Hydrogen Cyclotron
PHYS 4150 Final

Robert H. A. Sewell[1]†

[1]Department of Physics, University of Colorado, Boulder, CO 80302, USA

In this paper we will model and examine the characteristics of a Hydrogen cyclotron. A cyclotron is used to accelerate ions to high energies while still being contained to the device. This is done by applying a sinusoidal electric field at the cyclotron frequency of the particle so that the ion is always accelerated while still contained due to a constant B field in the device. For our model we developed a Fourth Order Runge-Kutta solver to characterize the Hydrogen atom's position based on Lorentz equation of motion. We also investigated the efficiency of the cyclotron depending on the driving frequency of the electric field.

1. Cyclotron Setup and Fourth Order Runge-Kutta Model

The cyclotron motion for a single Hydrogen atom is described by the Lorentz Force equation given by

$$m\frac{dv}{dt} = q(E + v \times B) \tag{1.1}$$

where m is the mass of the atom, v is the particle's velocity, q is the atomic charge, B is the constant magnetic field applied inside the device, and E is tunable oscillatory forcing electric field. This forcing electric field is applied in the x-direction and varies in time given by the complex form

$$\boldsymbol{E}(t) = E_0 e^{i\omega t}\hat{\boldsymbol{x}} \tag{1.2}$$

Equation 1.1 can be decomposed into its transverse components

$$\dot{v}_x = \frac{qB}{m}v_y + \frac{q}{m}E_x(t) \qquad \dot{v}_y = -\frac{qB}{m}v_x \tag{1.3}$$

We further decomposed these coupled differential equations to get four coupled linear ODEs in terms of the position of the particle. To do this we had the following substitutions

$$u = y \qquad w = \frac{dy}{dt} \qquad a = x \qquad b = \frac{dx}{dt} \tag{1.4}$$

Leading to the ODEs

$$\begin{aligned} \frac{du}{dt} &= w & \frac{da}{dt} &= b \\ \frac{dw}{dt} &= -\frac{qB}{m}b & \frac{db}{dt} &= \frac{qB}{m}w + \frac{q}{m}E_x(t) \end{aligned} \tag{1.5}$$

Using these coupled ODEs we implemented a fourth order Runge-Kutta iterative method to describe the velocity and position of the particle in the cyclotron. We used the Python programming language (Python Software Foundation, https://www.python.org/) to develop our simulation (see Appendix A full code).

† Email address for correspondence: robert.sewell@colorado.edu

As per the assignment instructions, we modeled the Hydrogen atom over 50 orbits using a magnetic field of 10 Tesla and allowing the electric field to oscillate between $\pm 1 \frac{N}{C}$ using a driving frequency the same as the cyclotron frequency ($\omega = \omega_c = \frac{qB}{m}$). We set the initial conditions so that the particle was at rest in the center of the cyclotron and, once again, all other constants (i.e. mass and charge) were set by the use of a Hydrogen ion.

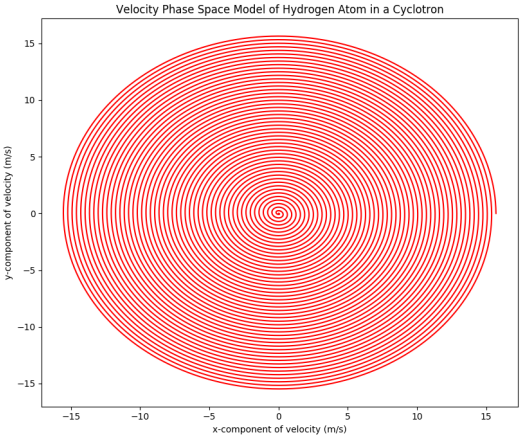

Figure 1: RK4 model of velocity

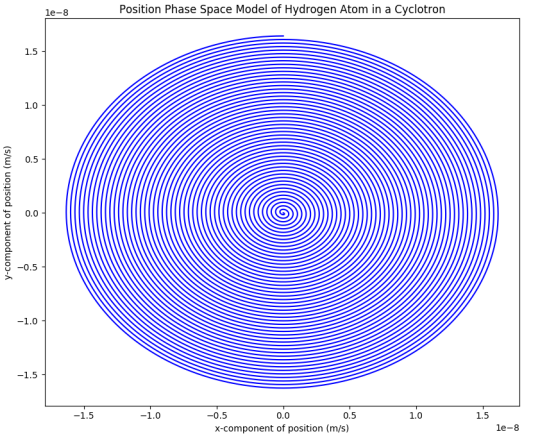

Figure 2: RK4 model of position

Above we can obviously see the cyclic motion of the particle in the cyclotron from the position phase space and we can also see the oscillatory components of the velocity which grows due to the acceleration from the forcing electric field.

2. Particle Velocity in a Cyclotron

As stated earlier, one of the advantages of a cyclotron is the ability to accelerate particles to high energies in a more compact space. However, we will look at the velocity of the particle in our model after 1000 orbits and compare it to the speed of a particle in a constant linear electric field with no interacting magnetic field to see which reaches a higher final velocity.
The force on a Hydrogen atom in linear electric field is given by

$$F = qE \implies \dot{v} = \frac{q}{m}E \tag{2.1}$$

For a particle that starts at rest, the same as in our cyclotron model, the final velocity the particle achieves is

$$v_f = \dot{v}t = \frac{qE}{m}t_f \tag{2.2}$$

The timescale for 1000 orbits, with a driving frequency the same as the cyclotron frequency of Hydrogen, is

$$t_f = 1000\frac{2\pi}{wc} = 1000\frac{2\pi m}{qB} \tag{2.3}$$

So our final velocity for a linearly accelerated particle (in a $1\frac{N}{C}E - field$) becomes

$$v_{f_{linear}} = 1000\frac{2\pi E}{B} \approx 628.32\frac{m}{s} \tag{2.4}$$

Now running our model for 1000 orbits (for 1×10^6 steps) we find the final components of our velocity to be

$$v_x \approx 314.16\frac{m}{s} \qquad v_y \approx 1.50 \times 10^{-5}\frac{m}{s} \tag{2.5}$$

Giving us a final total velocity of

$$v_{f_{cyclo}} = \sqrt{v_x^2 + v_y^2} \approx 314.16\frac{m}{s} \tag{2.6}$$

$$\frac{v_{f_{linear}}}{v_{f_{cyclo}}} = 2 \tag{2.7}$$

Looking at the relation between these to methods we can clearly see that in the same amount of time, the linearly accelerated particle is twice as fast as the particle accelerated in cyclotron. This is due to the fact that some of the ions energy in the cyclotron is put into turning due to the B-field. The advantage of the cyclotron, however, is that it is far more compact and easier to design than a linear accelerator that achieves the same speeds.

3. Driving Frequency of the Electric Field

To effectively accelerate particles in the cyclotron, the driving frequency of the electric field has to be very near the cyclotron frequency. This is because resonance is needed to accelerate the particle as it curls around the magnetic field.
To illustrate this we will alter the driving frequency to see how this effects resonance and ultimately the maximum velocity. Seen below are velocities and positions of the Hydrogen atom for 1000 orbits for different driving frequencies.

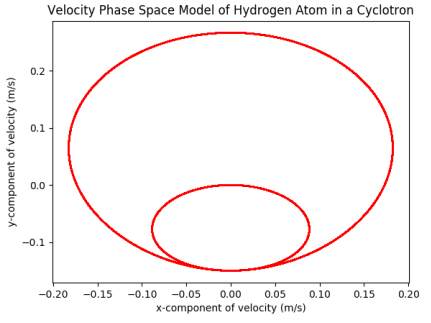

Figure 3: Velocity with driving frequency $\omega = 0.5\omega_c$

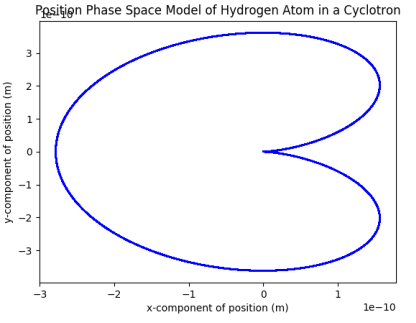

Figure 4: Position with driving frequency $\omega = 0.5\omega_c$

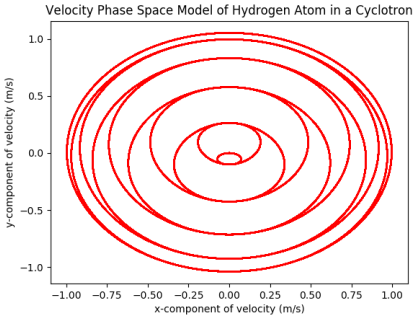

Figure 5: Velocity with driving frequency $\omega = 0.9\omega_c$

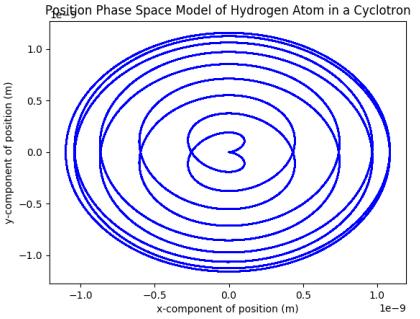

Figure 6: Position with driving frequency $\omega = 0.9\omega_c$

Here we can obviously see that resonance is not achieved in these cases. The maximum velocity achieved over 1000 orbits for a given driving frequency is summarized in the table below

Driving Frequency	Max Model Velocity
$\omega = 0.5\omega_c$	$0.27\frac{m}{s}$
$\omega = 0.9\omega_c$	$1.05\frac{m}{s}$
$\omega = 0.999\omega_c$	$100.05\frac{m}{s}$
$\omega = \omega_c$	$314.16\frac{m}{s}$

Table 1: Relation between E-field frequency and particle velocity

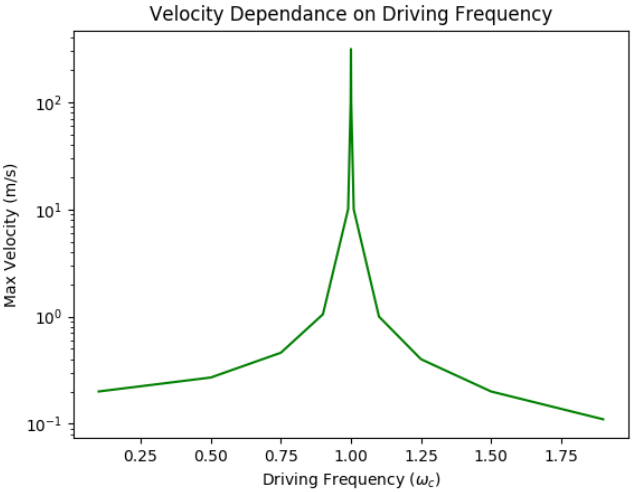

Figure 7: Log scale relation between driving frequency, in terms of cyclotron frequency, and particle velocity

We can clearly see that in order to get an effective cyclotron acceleration the driving frequency must be nearly that of the cyclotron frequency. Using our model we found that to get even 50% the max velocity of a cyclotron at ω_c requires a driving frequency

of $0.99936317\omega_c$. From this it is evident that the driving frequency must be set very accurately to get efficient acceleration of a particle in a cyclotron.

Appendix A

```
import sys
import math
import matplotlib.pyplot as plt
import numpy as np
import scipy as sci
# Title   : phys4150_final_extended.py
# Author  : Robert Sewell
# Date    : 12/16/17
# Synopsis: RK4 model of hydrogen cyclotron, extended to provide
#            velocities and positions.
# Keywords: runge-kutta
#
# Revisions:
# 12/16/17 Robert Sewell   Initial creation
#
#
# Usage   : python3 phys4150_final_extended
# Input   : None
# Output  : Plots of both particle velocties and position phase spaces
#
#
################################################################################
#Global declarations for input variable
vx0 = 0
vy0 = 0
x0 = 0
y0 = 0
E0 = 1
bfield = 10
mass = 1.6737236*10**(-27)
charge = 1.60217662*10**(-19)
wc = charge*bfield/mass
w = 2*wc #.999363179*wc for 1/2 max vel
tau=2*math.pi/wc
t0=0
tf=50*tau#1000*tau for Q2
n=10000 #1000000 for Q2
s=(tf-t0)/n
#print(charge*E0*tf/mass) linear velocity

#Electic field as a function of time
def Ef(t):
    return E0*math.cos(w*t)

#Coupled ODE definitions
```

```
def f(t,u,w,a,b):
    return w

def g(t,u,w,a,b):
    return -charge*bfield*b/mass

def h(t,u,w,a,b):
    return b

def q(t,u,w,a,b):
    return (charge*bfield*w)/mass+charge/mass*Ef(t)

#Main program loop
def main(argv):
    #Initial conditions of variable lists
    t=[t0]
    a=[x0]
    b=[vx0]
    u=[y0]
    w=[vy0]

    #Main RK4 loop
    for i in range(0,n):
        k1 = s*f(t[i], u[i], w[i], a[i], b[i])
        m1 = s*g(t[i], u[i], w[i], a[i], b[i])
        l1 = s*h(t[i], u[i], w[i], a[i], b[i])
        d1 = s*q(t[i], u[i], w[i], a[i], b[i])

        k2 = s*f(t[i] + s/2, u[i] + k1/2, w[i] + m1/2, a[i] + l1/2, b[i] + d1/2)
        m2 = s*g(t[i] + s/2, u[i] + k1/2, w[i] + m1/2, a[i] + l1/2, b[i] + d1/2)
        l2 = s*h(t[i] + s/2, u[i] + k1/2, w[i] + m1/2, a[i] + l1/2, b[i] + d1/2)
        d2 = s*q(t[i] + s/2, u[i] + k1/2, w[i] + m1/2, a[i] + l1/2, b[i] + d1/2)

        k3 = s*f(t[i] + s/2, u[i] + k2/2, w[i] + m2/2, a[i] + l2/2, b[i] + d2/2)
        m3 = s*g(t[i] + s/2, u[i] + k2/2, w[i] + m2/2, a[i] + l2/2, b[i] + d2/2)
        l3 = s*h(t[i] + s/2, u[i] + k2/2, w[i] + m2/2, a[i] + l2/2, b[i] + d2/2)
        d3 = s*q(t[i] + s/2, u[i] + k2/2, w[i] + m2/2, a[i] + l2/2, b[i] + d2/2)

        k4 = s*f(t[i] + s, u[i] + k3, w[i] + m3, a[i] + l3, b[i] + d3)
        m4 = s*g(t[i] + s, u[i] + k3, w[i] + m3, a[i] + l3, b[i] + d3)
        l4 = s*h(t[i] + s, u[i] + k3, w[i] + m3, a[i] + l3, b[i] + d3)
        d4 = s*q(t[i] + s, u[i] + k3, w[i] + m3, a[i] + l3, b[i] + d3)

        k = 1/6*(k1 + 2*k2 + 2*k3 + k4)
        m = 1/6*(m1 + 2*m2 + 2*m3 + m4)
        l = 1/6*(l1 + 2*l2 + 2*l3 + l4)
        d = 1/6*(d1 + 2*d2 + 2*d3 + d4)

        t.append(t[i] + s)
        u.append(u[i] + k)
```

```
            w.append(w[i] + m)
            a.append(a[i] + l)
            b.append(b[i] + d)
        #Plot velocities
        plt.figure()
        plt.plot(b,w,'r')#,'b.',markersize=2)
        plt.title('Velocity Phase Space Model of Hydrogen Atom in a Cyclotron')
        plt.ylabel('y-component of velocity (m/s)')
        plt.xlabel('x-component of velocity (m/s)')

        #Plot postions
        plt.figure()
        plt.plot(a,u,'b')#,'r.',markersize=2)
        plt.title('Position Phase Space Model of Hydrogen Atom in a Cyclotron')
        plt.ylabel('y-component of position (m)')
        plt.xlabel('x-component of position (m)')
        #y_int=sci.integrate.simps(vy,t)
        #x_int=sci.integrate.simps(vx,t)
        #print(y_int)
        #plt.figure()
        #plt.plot(y_int,x_int)
        #print(w[-1])
        #print(b[-1])

        #Given the maximum achieved velocity
        if(max(w)>max(b)):
            print(math.sqrt(w[w.index(max(w))]**2+b[w.index(max(w))]**2))
        else:
            print(math.sqrt(w[b.index(max(b))]**2+b[b.index(max(b))]**2))

        #For plotting velocity dependence on driving frequency
        #vel_list=[0.20,.27,.46,1.05,10.05,100.05,314.16,100,10,1,.4,.2,.11]
        #omeg_list=[.1,.5,.75,.9,.99,.999,1,1.001,1.01,1.1,1.25,1.5,1.9]
        #plt.figure()
        #plt.plot(omeg_list,vel_list,'g')#,'r.',markersize=2)
        #plt.yscale('log')
        #plt.title('Velocity Dependance on Driving Frequency')
        #plt.ylabel('Max Velocity (m/s)')
        #plt.xlabel(r'Driving Frequency ($\omega_c$)')

    if __name__ == "__main__":
        main(sys.argv)
```