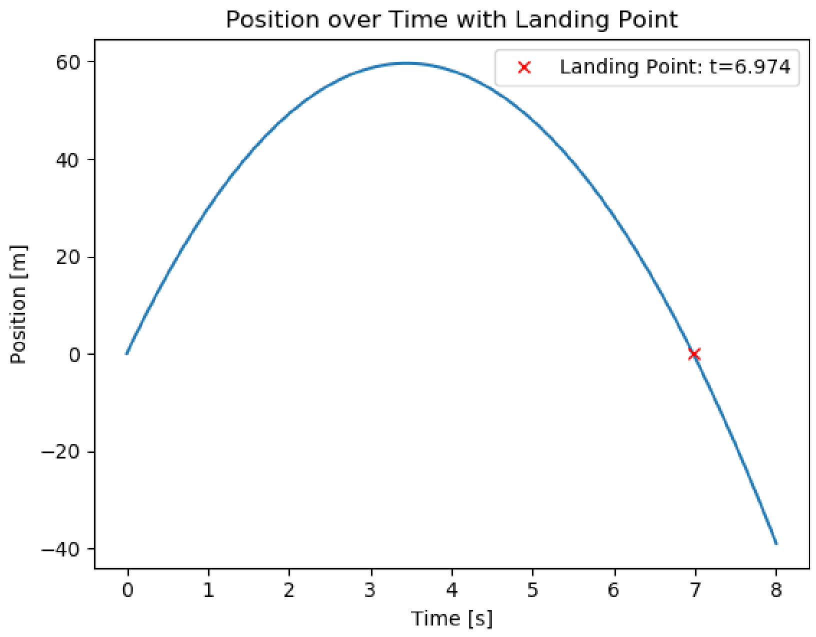
Homework 9

Ryan Branagan

March 28, 2019

Problem 1

```
#Ryan Branagan
#Collaborators: N/A
#Branagan_hw9_p1.py
#3/19/19

import numpy as np
import pylab as p
#%%
#This is when I did it by hand before
#print(Drag(6.97352815))

#Parameters
t0 = 0 # [s]
tf = 8 # [s]
points = 1000

#Function for motion in air
def Drag(t):
    B = 0.02 # [s^-1]
    g = 9.81 # [m/s^2]
    vy0 = 35. # [m/s]
    return -((g*t)/B)+((vy0+(g/B))/B)*(1-np.exp(-B*t))

#Bisection method
Guesses = np.array([[5,8],[6,7],[6.9,7]])
Cs = np.zeros(len(Guesses))
for i,G in enumerate(Guesses):
    a = G[0]
    b = G[1]
    while True:
```

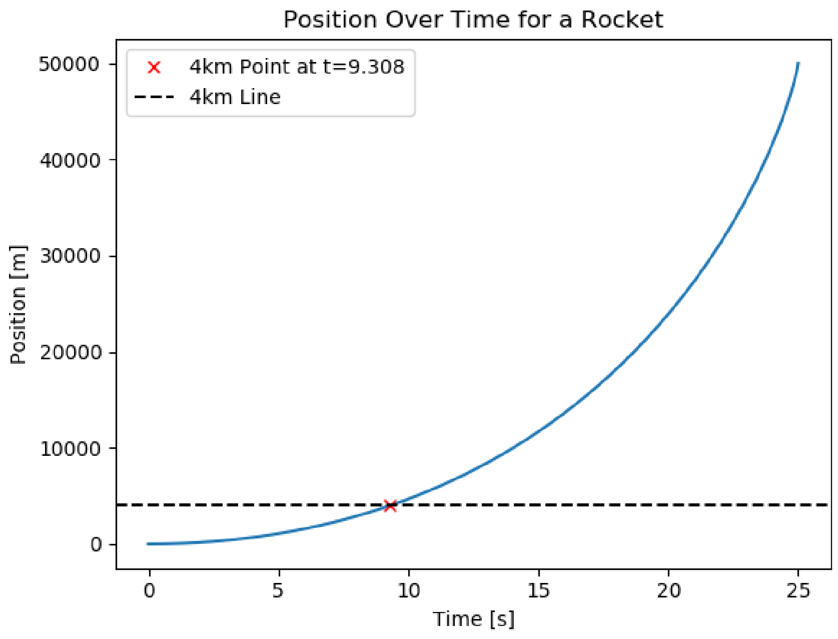


```
        c = (a+b)/2
        if np.abs(Drag(b)-Drag(a)) < 0.001:
            Cs[i] = c
            break
        if np.sign(Drag(c)) == np.sign(Drag(a)):
            a = c
        if np.sign(Drag(c)) == np.sign(Drag(b)):
            b = c

#Landing point
print("("+str(Cs[0])+","+str(Drag(Cs[0]))+")")

#Plotting
ts = np.linspace(t0,tf,points)

fig,ax = p.subplots(1,1)
ax.plot(ts,Drag(ts),"-")
ax.plot(Cs[0],Drag(Cs[0]),"rx",label="Landing Point: t="+str(round(Cs[0],3)))
ax.legend()
ax.set_title("Position over Time with Landing Point")
ax.set_xlabel("Time [s]")
ax.set_ylabel("Position [m]")
```

My final result as annotated on the graph is t=6.974. This routine uses the bisection method to get closer and closer to a zero. This method requires guesses of x-axis values on the left and right side of the zero. If we didn't mind waiting a few more seconds it is probably possible to push this to a further degree of accuracy all the way to the limit of machine precision.

Problem 2

```
#Ryan Branagan
#Collaborators: N/A
#Branagan_hw9_p2.py
#3/20/19

import numpy as np
import pylab as p
#%%
#Parameters
t0 = 0
tf = 24.999
points = 1000

ts = np.linspace(t0,tf,points)

#Functions
def X(t):
    x0 = 0
    v0 = 0
    m0 = 5000
    rho = 200
    u = 2000
    return x0+(u+v0)*t+u*((m0/rho)-t)*np.log(1-((rho*t)/m0))

def V(t):
    v0 = 0
    m0 = 5000
    rho = 200
    u = 2000
    return v0-u*np.log((m0-rho*t)/m0)

#Root finding
T = G = 2
i = 0
while True:
    if np.abs(X(T)-4000) < 0.001:
```



```
        break
    T = T - ((X(T)-4000)/V(T))
    i = i + 1

#Results
print("The time it takes for the rocket to reach 4km is: "+str(T)+"[s]")
print("Iterations: "+str(i)+"   Initial Guess: "+str(G))

#Plotting
fig, ax = p.subplots(1,1)
ax.plot(ts,X(ts))
ax.plot(T,X(T),"rx",label="4km Point at t="+str(round(T,3)))
ax.axhline(4000,color="k",linestyle="--",label="4km Line")
ax.legend()
ax.set_title("Position Over Time for a Rocket")
ax.set_xlabel("Time [s]")
ax.set_ylabel("Position [m]")
```

This routine uses Newton's method which like bisection requires guessing but this time only one guess. After a few attempts, I chose my guess to be two seconds because it gave me the best answer instead of a guess that was closer to the actual answer. This routine took eight iterations to come to an answer. The answer found was approximately 9.3 seconds. My graph has the point annotated but I though it would help illustrate where 4km was on the graph by adding a horizontal line.