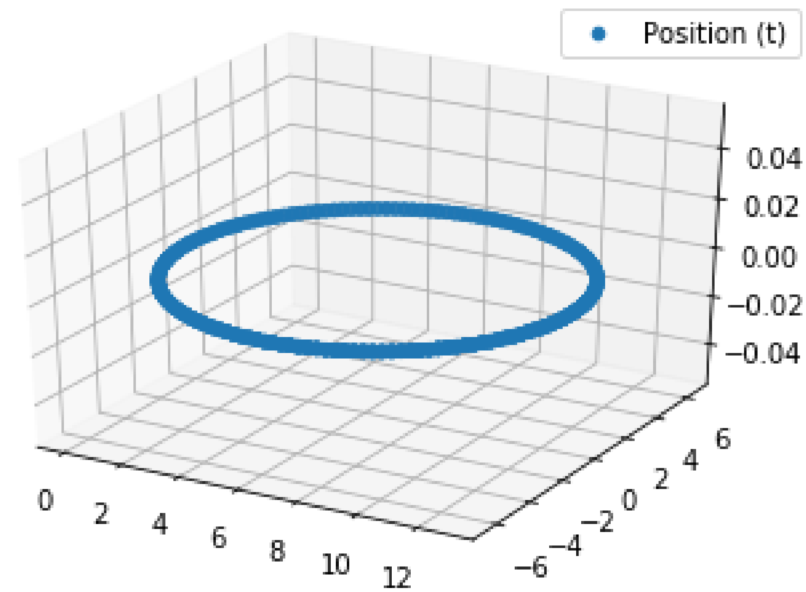
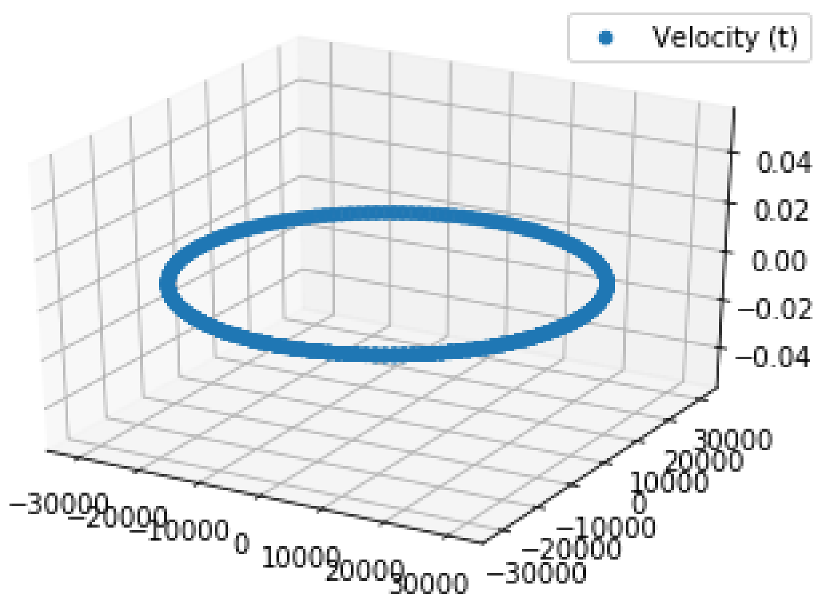
Assignment 6

March 19, 2018

In [4]:
```
#Exercise 1

from scipy.integrate import odeint
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import axes3d
import math

def mag_dip(y, t, B):
    p = y[:3]
    v = y[3:]
    a = np.cross(v, B)
    dydt = [v[0], v[1], v[2], a[0], a[1], a[2]]
    return dydt

def plot_path(sol):
    fig = plt.figure()
    ax = fig.gca(projection='3d')
    ax.scatter(sol[:,0], sol[:,1], sol[:,2], label='Position (t)')
    ax.legend(loc='best')

def plot_vel(sol):
    fig = plt.figure()
    ax = fig.gca(projection='3d')
    ax.scatter(sol[:,3], sol[:,4], sol[:, 5], label='Velocity (t)')
    ax.legend(loc='best')

# Alpha particle
q = 2*(1.6*(10**(-19)))
b = 10**(-4)
m = 6.644*(10**-27)
B = [0,0,q*b/m]

y0 = [0.0,0.0,0.0, 0, 31000, 0]
t = np.linspace(0, np.pi, 1000)

sol = odeint(mag_dip, y0, t, args=(B,))
```

```
plot_path(sol)
plot_vel(sol)
plt.show()
```

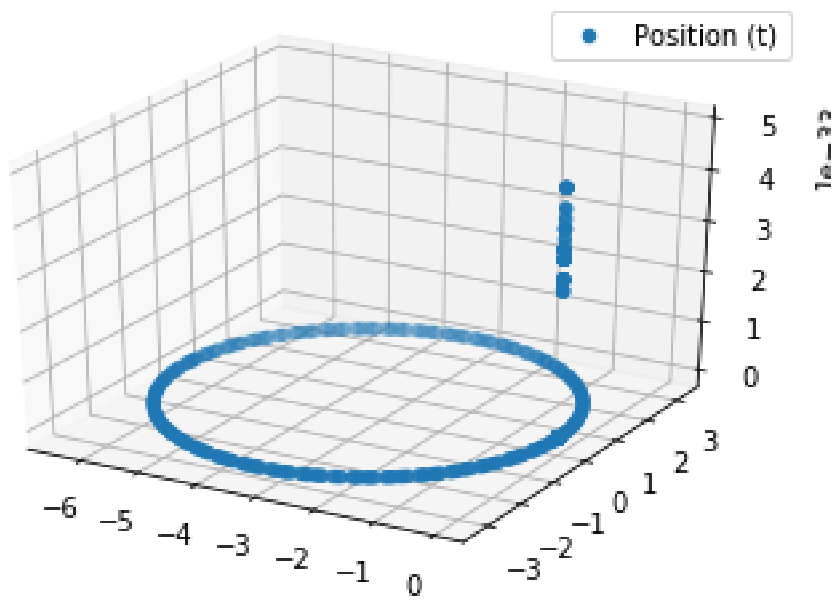


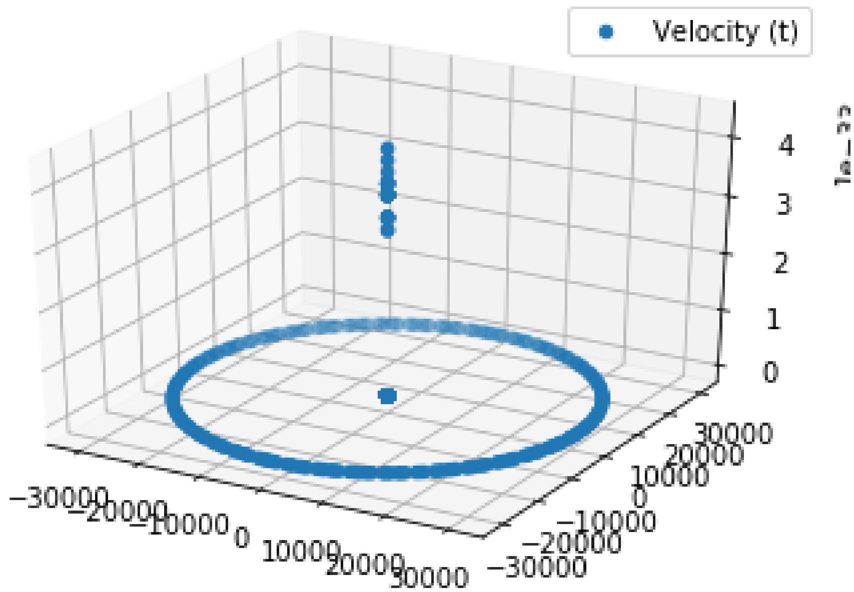



```
In [5]: # H- ion
        q = -1.6*(10**(-19))
```

```
b = 10**(-4)
m = 1.67*(10**-27)
B = [0,0,q*b/m]

y0 = [0.0,0.0,0.0, 0, 31000, 0]
t = np.linspace(0, np.pi, 1000)

sol = odeint(mag_dip, y0, t, args=(B,))
plot_path(sol)
plot_vel(sol)
plt.show()
```

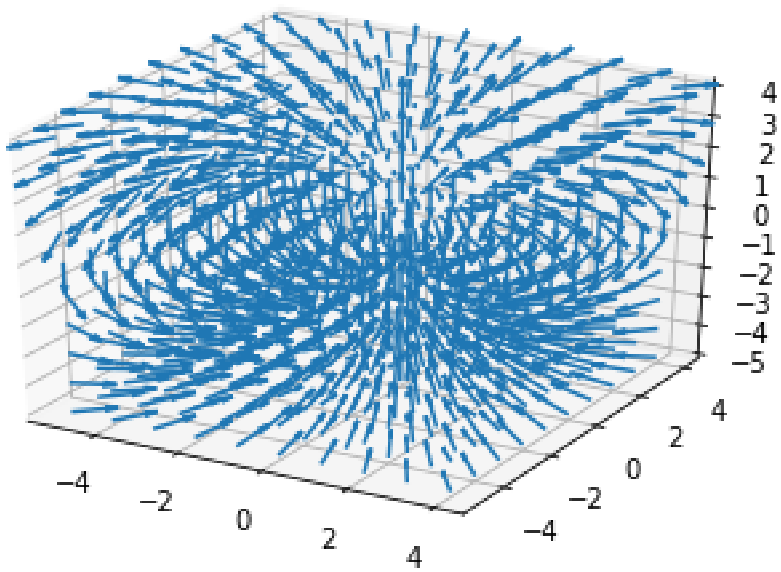
```
/Users/vishaal/Library/Python/2.7/lib/python/site-packages/scipy/integrate/odepack.py:218: ODE
  warnings.warn(warning_msg, ODEintWarning)
```

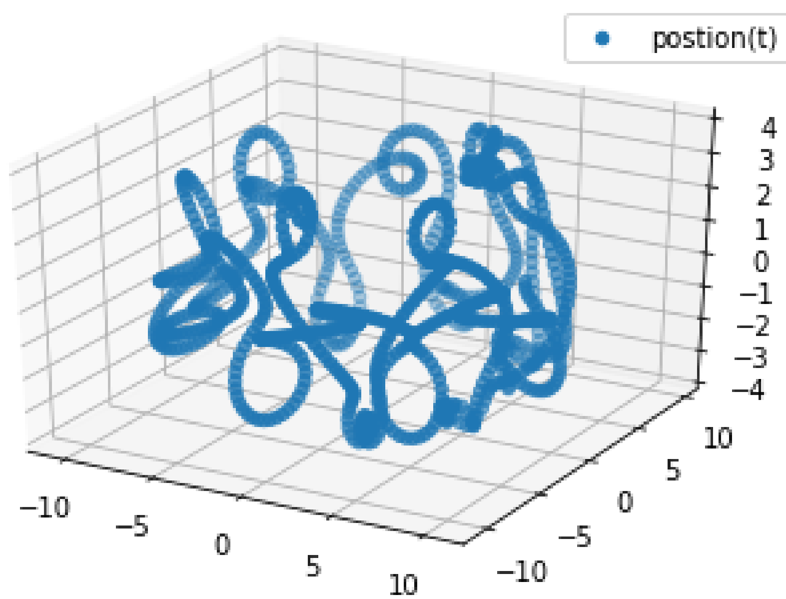


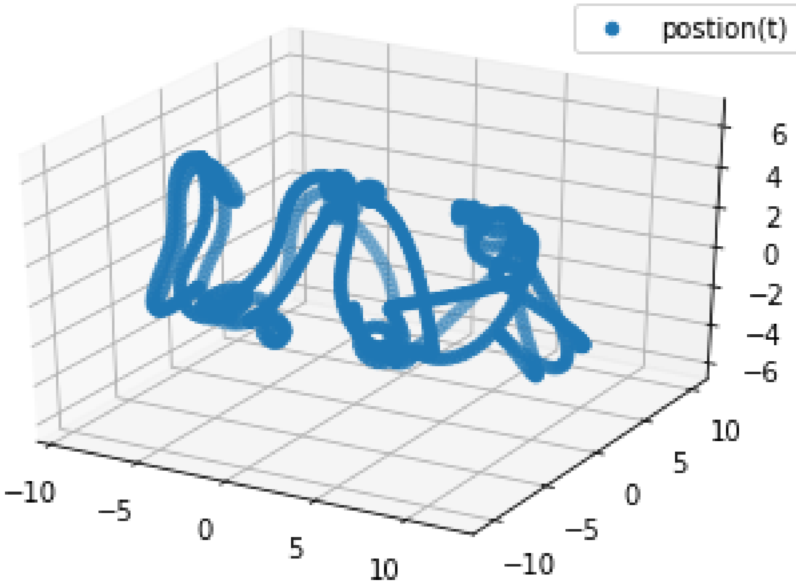



```
In [7]: #Exercise 2

        def dipole_field(x,y,z,deg, offset=0):
            r = (x**2 + y**2 + z**2)**0.5
            deg = math.radians(deg)
            sinx = math.sin(deg)
            cosx = math.cos(deg)
            mu_v = [0,(10**4)*sinx,(10**4)*cosx]

            if r == 0:
                B,C = np.dot(mu_v,[x,y,z]),0
            else:
                B = 3*(10**-7)*np.dot(mu_v,[x,y,z])/r**5
                C = (10**-7)/r**3

            b_v = [B*x - C*mu_v[0] , B*y - C*mu_v[1], B*(z-offset) - C*mu_v[2]]
            return b_v

        def mag_dip(y, t, q, m, deg, offset):
            vel_v = y[3:]
            x,y,z = y[:3]
            B_v = dipole_field(x,y,z, deg, offset)
            accel = (q/m)*np.cross(vel_v,B_v)
            dydt = [vel_v[0], vel_v[1], vel_v[2], accel[0], accel[1], accel[2]]
            return dydt

        def set_field_grid(): #(x,y,z):
```

```python
    x,y,z = [],[],[]
    u,v,w = [],[],[]
    for k in range(-5,5):
        for j in range(-5,5):
            for i in range(-5,5):
                t1,t2,t3 = dipole_field(i,j,k,0)
                u.append(t1)
                v.append(t2)
                w.append(t3)
                x.append(i)
                y.append(j)
                z.append(k)
    return x,y,z,u,v,w

def plot_dipole_magnetic_field():
    X,Y,Z,U,V,W = set_field_grid()
    fig = plt.figure()
    ax = fig.gca(projection='3d')
    ax.quiver(X, Y, Z, U, V, W, normalize= True)

def plot_dipole(sol):
    fig = plt.figure()
    ax = fig.gca(projection='3d')
    ax.scatter(sol[:, 0], sol[:, 1], sol[:, 2], label='Postion (t)')
    ax.legend(loc= 'best')

plot_dipole_magnetic_field()

# Alpha particle
q = 2*(1.6*(10**(-19)))
m = 6.644*(10**-27)
y0 = [0.0,-8.0,2, 0, 100, 0]
y1 = [0.0,-8.0,0, 0, 100, 0]
t = np.linspace(0, np.pi, 1000)

deg1, deg2 = 0, 23
sol0 = odeint(mag_dip, y0, t, args=(q, m, deg1, 0))
sol1 = odeint(mag_dip, y1, t, args=(q, m, deg2, 0))

plot_dipole(sol0)
plot_dipole(sol1)
plt.show()
```


4
3
2
1
0
−1
−2
−3
−4
−5
−4
−2
0
2
4
4
2
0
−2
−4


postion(t)
4
3
2
1
0
−1
−2
−3
−4
−10
−5
0
5
10
10
5
0
−5
−10




```
In [23]: #Exercise 3
         '''

         Couldn't get plot to work, sorry!
         :(

         '''

         def dipole_field_2(x,y,z,deg):
             r = (x**2 + y**2 + z**2)**0.5
             deg = math.radians(deg)
             sinx = math.sin(deg)
             cosx = math.cos(deg)
             mu_v = [0,(10**4)*sinx,(10**4)*cosx]

             if r == 0:
                 B,C = np.dot(mu_v,[x,y,z]),0
             else:
                 B = 3*(10**-7)*np.dot(mu_v,[x,y,z])/r**5
                 C = (10**-7)/r**3

             b_v = [B*x - C*mu_v[0] , B*y - C*mu_v[1], B*z - C*mu_v[2]]

             return b_v

         def set_bottlefield_grid(z_off1, z_off2):
             bound = 20
```

```
    y = [[j for j in range(-bound,bound)] for k in range(-bound,bound) ]
    z = [[k for j in range(-bound,bound)] for k in range(-bound,bound) ]
    v = [[ dipole_field_2(0,j,k - z_off1, 0)[1] + dipole_field_2(0,j,k - z_off2, 0) ]
    w = [[ dipole_field_2(0,j,k - z_off1, 0)[2] + dipole_field_2(0,j,k - z_off2, 0) ]

    y= np.array(y)
    z= np.array(z)
    w= np.array(w)
    v= np.array(v)

    return y,z,v,w

def plot_bottle(X,Y,U,V, title="Y vs Z plot for Magnetic bottle"):
    fig0, ax0 = plt.subplots()
    strm = ax0.streamplot(X, Y, U, V, color=U, linewidth=2, cmap=plt.cm.autumn)
    ax0.set_title(title)

y1,z1,v1,w1 = set_bottlefield_grid(10, -10)
plot_bottle(y1,z1,v1,w1)

def mag_dip_2(y, t, q, m, deg, offset1= 10, offset2=-10):
    v = y[3:]
    x,y,z = y[:3]

    B_v1 = dipole_field2(x,y,z - offset1, deg)
    B_v2 = dipole_field2(x,y,z - offset2, deg)
    B_v = [B_v1[0] + B_v2[0], B_v1[1] + B_v2[1], B_v1[2] + B_v2[2]]

    a = (q/m)*np.cross(v,B_v)
    dydt = [v[0], v[1], v[2], a[0], a[1], a[2]]

    return dydt

def plot_dip_2(sol):
    fig = plt.figure()
    ax = fig.gca(projection='3d')
    ax.scatter(sol[:, 0], sol[:, 1], sol[:, 2], label='Position (t)')
    ax.legend(loc= 'best')
    ax.set_title("Path in Magnetic Bottle")

def plot_z_vs_time(sol):
    fig, ax = plt.subplots()
    ax.scatter(t, sol[:, 2], label='z(t)')
    ax.legend(loc= 'best')
    ax.set_title("z(t)")

# Alpha particle
q = 2*(1.6*(10**(-19)))
```

```
m = 6.644*(10**-27)
y0 = [-5,0,0, 0, 0, 100]
t = np.linspace(0, np.pi, 1000)

deg1 = 0
sol0 = odeint(mag_dip_2, y0, t, args=(q,m,deg1,10,-10))

plot_dip_2(sol0)
plot_z_vs_time(sol0)
plt.show()
```

```
---------------------------------------------------------------------------

ValueError                                Traceback (most recent call last)

<ipython-input-23-7fe2b95abdcd> in <module>()
     36
     37 y1,z1,v1,w1 = set_bottlefield_grid(10, -10)
---> 38 plot_bottle(y1,z1,v1,w1)
     39
     40 def mag_dip_2(y, t, q, m, deg, offset1= 10, offset2=-10):


<ipython-input-23-7fe2b95abdcd> in plot_bottle(X, Y, U, V, title)
     32 def plot_bottle(X,Y,U,V, title="Y vs Z plot for Magnetic bottle"):
     33     fig0, ax0 = plt.subplots()
---> 34     strm = ax0.streamplot(X, Y, U, V, color=U, linewidth=2, cmap=plt.cm.autumn)
     35     ax0.set_title(title)
     36


/Users/vishaal/Library/Python/2.7/lib/python/site-packages/matplotlib/__init__.pyc in i
   1715                     warnings.warn(msg % (label_namer, func.__name__),
   1716                                   RuntimeWarning, stacklevel=2)
-> 1717             return func(ax, *args, **kwargs)
   1718         pre_doc = inner.__doc__
   1719         if pre_doc is None:


/Users/vishaal/Library/Python/2.7/lib/python/site-packages/matplotlib/axes/_axes.pyc i
   4619             zorder=zorder,
   4620             maxlength=maxlength,
-> 4621             integration_direction=integration_direction)
   4622         return stream_container
   4623     streamplot.__doc__ = mstream.streamplot.__doc__
```

```
/Users/vishaal/Library/Python/2.7/lib/python/site-packages/matplotlib/streamplot.pyc in
    114         if color.shape != grid.shape:
    115             msg = "If 'color' is given, must have the shape of 'Grid(x,y)'"
--> 116             raise ValueError(msg)
    117         line_colors = []
    118         color = np.ma.masked_invalid(color)


ValueError: If 'color' is given, must have the shape of 'Grid(x,y)'
```