Lesson 3

Ryan Branagan

February 28, 2019

Problem 1

```
#Ryan Branagan
#Collaborators: N/A
#Branagan_hw7_p1.py
#2/26/19

import numpy as np
import pylab as p
from scipy.integrate import odeint
#%%
def deriv(s,t,ma,mb,kl,km,kr):
    xa = s[0]
    va = s[1]
    xb = s[2]
    vb = s[3]
    deriv = [va,(-1*(kl*xa+km*(xa-xb))/(ma)),vb,(-1*(kr*xb+km*(xb-xa))/(mb))]
    return deriv

def DiffEq(si,t0,tf,points,deriv,params):
    ma = params[0]
    mb = params[1]
    kl = params[2]
    km = params[3]
    kr = params[4]
    t = np.linspace(t0,tf,points)
    s = odeint(deriv,si,t,args=(ma,mb,kl,km,kr))
    return t,s
#%%
#1
#Initial conditions
si = [5.,0.,-5.,0.]
```

```
t0 = 0
tf = 30
points = 3001
params = [1,1,0.5,1,1]

#Getting Points
t,solns = DiffEq(si,t0,tf,points,deriv,params)
xa = solns[:,0]
va = solns[:,1]
xb = solns[:,2]
vb = solns[:,3]

#Plotting
fig1,(pos,vel) = p.subplots(2,1,figsize=(6,8))
pos.plot(t,xa,label="Mass A")
pos.plot(t,xb,label="Mass B")
pos.set_title('Position of Both Masses Over Time')
pos.legend()
pos.set_ylabel('Position [m]')

vel.plot(t,va,label="Mass A")
vel.plot(t,vb,label="Mass B")
vel.set_title('Velocity of Both Masses Over Time')
vel.legend()
vel.set_xlabel('Time [s]')
vel.set_ylabel('Velocity [m/s]')
#%%
#2
#Initial conditions
si = [5.,0.,-5.,0.]
t0 = 0
tf = 30
points = 3001
params = [1,1000,0.5,1,1]

#Getting Points
t,solns = DiffEq(si,t0,tf,points,deriv,params)
xa = solns[:,0]
va = solns[:,1]
xb = solns[:,2]
vb = solns[:,3]

#Plotting
fig2,(pos,vel) = p.subplots(2,1,figsize=(6,8))
```

```
pos.plot(t,xa,label="Mass A")
pos.plot(t,xb,label="Mass B")
pos.set_title('Position of Both Masses Over Time')
pos.legend()
pos.set_ylabel('Position [m]')

vel.plot(t,va,label="Mass A")
vel.plot(t,vb,label="Mass B")
vel.set_title('Velocity of Both Masses Over Time')
vel.legend()
vel.set_xlabel('Time [s]')
vel.set_ylabel('Velocity [m/s]')
#%%
#3
#Initial conditions
si = [5.,0.,-5.,0.]
t0 = 0
tf = 30
points = 3001
params = [1,1,0.5,1,2]

#Getting Points
t,solns = DiffEq(si,t0,tf,points,deriv,params)
xa = solns[:,0]
va = solns[:,1]
xb = solns[:,2]
vb = solns[:,3]

#Plotting
fig3,(pos,vel) = p.subplots(2,1,figsize=(6,8))
pos.plot(t,xa,label="Mass A")
pos.plot(t,xb,label="Mass B")
pos.set_title('Position of Both Masses Over Time')
pos.legend()
pos.set_ylabel('Position [m]')

vel.plot(t,va,label="Mass A")
vel.plot(t,vb,label="Mass B")
vel.set_title('Velocity of Both Masses Over Time')
vel.legend()
vel.set_xlabel('Time [s]')
vel.set_ylabel('Velocity [m/s]')
#%%
#4
```

```
#Initial conditions
si = [5.,0.,-5.,0.]
t0 = 0
tf = 30
points = 3001
params = [1,20,5,2,5]

#Getting Points
t,solns = DiffEq(si,t0,tf,points,deriv,params)
xa = solns[:,0]
va = solns[:,1]
xb = solns[:,2]
vb = solns[:,3]

#Plotting
fig3,(pos,vel) = p.subplots(2,1,figsize=(6,8))
pos.plot(t,xa,label="Mass A")
pos.plot(t,xb,label="Mass B")
pos.set_title('Position of Both Masses Over Time')
pos.legend()
pos.set_ylabel('Position [m]')

vel.plot(t,va,label="Mass A")
vel.plot(t,vb,label="Mass B")
vel.set_title('Velocity of Both Masses Over Time')
vel.legend()
vel.set_xlabel('Time [s]')
vel.set_ylabel('Velocity [m/s]')
```

I thought the question asked to plot both the position and velocity, but it only asks for the position and I didn't fix it because I think it is useful to also see the velocity.

Problem 2

```
#Ryan Branagan
#Collaborators: N/A
#Branagan_hw7_p2.py
#2/27/19

import numpy as np
import pylab as p
from scipy.integrate import odeint
#%%
def deriv(s,t,ma,mb,kl,km,kr):
```

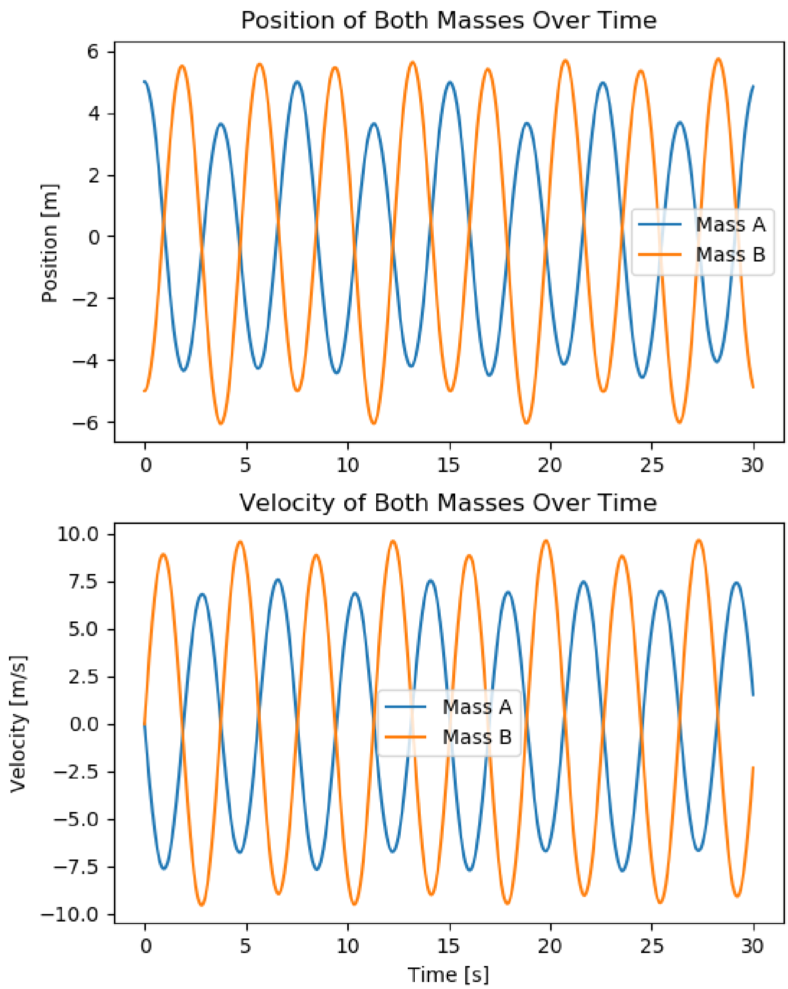


Figure 1: This is for equal masses and the left spring having half the k of the other two. If you look closely you can see how the two masses motions are coupled, they do not move together but you can see how where they intersect has a sinusoidal shape. So they oscillate back of forth pulling on each other.

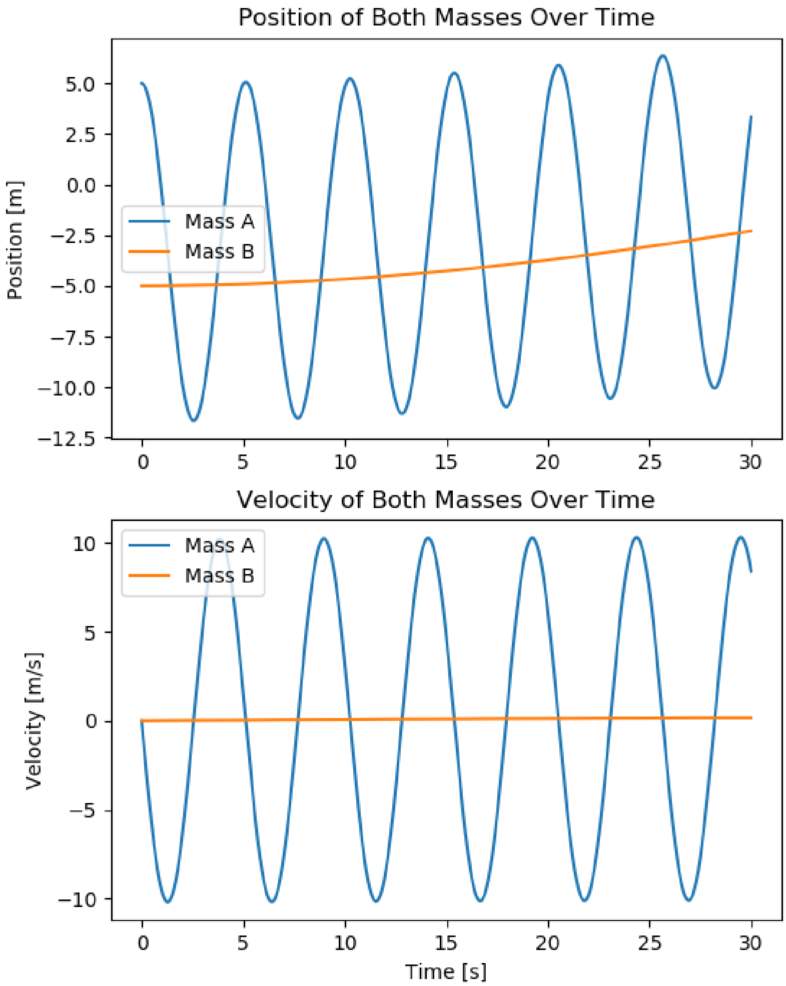


Figure 2: This has the same spring configuration as the first example, but mass B is 1000 times larger than A. In this example you can see how mass A's motion is connected to mass B. Mass B is very slowly being pulled to the right while mass A oscillates "normally" with respect to mass B.

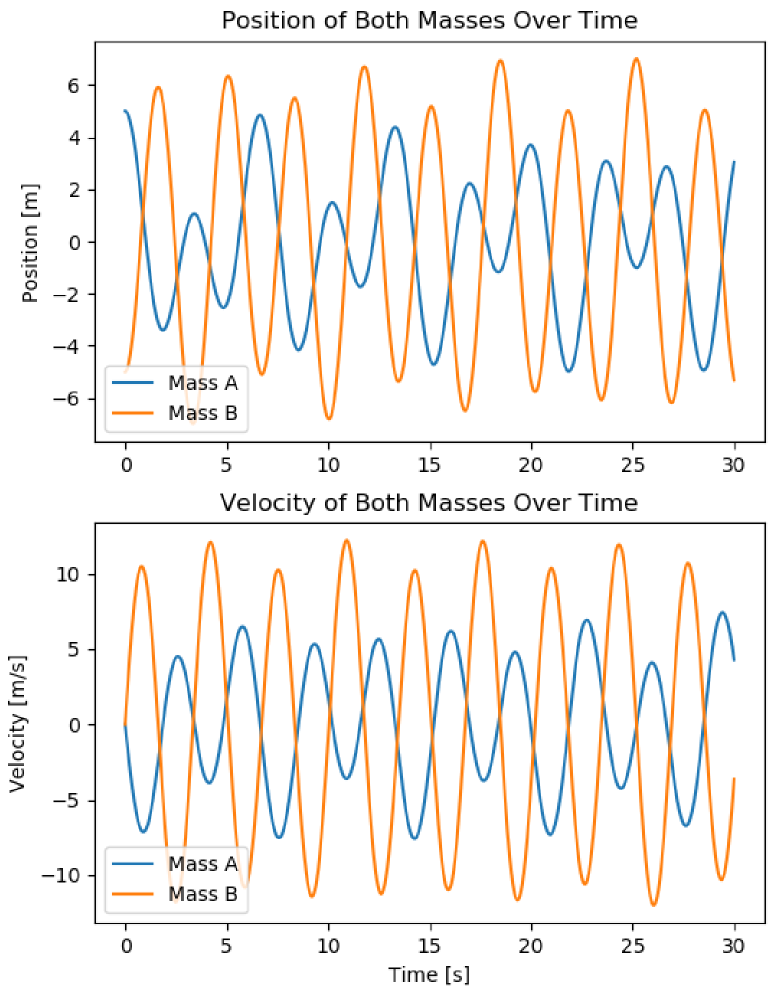


Figure 3: This example has equal masses and the spring constants double from left to right. In this case, mass B's motion is limited due to it having the stiffest spring connecting it to a wall, but mass A is very free to move with the loosest spring constant.

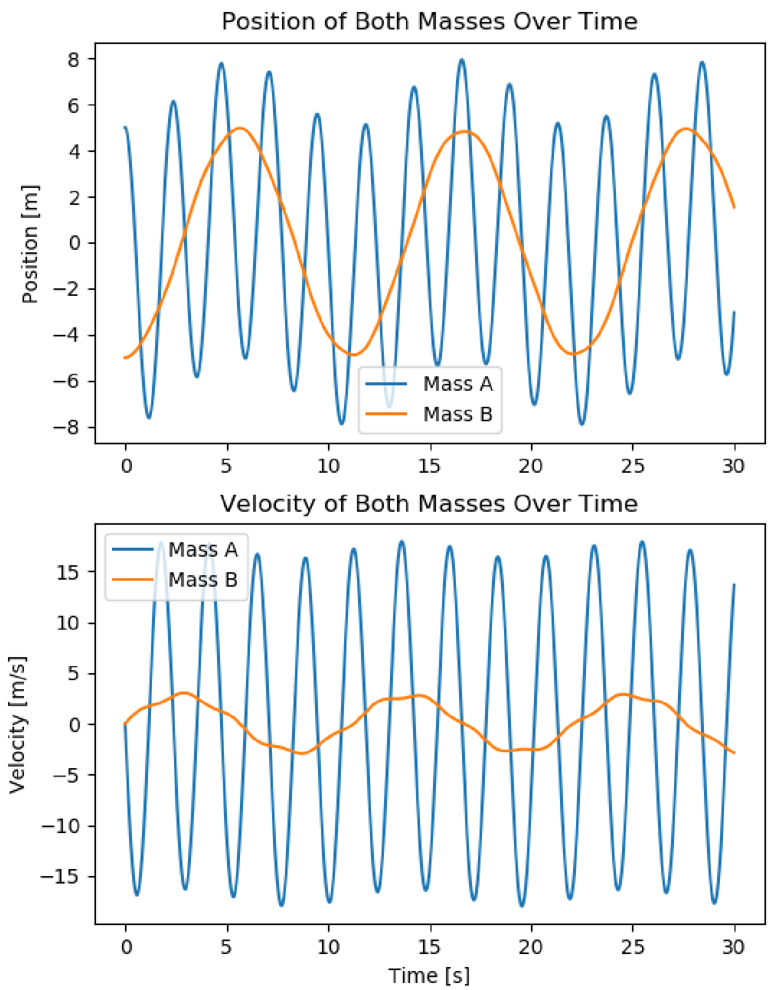


Figure 4: My random example is mass B being 20 times that of mass A, the left and right spring having the same spring constant while the middle spring is weaker. The motion in this case is very weird. The motion of mass A is heavily tied to mass B, however looking at the velocity graph you can see that mass A's motion does affect mass B altering what would be a perfectly sinusoidal shape.

```
    xa = s[0]
    va = s[1]
    xb = s[2]
    vb = s[3]
    deriv = [va,(-1*(kl*xa+km*(xa-xb))/(ma)),vb,(-1*(kr*xb+km*(xb-xa))/(mb))]
    return deriv

def DiffEq(si,t0,tf,points,deriv,params):
    ma = params[0]
    mb = params[1]
    kl = params[2]
    km = params[3]
    kr = params[4]
    t = np.linspace(t0,tf,points)
    s = odeint(deriv,si,t,args=(ma,mb,kl,km,kr))
    return t,s
#%%
#1
#Initial conditions
si = [0.,0.1,0.,0.1]
t0 = 0
tf = 30
points = 3001
params = [1,1,1,1,1]

#Getting Points
t,solns = DiffEq(si,t0,tf,points,deriv,params)
xa = solns[:,0]
va = solns[:,1]
xb = solns[:,2]
vb = solns[:,3]

#Plotting
fig1,(pos,vel) = p.subplots(2,1,figsize=(6,8))
pos.plot(t,xa,label="Mass A")
pos.plot(t,xb,label="Mass B")
pos.set_title('Position of Both Masses Over Time')
pos.legend()
pos.set_ylabel('Position [m]')

vel.plot(t,va,label="Mass A")
vel.plot(t,vb,label="Mass B")
vel.set_title('Velocity of Both Masses Over Time')
vel.legend()
```

```
vel.set_xlabel('Time [s]')
vel.set_ylabel('Velocity [m/s]')
#%%
#2
#Initial conditions
si = [0.,0.1,0.,-0.1]
t0 = 0
tf = 30
points = 3001
params = [1,1,1,1,1]

#Getting Points
t,solns = DiffEq(si,t0,tf,points,deriv,params)
xa = solns[:,0]
va = solns[:,1]
xb = solns[:,2]
vb = solns[:,3]

#Plotting
fig3,(pos,vel) = p.subplots(2,1,figsize=(6,8))
pos.plot(t,xa,label="Mass A")
pos.plot(t,xb,label="Mass B")
pos.set_title('Position of Both Masses Over Time')
pos.legend()
pos.set_ylabel('Position [m]')

vel.plot(t,va,label="Mass A")
vel.plot(t,vb,label="Mass B")
vel.set_title('Velocity of Both Masses Over Time')
vel.legend()
vel.set_xlabel('Time [s]')
vel.set_ylabel('Velocity [m/s]')
#%%
#3
#Initial conditions
si = [0.,-5,0.,5]
t0 = 0
tf = 30
points = 3001
params = [1,1,1,1,1]

#Getting Points
t,solns = DiffEq(si,t0,tf,points,deriv,params)
xa = solns[:,0]
```

```
va = solns[:,1]
xb = solns[:,2]
vb = solns[:,3]

#Plotting
fig3,(pos,vel) = p.subplots(2,1,figsize=(6,8))
pos.plot(t,xa,label="Mass A")
pos.plot(t,xb,label="Mass B")
pos.set_title('Position of Both Masses Over Time')
pos.legend()
pos.set_ylabel('Position [m]')

vel.plot(t,va,label="Mass A")
vel.plot(t,vb,label="Mass B")
vel.set_title('Velocity of Both Masses Over Time')
vel.legend()
vel.set_xlabel('Time [s]')
vel.set_ylabel('Velocity [m/s]')
#%%
#4
#Initial conditions
si = [0.,-20.,0.,0.01]
t0 = 0
tf = 30
points = 3001
params = [1,1,1,1,1]

#Getting Points
t,solns = DiffEq(si,t0,tf,points,deriv,params)
xa = solns[:,0]
va = solns[:,1]
xb = solns[:,2]
vb = solns[:,3]

#Plotting
fig3,(pos,vel) = p.subplots(2,1,figsize=(6,8))
pos.plot(t,xa,label="Mass A")
pos.plot(t,xb,label="Mass B")
pos.set_title('Position of Both Masses Over Time')
pos.legend()
pos.set_ylabel('Position [m]')

vel.plot(t,va,label="Mass A")
vel.plot(t,vb,label="Mass B")
```

```
vel.set_title('Velocity of Both Masses Over Time')
vel.legend()
vel.set_xlabel('Time [s]')
vel.set_ylabel('Velocity [m/s]')
```

Since I copied my work from the first question to do this one, these also have velocity graphs. If you keep the same proportions or just change signs you get similar results. For example the first random example has the same shape as the second given example but the masses motions are switched and go farther.

Problem 3

```
#Ryan Branagan
#Collaborators: N/A
#Branagan_hw7_p3.py
#2/27/19

import numpy as np
#%%
#Equations
#[1-cos(theta)+0]*[Fx]=[0]
#[0+sin(theta)+1]*[T]=[mg]
#[0+sin(theta)-1]*[Fy]=[0]
#%%
#Parameters
m = 14
L = 1.2
g = 9.81
theta = np.deg2rad(35)

#Matricies
A = np.matrix([[1,-np.cos(theta),0],[0,np.sin(theta),1],[0,np.sin(theta),-1]])
r = np.matrix([[0],[m*g],[0]])
v = np.linalg.solve(A,r)

#Answer
print('Fx is',v[0,0],'N')
print('T is',v[1,0],'N')
print('Fy is',v[2,0],'N')
```

I wrote the equations in the desired form in my code. My answers are approximately $F_x = 98.1N$, $T = 119.7N$, and $F_y = 68.8N$.

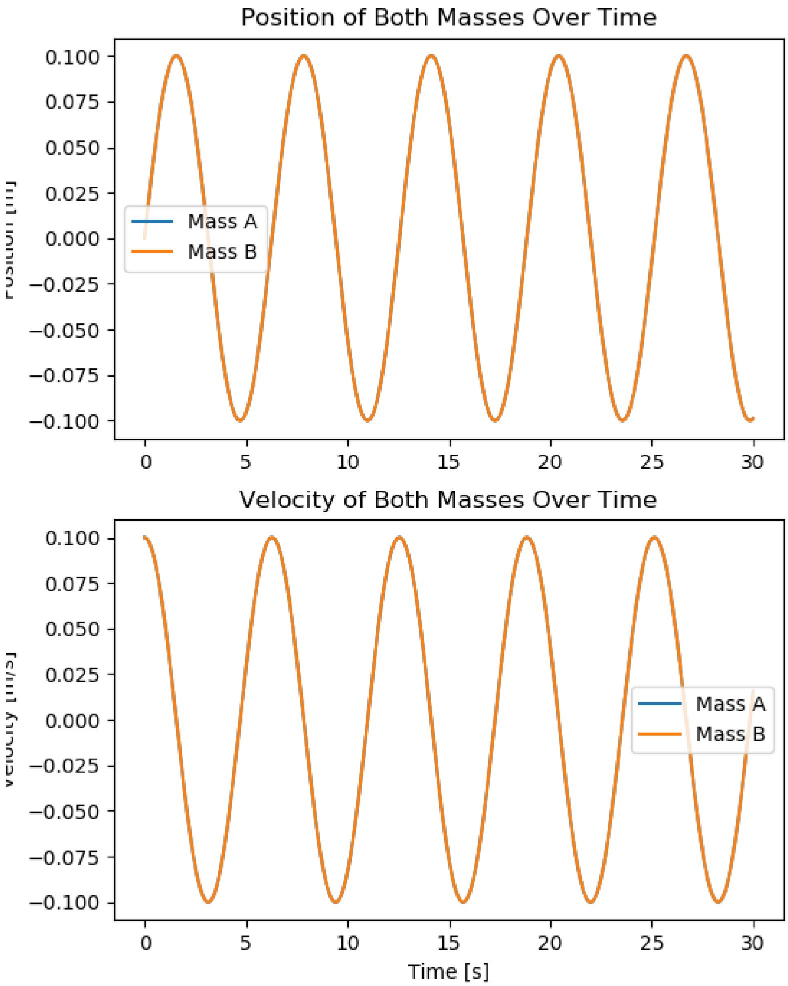


Figure 5: This is the given example with both masses starting at the same point and equal velocities.

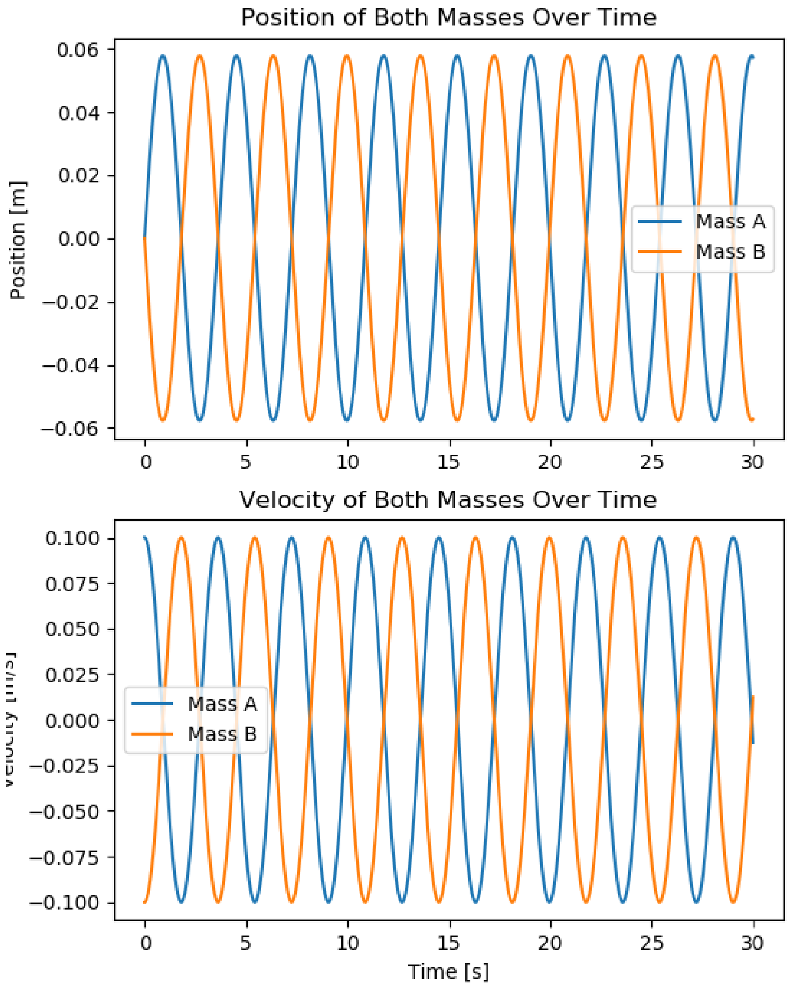


Figure 6: This is the given example with both masses starting at the same point and equal but opposite velocities.

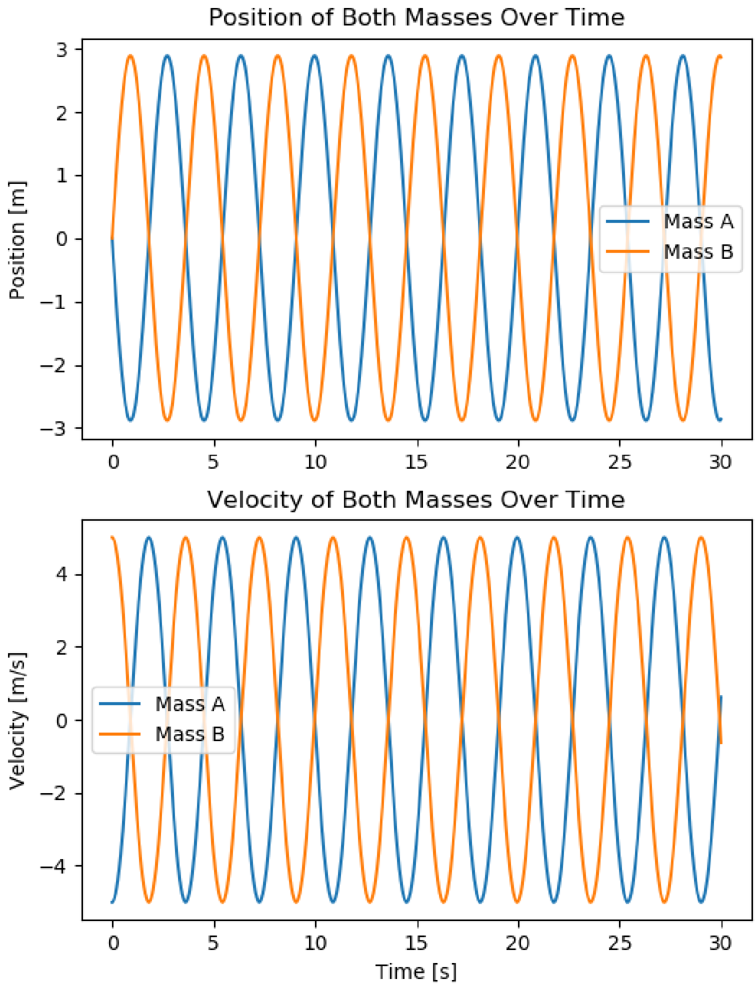


Figure 7: This is a random example with larger velocities and opposite signs of the second example still starting from the same point.

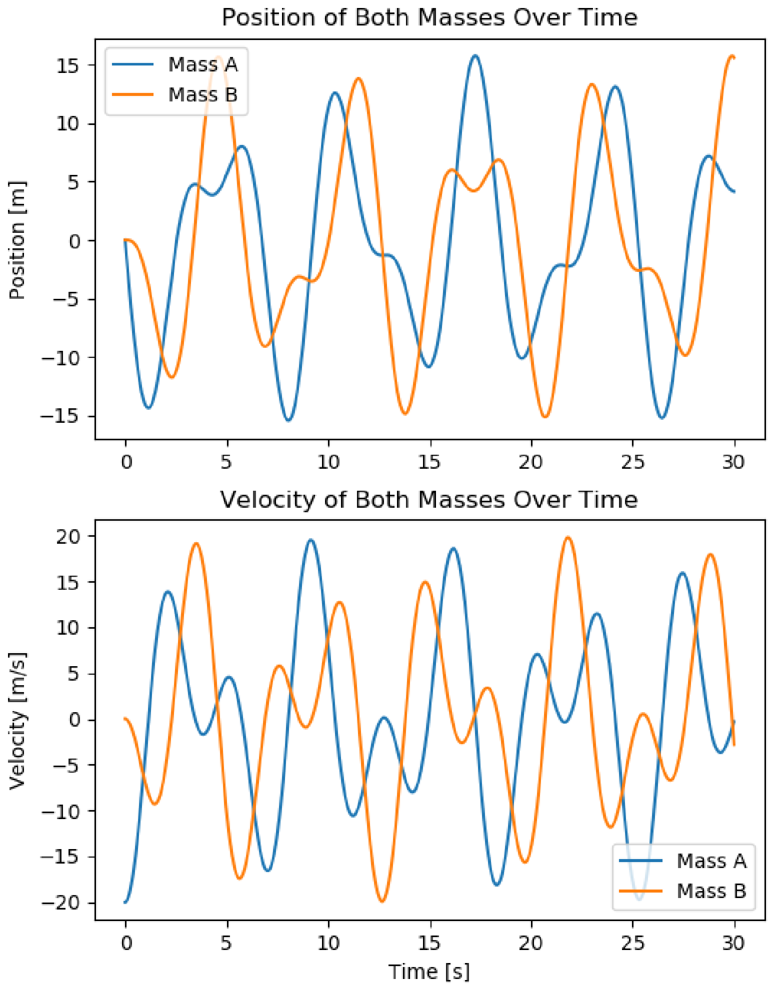


Figure 8: This is a random example with both masses starting from the same point but with mass A starting with high speed and mass B with no initial velocity.

Problem 4

```
#Ryan Branagan
#Collaborators: N/A
#Branagan_hw7_p4.py
#2/27/19

import numpy as np
#%%
#Equations
#[R1+0+R3]*[I1]=[E1]
#[0+R2+R3]*[I2]=[E2]
#[1+1-1]*[I3]=[0]
#%%
#Parameters
R1 = 100
R2 = 120
R3 = 65
E1 = 12
E2 = 9

#Matricies
A = np.matrix([[R1,0,R3],[0,R2,R3],[1,1,-1]])
r = np.matrix([[E1],[E2],[0]])
v = np.linalg.solve(A,r)

#Answer
print('I1 is',v[0,0]*1000,'mA')
print('I2 is',v[1,0]*1000,'mA')
print('I3 is',v[2,0]*1000,'mA')
```

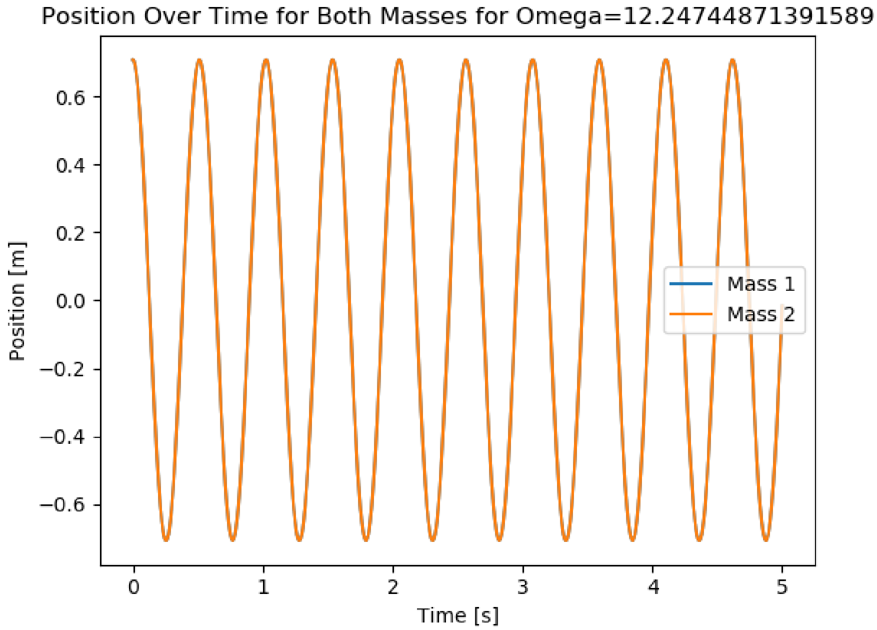
I decided to report the answers in milliamps since as expected the currents are small. My answers are approximately $I_1 = 62.2mA$, $I_2 = 26.8mA$, and $89mA$.

Problem 5

```
#Ryan Branagan
#Collaborators: N/A
#Branagan_hw7_p4.py
#2/27/19

import numpy as np
import pylab as p
#%%
```

```
#1
#Parameters
k = 15
m = 0.3
t0 = 0
tf = 5
points = 5001

#Input Matrix
M = np.matrix([[2*k,-k],[-k,2*k]])

#Solving
evals,evecs = np.linalg.eig(M)

ws = np.sqrt(evals/m)

w1 = ws[0]
C11 = evecs[0,0]
C12 = evecs[0,1]
w2 = ws[1]
C21 = evecs[1,0]
C22 = evecs[1,1]

#Equations
ts = np.linspace(t0,tf,points)
x11 = C11*np.cos(w1*ts)
x12 = C12*np.cos(w1*ts)
x21 = C21*np.cos(w2*ts)
x22 = C22*np.cos(w2*ts)

#Plottting
fig1,ax1 = p.subplots(1,1)
ax1.plot(ts,x11,label="Mass 1")
ax1.plot(ts,x12,label="Mass 2")
ax1.set_title('Position Over Time for Both Masses for Omega='+str(w1))
ax1.legend()
ax1.set_xlabel('Time [s]')
ax1.set_ylabel('Position [m]')

fig2,ax2 = p.subplots(1,1)
ax2.plot(ts,x21,label="Mass 1")
ax2.plot(ts,x22,label="Mass 2")
ax2.set_title('Position Over Time for Both Masses for Omega='+str(w2))
ax2.legend()
```

```
ax2.set_xlabel('Time [s]')
ax2.set_ylabel('Position [m]')
#%%
#2
#Answer in report
#%%
#3
#Parameters
k = 15
m = 0.3
t0 = 0
tf = 5
points = 5001

#Input Matrix
M = np.matrix([[2*k,-k],[-k,k]])

#Solving
evals,evecs = np.linalg.eig(M)

ws = np.sqrt(evals/m)

w1 = ws[0]
C11 = evecs[0,0]
C12 = evecs[0,1]
w2 = ws[1]
C21 = evecs[1,0]
C22 = evecs[1,1]

#Equations
ts = np.linspace(t0,tf,points)
x11 = C11*np.cos(w1*ts)
x12 = C12*np.cos(w1*ts)
x21 = C21*np.cos(w2*ts)
x22 = C22*np.cos(w2*ts)

#Plottting
fig3,ax3 = p.subplots(1,1)
ax3.plot(ts,x11,label="Mass 1")
ax3.plot(ts,x12,label="Mass 2")
ax3.set_title('Position Over Time for Both Masses for Omega='+str(w1))
ax3.legend()
ax3.set_xlabel('Time [s]')
ax3.set_ylabel('Position [m]')
```

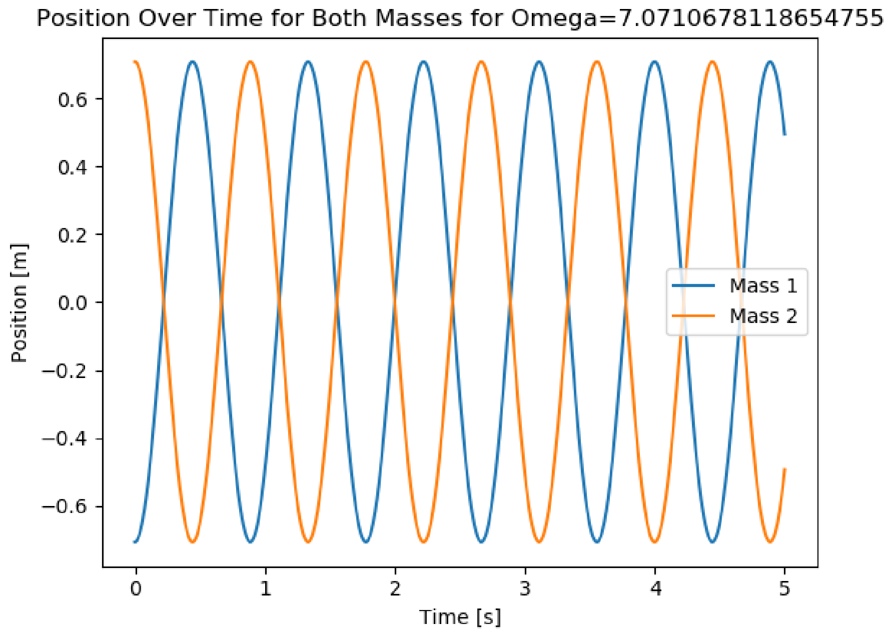


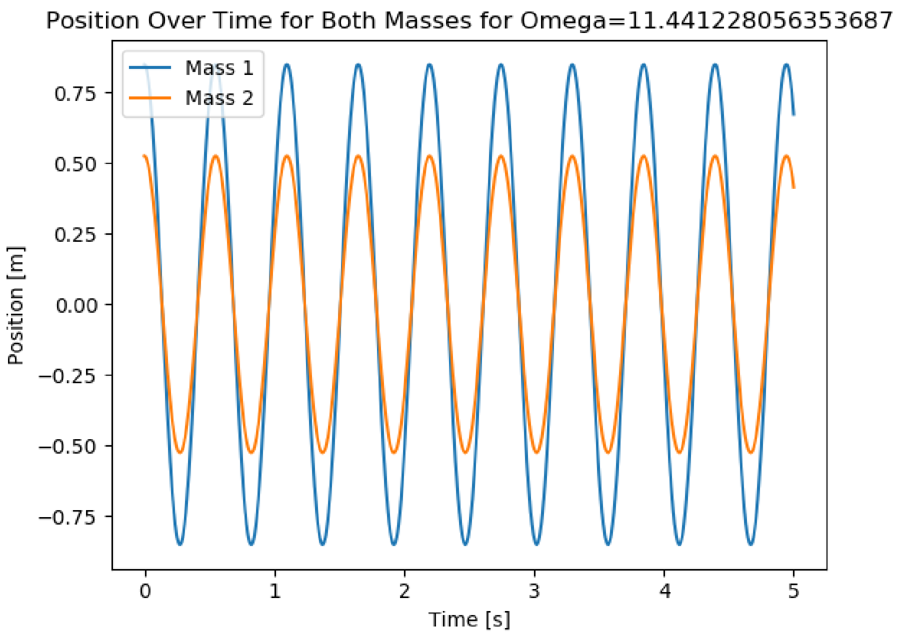

```
fig4,ax4 = p.subplots(1,1)
ax4.plot(ts,x21,label="Mass 1")
ax4.plot(ts,x22,label="Mass 2")
ax4.set_title('Position Over Time for Both Masses for Omega='+str(w2))
ax4.legend()
ax4.set_xlabel('Time [s]')
ax4.set_ylabel('Position [m]')
```

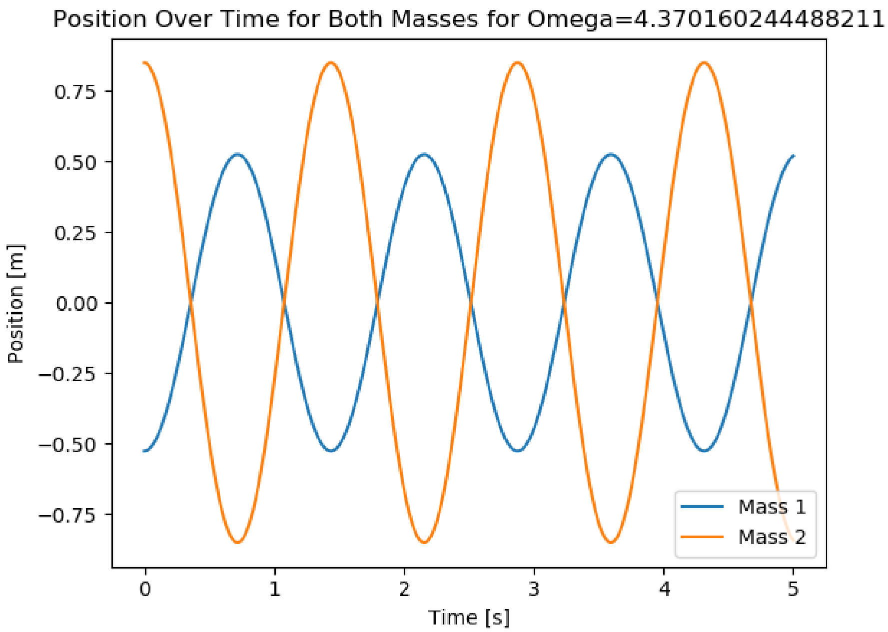
My obtained equations are $0.707cos(12.25t)$, $0.707cos(12.25t)$, $-0.707cos(7.07t)$, and $0.707cos(7.07t)$. The first normal mode has both masses moving together while the second normal mode has them moving perfectly opposite each other.

These normal modes correspond to the given examples from Problem 2 where both masses are starting from the same position and either have the same velocity or same speed but opposite direction.

The normal modes look very similar except that in each case one of the masses do not oscillate with the same amplitude as the other.


Position Over Time for Both Masses for Omega=7.0710678118654755
Position [m]
Time [s]
Mass 1
Mass 2


Position Over Time for Both Masses for Omega=11.441228056353687
Position [m]
Time [s]
Mass 1
Mass 2


Position Over Time for Both Masses for Omega=4.370160244488211
Position [m]
Time [s]
Mass 1
Mass 2
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
0
1
2
3
4
5