail.com, +1 (206) 551-1258.
[3]Carl Thomas: cjthoma@uw.edu
[4]Hunter Parker: hparker2@uw.edu
[5]Rebecca: munkr@uw.edu
[6]Jane Gunn: janegunn@uw.edu

- **mass**: amu
- **charge**: $e = 1e$
- **Boltzmann constant**: $k_B = 1k_B$
- energy: $\kappa = (\text{amu})(\mu\text{m}^2)(\mu\text{s}^{-2}) \approx 10.364\,253\,704\,3\,\text{neV}$
- frequency: MHz
- vacuum permittivity: $\epsilon_0 \approx 5.727\,660\,742\,3 \times 10^{-7}\, e^{-2}\mu\text{m}^{-3}\,\mu\text{s}^2\text{amu}$
- Coulomb constant: $k = \frac{1}{4\pi\epsilon_0} \approx 1.389\,353\,789\,02 \times 10^5\, e^2\mu\text{m}^3\,\mu\text{s}^{-2}\text{amu}^{-1}$
- temperature: $\theta = \kappa/k_B \approx 0.120\,272\,422\,607\,\text{mK}$
- force: $(\text{amu})(\mu\text{m})(\mu\text{s}^{-2}) \approx 1.660\,54 \times 10^{-21}\,\text{N}$
- damping parameter: $(\text{amu})(\mu\text{s}^{-1})$
- reduced plank constant: $\hbar \approx 0.063\,507\,799\,295\,889\,\kappa\,\mu\text{s}$

Generally, in our simulations we take $m = 137.327\,\text{amu}$ is the mass of a Barium ion. We take $\omega = 1\,\text{MHz}$.

Note: All physical quantities in the code are in terms of these natural units except where otherwise specified. For example, a variable called `temperature` will be measured in units of θ, whereas `tempeature_mK` would be in units of mK.

3 Theoretical Background

Note that this section is long, and the majority is *not* required for a deep and complete understanding of the code.

3.1 The problem

Consider a set of N charged particles of mass m in a potential $V(x,y) = \frac{1}{2}m\omega_x^2x^2 + \frac{1}{2}m\omega_y^2y^2$. Suppose without loss of generality that $\omega_x \geq \omega_y$. Define the *isotropy parameter* $\gamma = \frac{\omega_y}{\omega_x}$. As a result of the convention that $\omega_x \geq \omega_y$, we observe that $0 < \gamma \leq 1$. The *anisotropy parameter*, $\alpha \stackrel{\text{def}}{=} \frac{1}{\gamma}$ we will not use[7]. Note that $1 \leq \alpha < \infty$. For ease of notation, let us fix $\omega_x \stackrel{\text{def}}{=} \omega$ such that $\omega_y = \gamma\omega$. Thus the new potential is given by

$$V(x,y) = \frac{1}{2}m\omega^2x^2 + \frac{1}{2}m\omega^2\gamma^2y^2 = \frac{1}{2}m\omega^2(x^2 + (\gamma y)^2). \tag{6}$$

This can be written in terms of the position vector $\vec{r} = (x,y)^T$ by defining the matrix $\Gamma = \begin{bmatrix} 1 & 0 \\ 0 & \gamma^2 \end{bmatrix}$. This allows us to write

[7] In other texts and in the literature γ is called the anisotropy parameter, and is still fixed to lie between zero and 1. I prefer calling γ the isotropy parameter, because the larger γ is (the closer it is to 1) the more isotropic the potential

$$V(\vec{r}) = \frac{1}{2}m\omega^2(\vec{r}^T\Gamma\vec{r}). \tag{7}$$

The potential resulting from the Coulomb interaction between the charged particles is

$$V_c = \sum_{i>j}\sum_j \frac{k}{|\vec{r_i} - \vec{r_j}|} = \sum_{i>j}\sum_j \frac{k}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}} \tag{8}$$

for some constant $k = \frac{q^2}{4\pi\epsilon_0}$. Thus the Lagrangian of the system is

$$L = \sum_{i=1}^N \frac{1}{2}m(\dot{x_i}^2 + \dot{y_i}^2) - \sum_{i=1}^N \frac{1}{2}m\omega^2(x_i^2 + (\gamma y_i)^2) - \sum_{i>j}\sum_j \frac{k}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}. \tag{9}$$

$$L = \sum_{i=1}^N \frac{1}{2}m\dot{\vec{r_i}}^2 - \sum_{i=1}^N \frac{1}{2}m\omega^2(\vec{r_i}^T\Gamma\vec{r_i}) - \sum_{i>j}\sum_j \frac{k}{|\vec{r_i} - \vec{r_j}|} \tag{10}$$

This has generalized momenta of

$$\vec{p_i} \stackrel{\text{def}}{=} \frac{\partial L}{\partial \vec{r_j}} \tag{11}$$

Where we make use of the abuse of notation $\vec{x} = \frac{\partial Q}{\partial \vec{y}}$ as shorthand for $\vec{x}^{(i)} = \frac{\partial Q}{\partial \vec{y}^{(i)}}$ where $\vec{x}^{(i)}$ denotes the ith component of vector $\vec{x}$. This gives

$$\vec{p_i} = m\dot{\vec{r_j}} \tag{12}$$

and so the Hamiltonian of the system is given by

$$\boxed{H(\{\vec{r_i}\}, \{\vec{p_i}\}) = \sum_{i=1}^N \frac{\vec{p_i}^2}{2m} + \sum_{i=1}^N \frac{1}{2}m\omega^2(\vec{r_i}^T\Gamma\vec{r_i}) + \sum_{i>j}\sum_j \frac{k}{|\vec{r_i} - \vec{r_j}|}.} \tag{13}$$

where we will use the shorthands

$$T = \sum_{i=1}^N \frac{\vec{p_i}^2}{2m} \tag{14}$$

$$U_h = \sum_{i=1}^N \frac{1}{2}m\omega^2(\vec{r_i}^T\Gamma\vec{r_i}) \tag{15}$$

$$U_c = \sum_{i>j}\sum_j \frac{k}{|\vec{r_i} - \vec{r_j}|} \tag{16}$$

to denote the kinetic energy, harmonic potential energy, and Coulomb energy respectively. In this shorthand we have $H = T + U_h + U_c$.

3.2 Fixed points of the Hamiltonian

We now seek fixed points of the Hamiltonian. These satisfy

$$\dot{\vec{r}}_i = \frac{\partial H}{\partial \vec{p}_i} = \frac{\vec{p}_i}{m} = 0, \qquad \dot{\vec{p}}_i = -\frac{\partial H}{\partial \vec{r}_i} = 0. \tag{17}$$

Computing $\frac{\partial H}{\partial \vec{r}_i}$ can be split into two parts, $\frac{\partial U_h}{\partial \vec{r}_i}$ and $\frac{\partial U_c}{\partial \vec{r}_i}$:

$$\left(\frac{\partial U_h}{\partial \vec{r}_i}\right)^{(k)} = \frac{\partial U_h}{\partial \vec{r}_i^{(k)}} \tag{18}$$

$$= \sum_{l=1}^{N} \frac{1}{2} m\omega^2 \frac{\partial}{\partial \vec{r}_i^{(k)}} \sum_p \sum_q \vec{r}_l^{(p)} \Gamma_{pq} \vec{r}_l^{(q)} \tag{19}$$

$$= \sum_{l=1}^{N} \sum_p \sum_q \frac{1}{2} m\omega^2 (\delta_{li}\delta_{pk}\Gamma_{pq}\vec{r}_l^{(q)} + \delta_{li}\delta_{kq}\vec{r}_l^{(p)}\Gamma_{pq}) \tag{20}$$

$$= \frac{1}{2} m\omega^2 \left(\sum_q \Gamma_{kq}\vec{r}_i^{(q)} + \sum_p \vec{r}_i^{(p)}\Gamma_{pk} \right) \tag{21}$$

$$\boxed{\frac{\partial U_h}{\partial \vec{r}_i} = m\omega^2 \Gamma \vec{r}_i.} \tag{22}$$

where we have used the fact that $\Gamma = \Gamma^T$ is a symmetric matrix. Now turning to the Coulomb term we have:

$$\left(\frac{\partial U_c}{\partial \vec{r}_i}\right)^{(k)} = \frac{\partial U_c}{\partial r_i^{(k)}} \tag{23}$$

$$= \sum_{p>q}\sum_{q} \frac{\partial}{\partial r_i^{(k)}}\left[\frac{k}{\sqrt{(\vec{r}_p - \vec{r}_q)^2}}\right] \tag{24}$$

$$= \sum_{p>q}\sum_{q} -\frac{k}{2((\vec{r}_p - \vec{r}_q)^2)^{3/2}} \frac{\partial}{\partial r_i^{(k)}}(\vec{r}_p - \vec{r}_q)^2 \tag{25}$$

$$= -\frac{k}{2}\sum_{p>q}\sum_{q} \frac{1}{|\vec{r}_p - \vec{r}_q|^3} \sum_j 2(\vec{r}_p^{(j)} - \vec{r}_q^{(j)}) \frac{\partial}{\partial \vec{r}_i^{(k)}}\left(\vec{r}_p^{(j)} - \vec{r}_q^{(j)}\right) \tag{26}$$

$$= -k\sum_{p>q}\sum_{q}\sum_j \frac{(\vec{r}_p^{(j)} - \vec{r}_q^{(j)})}{|\vec{r}_p - \vec{r}_q|^3}(\delta_{ip}\delta_{jk} - \delta_{iq}\delta_{jk}) \tag{27}$$

$$= -k\sum_{p>q}\sum_{q} \frac{(\vec{r}_p^{(k)} - \vec{r}_q^{(k)})}{|\vec{r}_p - \vec{r}_q|^3}(\delta_{ip} - \delta_{iq}) \tag{28}$$

$$\frac{\partial U_c}{\partial \vec{r}_i} = -k\sum_{p>q}\sum_{q} \frac{(\vec{r}_p - \vec{r}_q)}{|\vec{r}_p - \vec{r}_q|^3}(\delta_{ip} - \delta_{iq}) \tag{29}$$

$$= -\frac{k}{2}\sum_{p\neq q}\sum_{q} \frac{(\vec{r}_p - \vec{r}_q)}{|\vec{r}_p - \vec{r}_q|^3}(\delta_{ip} - \delta_{iq}) \quad \text{(As symmetric in } p \text{ and } q\text{)} \tag{30}$$

$$= -\frac{k}{2}\left(\sum_{q\neq i} X_{iq} - \sum_{p\neq i} X_{pi}\right) \qquad \left(X_{pq} \overset{\text{def}}{=} \frac{(\vec{r}_p - \vec{r}_q)}{|\vec{r}_p - \vec{r}_q|^3}\right) \tag{31}$$

$$= -k\left(\sum_{q\neq i} X_{iq}\right) \quad \text{(As } X_{pq} \text{ is antisymmetric)} \tag{32}$$

$$= k\left(\sum_{q\neq i} X_{qi}\right) \tag{33}$$

$$= k\left(\sum_{q\neq i} \frac{(\vec{r}_q - \vec{r}_i)}{|\vec{r}_q - \vec{r}_i|^3}\right) \tag{34}$$

$$\boxed{\frac{\partial U_c}{\partial \vec{r}_i} = k\left(\sum_{q\neq i} \frac{(\vec{r}_q - \vec{r}_i)}{|\vec{r}_q - \vec{r}_i|^3}\right)} \tag{35}$$

Putting (22) and (35) together yields a condition for fixed points of the system:

$$0 = -\frac{\partial H}{\partial \vec{r}_i} = -\left(\frac{\partial U_h}{\partial \vec{r}_i} + \frac{\partial U_c}{\partial \vec{r}_i}\right) \tag{36}$$

$$0 = m\omega^2\Gamma\vec{r}_i + k\sum_{q\neq i} \frac{(\vec{r}_q - \vec{r}_i)}{|\vec{r}_q - \vec{r}_i|^3}. \tag{37}$$

That is to say, a fixed point of the system is a set of $\{\vec{r}_i\}$ such that for all $i = 1, \cdots, N$

$$\boxed{0 = m\omega^2 \Gamma \vec{r}_i + k \sum_{q \neq i} \frac{(\vec{r}_q - \vec{r}_i)}{|\vec{r}_q - \vec{r}_i|^3}}. \tag{38}$$

It will also be useful to note that (38) is a simplified form of the general case of the force on each particle given by

$$\dot{\vec{p}}_i = -\left(m\omega^2 \Gamma \vec{r}_i + k \sum_{q \neq i} \frac{(\vec{r}_q - \vec{r}_i)}{|\vec{r}_q - \vec{r}_i|^3} \right). \tag{39}$$

Note also, as we will use it for later:

$$\frac{\partial}{\partial \vec{r}_i} \frac{1}{|\vec{r}_p - \vec{r}_q|} = -\frac{(\vec{r}_p - \vec{r}_q)}{|\vec{r}_p - \vec{r}_q|^3} (\delta_{ip} - \delta_{iq}) \tag{40}$$

3.3 Symmetry properties of the solution set

As a check, suppose that $\gamma = 1$ (i.e. the system is rotationally symmetric). This means that $\Gamma = \mathbb{I}$, the identity matrix. We now expect that if $\{\vec{r}_i\}$ are a solution to (38), we should expect that $\{\vec{r}'_i\}$ are also a solution, where $\vec{r}'_i \overset{\text{def}}{=} R\vec{r}_i$ and R is a rotation matrix. Observe that

$$m\omega^2 R\vec{r}_i + k \sum_{q \neq i} \frac{(R\vec{r}_q - R\vec{r}_i)}{|R\vec{r}_q - R\vec{r}_i|^3} = R\left(m\omega^2 \vec{r}_i + k \sum_{q \neq i} \frac{(\vec{r}_q - \vec{r}_i)}{|\vec{r}_q - \vec{r}_i|^3} \right) \tag{41}$$

$$= 0 \tag{42}$$

as expected. In fact, we notice that the solution set is invariant under the action of any matrix M that commutes with Γ and preserves $(\vec{r}_q - \vec{r}_i)^2$. In other words, any unitary matrix M which commutes with Γ. Suppose $M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$. The fact that M must commute with Γ implies that either $\gamma = 1$ or $b = c = 0$. In the case of $\gamma \neq 1$ this means that M is a diagonal unitary matrix. This means that each diagonal entry of M (a and d) has complex norm 1. Assuming further that M is real gives that $a = \pm 1$ and $d = \pm 1$. This makes sense, as for $\gamma \neq 1$ the only symmetry transformations we expect on our solution set are similarity transformations of the ellipse, namely reflection about the y-axis ($M = \text{diag}(-1, 1)$), reflection about the x-axis ($M = \text{diag}(1, -1)$), rotation by π ($M = \text{diag}(-1, -1)$), and of course the identity ($M = \mathbb{I}$).

3.4 Special fixed points: equally spaced charges around the circle

Let us consider the special case of $\gamma = 1$. Let us further divide (38) by k and define $x = m\omega^2/k$. Equation (38) now reads

$$0 = x\vec{r}_i + \sum_{q \neq i} \frac{(\vec{r}_q - \vec{r}_i)}{|\vec{r}_q - \vec{r}_i|^3} \tag{43}$$

We conjecture that given N there exists a radius r such that $\vec{r}_j$ defined by

$$\vec{r}_j = r \begin{pmatrix} \cos(2\pi j/N) \\ \sin(2\pi j/N) \end{pmatrix} \tag{44}$$

form a solution to (43). That is to say we conjecture there exists an r such that equally spaced particles on a circle of radius r is a solution. Because the problem is rotationally symmetric, it suffices to check (43) for only one case (i.e. $i = N$) (or $i = 0$ if you prefer an abuse of notation). We may move to the complex plane by associating the x coordinate with the real part and the y coordinate with the imaginary part of a complex function. Thus $\vec{r}_q$ is associated to the scaled root of unity $re^{2\pi iq/N}$. This is equivalent to the claim that the complex function defined below has a root:

$$f(r) \stackrel{\text{def}}{=} xr + \sum_{q=1}^{N-1} \frac{(re^{2\pi iq/N} - r)}{|re^{2\pi iq/N} - r|^3} \tag{45}$$

$$f(r) = xr + \frac{1}{r^2} \sum_{q=1}^{N-1} \frac{(e^{2\pi iq/N} - 1)}{|e^{2\pi iq/N} - 1|^3} \tag{46}$$

Setting this equal to zero gives:

$$0 = xr^* + \frac{1}{r^{*2}} \sum_{q=1}^{N-1} \frac{(e^{2\pi iq/N} - 1)}{|e^{2\pi iq/N} - 1|^3} \tag{47}$$

$$0 = xr^{*3} + \sum_{q=1}^{N-1} \frac{(e^{2\pi iq/N} - 1)}{|e^{2\pi iq/N} - 1|^3} \tag{48}$$

$$r^* = \left(-\frac{1}{x} \sum_{q=1}^{N-1} \frac{(e^{2\pi iq/N} - 1)}{|e^{2\pi iq/N} - 1|^3} \right)^{1/3} \tag{49}$$

To simplify r^*, define $z_q = e^{2\pi iq/N} - 1$. Now observe that

$$|z_q|^2 = (e^{2\pi iq/N} - 1)(e^{-2\pi iq/N} - 1) \tag{50}$$

$$= 2 - e^{2\pi iq/N} - e^{-2\pi iq/N} \tag{51}$$

$$= 2 - 2\cos(2\pi q/N) \tag{52}$$

$$= 2(1 - \cos(2\pi q/N)) \tag{53}$$

$$|z_q|^2 = 4\sin^2(\pi q/N) \tag{54}$$

$$|z_q| = 2\sin(\pi q/N) \tag{55}$$

Thus we have

$$r^* = \left(-\frac{1}{x} \sum_{q=1}^{N-1} \frac{(e^{2\pi iq/N} - 1)}{8\sin^3(\pi q/N)} \right)^{1/3} \tag{56}$$

$$= -\frac{1}{2} \left(\frac{1}{x} \sum_{q=1}^{N-1} \frac{e^{2\pi iq/N} - 1}{\sin^3(\pi q/N)} \right)^{1/3}. \tag{57}$$

Now we notice that for each q, the corresponding $N-q$ term has numerator $e^{2\pi i(N-q)/N}-1 = \overline{e^{2\pi iq/N}-1}$ and so the imaginary parts cancel as $\sin(\pi q/N) = \sin(\pi(N-q)/N)$. Thus we may write

$$r^* = -\frac{1}{2}\left(\frac{1}{x}\sum_{q=1}^{N-1}\frac{\mathrm{Re}(e^{2\pi iq/N}-1)}{\sin^3(\pi q/N)}\right)^{1/3} \tag{58}$$

$$= -\frac{1}{2}\left(\frac{1}{x}\sum_{q=1}^{N-1}\frac{\cos(2\pi q/N)-1}{\sin^3(\pi q/N)}\right)^{1/3} \tag{59}$$

$$= \frac{1}{2}\left(\frac{1}{x}\sum_{q=1}^{N-1}\frac{2\sin^2(\pi q/N))}{\sin^3(\pi q/N)}\right)^{1/3} \tag{60}$$

$$= \frac{1}{2^{2/3}}\left(\frac{1}{x}\sum_{q=1}^{N-1}\frac{1}{\sin(\pi q/N)}\right)^{1/3} \tag{61}$$

$$\boxed{r^* = \left(\frac{1}{4x}\sum_{q=1}^{N-1}\frac{1}{\sin(\pi q/N)}\right)^{1/3}} \tag{62}$$

The above expression is pretty well approximated by $\left(\frac{N}{2\pi x}\ln(N)\right)^{\frac{1}{3}}$. The above expression can be visualized in Desmos, where it is clear there is always a root. Thus we have shown that for $\gamma = 1$ and all N, there exists r^* such that charges spaced equally around a circle of radius r^* is a fixed point of the system. It remains to be seen whether or not this fixed point is a point of stability or not. I have a conjecture that the above is a position of stability for $N \leq 6$, and is unstable for $N \geq 7$, but I have not succeeded in showing this algebraically.

3.4.1 Heuristic argument

Note that there is also an intuitive argument for why we expect there to exist a radius r^* such that a system of N ions in a $\gamma = 1$ potential spaced equally in a circle is in equilibrium. Consider such a system, of ions equally spaced at a radius r. Since the system is rotationally symmetric, the force on each ion must be the same, and the force on each ion must be either into the origin or out of the origin (in the $\hat{r}$ direction). For r sufficiently small, the ions are very close together, and the force on each ion must be outwards (in the $+\hat{r}$ direction). On the other hand, for r sufficiently large, the ions are far apart from each other and must be attracted together and the force must be inwards (in the $-\hat{r}$ direction). Since we expect the force on each ion as a function of r to be a continuous function, by the intermediate value theorem, there must be a value of r for which the force on each ion vanishes. Of course, we can still say nothing about the stability of such a point of equilibrium.

4 Convergence

To determine the optimal time step at which to run the simulation we look at how the temperature of the system evolves over time. It seems as though the larger the timestep, the faster the temperature of the system increases. For a timestep of $\delta t = 1\,\mathrm{ns}$ the simulation can run roughly $100\,\mu\mathrm{s}$ before

the temperature increases significantly. However for this case, the temperature of the simulation has a tendency to roughly double over the course of 1000 µs. That is, going from around 5 mK to 10 mK. When we implement laser cooling, this is no longer the case, and so we will stick to a timestep of $\delta t = 1$ ns. Further investigations should run a full convergence test.

5 Codebase Structure

The codebase consists of the following Python files:

- `natural_units.py` — defines physical constants and conversions.
- `utility_functions.py` — general-purpose numerical helpers.
- `laser.py` — models laser cooling physics.
- `simulation_module.py` — core simulation logic: initialization, dynamics, and visualization.
- `histogram_workers.py` — handles histogram production functions.
- `parralel_workers.py` — handles functions that are later parallelized.
- `parralel.py` — handles functions that run in parallel.
- `1_example_simple_simulation.ipynb` — example notebook for running a simple simulation.
- `2_example_quench_series.ipynb` — example notebook for running a quench series.
- `3_generating_equilibrium_configurations.ipynb` — example notebook for generating equilbirium configurations.

We now explain each of these in detail.

5.1 `natural_units.py`

This file defines physical constants in natural units used throughout the simulation. These constants are crucial for scaling energies, forces, and temperatures appropriately. For example:

```
hbar = 0.063507799295889 # Reduced Plank's constant((amu)(um^2)(us^-1))
kB = 1. # Boltzmann constant (in units of kB)
k = 1.38935378902e5 # Coulomb constant ((e^2)(um^3)(us^-2)(amu^-1))
epsilon_0 = 5.7276607423e-7 # Vacuum permitivity ((amu)(us)(um^-3)(e))
electron_charge = 1. # Electron charge (in units of e)
```

This file also includes various functions to convert values from natural units into more standard units. For example, the function `theta_to_mK` accepts a temperature in natural units of θ and returns the corresponding temperature in units of mK.

5.2 `utility_functions.py`

This file includes various functions that mainly serve to convert various possible list structures into other list structures. All the functions are well described in the file.

For example, one particular function, `base_radius` gives an implementation of (62).

5.3 `laser.py`

This file implements the `Laser` class, which handles Doppler cooling of the ions. A `Laser` object may be initialized with a wavelength and various other parameters, and may then be used to compute the Doppler cooling force on a set of ions at a particular time.

5.4 `simulation_module.py`

This contains various classes that handle the initialization, and running of a simulation. A more detailed description of all of the classes is included as a comment at the top of the file. To get familiar with the functions included in here, take a look at section 6.1 where we have included an example Jupyter Notebook that walks through using this file to run a simple simulation.

5.5 `histogram_workers.py`

This contains a number of helper functions that generate histograms (ion density maps) based on trajectory data.

5.6 `parralel_workers.py`

This contains a number of functions that perform a single action. Usually running the simulation once in a particular way. For example, performing a quench and running the simulation, or

5.7 `parralel.py`

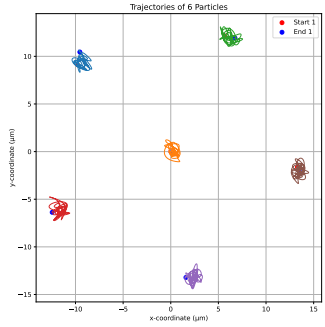
This contains all of the functions that use parallelization on execution.

6 Use Examples

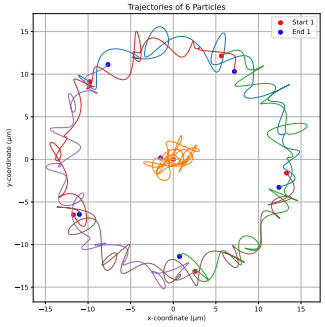
Included with the code are a handful of tutorial Jupyter Notebooks to be used to familiarize the user with the code. Certainly a minority of functions present in the code make an appearance in the tutorial notebooks, but most of the critical ones are present, and enough are used to as to allow the user to become familiar with the essentials.

6.1 Running a simple simulation

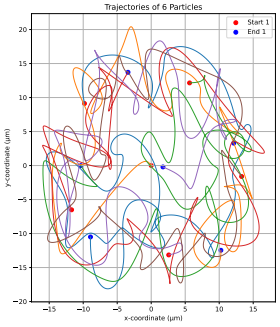
The Jupyter Notebook `1_example_simple_simulation.ipynb` gives a tutorial for using the code to run a simple simulation. As suggested in one of the markdown cells, by changing the initial temperature of the simulation, we can get the crystal of 6 ions to either stay frozen or to melt. Examples of those two conditions are given in Fig. 1. Generally the ion density maps acquired through `SimulationVisualizer().plot_density_map(state)` carry more useful information than the trajectory plots, especially as the length of the simulation gets large. For simulations of length 1000 µs or longer, the ion density maps are always preferred.



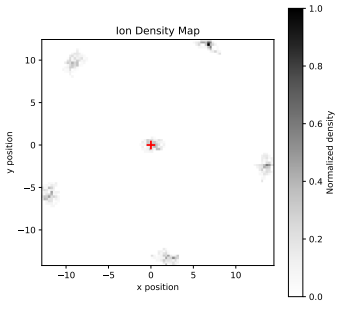
(a) Trajectory of localized ions at 10 mK



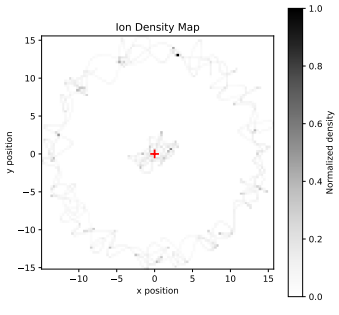
(b) Trajectory of partial melting at 50 mK



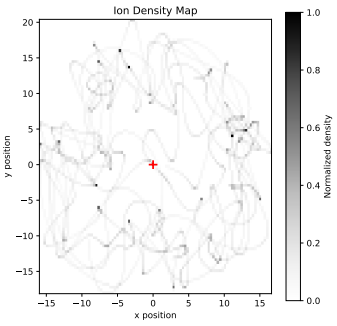
(c) Trajectory of full melting at 300 mK



(d) Density map of localized ions at 10 mK



(e) Density map of partial melting at 50 mK

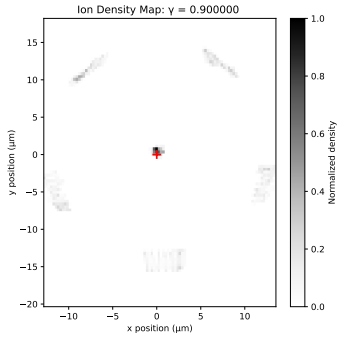


(f) Density map of full melting at 300 mK

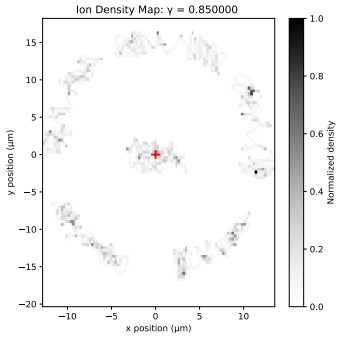
Figure 1: Simulations of 6 ions with an isotropy of 1 over 50 µs. Top row: trajectory maps; bottom row: ion density distributions.

6.2 Running a quench series

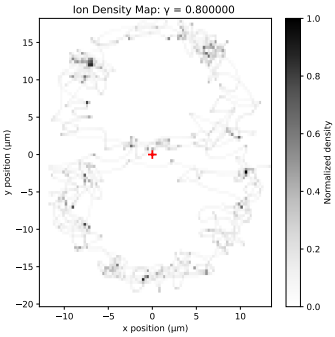
The Jupyter Notebook `2_example_quench_series.ipynb` gives a tutorial for using the code to run a series of quenches in parallel and animate the output. This notebook is computationally much more intensive than the one in Section 6.1 and relies on parallelized computation. In my experience, the fact that you will be running this code on a different machine to me means it is likely you will run into some kind of problem with the parallelized computation that might be an issue. This notebook simulates the system under 30 different quenches, each from $\gamma = 1$ to different values of $\gamma' \in [0.75, 0.9]$ for 100 µs. For example, below we display ion density, and temperature plots, for quenches to $\gamma = 0.9, 0.85$, and 0.8 in Fig. 2. We also present exactly the same data again, except under a simulation for 2000 µs seconds in Fig. 3 just to compare (though the Jupyter Notebook won't produce that by default).



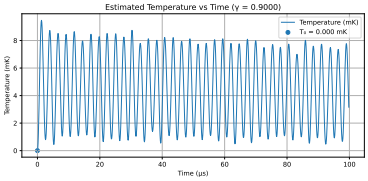
(a) Density map of quench to $\gamma = 0.9$.



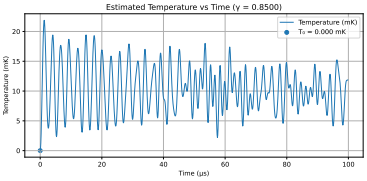
(b) Density map of quench to $\gamma = 0.85$.



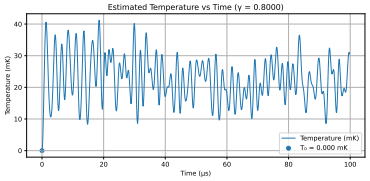
(c) Density map of quench to $\gamma = 0.8$.



(d) Temperature over time of quench to $\gamma = 0.9$.

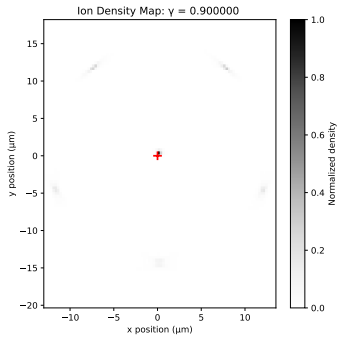


(e) Temperature over time of quench to $\gamma = 0.85$.

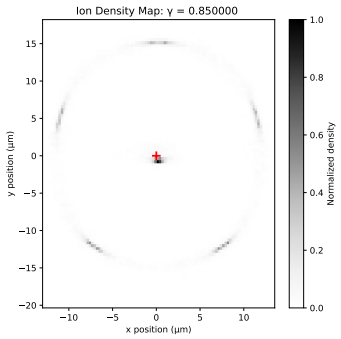


(f) Temperature over time of quench to $\gamma = 0.8$.

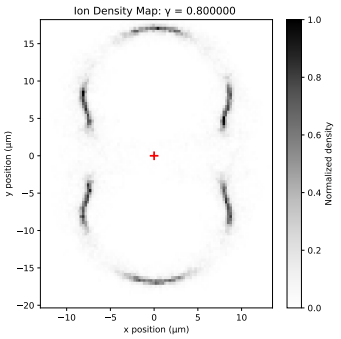
Figure 2: Ion density maps and temperature plots of quenches to three different isotropies. All simulations run over 100 µs with laser cooling.



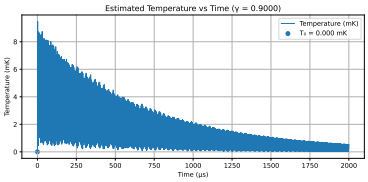
(a) Density map of quench to $\gamma = 0.9$.



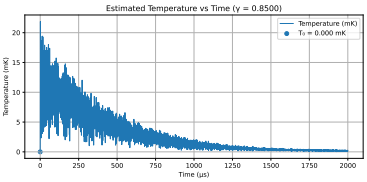
(b) Density map of quench to $\gamma = 0.85$.



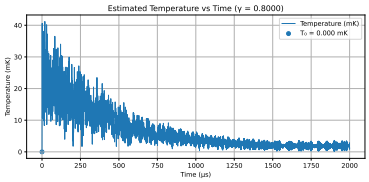
(c) Density map of quench to $\gamma = 0.8$.



(d) Temperature over time of quench to $\gamma = 0.9$.



(e) Temperature over time of quench to $\gamma = 0.85$.



(f) Temperature over time of quench to $\gamma = 0.8$.

Figure 3: Ion density maps and temperature plots of quenches to three different isotropies. All simulations run over 2000 μs with laser cooling. The decrease in temperature from the laser cooling is clear.

6.3 Generating equilibrium configurations

The Jupyter Notebook `3_generating_equilibrium_configurations.ipynb` gives a tutorial for using the code to generate equilibrium configurations to use as a starting point for running quench simulations. A number of pre-generated equilibrium configurations for various numbers of ions and various isotropy parameters are given in the code in the `input` folder.

7 Acknowledgements

We would like to thank Boris Blinov for his supervision. We also owe a huge debt of gratitude to Boris Pashinskii for the code that was passed to me. Much of this code grew out of beautiful code written originally by him.

References

[1] S. Ejtemaee and P. C. Haljan. "Spontaneous nucleation and dynamics of kink defects in zigzag arrays of trapped ions". In: *Physical Review A* 87.5 (2013), p. 051401. DOI: 10.1103/PhysRevA.87.051401.

[2] Boris V. Pashinsky, Alexander Kato, and Boris B. Blinov. "Structural Transitions and Melting of Two-Dimensional Ion Crystals in RF Traps". In: *Entropy* 27.4 (2025). © 2025 by the authors. Distributed under the terms of the Creative Commons Attribution (CC BY) license, p. 325. DOI: 10.3390/e27040325. URL: https://doi.org/10.3390/e27040325.