Cyclotron Orbit Simulation Using the Boris Algorithm

Robert Buckley

Abstract

Can cyclotrons effectively accelerate charged particles? How successful is the Boris integration algorithm at simulating cyclotrons? We modeled and implemented iPython simulations for a non-relativistic cyclotron using both the traditional Euler differential equation algorithm and the Boris integration algorithm. We found that only the Boris integration algorithm could conserve energy during the cyclotron simulation. Examining the final simulation, we found that cyclotrons are an excellent way to accelerate particles with limited space.

Introduction and Motivation

Cyclotrons have been a staple of physics for decades. They can generate high-energy beams at a small cost and with little space. It's important to understand cyclotrons because the high-energy beams that they generate are a staple in nuclear physics experiments.[1] Also, cyclotrons can be used in radiation therapy for cancer.[2] While the behavior of cyclotrons is already understood in the physics community, I want to better understand cyclotrons to appreciate their historical significance and apply them to the field of biomedical engineering.

Since cyclotrons accelerate a charged particle in periodic circles, it's important to simulate them accurately. In an inaccurate simulation, a small amount of error per cycle would quickly compound and cause the particle to spiral out of control. For this reason, it's well known that the simplest Euler method and Riemann sum approach to solving the cyclotron's differential equations fails. It is well known that the Boris algorithm performs better for particle simulations.[3] This project is essential to our education in that it teaches us how to simulate particle motion, which is a necessary skill as a physics student.

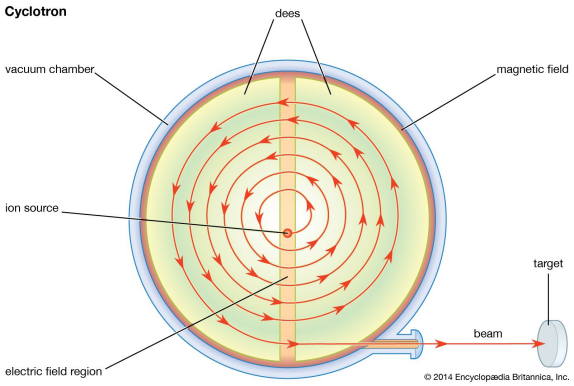
Background

[1] http://www.int.washington.edu/NNPSS/2016/lectures/Cyclotrons_for_Nuclear_Physics.pdf, National Nuclear Physics Summer School at the University of Washington, Cyclotrons for Nuclear Physics: Past, Present, Future

[2] https://www.uab.edu/news/health/item/10246-uab-s-newest-cancer-killing-cyclotron-installed, University of Alabama Birmingham, UAB's Newest Cancer-Killing Cyclotron Installed

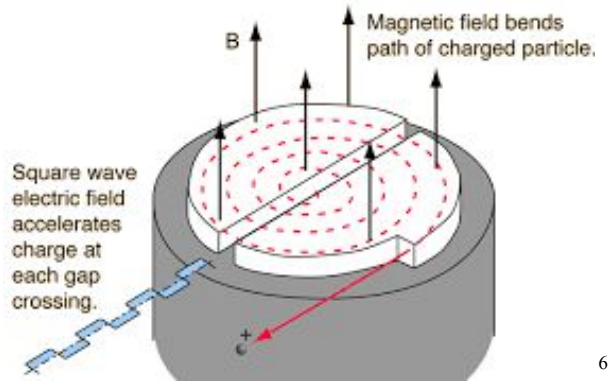
[3] https://www.particleincell.com/2011/vxb-rotation/, Particle In Cell Consulting LLC, Particle Push in Magnetic Field (Boris Method)

A cyclotron accelerates charged particles into spiraling, semicircle paths using the laws of electricity and magnetism. The cyclotron consists of two D-shaped pieces of metal known as "dees." A high frequency alternating voltage is applied to both dees in order to create a constant magnetic field perpendicular of the dees. When a charged particle is placed in one of the Dees, it travels in spirals from a Lorentz force perpendicular to the direction of the particle's motion and parallel to its plane of motion. The particle's speed increases when it is between the two Dees. Between the dees, the cyclotron creates a square wave electric field with a resonant frequency so that the force is in the same direction as the particle's velocity. At the edge of a dee, there is a deflection plate that sends the particle out of the cyclotron in a straight line.[4]



[5]

Figure 1



[6]

Figure 2

Setup

Mathematical Model

We can model a particle's motion inside the cyclotron using the following differential equations derived from the Lorentz force:

$$\vec{F} = q(\vec{E} + \vec{v} \times \vec{B})$$

$$\frac{\mathrm{d}\vec{v}}{\mathrm{d}t} = \frac{q}{m}(\vec{E} + \vec{v} \times \vec{B})$$

$$\frac{\mathrm{d}\vec{r}}{\mathrm{d}t} = \vec{v}$$

As for the square wave electric field, it's resonant frequency is:[7]

---

[4] https://en.wikipedia.org/wiki/Cyclotron, Wikipedia, Cyclotron
[5] https://www.britannica.com/science/dee, Encyclopedia Britannica, Dee
[6] http://hyperphysics.phy-astr.gsu.edu/hbase/magnetic/cyclot.html, Georgia State University, Cyclotron
[7] https://www.didaktik.physik.uni-muenchen.de/elektronenbahnen/en/b-feld/anwendung/zyklotron2.php, LMU Munich, Calculation of the Cyclotron frequency (non-relativistic)

$$f = \frac{q|\vec{B}|}{2\pi m}$$

However, we can add a layer of abstraction to simplify the square wave electric field. Instead of simulating a radio wave with a given frequency, we can instead apply a constant electric force in the direction of motion when the particle is between the dees. The resonant frequency is still useful, however, in order to calculate timestep since the frequency corresponds to one loop in the particle's motion.

To simulate the motion of a particle in a cyclotron, we need to solve the above differential equations. As a start, we can update force, acceleration, velocity, and position explicitly over many small time steps according to the above equations using the Euler method (also known as Riemann sums).

To increase our simulation accuracy, the Boris algorithm offers a better update equation by cancelling out the electric field in the Lorentz Force. We describe the process for deriving the Boris algorithm below:[8]

The Boris algorithm introduces two intermediate velocities $\vec{v}^-$ and $\vec{v}^+$:

$$\vec{v}^- = \vec{v}^t + \frac{q\triangle t}{2m}\vec{E}^+$$

$$\vec{v}^+ = \vec{v}^{t+\triangle t} - \frac{q\triangle t}{2m}\vec{E}^+$$

The Boris algorithm then cancels out the electric field:

$$\frac{\vec{v}^+ - \vec{v}^-}{\triangle t} = \frac{e}{m}(\frac{\vec{v}^+ - \vec{v}^-}{2} \times \vec{B})$$

The Boris algorithm then makes the final update time independent:

$$\vec{v}^+ = \vec{v}^- + (\vec{v}^- + \vec{v}^- \times \vec{t}) \times s \text{ where } t = \frac{q\vec{B}\triangle t}{2m} \text{ and } s = \frac{2t}{1+t^2}$$

Note that the final result of the Boris algorithm is, by design, time independent. That means that it is difficult to adapt the Boris algorithm to make it accurate when accounting for relativity. So, in our iPython simulation, we ignore relativistic effects.

Python Implementation

[8] http://phoenix.ps.uci.edu/zlin/bib/weixs15.pdf, University of California Irvine, Method to Integrate Full Particle Orbit in Toroidal Plasmas

We first simulate a particle inside the cyclotron by iteratively solving the differential equation using the Euler method of Riemann sums. Each call to update_standard will perform an update over a given time step with the provided parameters. We make use of the np.cross function to efficiently compute the cross product of velocity and magnetic field. All vectors are 3D so that the cross products function correctly.

```
def update_standard(self, gap, e_magnitude, B, dt):
    '''Updates the position and velocity of the particle inside a cyclotron over some timestep dt
    with given gap, constant magnetic field, and square wave electric field with given magnitude.
    Uses the standard euler method/riemann sum for the corresponding differential equation.'''
    E = self.getE(gap, e_magnitude)
    force = self.charge * (E + np.cross(self.vel, B))
    self.accel = force / self.mass
    self.vel = self.vel + self.accel * dt
    self.pos = self.pos + self.vel * dt
```

The electric field E is defined by the function getE(), shown below. The function returns an electric field vector with the given magnitude in the direction of velocity if the particle is between the dees and an electric field vector with magnitude zero if it is not between the dees.

```
def getE(self, gap, e_magnitude):
    '''Returns the an eletric field vector in the same direction as velocity and with given
    magnitude if the particle is between the dees of the cyclotron to simulate a square wave
    electric field with a frequency so that it always increases the speed of the particle'''
    if abs(self.pos[1]) < gap / 2:
        if self.vel[1] > 0:
            return np.array([0, e_magnitude, 0])
        elif self.vel[1] < 0:
            return np.array([0, -1 * e_magnitude, 0])
    return 0
```

To achieve more accurate results, we implemented the Boris algorithm. The implementation follows the mathematical model described earlier in the report. We use the Boris algorithm to calculate the new velocity and the Euler method of Riemann sums to calculate the new position.

```
def update_boris(self, gap, e_magnitude, B, dt):
    '''Updates the position and velocity of the particle inside a cyclotron over some timestep dt
    with given gap, constant magnetic field, and square wave electric field with given magnitude.
    Uses the boris algorithm to calculate the update.'''
    E = self.getE(gap, e_magnitude)
    self.vel = self.vel + (self.charge * E * dt) / (2 * self.mass)
    t = (self.charge * B * dt) / (2 * self.mass)
    v_plus = self.vel + np.cross(self.vel, t)
    s = (2 * t) / (1 + np.linalg.norm(t)**2)
    self.vel = self.vel + np.cross(v_plus, s)
    self.vel = self.vel + (self.charge * E * dt) / (2 * self.mass)
    self.pos = self.pos + self.vel * dt
```

We used the following constants for our simulations, which can be easily changed by the programmer. We chose the total simulation to last ten periods and performed 1000 total iterations (additional iterations showed minimal change in results).

```
num_iterations = 1000
mass = 1.6726e-25 #kg
charge = 1.6e-19 #C
B = np.array([0, 0, 1]) #T
gap = .0001 #m

frequency = (charge * np.linalg.norm(B)) / (2 * np.pi * mass) #hz
period = 1 / frequency #s
time = 10 * period #s
dt = time / num_iterations #s
t = np.arange(0, time, dt)
rect_height = 100
```

Then, we call the particle's update equation using object oriented programming to minimize duplicate code. We keep track of the particle's position, velocity, and energy at each timestamp so that we can create meaningful plots.

```
def generate_plot(boris, e_magnitude):
    '''Iteratively calls the update function of a particle constructed with given simulation method
    (boris=false is euler and boris=true is boris) and plots the proton's path, velocity, and energy'''
    xpos, ypos, zpos = np.zeros(num_iterations), np.zeros(num_iterations), np.zeros(num_iterations)
    xvel, yvel, zvel = np.zeros(num_iterations), np.zeros(num_iterations), np.zeros(num_iterations)
    energy = np.zeros(num_iterations)
    p = Particle(mass, charge, 0, 0, 0, 100, 0, 0, accurate=boris)
    for i in range(num_iterations):
        p.update(gap, e_magnitude, B, dt)
        xpos[i], ypos[i], zpos[i] = p.pos
        xvel[i], yvel[i], zvel[i] = p.vel
        energy[i] = (mass /2) * (np.linalg.norm(p.vel) ** 2)
```

Results and Conclusions

We set up and ran four simulations of a proton in a cyclotron. Specifically, we simulated a proton in a cyclotron using the standard algorithm with no electric field, using the Boris algorithm with no electric field, using the standard algorithm with a 1000T electric field, and using the Boris algorithm with a 1000T electric field.

We found that the standard Euler algorithm did not correctly simulate a proton a cyclotron while the Boris algorithm was successful.

Examining the position graph for the standard algorithm simulation with no electric field (figure 3), we see that the particle spirals out of control when it should maintain an elliptical path. In addition, its energy seems to increase exponentially while it should remain constant. This is not physically possible since no energy is being added to the system without an electric field present.

Both the particle's x and y velocities increase despite there being no electric force to accelerate the particle. We conclude that the standard simulation failed.

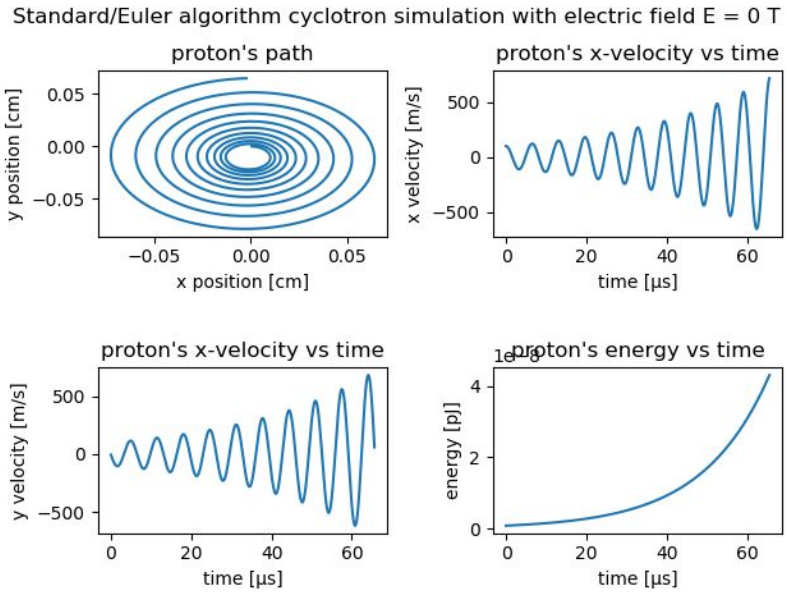


Figure 3

In contrast, the position graph for the Boris simulation with no electric field (figure 4) remains in an unchanging elliptical path for all 10 cycles. In addition, its energy remains constant at its near zero initial value. And again, as expected, the particle's x and y velocities oscillate as it travels the loop while its amplitude remains constant.

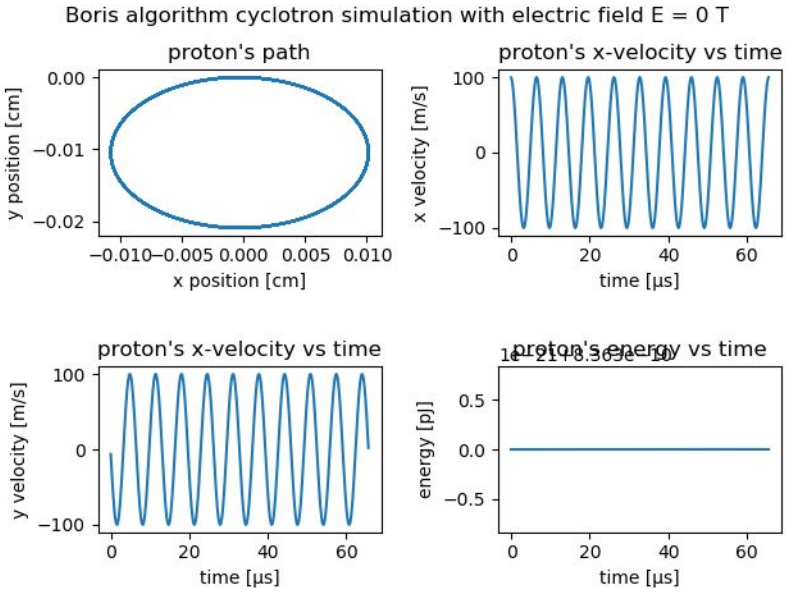


Figure 4

Now looking at the case where there is a non zero electric field between the dees, we see that the standard algorithm again fails. In figure 5 below, the energy vs. time graph is again growing

exponentially. However, the proton's energy should only increase when it is between the dees (pictured as the red sliver in the position graph). Since the standard algorithm fails to conserve energy, it must be incorrect.

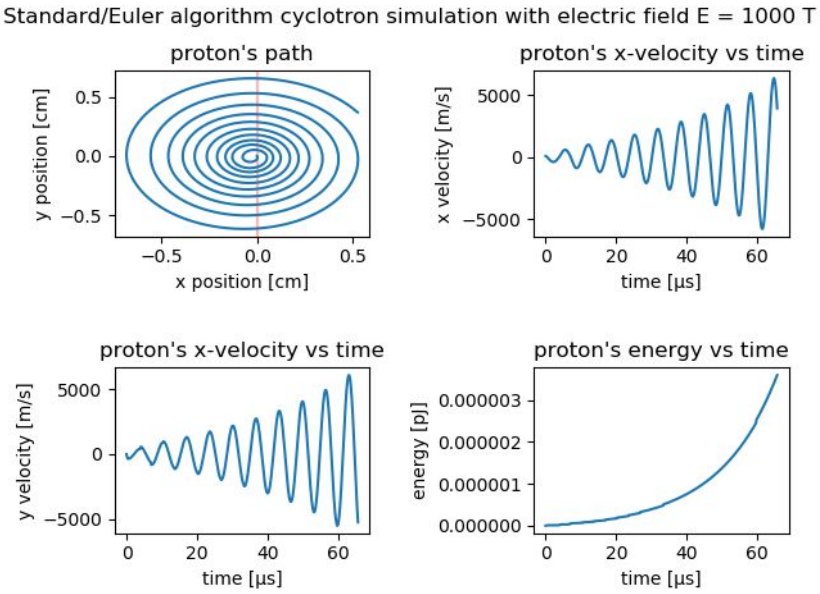


Figure 5

Finally, we examine the cyclotron with a non-zero electric field between the dees as simulated with the Boris algorithm. The results are consistent with the laws of physics. We see the proton's energy increasing when inside the red rectangle representing the gap between the dees. And the proton's speed increases overtime as the cyclotron is designed to do. Notice that the gaps between the loops of the proton's path decrease as the proton spirals outwards and speeds up, which we also expected.

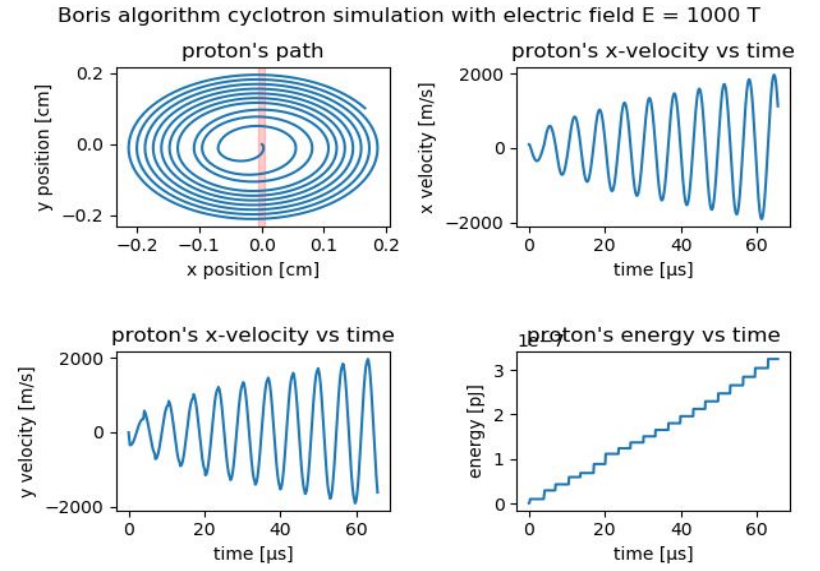


Figure 6

We can use the Boris algorithm simulation with non-zero electric to see how effective cyclotrons are. The cyclotron clearly increases the speed of a charged particle. The speed of the particle exiting a cyclotron is limited by the magnitude of its electric field and relativity, not the design of the cyclotron itself. So, the cyclotron has no issues concerning maximum speed. As for space required, the simulation suggests that the cyclotron can accelerate particles without much space needed. Because the loops in the particle's path are circular and increasingly less spaced apart, the proton can travel a long distance (and experience a lot of acceleration) without undergoing much total displacement. In the example, less than 1 cm width and height is used.

Moving forward, we would like to investigate the relativistic effects present in cyclotrons to find the maximum speed that a proton can reach in a cyclotron. However, due to time and knowledge constraints, we chose to not investigate relativity in this report. Simulating a particle in a cyclotron while accounting for relativity would be much more difficult because it would require a completely different integration algorithm or many modifications to the Boris algorithm likely outside the scope of this course. In summary, cyclotrons are a highly effective way to accelerate charged particles and the Boris algorithm outperforms standard integration when simulating charged particle motion under the Lorentz force.