I. METHODS

When we study the Penning traps effect on a particle with a charge q, we need to consider the forces acting on the particle. We can use Newton's second law (**??**) to determine the position of a particle as a function of time. In addition, we introduce the Lorentz force (**??**), which describes the force on the particle. The position can be described by

$$m\ddot{\mathbf{r}} = (q\mathbf{E} + q\mathbf{v} \times \mathbf{B}). \tag{1}$$

Using eq. (**??**) we derive the differential equations in sec. **??**, and can rewrite them as

$$\ddot{x} - \omega_0\dot{y} - \frac{1}{2}\omega_z^2 x = 0, \tag{2}$$

$$\ddot{y} - \omega_0\dot{x} - \frac{1}{2}\omega_z^2 y = 0, \tag{3}$$

$$\ddot{z} + \omega_z^2 z = 0, \tag{4}$$

We find the general solution for eq. (**??**)

$$z(t) = c_1 e^{i\omega_z t} + c_2 e^{-i\omega_z t}, \tag{5}$$

derived in sec. **??**. Continuing, we will use a Calcium ion with a single positive charge. That is, we assume the charge of the particle is $q > 0$.

Since eq. (**??**) and eq. (**??**) are coupled, we want to rewrite them as a single differential equation. We derive this in sec. **??**, and the resulting equation is given by

$$\ddot{f} + i\omega_0\dot{f} - \frac{1}{2}\omega_z^2 f = 0. \tag{6}$$

Eq. (**??**) has a general solution given by

$$f(t) = A_+ e^{-i(\omega_+ t + \phi_+)} + A_- e^{-i(\omega_- t + \phi_-)}. \tag{7}$$

The amplitude A_+ and A_- are positive, the phases ϕ_+ and ϕ_- are constant, and the rate is given by

$$\omega_\pm = \frac{\omega_0 \pm \sqrt{\omega_0^2 - 2\omega_z^2}}{2}.$$

We find the physical coordinates at a given time t using

$$x(t) = \mathrm{Re} f(t), \quad y(t) = \mathrm{Im} f(t),$$

and eq. (**??**). We can rearrange the right hand side of the derived expression in **??**, and find the physical coordinates

$$x(t) = A_+ \cos(\omega_+ t + \phi_+) + A_- \cos(\omega_- t + \phi_-) \tag{8}$$

$$y(t) = -A_+ \sin(\omega_+ t + \phi_+) - A_- \sin(\omega_- t + \phi_-) \tag{9}$$

However, to obtain a bound on the particle's movement in the radial direction, we have to ensure that

$$|f(t)| = \sqrt{(x(t))^2 + (y(t))^2}.$$

When $t \to \infty$, the upper and lower limits are

$$R_+ = A_+ + A_- \tag{10}$$

$$R_- = |A_+ - A_-|, \tag{11}$$

derived in sec. **??**.

To obtain the bounded solution, we need to consider the expression

$$\alpha = (\omega_+ - \omega_-)t + (\phi_+ - \phi_-).$$

When $t \to \infty$ the constant phases will not affect the expression, we obtain a bounded solution if we consider the rate such that $|f(t)| < \infty$. That is, we need $\omega_0^2 - 2\omega_z^2$ in eq. (**??**) to be real.

$$\omega_0^2 - 2\omega_z^2 = 0, \quad \omega_0 > 2\omega_z$$

$$\omega_\pm = \frac{\omega_0 \pm \sqrt{\omega_0^2 - 2\omega_z^2}}{2} \tag{12}$$

We find the solution Physical properties given by newtons second law (**??**)

The particle moves and its position can be determined using newton. where the electric field

Constant	Value	
B_0	$1.00T$	$9.65 \times 10^1 \frac{u}{(\mu s)e}$
V_0	$25.0mV$	$2.41 \times 10^6 \frac{u(\mu m)^2}{(\mu s)^2 e}$
d	$500\ \mu m$	

TABLE I. Default configuration of the Penning trap, where the value of T and V can be found in table **??**.

A. Dealing with a multi–particle system

For a multi-particles system, we need to modify $\mathbf{F}$ to account for the force of other particles in the system acting upon each other. To do that, we add another term to $\mathbf{F}$

$$\mathbf{F}_i(t, \mathbf{v}_i, \mathbf{r}_i) = q_i\mathbf{E}(t, \mathbf{r}_i) + q_i\mathbf{v}_i \times \mathbf{B} - \mathbf{E}_p(t, \mathbf{r}_i), \tag{13}$$

where i and j are particle indices and

$$\mathbf{E}_p(t, \mathbf{r}_i) = q_i k_e \sum_{j \neq i} q_j \frac{\mathbf{r_i} - \mathbf{r_j}}{|\mathbf{r_i} - \mathbf{r_j}|^3}. \tag{14}$$

Newton's second law for a particle i is then

$$\frac{d^2\mathbf{r}_i}{dt^2} = \frac{\mathbf{F}_i\left(t, \frac{d\mathbf{r}_i}{dt}, \mathbf{r_i}\right)}{m_i}, \tag{15}$$

We can then rewrite the second order ODE from equation **??** into a set of coupled first order ODEs. We now rewrite Newton's second law of motion as

$$\begin{aligned} \frac{d\mathbf{r}_i}{dt} &= \mathbf{v}_i \\ \frac{d\mathbf{v}_i}{dt} &= \frac{\mathbf{F}_i(t, \mathbf{v}_i, \mathbf{r}_i)}{m_i}. \end{aligned} \tag{16}$$

B. Forward Euler

For a particle i, the forward Euler method for a coupled system is expressed as

$$\begin{aligned}\mathbf{r}_{i,j+1} &= \mathbf{r}_{i,j} + h \cdot \frac{d\mathbf{r}_{i,j}}{dt} = \mathbf{r}_{i,j} + h \cdot \mathbf{v}_{i,j} \\ \mathbf{v}_{i,j+1} &= \mathbf{v}_{i,j} + h \cdot \frac{\mathbf{v}_{i,j}}{dt} = \mathbf{v}_{i,j} + h \cdot \frac{\mathbf{F}\left(t_j, \mathbf{v}_{i,j}, \mathbf{r}_{i,j}\right)}{m},\end{aligned} \tag{17}$$

for particle i where j is the current time step of the particle, m is the mass of the particle, and h is the step length.

When dealing with a multi-particle system, we need to ensure that we do not update the position of any particles until every particle has calculated their next step. An easy way of doing this is to create a copy of all the particles, then update the copy, and when all the particles have calculated their next step, simply replace the particles with the copies. Algorithm **??** provides an overview on how that can be achieved.

Algorithm 1 Forward Euler method

procedure EVOLVE FORWARD EULER($particles, dt$)
  $N \leftarrow$ Number of particles in $particles$
  $a \leftarrow$ Calculate $\frac{\mathbf{F_i}}{m_i}$ for each particle in $particles$
  for $i = 1, 2, \ldots, N$ **do**
    $particles_i.\mathbf{r} \leftarrow particles_i.\mathbf{r} + dt \cdot particles_i.\mathbf{v}$
    $particles_i.\mathbf{v} \leftarrow particles_i.\mathbf{v} + dt \cdot a_i$

C. 4th order Runge-Kutta

For a particle i, we can express the 4th order Runge-Kutta (RK4) method as

$$\begin{aligned}\mathbf{v}_{i,j+1} &= \mathbf{v}_{i,j} + \frac{h}{6}\left(\mathbf{k}_{\mathbf{v},1,i} + 2\mathbf{k}_{\mathbf{v},2,i} + 2\mathbf{k}_{\mathbf{v},3,i} + \mathbf{k}_{\mathbf{v},4,i}\right) \\ \mathbf{r}_{i,j+1} &= \mathbf{r}_{i,j} + \frac{h}{6}\left(\mathbf{k}_{\mathbf{r},1,i} + 2\mathbf{k}_{\mathbf{r},2,i} + 2\mathbf{k}_{\mathbf{r},3,i} + \mathbf{k}_{\mathbf{r},4,i}\right),\end{aligned} \tag{18}$$

where

$$\begin{aligned}\mathbf{k}_{\mathbf{v},1,i} &= \frac{\mathbf{F}_i(t_j, \mathbf{v}_{i,j}, \mathbf{r}_{i,j})}{m} \\ \mathbf{k}_{\mathbf{r},1,i} &= \mathbf{v}_{i,j} \\ \mathbf{k}_{\mathbf{v},2,i} &= \frac{\mathbf{F}_i(t_j + \frac{h}{2}, \mathbf{v}_{i,j} + h \cdot \frac{\mathbf{k}_{\mathbf{v},1,i}}{2}, \mathbf{r}_{i,j} + h \cdot \frac{\mathbf{k}_{\mathbf{r},1,i}}{2})}{m} \\ \mathbf{k}_{\mathbf{r},2,i} &= \mathbf{v}_{i,j} + h \cdot \frac{\mathbf{k}_{\mathbf{v},1,i}}{2} \\ \mathbf{k}_{\mathbf{v},3,i} &= \frac{\mathbf{F}_i(t_j + \frac{h}{2}, \mathbf{v}_{i,j} + h \cdot \frac{\mathbf{k}_{\mathbf{v},2,i}}{2}, \mathbf{r}_{i,j} + h\frac{\cdot \mathbf{k}_{\mathbf{r},2,i}}{2})}{m} \\ \mathbf{k}_{\mathbf{r},3,i} &= \mathbf{v}_{i,j} + h \cdot \frac{\mathbf{k}_{\mathbf{v},2,i}}{2} \\ \mathbf{k}_{\mathbf{v},4,i} &= \frac{\mathbf{F}_i(t_j + h, \mathbf{v}_{i,j} + h \cdot \mathbf{k}_{\mathbf{v},3,i}, \mathbf{r}_{i,j} + h \cdot \mathbf{k}_{\mathbf{r},3,i})}{m} \\ \mathbf{k}_{\mathbf{r},4,i} &= \mathbf{v}_{i,j} + h \cdot \frac{\mathbf{k}_{\mathbf{v},1,i}}{2}.\end{aligned} \tag{19}$$

In order to find each $\mathbf{k}_{\mathbf{r},i}$ and $\mathbf{k}_{\mathbf{v},i}$, we need to first compute all $\mathbf{k}_{\mathbf{r},i}$ and $\mathbf{k}_{\mathbf{v},i}$ for all particles, then we can update the particles in order to compute $\mathbf{k}_{\mathbf{r},i+1}$ and $\mathbf{k}_{\mathbf{v},i+1}$. In order for the algorithm to work, we need to save a copy of each particle before starting so that we can update the particles correctly for each step.

This approach would require 8 loops to be able to complete the calculation since we cannot update the particles until after all $\mathbf{k}$ values have been computed, however if we create a temporary array that holds particles, we can put the updated particles in there, and then use that array in the next loop, and would reduce the required amount of loops down to 4.

D. Testing the simulation

E. Relative error and error convergance rate

F. Tools

The numerical methods implemented in C++, are parallelized using OpenMP [**?**]. We used the Python library matplotlib [**?**] to produce all the plots, and seaborn [**?**] to set the theme in the figures.

Algorithm 2 RK4 method

procedure EVOLVE RK4($particles, dt$)
  $N \leftarrow$ Number of particles inside the Penning trap
  $orig\_p \leftarrow$ Copy of particles
  $tmp\_p \leftarrow$ Array of particles of size N
  $\mathbf{k_r} \leftarrow$ 2D array of vectors of size $4 \times N$
  $\mathbf{k_v} \leftarrow$ 2D array of vectors of size $4 \times N$
  for $i = 1, 2, \ldots, N$ **do**
    $\mathbf{k}_{\mathbf{r},1,i} \leftarrow particles_i.\mathbf{v}$
    $\mathbf{k}_{\mathbf{v},1,i} \leftarrow \frac{\mathbf{F}_i}{m_i}$
    $tmp\_p_i.\mathbf{r} \leftarrow orig\_p_i.\mathbf{r} + \frac{dt}{2} \cdot \mathbf{k}_{\mathbf{r},1,i}$
    $tmp\_p_i.\mathbf{v} \leftarrow orig\_p_i.\mathbf{v} + \frac{dt}{2} \cdot \mathbf{k}_{\mathbf{v},1,i}$
  $particles \leftarrow tmp\_p$ ▷ Update particles
  for $i = 1, 2, \ldots, N$ **do**
    $\mathbf{k}_{\mathbf{r},2,i} \leftarrow particles_i.\mathbf{v}$
    $\mathbf{k}_{\mathbf{v},2,i} \leftarrow \frac{\mathbf{F}_i}{m_i}$
    $tmp\_p_i.\mathbf{r} \leftarrow orig\_p_i.\mathbf{r} + \frac{dt}{2} \cdot \mathbf{k}_{\mathbf{r},2,i}$
    $tmp\_p_i.\mathbf{v} \leftarrow orig\_p_i.\mathbf{v} + \frac{dt}{2} \cdot \mathbf{k}_{\mathbf{v},2,i}$
  $particles \leftarrow tmp\_p$ ▷ Update particles
  for $i = 1, 2, \ldots, N$ **do**
    $\mathbf{k}_{\mathbf{r},3,i} \leftarrow particles_i.\mathbf{v}$
    $\mathbf{k}_{\mathbf{v},3,i} \leftarrow \frac{\mathbf{F}_i}{m}$
    $tmp\_p_i.\mathbf{r} \leftarrow orig\_p_i.\mathbf{r} + dt \cdot \mathbf{k}_{\mathbf{r},3,i}$
    $tmp\_p_i.\mathbf{v} \leftarrow orig\_p_i.\mathbf{v} + dt \cdot \mathbf{k}_{\mathbf{v},3,i}$
  $particles \leftarrow tmp\_p$ ▷ Update particles
  for $i = 1, 2, \ldots, N$ **do**
    $\mathbf{k}_{\mathbf{r},4,i} \leftarrow particles_i.\mathbf{v}$
    $\mathbf{k}_{\mathbf{v},4,i} \leftarrow \frac{\mathbf{F}}{m}$
    $tmp\_p_i.\mathbf{r} \leftarrow orig\_p_i.\mathbf{r} + \frac{dt}{6} \cdot (\mathbf{k}_{\mathbf{r},1,i} + \mathbf{k}_{\mathbf{r},2,i} + \mathbf{k}_{\mathbf{r},3,i} + \mathbf{k}_{\mathbf{r},4,i})$
    $tmp\_p_i.\mathbf{v} \leftarrow orig\_p_i.\mathbf{v} + \frac{dt}{6} \cdot (\mathbf{k}_{\mathbf{v},1,i} + \mathbf{k}_{\mathbf{v},2,i} + \mathbf{k}_{\mathbf{v},3,i} + \mathbf{k}_{\mathbf{v},4,i})$
  $particles \leftarrow tmp\_p$ ▷ Final update

F in the algorithm does not take any arguments as it uses the velocities and positions of the particles inside the array *particles* to calculate the total force acting on particle i.