ASEN 3801-002 Lab 1

Hannah Priddy, Ben Reynolds, Jared Steffen, Chloe Zentner

Problem 1

1.a

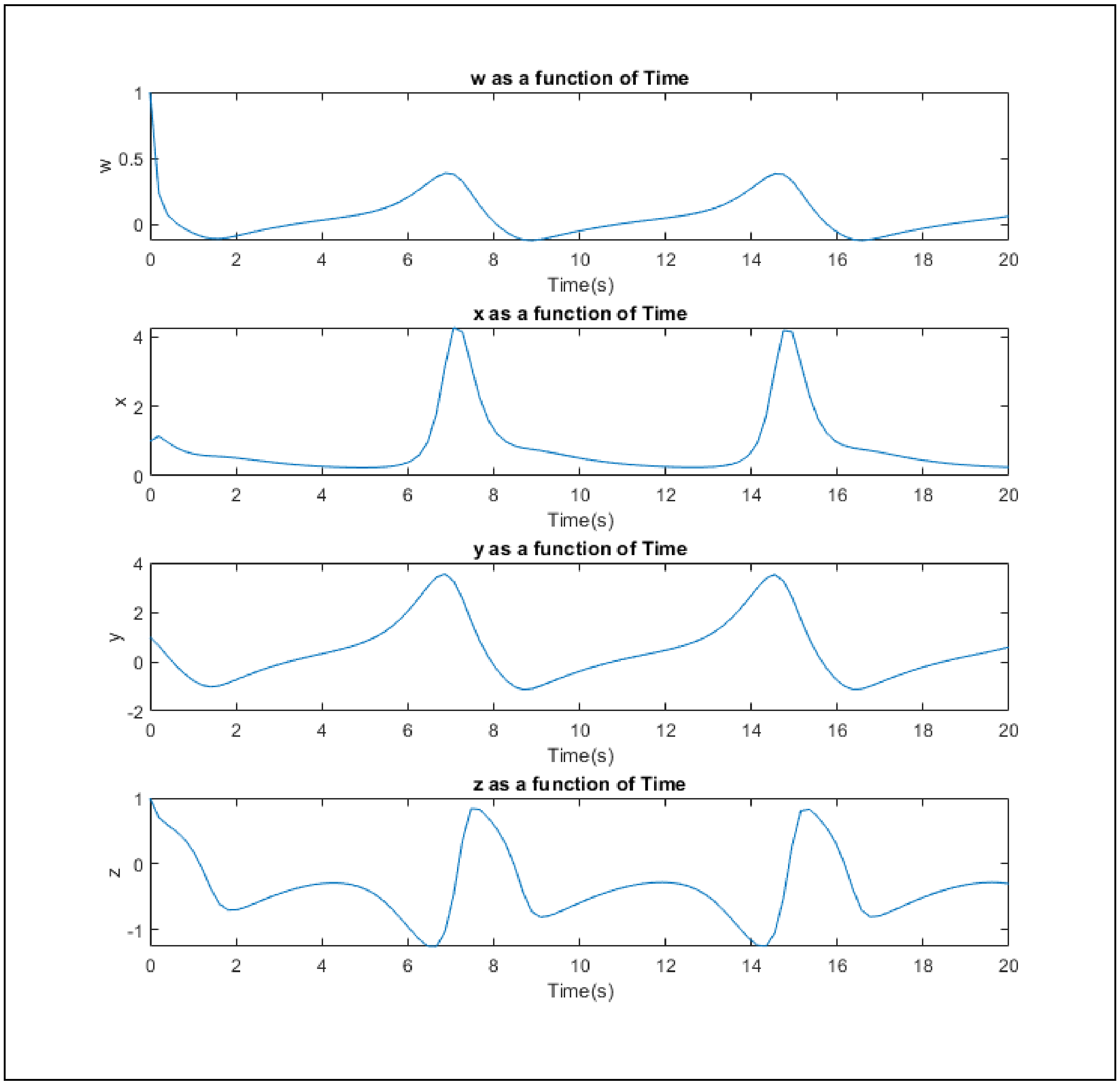

Figure 1: Set of Four plots created with ODE45 as a function of time

For our initial conditions we set it to [1;1;1;1] and used ODE45 with a time span of 20 to produce these plots. Since they all used the same variables, we see that they all have bumps around the same time and all trend the same way. Code is in appendix 1.a

1.b

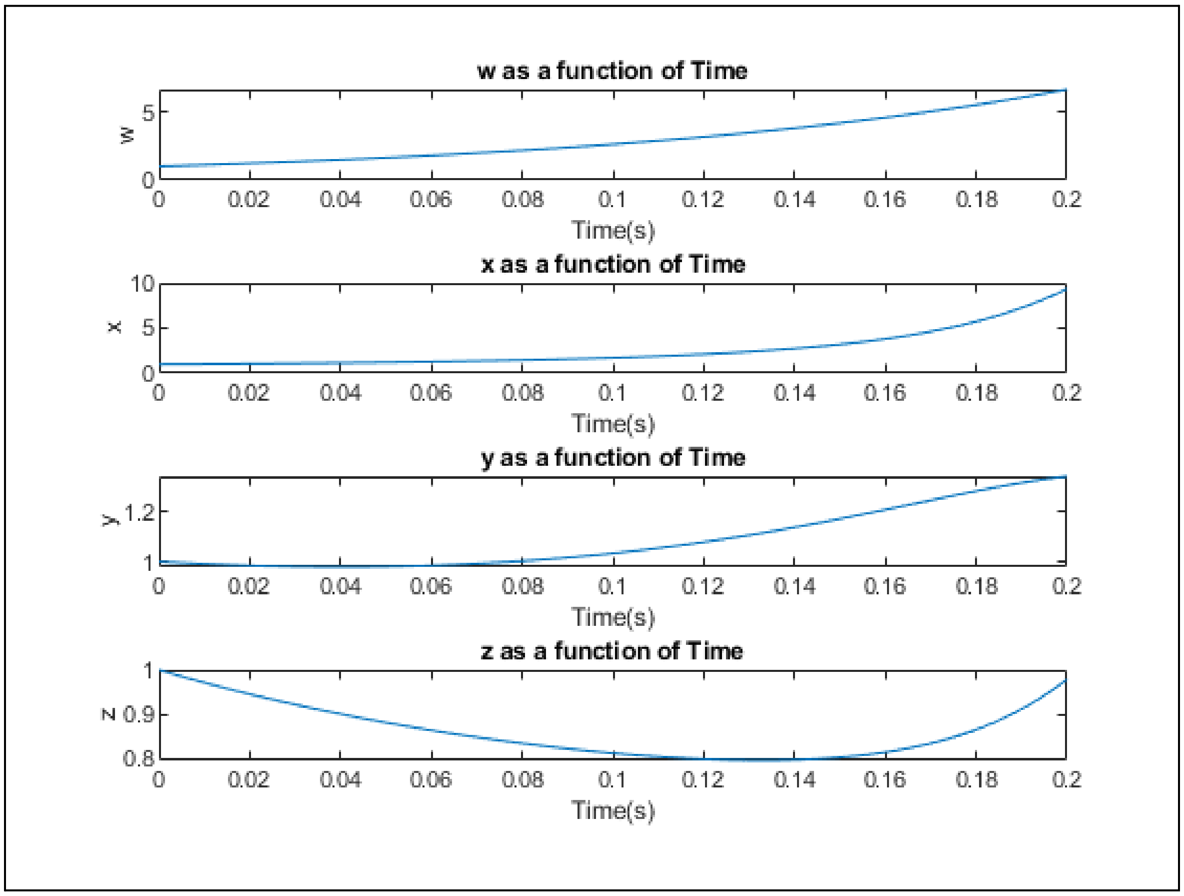

Figure 2: Set of Four plots created with ODE45 as a function of time

For this section we changed our time span to 0.2 seconds, and also changed our w dot to 9w +y. With this change we saw that instead of initially decreasing, w dot increased from the beginning. If we increased our time span (which we do later) we predicted that it would go off into infinity. This would also change the other graphs in a drastic way since they are all dependent on the same x,y,z, and w.

1.c

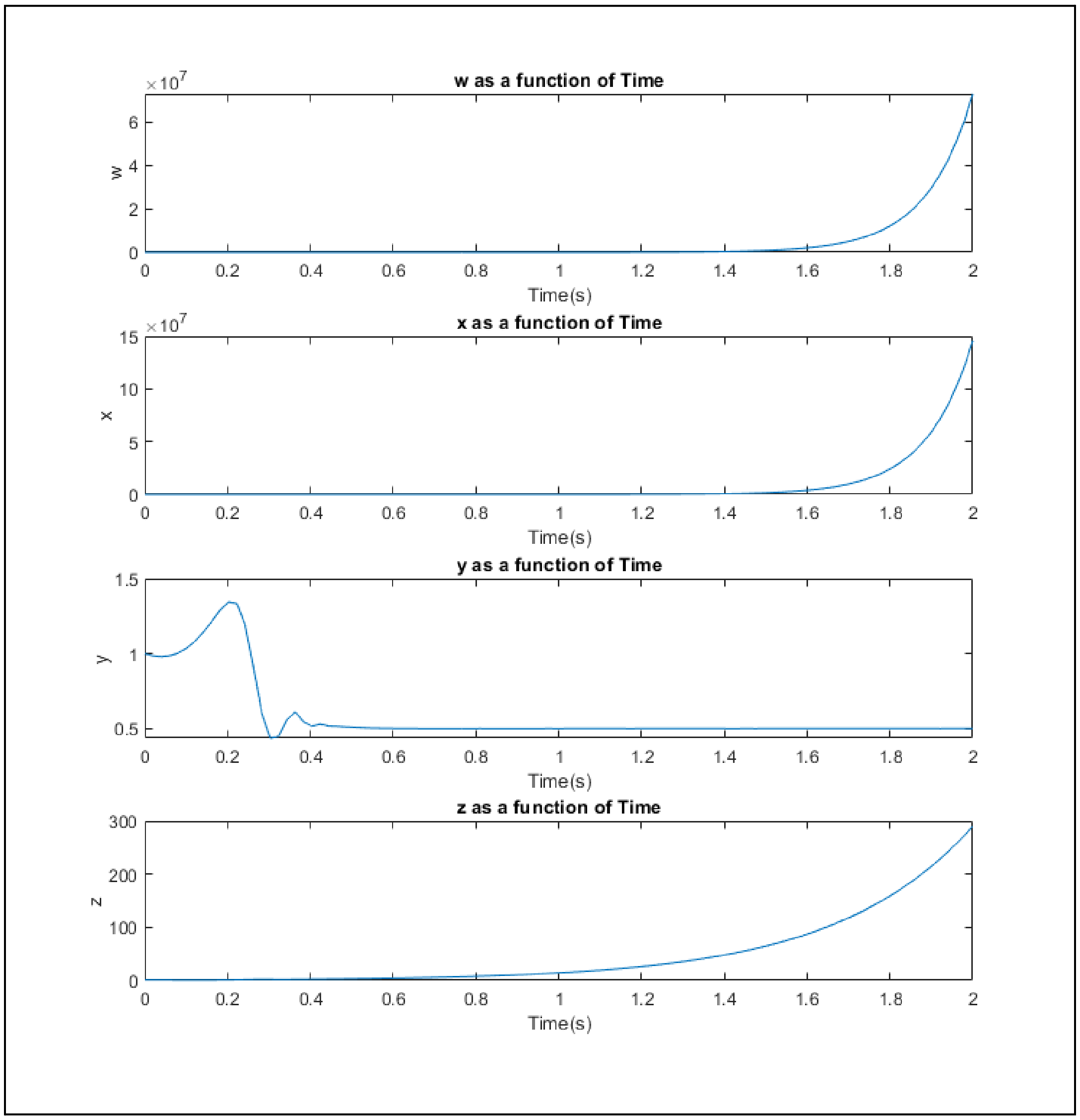

Figure 3: Set of four plots displaying the effects of wind on magnitude of landing position

The simulation, when set to simulate 2 seconds of adjustments, took a very long time to complete the calculations. This is for a couple of different reasons. First, The system is no longer repetitive. In part 1b, it can be seen that the equations follow a repeated pattern that is not present in part c. This requires more calculation, with different sensitivity to maintain accuracy. The second reason, and most likely to have the greatest impact, would be the initial scale of changes in the equation. The values of w, x, and z, change minutely for the first 1.6 seconds of the simulation. With the changes being so subtle, the small variations are statistically significant in the ODE45 matlab function. This requires more iterations of the equations, as an accurate representation of the progress of the system of differential equations had to contain any statistically significant, nonlinear change. Without a pattern and with such a small initial scale, these threaten to time out the MATLAB coderunner.

Problem 2

2.b

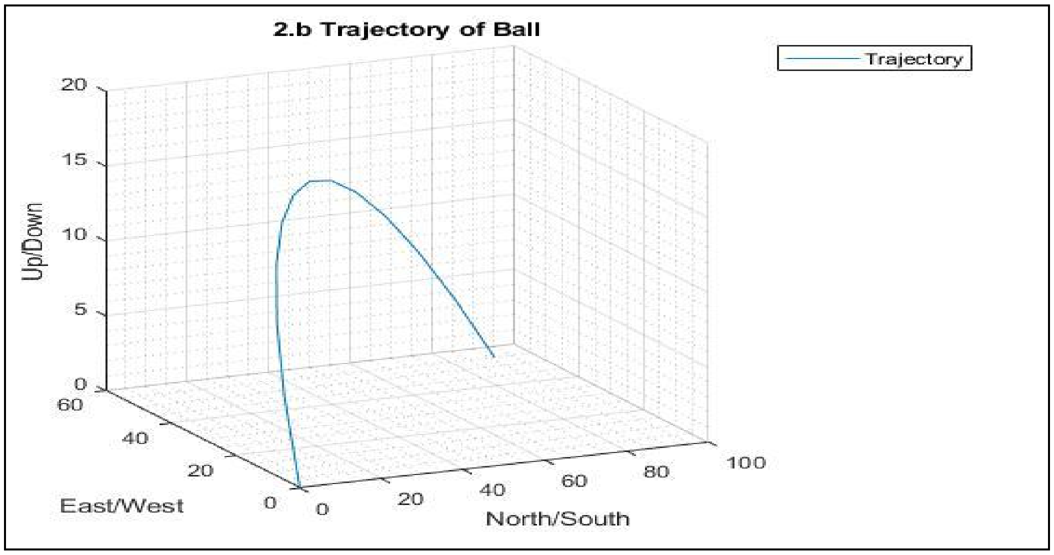

Figure 4: Trajectory of Ball with Wind

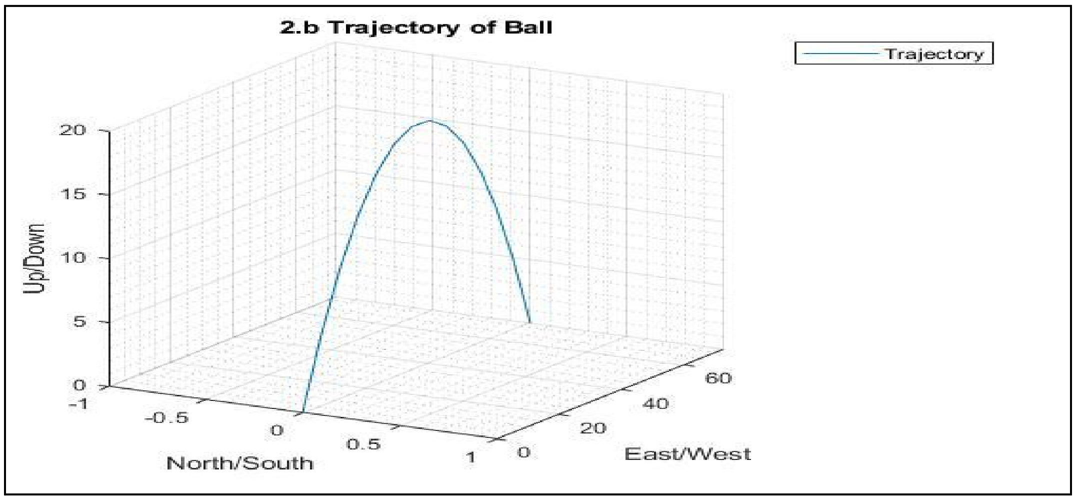

Figure 5: Trajectory of Ball without Wind

In Figure 5 we see the trajectory of the ball without wind. And these results make sense because when launching a ball it takes on a parabola shape because of velocity, gravity and drag. And since it has no initial velocity in the North/South direction it stays on zero. While once we add wind (Figure 4) we see that the trajectory changes and ends with position in the North/South direction. This also makes sense because once we add wind, it adds another vector pushing the object. Additionally it doesn't go as high or as far in the East/West direction because we have increased the drag with the wind vector.

2.c

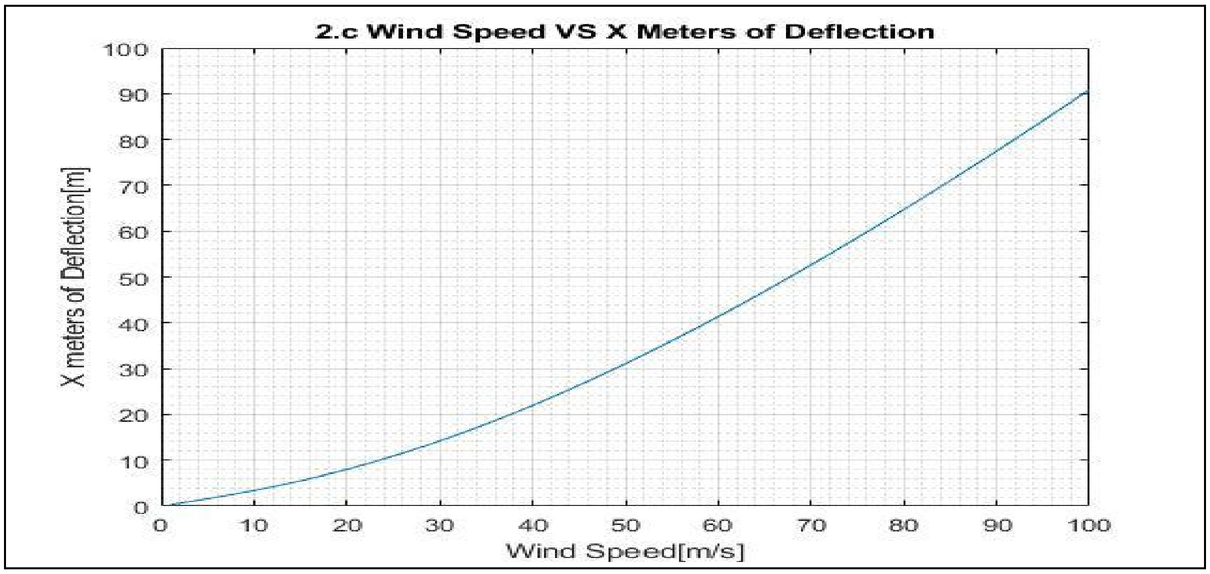

Figure 6: Wind Speed vs X-Meters of Deflection

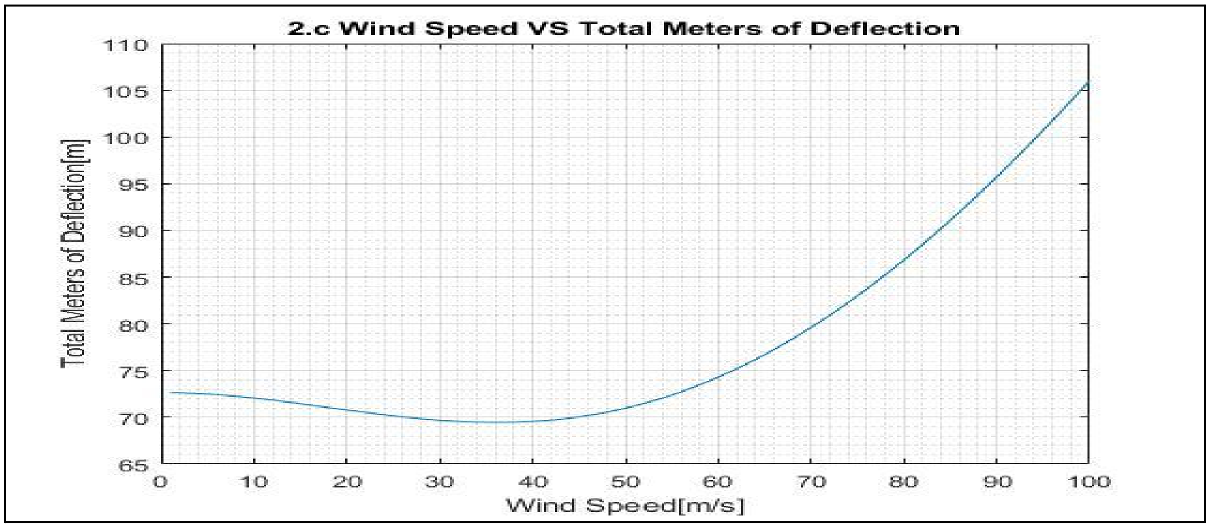

Figure 7: Wind Speed vs Total Meters of Deflection

In section 2.c, we looked at the sensitivity of the landing location due to wind. Figure 6 shows the deflection purely in the X direction, while Figure 7 shows the total deflection. With a wind blowing in the X direction (North), we see an increase in the X direction deflection as the wind speed increases. However, in terms of total deflection, we see an initial decrease in landing location, which is then followed by an increase in landing location distance. This is due to the fact that at low wind speeds, the ball is fighting against the wind for a longer duration of time, resulting in a smaller landing distance. As wind speed increases, the ball quickly aligns with the wind vector. When this occurs, the ball is no longer fighting against the wind, but is being pushed by the wind, resulting in a larger landing distance.

2.d

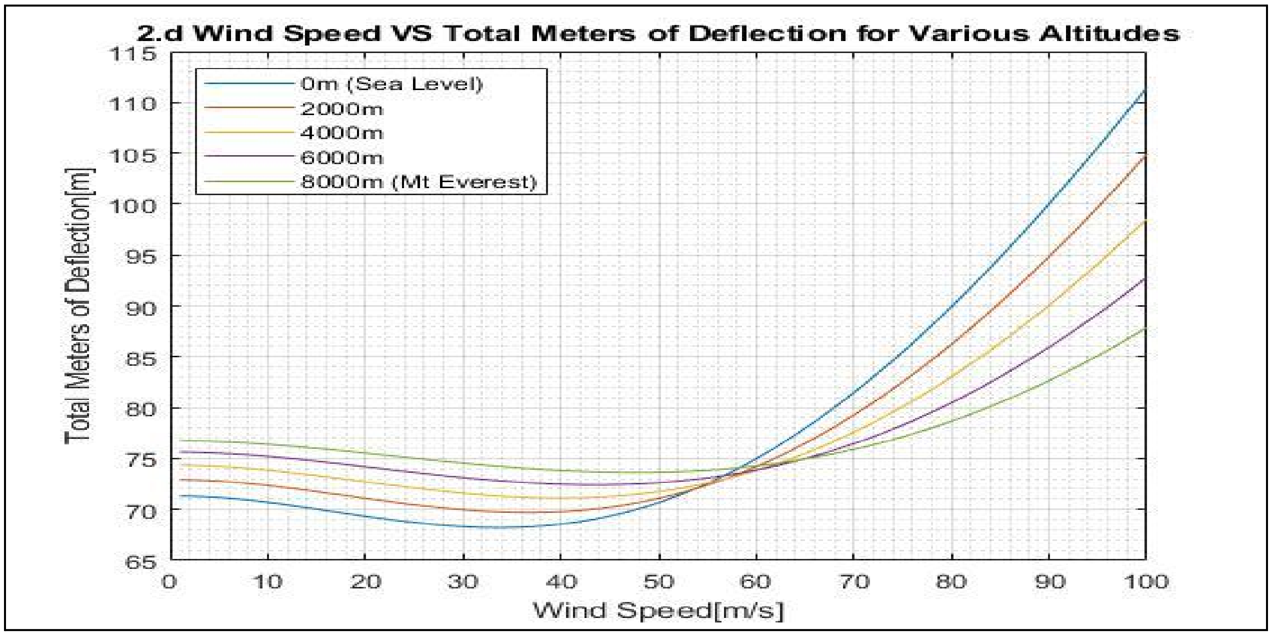

Figure 8: Total Meters of Deflection vs Wind Speed at Various Altitudes

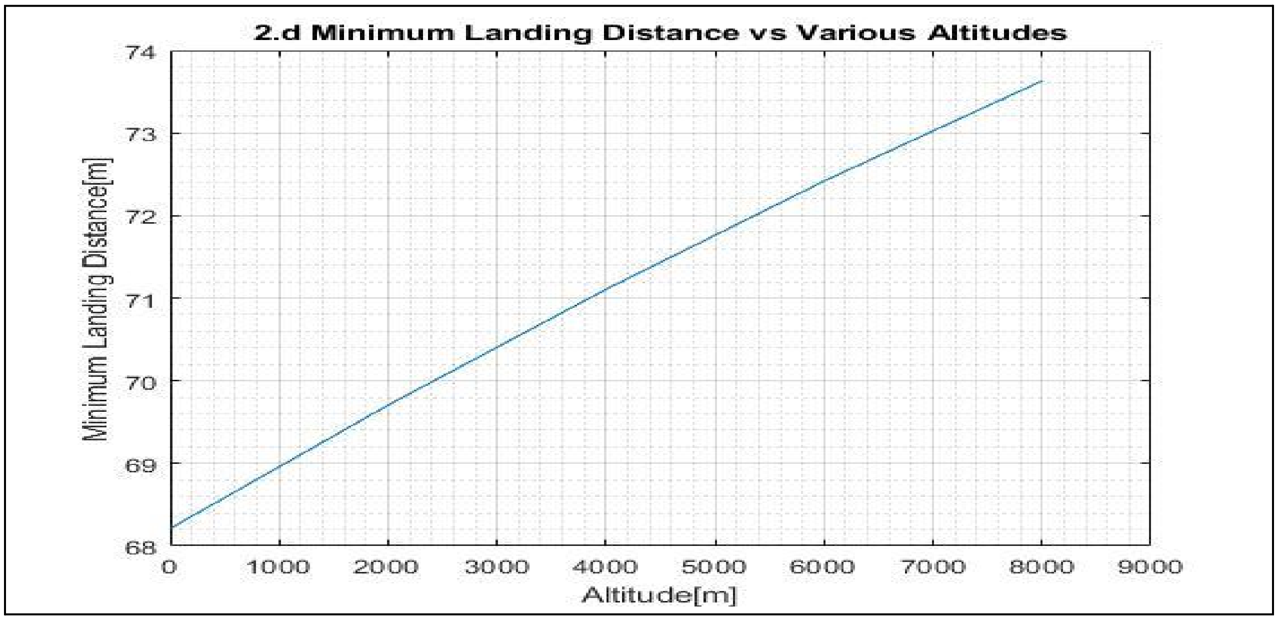

Figure 9: Minimum Landing Distance vs Altitude

In section 2.d of our analysis, we aimed to demonstrate the influence of altitude on landing location concerning various wind speeds and ascertain the minimum landing distance. Figure 8 encompasses altitudes ranging from 0 to 8000 meters, with wind speeds increasing in 2000-meter increments from 0 to 100 m/s. Notably, our findings reveal a distinct pattern: at lower wind speeds, lower altitudes correlate with shorter landing distances, while at higher wind speeds, lower altitudes result in longer landing distances. This phenomena can be explained by the fact that at high altitude and high wind speeds, the wind will have less of an effect on the ball due to lower air density, while at low altitude and high wind speeds, the wind has more of an effect on the ball. Furthermore, the minimum landing distance also escalates with altitude, but what's interesting is that this minimum distance occurs at progressively higher wind speed values as altitude rises. This effect is primarily due to the heightened impact of wind drag experienced at lower altitudes. These trends are clearly evident in both Figures 8 and 9 of our analysis.

2.e

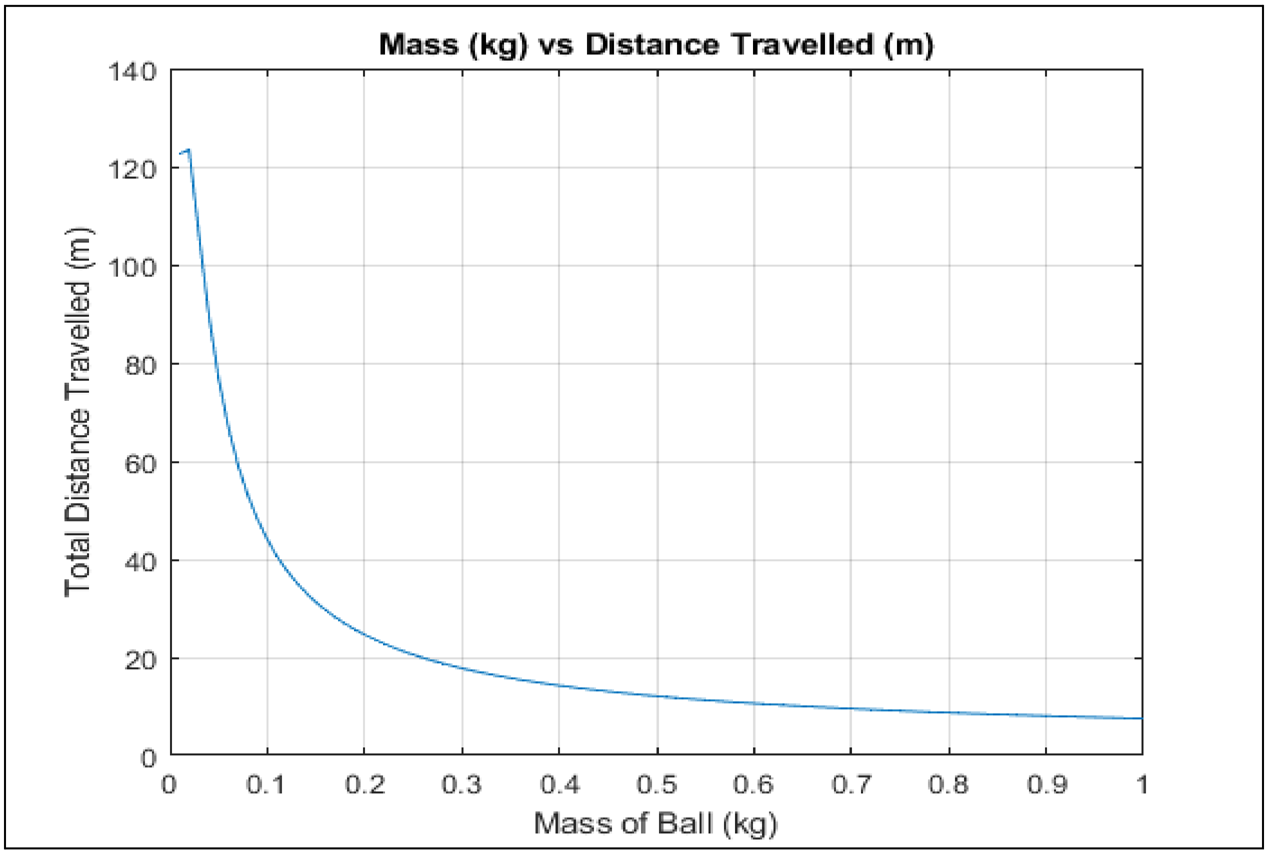

Figure 10: Mass vs Distance Traveled with Constant Kinetic Energy

Kinetic energy was held constant at 20 kg m^2/ s^2 by varying initial velocity with mass. As a result, the balls with greater mass traveled less distance overall. Because of the exponential drop in initial velocity, the overall momentum of the objects decreased, making it more difficult for the ball to overcome gravity over time.

Team Member Participation

	Plan	Model	Experiment	Results	Report	Code	ACK
Name							
Hannah Priddy	1	1	X	0	1	2	X
Ben Reynolds	1	1	X	1	1	1	X
Jared Steffen	1	1	X	1	1	1	X
Chloe Zentner	1	1	X	1	1	1	X

Appendix 1

```
% Hannah Priddy, Chloe Zentner, Ben Reynolds, Jared Steffen
% ASEN 3801
% Lab 1 Part 1
% Created: 9/8/23
% Last edited: 9/13/23
clear all; close all; clc; %housekeeping
tspan = linspace(0,2); % changes from different parts goes from 20 to .2 to 2
initial = [1; 1; 1; 1]; %initial values we set
[time, statearray] = ode45(@odefunction, tspan, initial);
figure(1)
subplot(4,1,1)
plot(time,statearray(:,1))
title("w as a function of Time")
xlabel("Time(s)")
ylabel("w")
subplot(4,1,2)
plot(time,statearray(:,2))
title("x as a function of Time")
xlabel("Time(s)")
ylabel("x")
subplot(4,1,3)
plot(time,statearray(:,3))
title("y as a function of Time")
xlabel("Time(s)")
ylabel("y")
```

```
subplot(4,1,4)
plot(time,statearray(:,4))
title("z as a function of Time")
xlabel("Time(s)")
ylabel("z")
%Function called in the ODE45 used given equations that needs to be
%integrated through
function derivatives = odefunction(tspan, init)
wdot = 9*init(1) + init(3); % We change this from part 1.a to 1.b
xdot = 4*init(1)*init(2)*init(3) - init(2).^2;
ydot = 2*init(1) - init(2) - 2*init(4);
zdot = init(2)*init(3) - init(3).^2 - 3*init(4).^3;
derivatives = [wdot; xdot; ydot; zdot];
end
```

Appendix 2

```
% Hannah Priddy, Chloe Zentner, Ben Reynolds, Jared Steffen
% ASEN 3801
% Lab 1
% Last Edited: 9/13/23
clear; close all; clc;
% initializing all variables, and constants
Initalm = .050; %kg
g = [0;0;9.81];%m/s^2
d = .02; %m
A = pi*(d/2).^2; %m^2
Cd = 0.6;
rho_Boulder = stdatmo(1655,0);%kg/m^3
X_inital= [0,0,0,0,20,-20];
options = odeset('Events',@myEventsFcn); %sets options for ODE45
%% Beginning of Part 2 B:
% which iterates through different wind vectors to see how wind changes final
position of the ball
%Pre-allocate
MagDispP2b = zeros(1,100);
xdispP2b = zeros(1,100);
min_landing = zeros(1,5);
WindSpeeds = 1:100;%m/s
for R = WindSpeeds
wind = [R;0;0]; %
xinitP2b = X_inital;
tspan = linspace(0,30);
[time, positionvecP2b] = ode45(@(t,x) objectEOM(t,x, rho_Boulder, Cd, A,
Initalm, g, wind), tspan, xinitP2b, options);
MagDispP2b(R) = norm(positionvecP2b(end,1:2));
xdispP2b(R) = positionvecP2b(end,1) ;
end
% Plots the trajectory of the Ball
```

```
figure(1)
plot3(positionvecP2b(:,1),positionvecP2b(:,2), -positionvecP2b(:,3))
xlabel('North/South')
ylabel('East/West')
zlabel('Up/Down')
legend('Trajectory')
title('2.b Trajectory of Ball')
grid on; grid minor;
%% Lab 1 part 2.c
%Plots different wind speeds by the total displacement of the ball
figure(2)
plot(WindSpeeds,MagDispP2b)
xlabel('Wind Speed[m/s]')
ylabel('Total Meters of Deflection[m]')
title('2.c Wind Speed VS Total Meters of Deflection')
grid on; grid minor;
%Plots the wind speeds versus the x displacement
figure(3)
plot(WindSpeeds,xdispP2b)
xlabel('Wind Speed[m/s]')
ylabel('X meters of Deflection[m]')
title('2.c Wind Speed VS X Meters of Deflection')
grid on; grid minor;
%% Lab 1 part 2.d
j = 1;
for i = 1:2000:8001
rhoP2d = stdatmo(i,0);
for R = WindSpeeds
windP2d = [R;0;0]; %
xinitP2d = X_inital;
tspan = linspace(0,30);
[time, positionvecP2d] = ode45(@(t,x) objectEOM(t,x, rhoP2d, Cd, A, Initalm, g,
windP2d), tspan, xinitP2d, options);
MagDispP2d(R) = norm(positionvecP2d(end,1:2));
xdispP2d(R) = positionvecP2d(end,1);
end
min_landing(j) = min(MagDispP2d);
j = j+1;
figure(4)
plot(WindSpeeds,MagDispP2d)
hold on
end
hold off
xlabel('Wind Speed[m/s]')
ylabel('Total Meters of Deflection[m]')
title('2.d Wind Speed VS Total Meters of Deflection for Various Altitudes')
legend('0m (Sea Level)','2000m','4000m','6000m','8000m (Mt
Everest)','Location','NorthWest')
grid on;grid minor;
```

```
%Plots the altitude versus the minimum landing distance
figure(5)
plot(1:2000:8001,min_landing)
ylabel('Minimum Landing Distance[m]')
xlabel('Altitude[m]')
title('2.d Minimum Landing Distance vs Various Altitudes')
grid on; grid minor;
%% Lab 1 part 2.e
%iterates through mass to see if we kept KE the same what would happen to
%The total distance traveled.
massP2e = linspace(.01,1);
for R = 1:100
wind = [0;0;0];
Ke_initial=.5*Initalm*norm(X_inital)^2;
vx =sqrt(((Ke_initial)/massP2e(R)));
xinitP2e = [0; 0; 0; 0; vx; -vx];
% Ke_final= .5*massP2e(R)*norm(xinit)^2;
tspan = linspace(0,30);
[time, positionvecP2e] = ode45(@(t,x) objectEOM(t,x, rho_Boulder, Cd, A,
massP2e(R), g, wind), tspan, xinitP2e, options);
MagP2e(R) = norm(positionvecP2e(end,:));
end
plot(massP2e, MagP2e)
ylabel('Total Distance Traveled (m)')
xlabel('Mass of Ball (Kg)')
title('Mass of Ball (Kg) vs Total Distance Traveled (m)')
%% Functions
%Creates the function that runs through ODE45 to get the position and
%velocity vectors
function xdot = objectEOM(~,x,rho,Cd,A,m,g,wind)
relvelocity = -wind + x(4:6);
airspeed = norm(relvelocity);
Dragscalar = -0.5*Cd*A*rho * airspeed^2;
Drag = Dragscalar*relvelocity/airspeed;
fgrav = m*g;
forces = (Drag+fgrav)/m;
xdot = [x(4:6); forces];
end
% Cretes the function to terminate ODE45 When the ball hits the ground
function [value,isterminal,direction] = myEventsFcn(~,y)
value = y(3);
isterminal = 1;
direction = 0;
end
```