Exam 2

Ryan Branagan

March 26, 2019

Problem 1

```
#Ryan Branagan
#Collaborators: Mason Lovejoy-Johnson, Steven Boswell, Jack Featherstone, and Grant Sher
#Branagan_ex2_p1.py
#3/21/19

import numpy as np
import pylab as p
from scipy.integrate import odeint
#%%
def deriv(s,t,ma,mb,kl,km,kr,B,w):
    xa = s[0]
    va = s[1]
    xb = s[2]
    vb = s[3]
    deriv = [va,((-1*(kl*xa+km*(xa-xb))-B*va+(0.1*np.sin(w*t)))/(ma)),vb,((-1*(kr*xb+km*
    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]
    B = params[5]
    w = params[6]
    t = np.linspace(t0,tf,points)
    s = odeint(deriv,si,t,args=(ma,mb,kl,km,kr,B,w))
    return t,s
#%%
#1
```

```
#Initial conditions
si = [0.,0.,0.,0.]
t0 = 0
tf = 200
points = 2001

#Getting Points
ws = np.linspace(0,2,21)
for i,w in enumerate(ws):
    params = [1,1,1,1,1,0.1,w]
    t,solns = DiffEq(si,t0,tf,points,deriv,params)
    xa = solns[:,0]
    va = solns[:,1]
    xb = solns[:,2]
    vb = solns[:,3]

    #Plotting
    i,pos = p.subplots(1,1)
    pos.plot(t,xa,"r-",label="Mass A")
    pos.plot(t,xb,"b-",label="Mass B")
    pos.set_ylim([-0.55,0.55])
    pos.set_title('Position of Both Masses Over Time')
    pos.legend()
    pos.set_xlabel('Time [s]')
    pos.set_ylabel('Position [m]')
    pos.annotate('Omega='+str(round(w,2)),xy=(10,10),xycoords='figure pixels')
```

The system has the largest response at ω=1 [rad/s]. I would guess that this is the system's resonance frequency. More generally, I think that this frequency matches the frequency of oscilation for the sytem so that the driving force changes direction at the same time as the spring force.

A quick note, this code produces 21 graphs corresponding to frequency values going from 0 to 2 in steps of 0.1. The twenty-first graph for ω=2[rads/s] has been omitted.

Problem 2

```
#Ryan Branagan
#Collaborators: Mason Lovejoy-Johnson, Steven Boswell, Jack Featherstone, and Grant Sherr
#Branagan_ex2_p2.py
#3/21/19

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

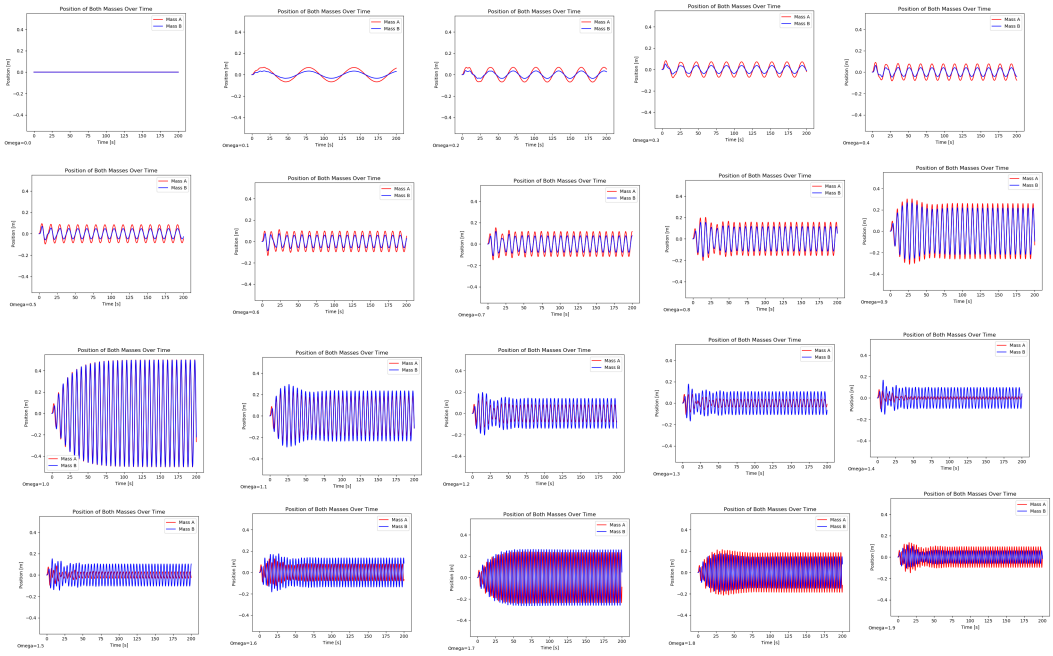


Figure 1: I used Microsoft Paint to put all the graphs together. I didn't have enough time to use subplots to do this in Python nor to do it well in Microsoft Paint.

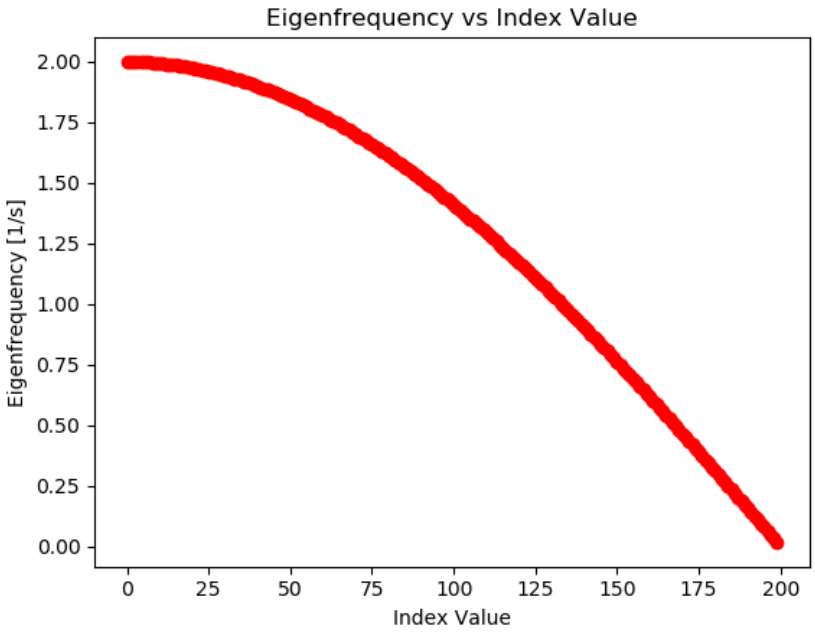
```
#Parameters
N = 200
k = 1 #[N/m]
m = 1 #[kg]

#Creating an array to do Linear Algebra on
One = np.ones(N)
One2 = np.ones(N-1)
A = np.diagflat((-2*k*One))
B = np.diagflat((k*One2),k=1)
C = np.diagflat((k*One2),k=-1)
Springs = A+B+C

#Doing Linear Algebra
EigV,EigVec = np.linalg.eigh(Springs)

#Finding the eigenfrequencies
w = np.sqrt(-1*EigV/m)

#Eigen Vectors
Mode1 = EigVec[:,0]
```

```
Mode2 = EigVec[:,1]
Mode99 = EigVec[:,98]
Mode100 = EigVec[:,99]
Mode199 = EigVec[:,198]
Mode200 = EigVec[:,199]


#Plotting
Index = np.arange(0,N,1)
fig1,freq = p.subplots(1,1)
freq.plot(Index,w,"ro")
freq.set_title("Eigenfrequency vs Index Value")
freq.set_xlabel("Index Value")
freq.set_ylabel("Eigenfrequency [1/s]")

fig2,((M1,M2),(M99,M100),(M199,M200)) = p.subplots(3,2,figsize=(12,10))
M1.plot(Mode1,"r-")
M1.set_title("M1")
M2.plot(Mode2,"o-")
M2.set_title("M2")
M99.plot(Mode99,"y-")
M99.set_title("M99")
M100.plot(Mode100,"g-")
M100.set_title("M100")
M199.plot(Mode199,"b-")
M199.set_title("M199")
M200.plot(Mode200,"m-")
M200.set_title("M200")

fig3,Art = p.subplots(1,1)
Art.plot(EigVec)
```

If this system could move in the x-y plane then it might be a good approximation of a standing wave on a string.

Problem 3

```
#Ryan Branagan
#Collaborators: Mason Lovejoy-Johnson, Steven Boswell, Jack Featherstone, and Grant Sherr
#Branagan_ex2_p3.py
#3/21/19

import numpy as np
import pylab as p
```




```
#%%
#Import Data
#Sim1
#Time 1
S1T1 = np.loadtxt("C:\\Users\\ryan-\\Documents\\251\\Exam 2\\Data\\Simdata-1\\t1.dat")
FID11 = S1T1[:,0]
X11 = S1T1[:,1]
StartPX11 = X11[0::16]
EndPX11 = X11[15::16]
Y11 = S1T1[:,2]
StartPY11 = Y11[0::16]
EndPY11 = Y11[15::16]
#Time 2
S1T2 = np.loadtxt("C:\\Users\\ryan-\\Documents\\251\\Exam 2\\Data\\Simdata-1\\t2.dat")
FID12 = S1T2[:,0]
X12 = S1T2[:,1]
StartPX12 = X12[0::16]
EndPX12 = X12[15::16]
Y12 = S1T2[:,2]
StartPY12 = Y12[0::16]
EndPY12 = Y12[15::16]
#Time 3
S1T3 = np.loadtxt("C:\\Users\\ryan-\\Documents\\251\\Exam 2\\Data\\Simdata-1\\t3.dat")
```

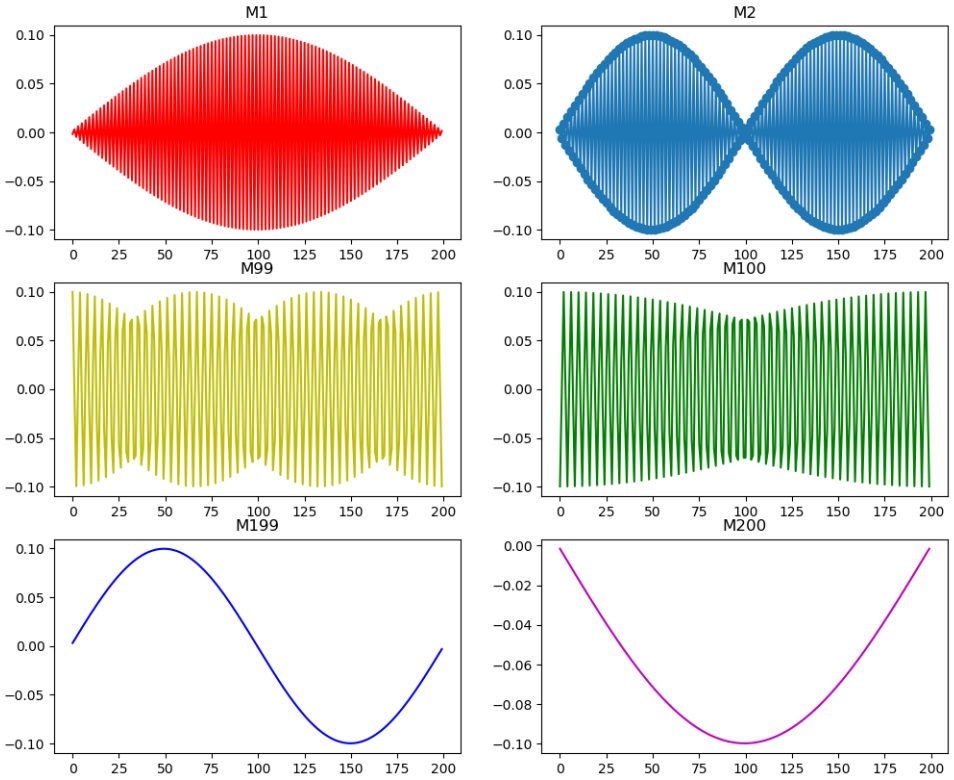


Figure 2: This graph corresponds to the normal modes of the system, specifically the fist, second, ninety-ninth, hundredth, one-hundred and ninety-ninth, and two-hundredth normal modes of the system. It did not make sense to me to give the graphs x and y labels since we are not plotting x and y values.

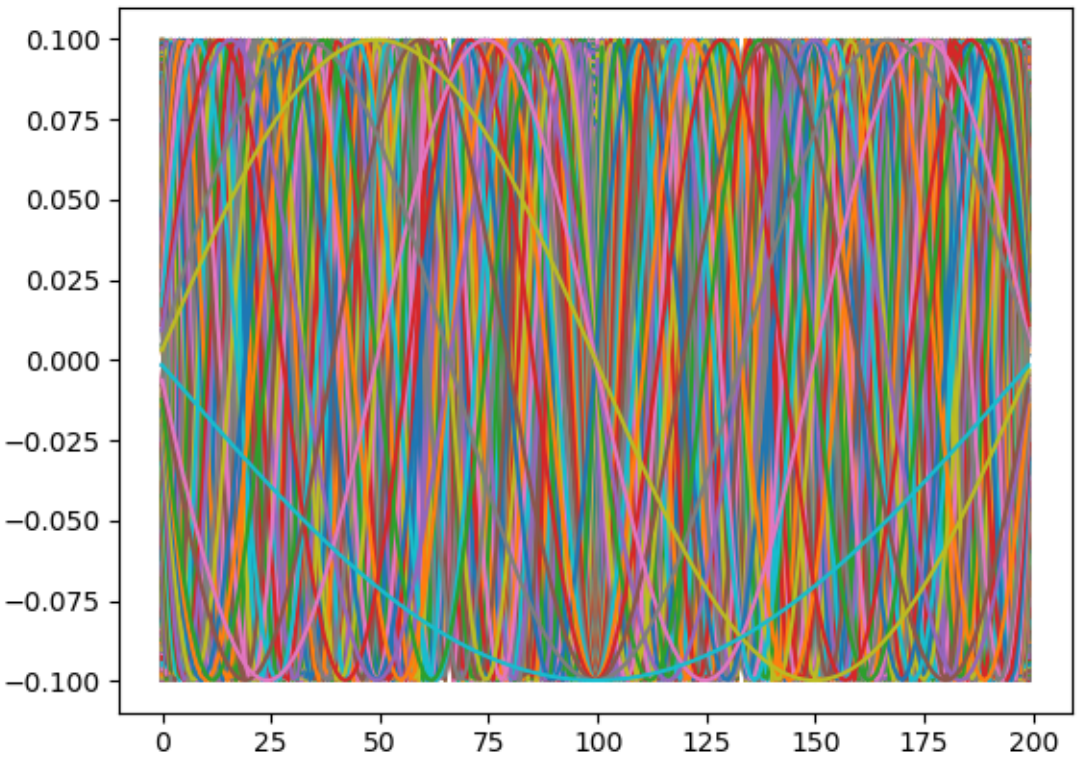


Figure 3: This graph is nonsense but I think it looks like modern art and it makes me happy so I'm including it in this report.

```
FID13 = S1T3[:,0]
X13 = S1T3[:,1]
StartPX13 = X13[0::16]
EndPX13 = X13[15::16]
Y13 = S1T3[:,2]
StartPY13 = Y13[0::16]
EndPY13 = Y13[15::16]
#Time 4
S1T4 = np.loadtxt("C:\\Users\\ryan-\\Documents\\251\\Exam 2\\Data\\Simdata-1\\t4.dat")
FID14 = S1T4[:,0]
X14 = S1T4[:,1]
StartPX14 = X14[0::16]
EndPX14 = X14[15::16]
Y14 = S1T4[:,2]
StartPY14 = Y14[0::16]
EndPY14 = Y14[15::16]
#Time 5
S1T5 = np.loadtxt("C:\\Users\\ryan-\\Documents\\251\\Exam 2\\Data\\Simdata-1\\t5.dat")
FID15 = S1T5[:,0]
X15 = S1T5[:,1]
StartPX15 = X15[0::16]
EndPX15 = X15[15::16]
Y15 = S1T5[:,2]
StartPY15 = Y15[0::16]
EndPY15 = Y15[15::16]

#Sim2
#Time 1
S2T1 = np.loadtxt("C:\\Users\\ryan-\\Documents\\251\\Exam 2\\Data\\Simdata-2\\t1.dat")
FID21 = S2T1[:,0]
X21 = S2T1[:,1]
StartPX21 = X21[0::16]
EndPX21 = X21[15::16]
Y21 = S2T1[:,2]
StartPY21 = Y21[0::16]
EndPY21 = Y21[15::16]
#Time 2
S2T2 = np.loadtxt("C:\\Users\\ryan-\\Documents\\251\\Exam 2\\Data\\Simdata-2\\t2.dat")
FID22 = S2T2[:,0]
X22 = S2T2[:,1]
StartPX22 = X22[0::16]
EndPX22 = X22[15::16]
Y22 = S2T2[:,2]
StartPY22 = Y22[0::16]
```

```
EndPY22 = Y22[15::16]
#Time 3
S2T3 = np.loadtxt("C:\\Users\\ryan-\\Documents\\251\\Exam 2\\Data\\Simdata-2\\t3.dat")
FID23 = S2T3[:,0]
X23 = S2T3[:,1]
StartPX23 = X23[0::16]
EndPX23 = X23[15::16]
Y23 = S2T3[:,2]
StartPY23 = Y23[0::16]
EndPY23 = Y23[15::16]
#Time 4
S2T4 = np.loadtxt("C:\\Users\\ryan-\\Documents\\251\\Exam 2\\Data\\Simdata-2\\t4.dat")
FID24 = S2T4[:,0]
X24 = S2T4[:,1]
StartPX24 = X24[0::16]
EndPX24 = X24[15::16]
Y24 = S2T4[:,2]
StartPY24 = Y24[0::16]
EndPY24 = Y24[15::16]
#Time 5
S2T5 = np.loadtxt("C:\\Users\\ryan-\\Documents\\251\\Exam 2\\Data\\Simdata-2\\t5.dat")
FID25 = S2T5[:,0]
X25 = S2T5[:,1]
StartPX25 = X25[0::16]
EndPX25 = X25[15::16]
Y25 = S2T5[:,2]
StartPY25 = Y25[0::16]
EndPY25 = Y25[15::16]

#Sim3
#Time 1
S3T1 = np.loadtxt("C:\\Users\\ryan-\\Documents\\251\\Exam 2\\Data\\Simdata-3\\t1.dat")
FID31 = S3T1[:,0]
X31 = S3T1[:,1]
StartPX31 = X31[0::16]
EndPX31 = X31[15::16]
Y31 = S3T1[:,2]
StartPY31 = Y31[0::16]
EndPY31 = Y31[15::16]
#Time 2
S3T2 = np.loadtxt("C:\\Users\\ryan-\\Documents\\251\\Exam 2\\Data\\Simdata-3\\t2.dat")
FID32 = S3T2[:,0]
X32 = S3T2[:,1]
StartPX32 = X32[0::16]
```

```
EndPX32 = X32[15::16]
Y32 = S3T2[:,2]
StartPY32 = Y32[0::16]
EndPY32 = Y32[15::16]
#Time 3
S3T3 = np.loadtxt("C:\\Users\\ryan-\\Documents\\251\\Exam 2\\Data\\Simdata-3\\t3.dat")
FID33 = S3T3[:,0]
X33 = S3T3[:,1]
StartPX33 = X33[0::16]
EndPX33 = X33[15::16]
Y33 = S3T3[:,2]
StartPY33 = Y33[0::16]
EndPY33 = Y33[15::16]
#Time 4
S3T4 = np.loadtxt("C:\\Users\\ryan-\\Documents\\251\\Exam 2\\Data\\Simdata-3\\t4.dat")
FID34 = S3T4[:,0]
X34 = S3T4[:,1]
StartPX34 = X34[0::16]
EndPX34 = X34[15::16]
Y34 = S3T4[:,2]
StartPY34 = Y34[0::16]
EndPY34 = Y34[15::16]
#Time 5
S3T5 = np.loadtxt("C:\\Users\\ryan-\\Documents\\251\\Exam 2\\Data\\Simdata-3\\t5.dat")
FID35 = S3T5[:,0]
X35 = S3T5[:,1]
StartPX35 = X35[0::16]
EndPX35 = X35[15::16]
Y35 = S3T5[:,2]
StartPY35 = Y35[0::16]
EndPY35 = Y35[15::16]
#%%
#Fixing Boundaries
S1X = [X11,X12,X13,X14,X15]
S2X = [X21,X22,X23,X24,X25]
S3X = [X31,X32,X33,X34,X35]
SuperX = [S1X,S2X,S3X]
for Sim in SuperX:
    for Time in Sim:
        for x in Time:
            if x > 5.:
                x = x - 10
            if x < -5.:
                x = x + 10
```

```
S1Y = [Y11,Y12,Y13,Y14,Y15]
S2Y = [Y21,Y22,Y23,Y24,Y25]
S3Y = [Y31,Y32,Y33,Y34,Y35]
SuperY = [S1Y,S2Y,S3Y]
for Sim in SuperY:
    for Time in Sim:
        for y in Time:
            if y > 5.:
                y = y - 10
            if y < -5.:
                y = y + 10
#%%
#Calculating Relevant Values
#Finding End to End Distance
def Dist(X1,X2,Y1,Y2):
    return np.sqrt(((X2-X1)**2)+((Y2-Y1)**2))

#End to End Distance: Sim-1
EndD11 = Dist(StartPX11,EndPX11,StartPY11,EndPY11)
EndD12 = Dist(StartPX12,EndPX12,StartPY12,EndPY12)
EndD13 = Dist(StartPX13,EndPX13,StartPY13,EndPY13)
EndD14 = Dist(StartPX14,EndPX14,StartPY14,EndPY14)
EndD15 = Dist(StartPX15,EndPX15,StartPY15,EndPY15)

#End to End Distance: Sim-2
EndD21 = Dist(StartPX21,EndPX21,StartPY21,EndPY21)
EndD22 = Dist(StartPX22,EndPX22,StartPY22,EndPY22)
EndD23 = Dist(StartPX23,EndPX23,StartPY23,EndPY23)
EndD24 = Dist(StartPX24,EndPX24,StartPY24,EndPY24)
EndD25 = Dist(StartPX25,EndPX25,StartPY25,EndPY25)

#End to End Distance: Sim-3
EndD31 = Dist(StartPX31,EndPX31,StartPY31,EndPY31)
EndD32 = Dist(StartPX32,EndPX32,StartPY32,EndPY32)
EndD33 = Dist(StartPX33,EndPX33,StartPY33,EndPY33)
EndD34 = Dist(StartPX34,EndPX34,StartPY34,EndPY34)
EndD35 = Dist(StartPX35,EndPX35,StartPY35,EndPY35)

#Compute Average of end to end distance
def Average(x):
    return ((np.sum(x))/(len(x)))

AvEED11 = Average(EndD11)
```

```
AvEED12 = Average(EndD12)
AvEED13 = Average(EndD13)
AvEED14 = Average(EndD14)
AvEED15 = Average(EndD15)
AvEnd1 = [AvEED11,AvEED12,AvEED13,AvEED14,AvEED15]

AvEED21 = Average(EndD21)
AvEED22 = Average(EndD22)
AvEED23 = Average(EndD23)
AvEED24 = Average(EndD24)
AvEED25 = Average(EndD25)
AvEnd2 = [AvEED21,AvEED22,AvEED23,AvEED24,AvEED25]

AvEED31 = Average(EndD31)
AvEED32 = Average(EndD32)
AvEED33 = Average(EndD33)
AvEED34 = Average(EndD34)
AvEED35 = Average(EndD35)
AvEnd3 = [AvEED31,AvEED32,AvEED33,AvEED34,AvEED35]

#Finding Angles
def Angle(X1,X2,Y1,Y2):
    return np.arctan2((Y2-Y1),(X2-X1))

#Filament Angle Sim-1
Ang11 = Angle(StartPX11,EndPX11,StartPY11,EndPY11)
Ang12 = Angle(StartPX12,EndPX12,StartPY12,EndPY12)
Ang13 = Angle(StartPX13,EndPX13,StartPY13,EndPY13)
Ang14 = Angle(StartPX14,EndPX14,StartPY14,EndPY14)
Ang15 = Angle(StartPX15,EndPX15,StartPY15,EndPY15)

#Filament Angle Sim-2
Ang21 = Angle(StartPX21,EndPX21,StartPY21,EndPY21)
Ang22 = Angle(StartPX22,EndPX22,StartPY22,EndPY22)
Ang23 = Angle(StartPX23,EndPX23,StartPY23,EndPY23)
Ang24 = Angle(StartPX24,EndPX24,StartPY24,EndPY24)
Ang25 = Angle(StartPX25,EndPX25,StartPY25,EndPY25)

#Filament Angle Sim-3
Ang31 = Angle(StartPX31,EndPX31,StartPY31,EndPY31)
Ang32 = Angle(StartPX32,EndPX32,StartPY32,EndPY32)
Ang33 = Angle(StartPX33,EndPX33,StartPY33,EndPY33)
Ang34 = Angle(StartPX34,EndPX34,StartPY34,EndPY34)
Ang35 = Angle(StartPX35,EndPX35,StartPY35,EndPY35)
```

```
#Standard Deviation
AngDe11 = np.std(Ang11)
AngDe12 = np.std(Ang12)
AngDe13 = np.std(Ang13)
AngDe14 = np.std(Ang14)
AngDe15 = np.std(Ang15)
Ang1 = [AngDe11,AngDe12,AngDe13,AngDe14,AngDe15]

AngDe21 = np.std(Ang21)
AngDe22 = np.std(Ang22)
AngDe23 = np.std(Ang23)
AngDe24 = np.std(Ang24)
AngDe25 = np.std(Ang25)
Ang2 = [AngDe21,AngDe22,AngDe23,AngDe24,AngDe25]

AngDe31 = np.std(Ang31)
AngDe32 = np.std(Ang32)
AngDe33 = np.std(Ang33)
AngDe34 = np.std(Ang34)
AngDe35 = np.std(Ang35)
Ang3 = [AngDe31,AngDe32,AngDe33,AngDe34,AngDe35]

#Average of an average of an approximation
#Define a function to separate the filament coordinates
def GetCoord(Xarr,Yarr):
    a = [np.zeros(16) for i in range(300)]
    b = [np.zeros(16) for i in range(300)]
    for i in range(300):
        a[i] = Xarr[(16*i):(16*(i+1))]
        b[i] = Yarr[(16*i):(16*(i+1))]
    return a,b

#Separating filaments
FilCoord11 = GetCoord(X11,Y11)
FilCoord12 = GetCoord(X12,Y12)
FilCoord13 = GetCoord(X13,Y13)
FilCoord14 = GetCoord(X14,Y14)
FilCoord15 = GetCoord(X15,Y15)

FilCoord21 = GetCoord(X21,Y21)
FilCoord22 = GetCoord(X22,Y22)
FilCoord23 = GetCoord(X23,Y23)
FilCoord24 = GetCoord(X24,Y24)
```

```
FilCoord25 = GetCoord(X25,Y25)

FilCoord31 = GetCoord(X31,Y31)
FilCoord32 = GetCoord(X32,Y32)
FilCoord33 = GetCoord(X33,Y33)
FilCoord34 = GetCoord(X34,Y34)
FilCoord35 = GetCoord(X35,Y35)

#Define a function to get the average average
def Average2(Coords):
    X = Coords[0]
    Y = Coords[1]
    Z = np.zeros(len(X))
    for i in range(len(X)):
        Avg = ((np.sum(np.sqrt((X[i]**2)+(Y[i]**2))))/(len(X[i])))
        Z[i] = Avg
    FinAvg = ((np.sum(Z))/len(Z))
    return FinAvg

#Get the Radius of Gyration for each time
RG11 = Average2(FilCoord11)
RG12 = Average2(FilCoord12)
RG13 = Average2(FilCoord13)
RG14 = Average2(FilCoord14)
RG15 = Average2(FilCoord15)
RG1 = [RG11,RG12,RG13,RG14,RG15]

RG21 = Average2(FilCoord11)
RG22 = Average2(FilCoord22)
RG23 = Average2(FilCoord23)
RG24 = Average2(FilCoord24)
RG25 = Average2(FilCoord25)
RG2 = [RG21,RG22,RG23,RG24,RG25]

RG31 = Average2(FilCoord31)
RG32 = Average2(FilCoord32)
RG33 = Average2(FilCoord33)
RG34 = Average2(FilCoord34)
RG35 = Average2(FilCoord35)
RG3 = [RG31,RG32,RG33,RG34,RG35]
#%%
#Plotting
Time = [1,2,3,4,5]
Part1, ((S1T1,S1T2,S1T3,S1T4,S1T5),(S2T1,S2T2,S2T3,S2T4,S2T5),(S3T1,S3T2,S3T3,S3T4,S3T5)
```

```
Part1.suptitle("Distribution of End to End Distance for Each Sim and Time")
S1T1.set_title("Sim-1 Time 1")
S1T1.hist(EndD11)
S1T2.set_title("Sim-1 Time 2")
S1T2.hist(EndD12)
S1T3.set_title("Sim-1 Time 3")
S1T3.hist(EndD13)
S1T4.set_title("Sim-1 Time 4")
S1T4.hist(EndD14)
S1T5.set_title("Sim-1 Time 5")
S1T5.hist(EndD15)

S2T1.set_title("Sim-2 Time 1")
S2T1.hist(EndD21)
S2T1.set_ylabel("Distribution")
S2T2.set_title("Sim-2 Time 2")
S2T2.hist(EndD22)
S2T3.set_title("Sim-2 Time 3")
S2T3.hist(EndD23)
S2T4.set_title("Sim-2 Time 4")
S2T4.hist(EndD24)
S2T5.set_title("Sim-2 Time 5")
S2T5.hist(EndD25)

S3T1.set_title("Sim-3 Time 1")
S3T1.hist(EndD31)
S3T2.set_title("Sim-3 Time 2")
S3T2.hist(EndD32)
S3T3.set_title("Sim-3 Time 3")
S3T3.hist(EndD33)
S3T3.set_xlabel("End to End Distance")
S3T4.set_title("Sim-3 Time 4")
S3T4.hist(EndD34)
S3T5.set_title("Sim-3 Time 5")
S3T5.hist(EndD35)

Part12, (End1,End2,End3) = p.subplots(1,3,figsize=(15,5))
Part12.suptitle("Average End to End Distance Over Time for Each Sim")
End1.plot(Time,AvEnd1,"o-")
End1.set_title("Sim-1")
End2.plot(Time,AvEnd2,"o-")
End2.set_title("Sim-2")
End2.set_xlabel("Time")
End2.set_ylabel("Average End to End Distance")
```

```
End3.plot(Time,AvEnd3,"o-")
End3.set_title("Sim-3")

Part2, ((S1T1,S1T2,S1T3,S1T4,S1T5),(S2T1,S2T2,S2T3,S2T4,S2T5),(S3T1,S3T2,S3T3,S3T4,S3T5)
Part2.suptitle("Distribution of Angle for Each Sim and Time")
S1T1.set_title("Sim-1 Time 1")
S1T1.hist(Ang11)
S1T2.set_title("Sim-1 Time 2")
S1T2.hist(Ang12)
S1T3.set_title("Sim-1 Time 3")
S1T3.hist(Ang13)
S1T4.set_title("Sim-1 Time 4")
S1T4.hist(Ang14)
S1T5.set_title("Sim-1 Time 5")
S1T5.hist(Ang15)

S2T1.set_title("Sim-2 Time 1")
S2T1.hist(Ang21)
S2T1.set_ylabel("Distribution")
S2T2.set_title("Sim-2 Time 2")
S2T2.hist(Ang22)
S2T3.set_title("Sim-2 Time 3")
S2T3.hist(Ang23)
S2T4.set_title("Sim-2 Time 4")
S2T4.hist(Ang24)
S2T5.set_title("Sim-2 Time 5")
S2T5.hist(Ang25)

S3T1.set_title("Sim-3 Time 1")
S3T1.hist(Ang31)
S3T2.set_title("Sim-3 Time 2")
S3T2.hist(Ang32)
S3T3.set_title("Sim-3 Time 3")
S3T3.hist(Ang33)
S3T3.set_xlabel("Distribution of Angle")
S3T4.set_title("Sim-3 Time 4")
S3T4.hist(Ang34)
S3T5.set_title("Sim-3 Time 5")
S3T5.hist(Ang35)

Part22, (AngA1,AngA2,AngA3) = p.subplots(1,3,figsize=(15,5))
Part22.suptitle("Standard Deviation of Angle Over Time for Each Sim")
AngA1.plot(Time,Ang1,"o-")
AngA1.set_title("Sim-1")
```

```
AngA2.plot(Time,Ang2,"o-")
AngA2.set_title("Sim-2")
AngA2.set_xlabel("Time")
AngA2.set_ylabel("Standard Deviation of Angle")
AngA3.plot(Time,Ang3,"o-")
AngA3.set_title("Sim-3")

Part3,(RGS1,RGS2,RGS3) = p.subplots(1,3,figsize=(15,5))
Part3.suptitle("Radius of Gyration Over Time for Each Sim")
RGS1.plot(Time,RG1,"o-")
RGS1.set_title("Sim-1")
RGS2.plot(Time,RG2,"o-")
RGS2.set_title("Sim-2")
RGS2.set_xlabel("Time")
RGS2.set_ylabel("Radius of Gyration")
RGS3.plot(Time,RG3,"o-")
RGS2.set_title("Sim-3")
```

This question was extremely long and tedious. I would not be surprised if I accidentally put a wrong number and messed up somewhere. Although I did things sloppily since it has gotten late, I think the Python and coding part of it is correct. This problem was very interesting in the fact that we got to work with actual data, but if I did this again I would try my best to make it in a way where I didn't have to copy and paste something 15 times then change a few numbers. Overall, this question makes me appreciate the modern art from Problem 2 even more and makes me want to go look at it.

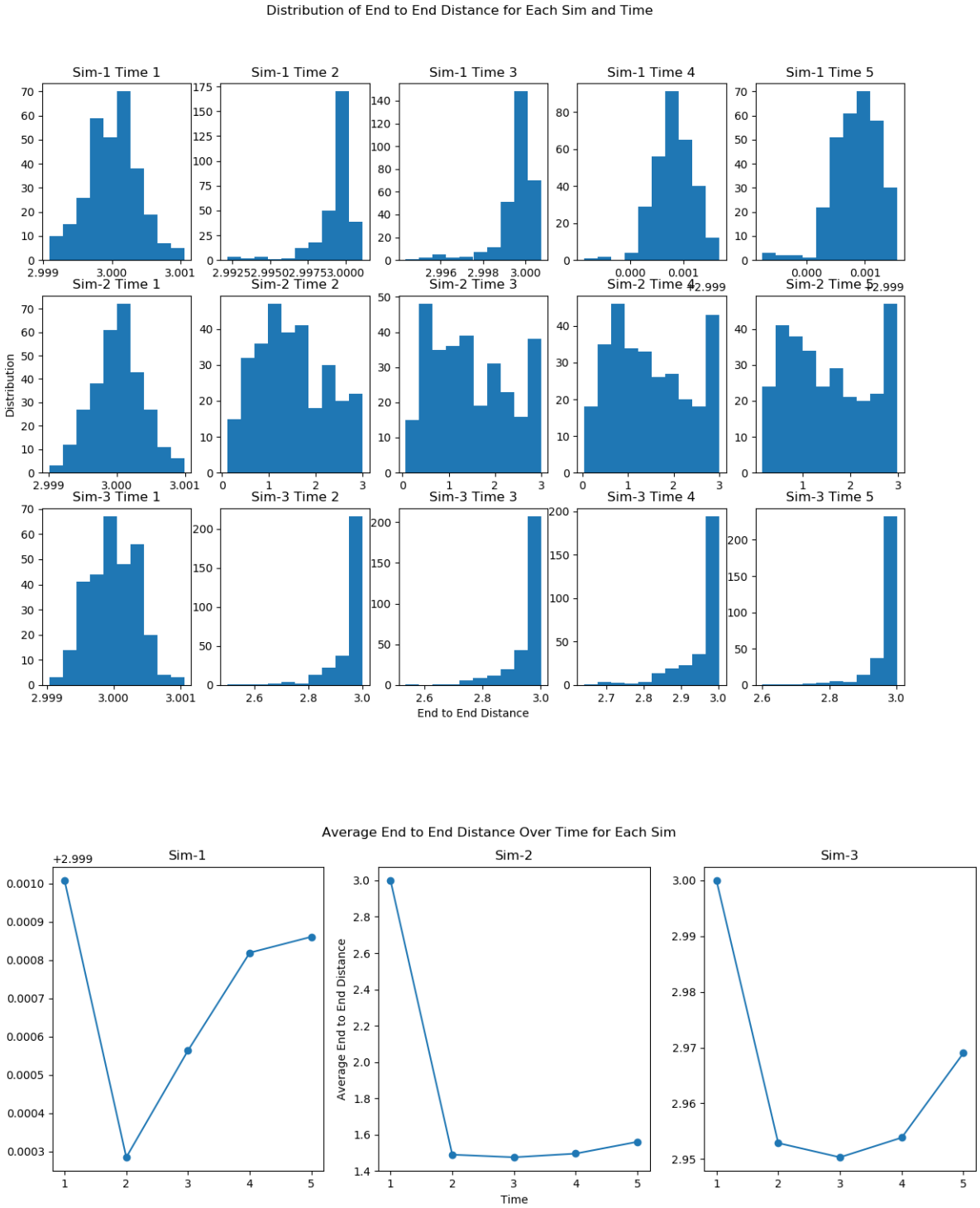


Figure 4: I know that technically it is not correct to plot lines, it should be only dots, but to better understand the data I plotted dots and lines.

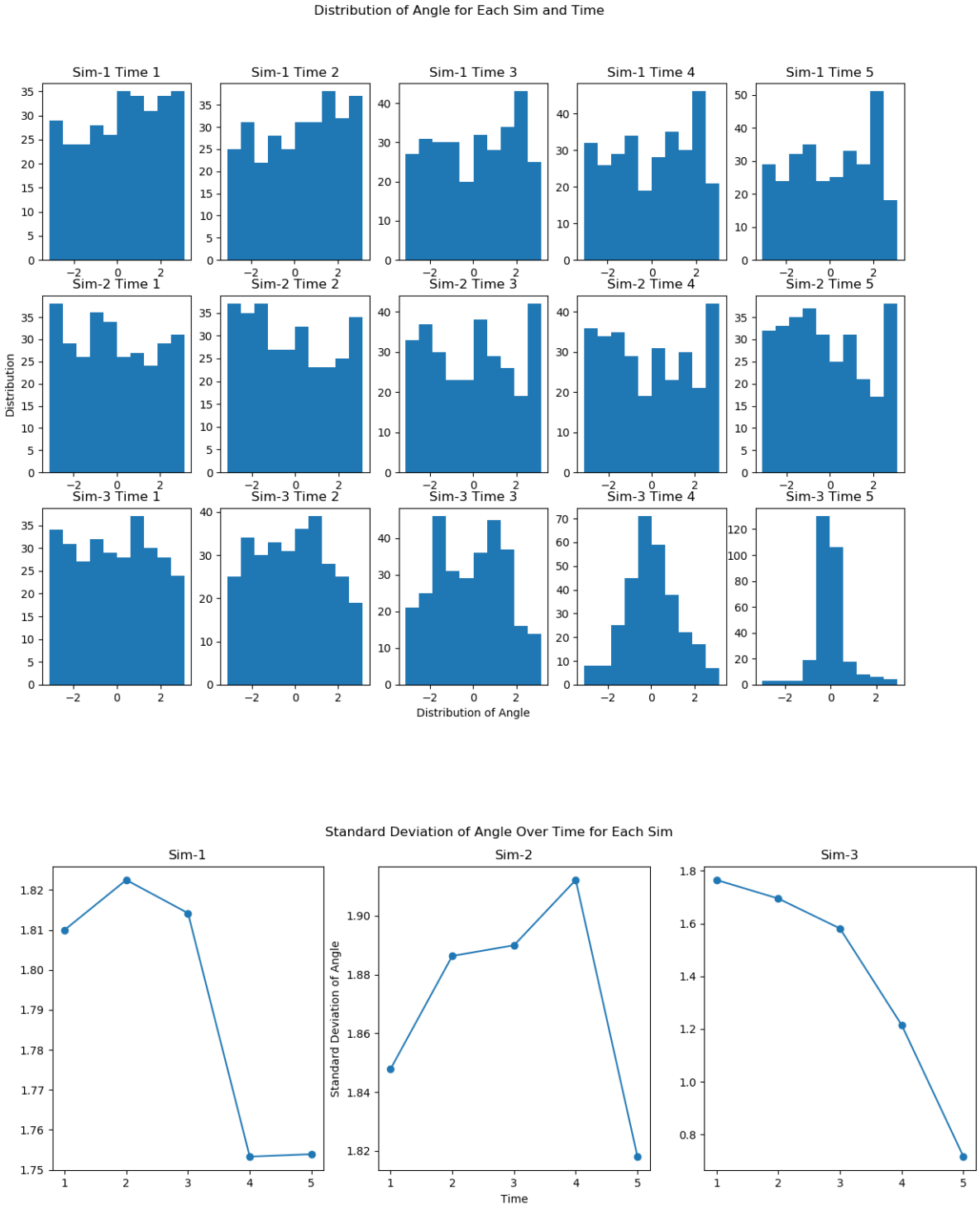


Figure 5: I know that technically it is not correct to plot lines, it should be only dots, but to better understand the data I plotted dots and lines.

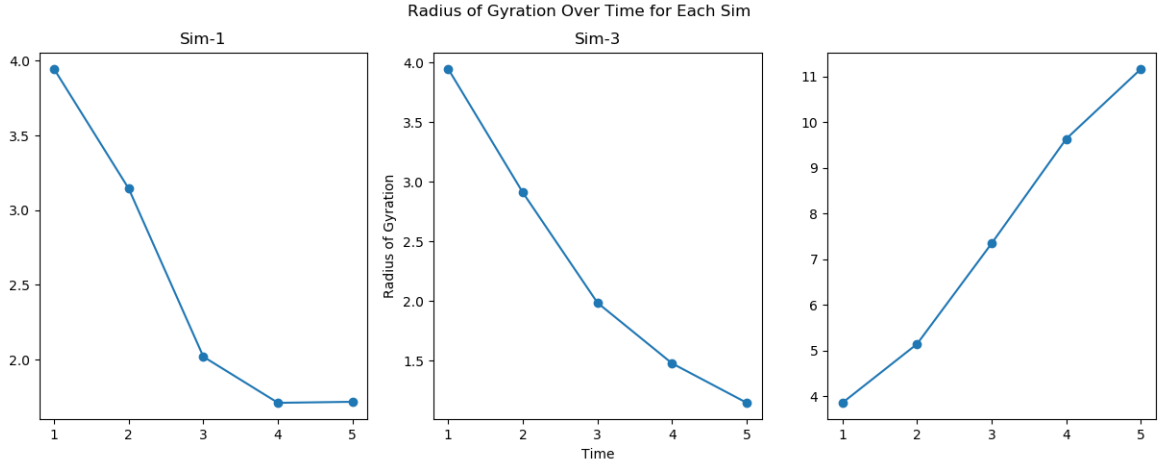


Figure 6: I know that technically it is not correct to plot lines, it should be only dots, but to better understand the data I plotted dots and lines.