JF COMPUTATIONAL LABORATORY REPORT SHEET
The Trajectory of a Projectile with Friction

This should be filled in using Word submitted ***as a hardcopy with graphs and commented source code****. Please don't forget to include your name and student number.*

Exercise 3: Projectile trajectories	Week: 1
Student Name: Kevin Grainger	Date: 09/02/2021
Laboratory Number :	Student number: 20333346

Abstract (no more than 200 words) The purpose of this exercise is to accurately simulate graph the path of a projectile through both a vacuum and a light atmosphere using python. This is done both analytically using the equations of motion and numerically using integration. The results of these methods are compared. (with same parameters used for each) Next the optimal launch velocity and launch angle is found using a loop. Both for a scenario with and without air resistance. The relationship between launch velocity and range is graphed both with and without air resistance. As is the relationship between launch angle and range. It is concluded that: The range of the projectile is proportional to the square of the launch velocity. The optimal launch velocity is the fastest possible. The optimal launch angle is 45 degrees when air resistance is not considered but is 33.75 degrees in our thin atmosphere. The flight path in a vacuum is symmetrical whereas with air resistance it 'compressed' on the right=hand side.	[4]

Section 1.2 Projectile motion without friction

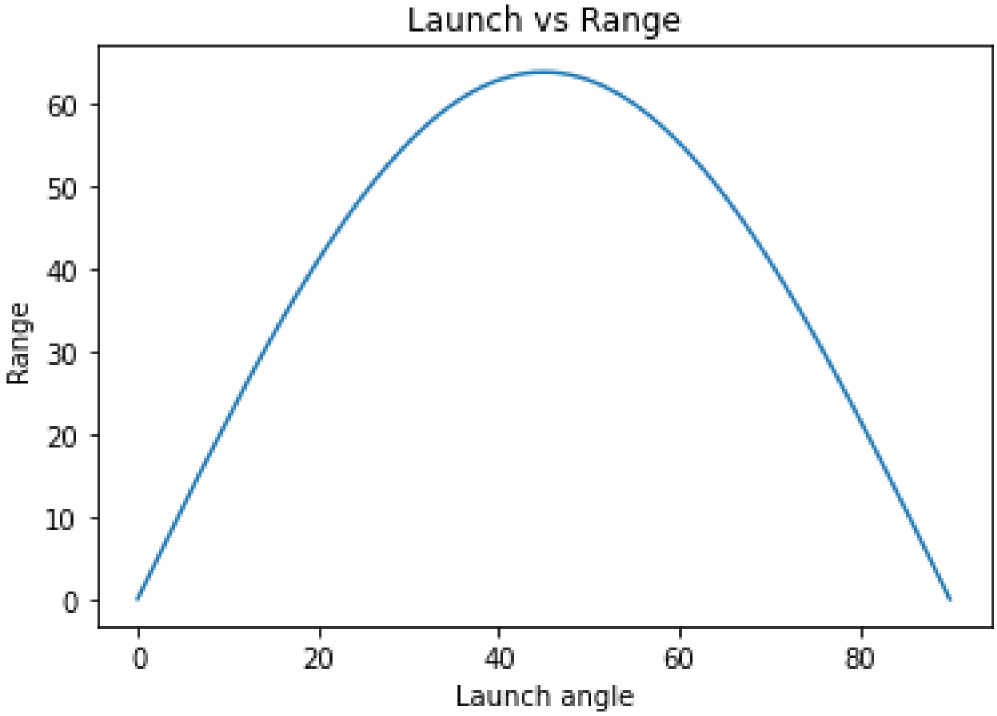
What is the time of flight for the parameters shown on the right ?	Initial velocity = 40 m/s Launch angle = $\pi/5$ (=36^0) Time of flight = 4.793 seconds	[1]
Choose another initial velocity (<40 m/s), and angle to compute the flight time.	Initial velocity =25 m/s Launch angle = pi/6 (=30°) Time of flight = 2.548 seconds	[1]
How does your numerically determined time of flight compare to the analytic solution ?	2.548 seconds – analytical solution 2.550 seconds-numerical solution The solution are identical to two decimal places, the discrepancy may be due the time step.	[1]

Section 2.1

What are the optimum launch angle and the	Initial velocity= 30 Optimum angle θ_{max} = 45.151 ≈45°	[1]

corresponding range for a given initial velocity?	Maximum range = 87.301	
Does the optimum angle depend on the initial velocity?	No, the optimium angle for projection is independent of the launch velocity. No matter what velocity is chosen the optimal angle is always = 45°	[1]

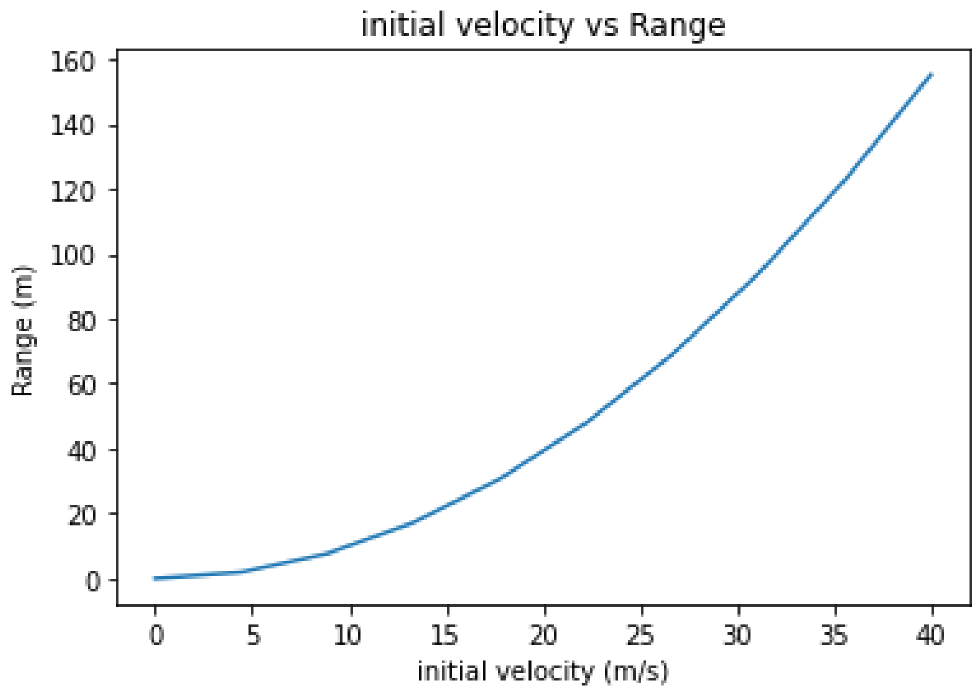
Attach a plot of range versus launch angle for a given initial velocity [1]
Initial velocity chosen = 25m/s



Section 2.2

Is the dependence of the maximum range on the initial velocity a straight line?	No, as the range is related to v^2 rather than just v. We know this by newtons equations of motion which give the range of a projectile as: v0^2(sin(2α)/g thus the range ∝ v0^2	[1]
How does the range depend on the initial velocity?	the range ∝ v0^2	[1]

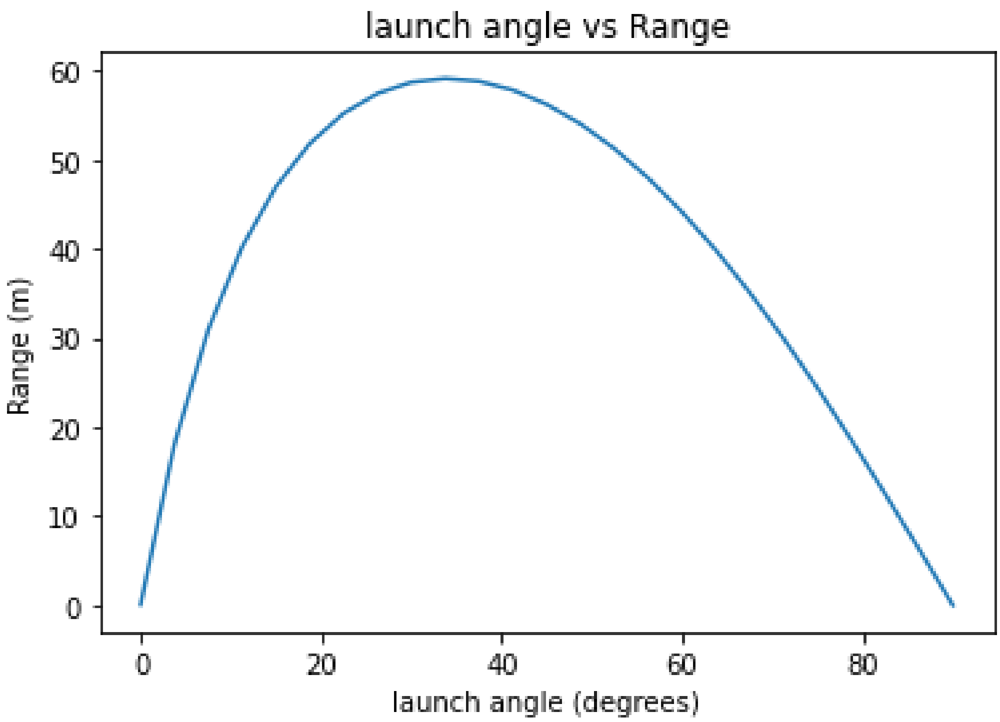
Attach a plot of the maximum range versus initial velocity [1]



Section 3.1 Projectile motion with height dependent air drag

Explain why the plots of *y* versus *t* and x versus t look like they do:	The x vs t plot likes like a sqrt x graph, this is because the horizontal velocity is continually being slowed, thus less and less ground is being covered. The y vs t graph is slightly tilted to the left because of how air resistance affects the projectile differently at 40m/s at ground level.	[1]
Explain why the plot of *y* versus *x* looks like it does:	The graph is almost compressed on the right-hand side. This because the air resistance continually decelerates the horizontal velocity of the object until it is basically =0, when it then drops nearly vertically from the sky.	[1]
Compute the range of the projectile for some launch angle and initial velocity:	Initial velocity=40 Angle=pi/4 to 45° Range distance = 56.232 meters	[1]
What are the optimum launch angle and the corresponding range for a given initial velocity?	Optimum angle θ_{max} = 33.75° Maximum range = 59.142 The velocity chosen was 40 m/s.	[1]

Attach a plot of range versus launch angle (γ=0.005, height=100) for an initial velocity of 40 m/s [1]:



Additional marks:
Tidy workbook [1]
Graphs correctly labeled [1]

CODE:
<u>Part 1.1</u>

```
import matplotlib.pyplot as plt
import numpy as np
import math
#all the necessary modules

g=9.81 #grav constant
dt=1e-3 #delta t or out smallest time unit
v0=25 #the initial speed
#all the necessary constants

angle=math.pi/6 #the projection angle of 45 degrees or pi/4
time=np.arange(0,10,dt) #gives an axis of evenly spaced values in the-
                        #range (0,10,dt), this is the time axis
#we now must define the functions for the x and y velocity
vx0=math.cos(angle)*v0
vy0=math.sin(angle)*v0
#these form the parabola of projection

#These enable us to find the location of the object using a time variable
xa=vx0*time #derivated from Newtons eqs of motion
ya=-0.5*g*time**2+vy0*time # derived from Newtons eqs of motion
```

```
fig1=plt.figure() #this is the picture of the plot
plt.ylim(0,max(ya)+15)#so the projectile does go through the floor
#the y axis caps just above the projectiles max-height
#we can thus see both the max-height and range clearly

plt.plot(xa,ya)#This plots x-position against y-position
#Label the axis
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.show() ;
print ((v0**2*math.sin(2*angle))/(g*vx0) #the range formula from eqs of motion,
divided by the horizontal velocity
```

Part 1.2

```
import matplotlib.pyplot as plt
import numpy as np
import math


g=9.81 #grav constant
dt=1e-3 #delta t or out smallest time unit
v0=40 #the initial speed
#all the nessicary constants

angle=math.pi/5 #the projection angle of 45 degrees or pi/4
time=np.arange(0,10,dt) #gives an axis of evenly spaced values in the-
                        #range (0,10,dt), this is the time axis

def trajectory(angle, v0): #This is the trajectory function
    pass #next we define v0 further into it's components
    vx0=math.cos(angle)*v0

    vy0=math.sin(angle)*v0

    x=np.zeros(len(time)) #creates an array of zeros of thr same lenght-
    y=np.zeros(len(time)) #as the array 'time'
    #the first term of the function must be the starting point and the
    #second comes their starting velocities
    x[0],y[0]=0,0 #this makes [0,0] the origin
    x[1],y[1]=x[0]+vx0*dt,y[0]+vy0*dt #this is their starting velocities


    i=1 #i is the variable for the function which starts at 1
    while y[i]>=0:#so the projectile doesn't go through the floor
        pass
        x[i+1]=(2*x[i]-x[i-1])#derived from integration
        y[i+1]=(2*y[i]-y[i-1])-g*dt**2
```

```
        i=i+1 #describes the next term of the while loop
        #it uses the prior two points to determine the third

    x=x[0:i+1]
    y=y[0:i+1] #trims the arrays to size
    return x,y, (dt*i), x[i]
#(dt*i) gives the flight time
#x[i] gives the range
x, y, duration, distance = trajectory(angle, v0)

print ('Distance:' , distance)
print ('Duration:', duration)
```

Part 2.1 Finding the max range of the Projectile

To find the optimal Angle:

```
import matplotlib.pyplot as plt
import numpy as np
import math


g=9.81 #grav constant
dt=1e-3 #delta t or out smallest time unit
v0=30 #the initial speed
#all the nessicary constants

angle=math.pi/5 #the projection angle of 45 degrees or pi/4
time=np.arange(0,10,dt) #gives an axis of evenly spaced values in the-
                        #range (0,10,dt), this is the time axis

def trajectory(angle, v0): #This is the trajectory function
    pass #next we define v0 further into it's components
    vx0=math.cos(angle)*v0

    vy0=math.sin(angle)*v0

    x=np.zeros(len(time)) #creates an array of zeros of thr same lenght-
    y=np.zeros(len(time)) #as the array 'time'
    #the first term of the function must be the starting point and the
    #second comes their starting velocities
    x[0],y[0]=0,0 #this makes [0,0] the origin
    x[1],y[1]=x[0]+vx0*dt,y[0]+vy0*dt #this is their starting velocities


    i=1 #i is the variable for the function which starts at 1
    while y[i]>=0:#so the projectile doesn't go through the floor
        pass
```

```
        x[i+1]=(2*x[i]-x[i-1])#derived from integration
        y[i+1]=(2*y[i]-y[i-1])-g*dt**2

        i=i+1 #describes the next term of the while loop
        #it uses the prior two points to determine the third

    x=x[0:i+1]
    y=y[0:i+1] #trims the arrays to size
    return x,y, (dt*i), x[i]
#(dt*i) gives the flight time
#x[i] gives the range
x, y, duration, distance = trajectory(angle, v0)

print ('Distance:' , distance)
print ('Duration:', duration)

n=300 #The number of points in the linspace
angles=np.linspace(0,math.pi/2,n)#Set 90 degrees as the max launch angle
maxrange=np.zeros(n)#primes the max range array

for i in range(n):     #loop to determine the optimum launch angle
    pass
    x,y,duration,maxrange[i]=trajectory(angles[i], v0)

angles=angles/2/math.pi*360 #converts radians to degrees
print("opt angle:",angles[np.where(maxrange==np.max(maxrange))])
figure1=plt.figure()
plt.plot(angles,maxrange) #plots launch angle against range
plt.xlabel('Launch angle')
plt.ylabel('Range')
plt.title('Launch vs Range')
plt.show
```

2.2 The dependence of range on initial velocity

To find the optimum Launch Velocity:

```
import matplotlib.pyplot as plt
import numpy as np
import math


g=9.81 #grav constant
dt=1e-3 #delta t or out smallest time unit
v0=40 #the initial speed
#all the nessicary constants

angle=math.pi/5 #the projection angle of 45 degrees or pi/4
time=np.arange(0,10,dt) #gives an axis of evenly spaced values in the-
                        #range (0,10,dt), this is the time axis

def trajectory(angle, v0): #This is the trajectory function
```

```
    pass #next we define v0 further into it's components
    vx0=math.cos(angle)*v0

    vy0=math.sin(angle)*v0

    x=np.zeros(len(time)) #creates an array of zeros of thr same lenght-
    y=np.zeros(len(time)) #as the array 'time'
    #the first term of the function must be the starting point and the
    #second comes their starting velocities
    x[0],y[0]=0,0 #this makes [0,0] the origin
    x[1],y[1]=x[0]+vx0*dt,y[0]+vy0*dt #this is their starting velocities


    i=1 #i is the variable for the function which starts at 1
    while y[i]>=0:#so the projectile doesn't go through the floor
        pass
        x[i+1]=(2*x[i]-x[i-1])#derived from integration
        y[i+1]=(2*y[i]-y[i-1])-g*dt**2

        i=i+1 #describes the next term of the while loop
        #it uses the prior two points to determine the third

    x=x[0:i+1]
    y=y[0:i+1] #trims the arrays to size
    return x,y, (dt*i), x[i]
#(dt*i) gives the flight time
#x[i] gives the range
x, y, duration, distance = trajectory(angle, v0)

print ('Distance:' , distance)
print ('Duration:', duration)

n=10 #No of points in linspace
v0=np.linspace(0,40,n) #sets max velocity to 40m/s
maxrange=np.zeros(n)

for i in range(n): #loop to find opt velocity
        x,y,duration,maxrange[i]=trajectory(angle, v0[i])


print("opt velo:",v0[np.where(maxrange==np.max(maxrange))])

plt.plot(v0,maxrange) #plot launch velocity against range
plt.xlabel('initial velocity (m/s)')
plt.ylabel('Range (m)')
plt.title('initial velocity vs Range')
plt.show
```

3.1 Projectile with friction dependent on height.

```
import matplotlib.pyplot as plt
import numpy as np
import math
#the necessary constants
g=9.81#gravity
dt=1e-3#our time step or smallest time interval
v0=40#the initial velocity
angle=0.58904862#launch angle
F=0.005 #our friction coefficient
height=100#height at which friction is zero
time=np.arange(0,10,dt)#gives an axis of evenly spaced values in the-
                        #range (0,10,dt), this is the time axis

def traj_fric(angle,v0): #our trajectory function
    v0x=math.cos(angle)*v0#x component of initial velocity
    v0y=math.sin(angle)*v0#y component of initial velocity
    #we now must prime our position arrays
    x=np.zeros(len(time))#creates an array of zeros of the same-
    y=np.zeros(len(time))#lenght as 'time'

    #we must define the first and second terms of the function so as to-
    #be able find the third and so on
    x[0],y[0]=0,0 #sets the objects starting position to [0,0]
    x[1],y[1]=x[0]+v0x*dt,y[0]+v0y*dt#these are the x,y positions of the-
    #the object after time period dt


    i=1

    while y[i]>=0:
        f=0.5*F*(height-y[i])*dt #the air resistance equation
        #Our x,y position functions derived from integration
        x[i+1]=((2*x[i]-x[i-1]+(f)*x[i-1])/(1+f))
        y[i+1]=((2*y[i]-y[i-1]+(f)*y[i-1]-(g*dt**2))/(1+f))
        i=i+1#describes the second term of the while loop
        #we use the first and second points to find the third, then the-
        #second and third points to find the fourth and so on.

    x=x[0:i+1]#trim the arrays to size
    y=y[0:i+1]

    return x,y,(dt*i),x[i]
    #(dt*i) gives the flight time
    #x[i] gives the range
#------------------------------------
#Here we are graphing the x-coordinated against the y-cordinates
#ie the trajectory of the object
x,y,duration,distance=traj_fric(angle,v0)
print ('Distance_f:',distance)
```

```
print ('duration_f:',duration)


#Firstly we plt x and y against time
Time=np.arange(0,duration+0.001,dt)#New time array as the other won't work
#Change 'duration' from a float to an array
#we have to create a time array of the same size as the array x
#by trial and error I found that adding 0.001 or dt to the duration works.

#------------------------------------
#This plots the trajectory of the projectile
figure1=plt.figure()#creates a unique graph, Graph No.1
plt.plot(x,y)
plt.xlabel('Horizontal position')
plt.ylabel('Vertical position')
plt.title('Trajectory in thin atmostsphere')
plt.show

#------------------------------------
#This plots the x position over time using the new time array
figure4=plt.figure()
plt.plot(Time,x)
plt.xlabel('Time')
plt.ylabel('X-position')
plt.title('Horizontal Position over time')
#This plots the y position over time
figure5=plt.figure()
plt.plot(Time,y)
plt.xlabel('Time')
plt.ylabel('y-position')
plt.title('Vertical Position over time')
plt.show
#-------------------------------------
#here we create a loop to find the optimium angle at which to launch the-
#object in the thin atmosphere
n=25#the number of points in the linspace
angles=np.linspace(0,math.pi/2,n)#defines a space with a max angle of 90 deg.
maxrange=np.zeros(n)#primes the maxrange array
for i in range (n):#loop to find the opt launch angle
    x,y,duration,maxrange[i]=traj_fric(angles[i],v0)#labels the outputs
angles=angles/2/math.pi*360#converts radians to degrees
print ("Opt angle in atmosphere",angles[np.where(maxrange==np.max(maxrange))])

figure2=plt.figure()#gives separate graph, graph No.2
plt.plot(angles,maxrange) #plots angle against range
plt.xlabel('launch angle (degrees)')
plt.ylabel('Range (m)')
plt.title('launch angle vs Range')
plt.show
#-------------------------------------
```

```
#Here we create a loop to determine the optimum initial in the thin atmosphere
n=25#no. of points
v0=np.linspace(0,40,n)#sets max initial velocity to 40m/s
maxrange=np.zeros(n)
for i in range (n): #loop to find the opt velocity
    x,y,duration,maxrange[i]=traj_fric(angle,v0[i])
print ("Opt velocity in atmosphere",v0[np.where(maxrange==np.max(maxrange))])

figure3=plt.figure()# Graph No.4
plt.plot(v0,maxrange) #plots velocity against range
plt.xlabel('initial velocity')
plt.ylabel('Range')
plt.title('initial velocity vs Range')
plt.show
#--------------------------------------
```