

UNIVERSITY OF COLORADO - BOULDER

ASEN 2004 - VEHICLE DESIGN AND PERFORMANCE

LAB 3

---

# Water Bottle Rocket Final Executive Summary

---

*Authors:*

Cole MACPHERSON<sup>a</sup>

<sup>a</sup>SID: 108521329

*Instructors:*

Torin CLARK

Aaron JOHNSON

John MAH

April 29 2020



College of Engineering & Applied Science  
UNIVERSITY OF COLORADO **BOULDER**

## Contents

<b>Nomenclature</b>	<b>2</b>
<b>I Methods</b>	<b>3</b>
<b>II MATLAB Modeling</b>	<b>3</b>
<b>III Final Modified Rocket Design</b>	<b>3</b>
<b>IV Flight Predictions</b>	<b>4</b>
<b>V Lessons Learned</b>	<b>6</b>
<b>References</b>	<b>7</b>
<b>Appendix A: Code Flow Charts</b>	<b>8</b>
<b>Appendix B: MATLAB Code</b>	<b>12</b>
Main Script . . . . .	12
ode45 Function . . . . .	16
Sensitivity Analysis . . . . .	19
Error Ellipses . . . . .	26

## Nomenclature

$A_t$	=	area of throat
$C_D$	=	drag coefficient
$c_d$	=	discharge coefficient
$d_b$	=	diameter of the bottle
$d_t$	=	diameter of the bottle's throat
$g$	=	specific heat ratio
$\dot{m}$	=	mass flow rate
$m_w$	=	mass of water
$p$	=	air pressure in the bottle
$p_a$	=	ambient air pressure
$p_e$	=	exit air pressure
$p_{air}^i$	=	initial air pressure in the bottle
$T$	=	thrust
$T_{air}$	=	temperature of the air
$v$	=	volume of air in the bottle
$v_{air}^i$	=	initial volume of air in the bottle
$V_e$	=	exit velocity
$\rho_{air}$	=	density of the air
$\rho_e$	=	exit density
$\rho_w$	=	density of water
$\theta$	=	launch angle

## I. Methods

During the flight of a water bottle rocket, the rocket goes through 3 unique phases. The first phase is the water thrust stage. This stage occurs when the thrust of the rocket is solely provided by the expulsion of water from the bottle. During this phase, the water thrust, gravity, and drag are the only forces that act on the bottle throughout its flight. Utilizing this knowledge of the forces acting during this phase, a model can be created that will track key, changing, variables throughout the flight. These changing variables are the air pressure within the bottle and the mass of water in the rocket. The changing air pressure can be found relatively easy by utilizing (1).

$$\frac{p}{p_{air}^i} = \left( \frac{v_{air}^i}{v} \right)^g \quad (1)$$

The change in mass can be modelled using equation (2).

$$\dot{m} = -c_d A_t \sqrt{2\rho_w (p - p_a)} \quad (2)$$

With these two changing parameters found we can find the thrust using equation (3)

$$T = 2c_d A_t (p - p_a) \quad (3)$$

The second phase is the air thrust phase. This phase can be identified when the thrust force is created from the air in the bottle. During this phase, the only variable changing is the mass of the bottle as the air is expelled. The change in mass of the bottle can be shown by equation (4)

$$\dot{m} = -c_d \rho_e A_t V_e \quad (4)$$

The thrust equation from the first phase must now be adapted as the pressure in the bottle is no longer constant. The adaptations result in equation (5) for the thrust representation of the second phase.

$$T = -\dot{m} V_e + (p_a - p_e) A_t \quad (5)$$

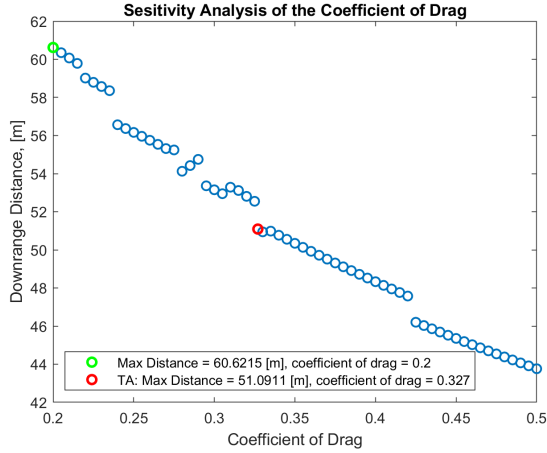
The final phase is the ballistic phase. The only forces acting during this phase are gravity and drag. There is no more thrust as all of the mass and air have been expelled at this point. There are no changing parameters and as such, the bottle rocket falls on a ballistic trajectory.

## II. MATLAB Modeling

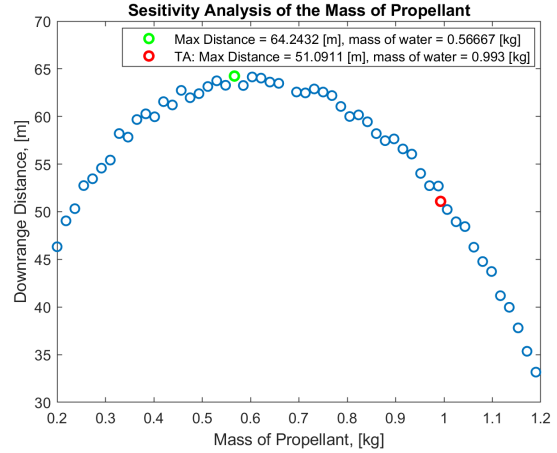
To validate the model code, variables were changed whose influence on the flight of the rocket was known ahead of the test. Variables such as coefficient of drag, gravity, the mass of the bottle, etc. When testing the coefficient of drag it was expected that the coefficient of drag was inversely related to the distance traveled. The same relationship was expected from both gravity and the mass of the bottle as well. After testing it was confirmed that the variables did indeed have an inverse relationship with the distance traveled. Further tests were conducted along the coding process such as changing the initial gage pressure in the bottle and increasing the launch angle. Gage pressure was expected to be directly related to the distance traveled and increasing the launch angle was expected to launch the rocket up very high and have a very small downrange distance. Both of these expected outcomes were found to be true for the model after testing. Based on these tests and more, the model was deemed to be accurate and correct and therefore could accurately model the flight of the rocket. The flow charts in Appendix A, figures 7 through 10, show the coding logic for the MATLAB files used for this lab.

## III. Final Modified Rocket Design

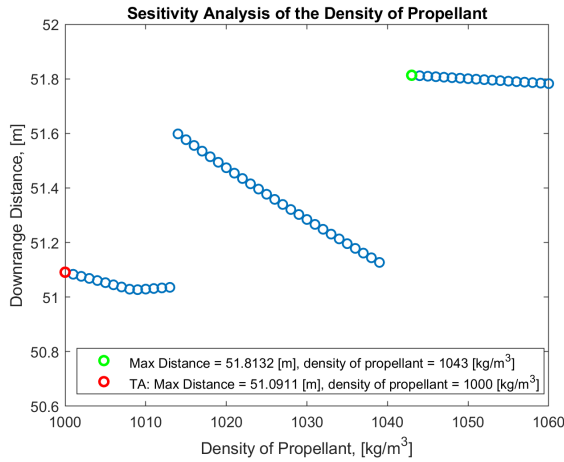
To come up with a final rocket design, a sensitivity analysis was performed to determine which of the 5 parameters was best to change to maximize distance traveled. Figures 1 thru 4 show how downrange distance was affected as by each parameter. Of the 5 parameters that could be changed, four were tested as the fifth, the temperature of the water, did not apply to the model that was created. Changing the mass of water used appeared to be the best parameter to alter as it offered the greatest gains from the TA baseline rocket.



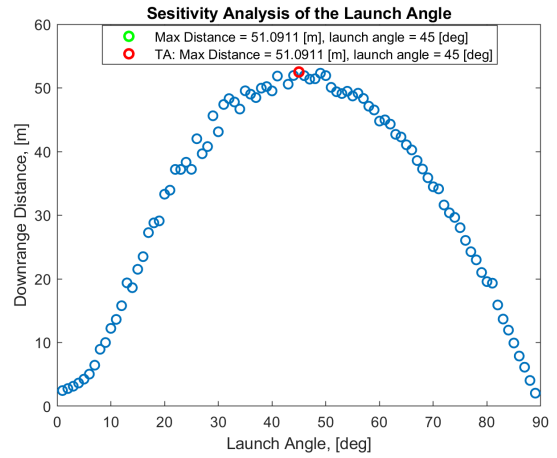
**Fig. 1 Sensitivity Analysis of the Coefficient of Drag**



**Fig. 2 Sensitivity Analysis of the Propellant Mass**



**Fig. 3 Sensitivity Analysis of the Propellant Density**



**Fig. 4 Sensitivity Analysis of the Launch Angle**

Based on the sensitivity analysis performed the launch angle was already in the optimal position for the maximum downrange distance and thus, that was not the parameter to change. When looking at how the density affects the maximum range, it was clear that the gains from increasing the density were extremely small. As such density was not going to be changed. When thinking about changing the coefficient of drag it was hard to believe a bottle rocket with a coefficient of drag of 0.2 or less was reasonable, and as such it's attainability was pulled into question moving forward. Between changing the coefficient of drag and propellant mass, the propellant mass was the clear choice as it offered an additional four meters to the max distance and the mass value required was reasonable. The new mass of propellant was to be 566.67 grams of water and that mass would take the rocket to a final distance of 64.2432 meters downrange in a no-wind environment. Based on the water having a density of  $1000 \frac{g}{L}$ , 566.67 grams of water would equate to 0.56667 liters. Since we are using 2-liter bottles this mass of water was deemed reasonable as it was not more than 2 liters.

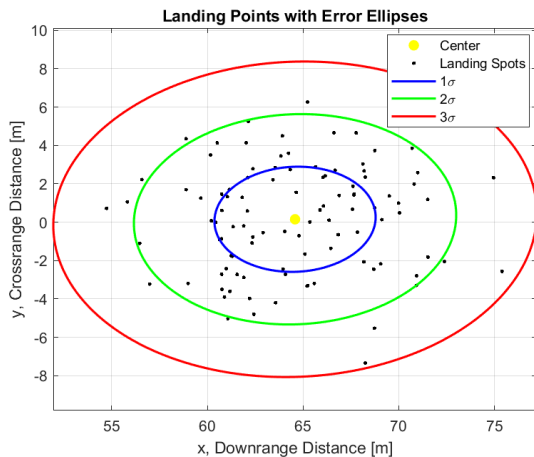
## IV. Flight Predictions

Normalized Error

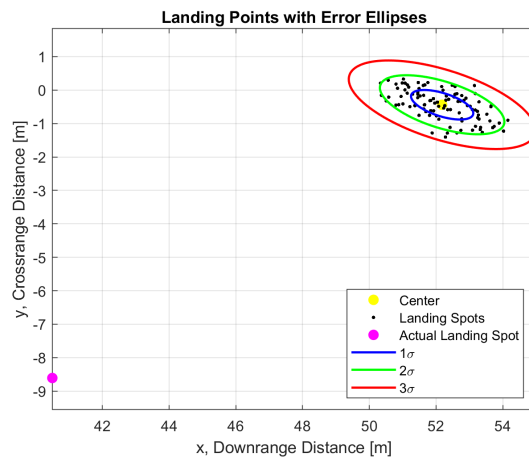
- $C_D = 0.327 \pm 0.01625$
- $d_t = 2.1 \pm 0.5$  cm
- $d_b = 10.5 \pm 0.5$  cm
- $p = 40.5 \pm 0.5$  psi

- $\theta = 45 \pm 1$  deg
- Random Error
- $m_w = 566.67 \pm 0.05$  g
- $\rho_{air} = 1.17358 \pm 0.05 \frac{kg}{m^3}$
- $p_a = 99640 \pm 5$  Pa
- $T_{air} = 22 \pm 0.5$  °C

When looking at errors, two types had to be considered, normalized and random error. Normal error is the error found through a normal distribution, which means it is likely that the value is very close to being accurate and has very small deviations from the actual value. Random error is where the error is truly not known and the certainty of the error can not be accurately determined as such a range is picked that the value could be within. The values associated with the normalized error were placed under this section because there is a human that is choosing the values and humans tend to be more precise when measuring. The coefficient of drag error was picked because of the MATLAB figure provided of the coefficient of drag and this is a value that was found by humans as such it is a normalized error. For the rest of the normalized errors, the errors associated with each were chosen based on the precision of the measuring instrument. Rulers and pressure gauges typically have a precision of 1 cm and 1 psi, respectively. Humans will most likely round to the nearest tick mark and as result could be off by 0.5 from the true value and is why 0.5 was chosen as the error for the bottle and throat diameter, and gauge pressure in the bottle. For the launch angle, the value was likely very precise as humans placed it. One degree is a fair error of the launch angle as the launch stand could have blocked the measuring device as it neared the desired value and therefore hindering accuracy. For the random error values, the mass is limited by the precision of the scale and it is most likely that the scale has a one decimal precision as such half of that, 0.05 g, is a reasonable error. Similarly to the mass of water, for the temperature of the air and air pressure, these readings are found from an instrument that measures the values. Thus the error is half of the precision of the measurement device. For the temperature and air pressure, this is 0.5 °C and 5 Pa, respectively. For air density, the precision of the ambient pressure was considered as it is used to calculate density. From this calculation the  $0.05 \frac{kg}{m^3}$  value was found. The final rocket design was given random wind values from 0 to 10 mph



**Fig. 5 Error Ellipses for Designed Rocket**



**Fig. 6 Error Ellipses for TA Gold Rocket**

Figures 5 and 6 show the error ellipses based on the errors stated from the beginning of this section. The two figures display the landing point of 100 simulations of the rocket launch with the errors included. The model did not accurately predict the landing spot of the gold rocket and it is believed that the fins were not perfect and caused a deviation cross-range that the model can't predict. Along with the errors ellipses for three standard deviations. The third standard deviation ellipse of the final design rocket has a semi-major axis of 12.6353 meters, a semi-minor axis of 8.2252 meters, and an area of 326.4994 square meters.

## **V. Lessons Learned**

Through this lab, a greater understanding of the complexity of models was gained. These models are hard to design and take a while to fabricate, as such time management is an important aspect of the fabrication process to ensure deadlines are met in a timely manner. Not only is time management important but also an understanding of every type of model that could be created is important. There could be a model that is easier to create but less accurate or a model that is extremely accurate but difficult to create and the model should be picked based on what requirements. In the future, a different model could be derived to compare to the current model. Along with these two items, it is important to fully understand the material that you are modeling as it will help troubleshoot and the overall creation of the model. All in all, this lab was a fun and challenging learning experience that required a full understanding of the material and great time management skills, and in the end, improved these skills and knowledge of the topic.

## References

- [1] ASEN 2004 Lab Document  
Anderson, Allie, "ASEN 2012 Project 2 Bottle Rocket Modeling.pdf," Spring 2020. University of Colorado Boulder.
- [2] ASEN 2004 Lab Document  
Jonhson, Aaron, "ASEN 2004 Water Bottle Rocket Lab.pdf," Spring 2020. University of Colorado Boulder.
- [3] ASEN 2004 Book  
Sellers. *Understanding Space: An Introduction to Astronautics*, thrid ed., The McGraw-Hill Companies, Inc., 2005.



## Appendix A: Code Flow Charts

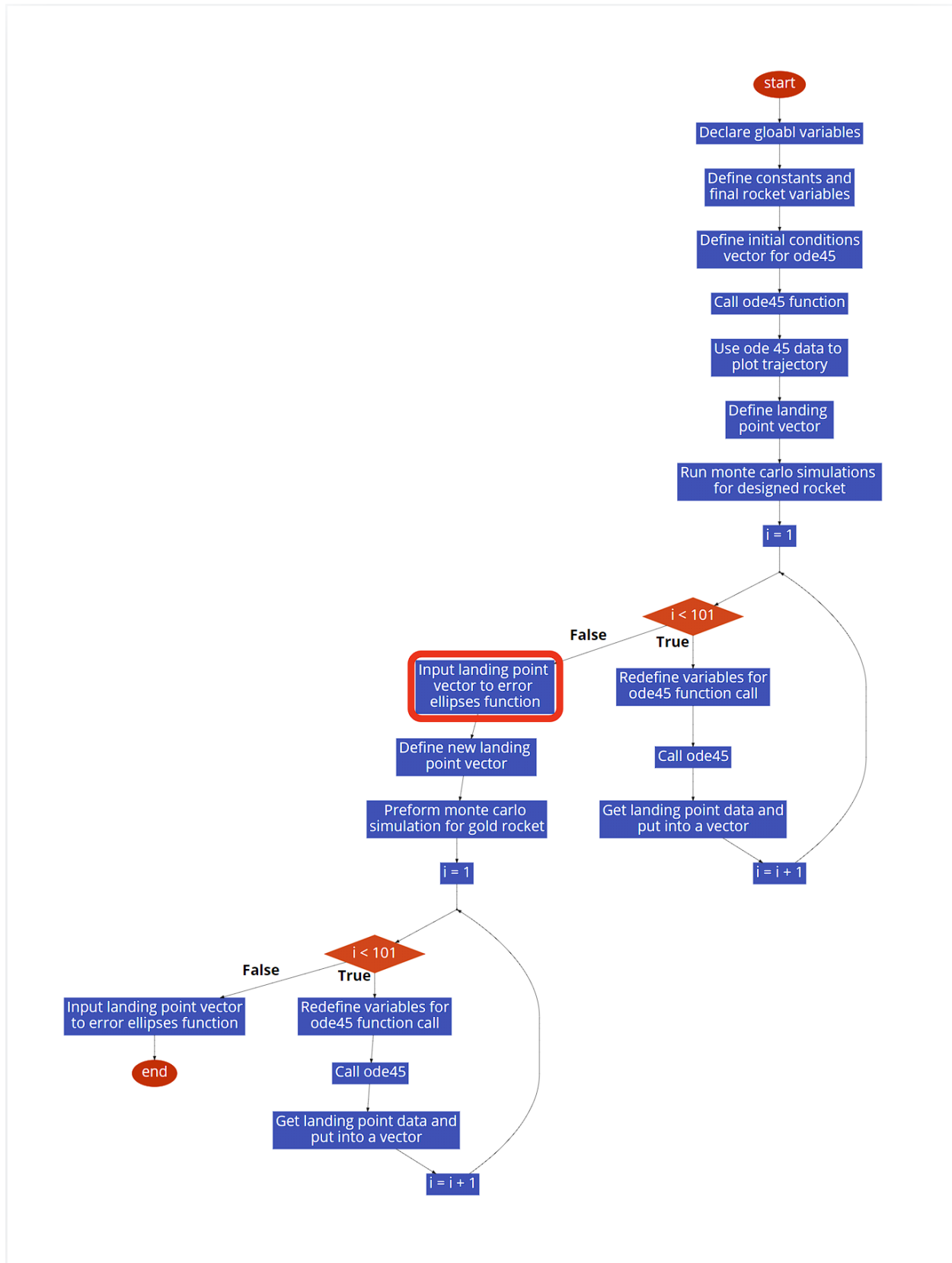
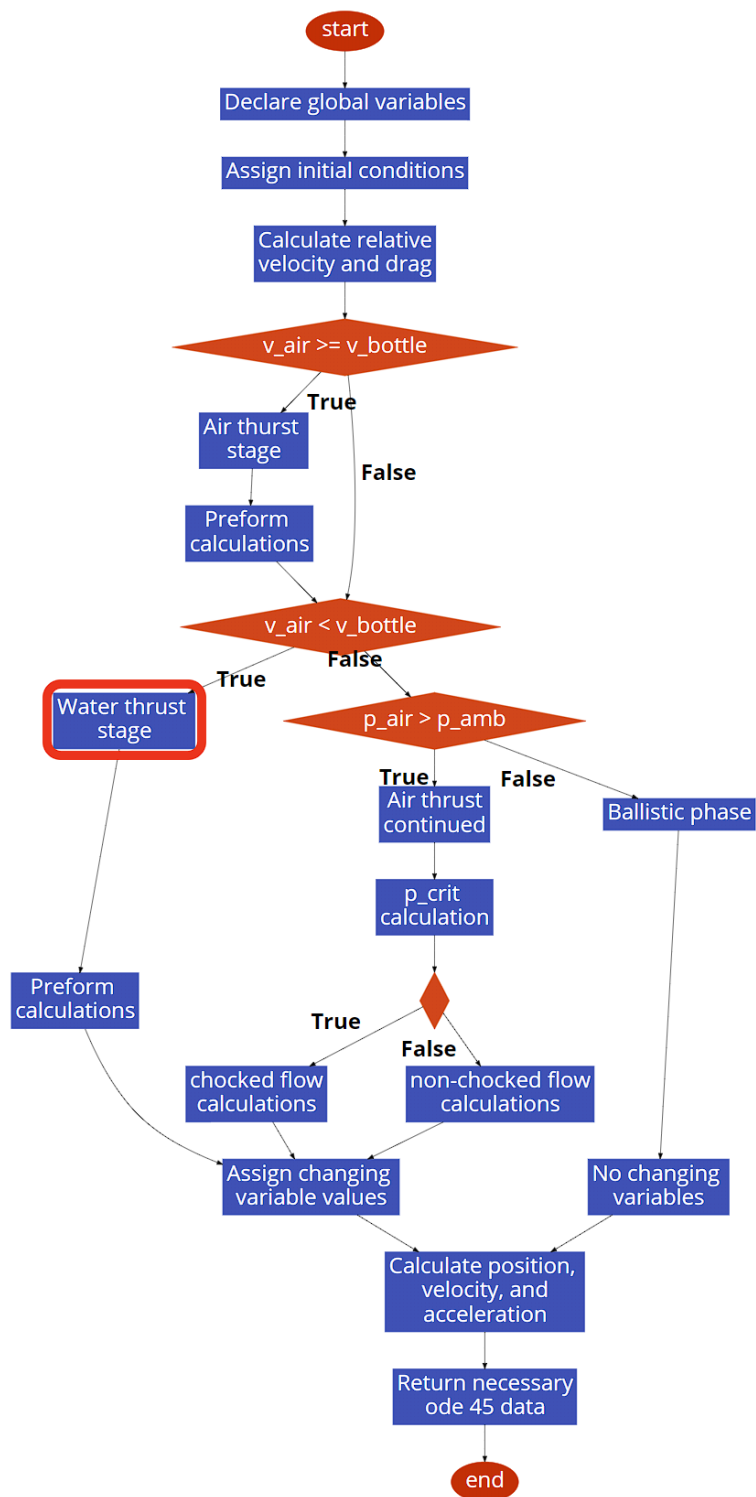
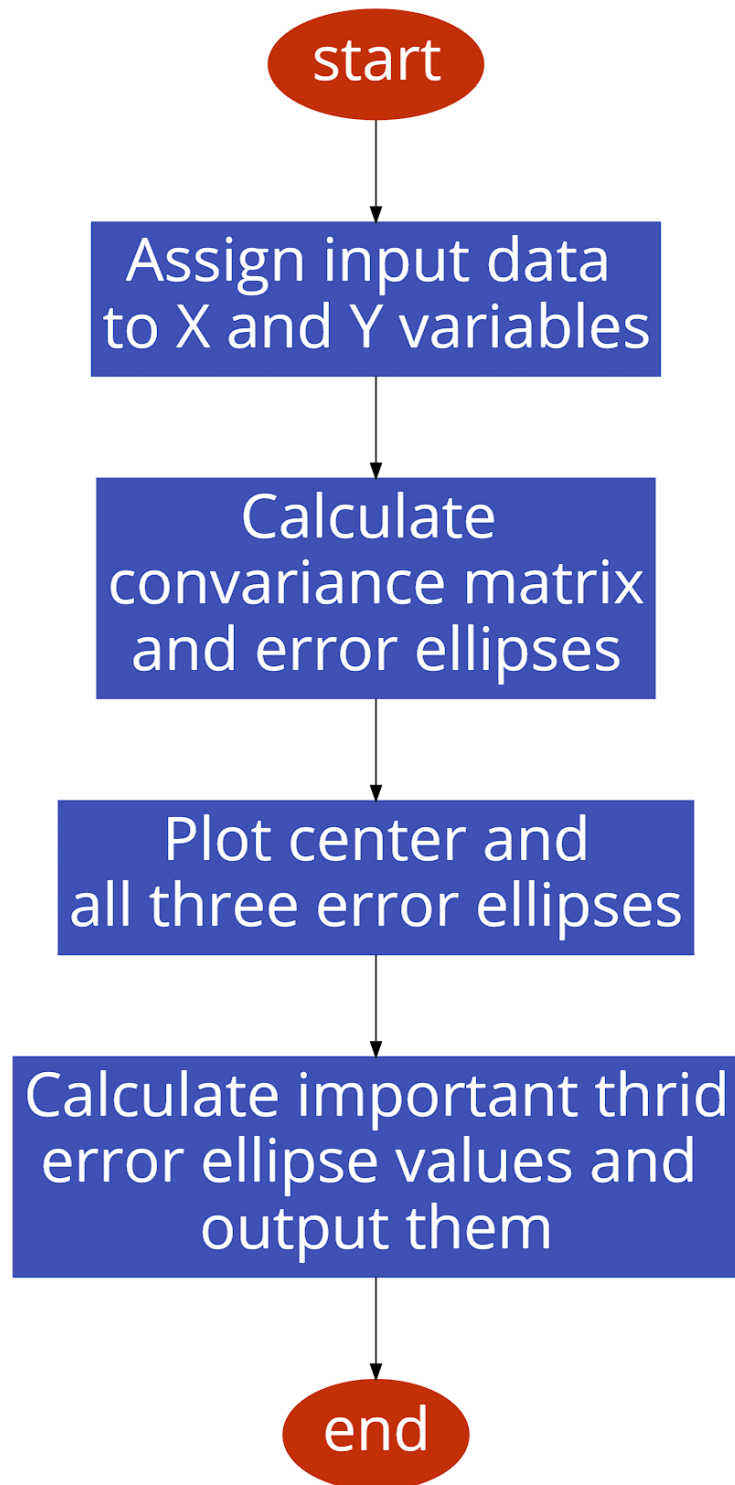


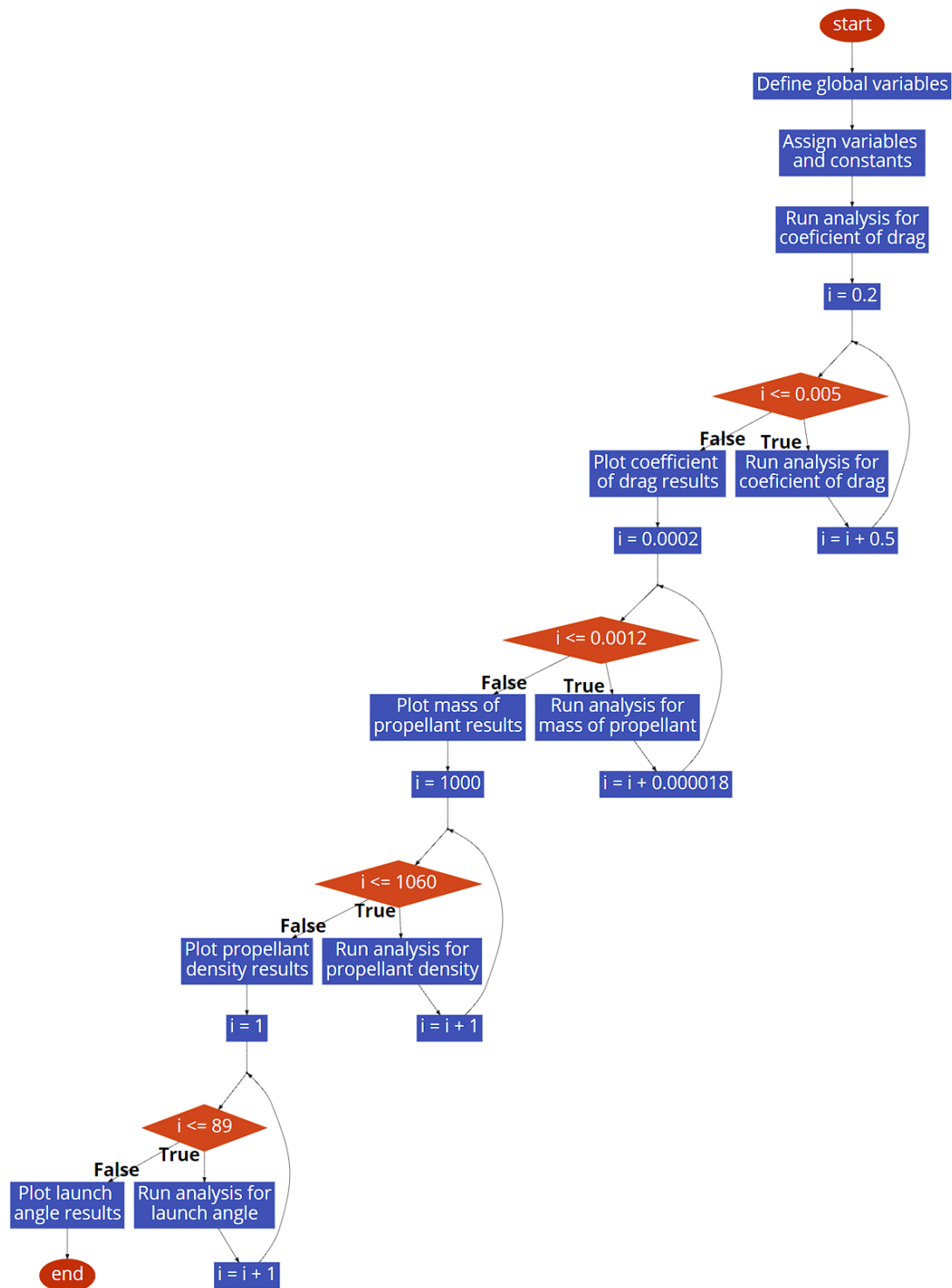
Fig. 7 Flow Chart for the Main Script



**Fig. 8 Flow Chart for the ode45 Function**



**Fig. 9 Flow Chart for the Error Ellipse Function**



**Fig. 10 Flow Chart for the Sensitivity Analysis Function**

## Appendix B: MATLAB Code

### Main Script

```
1 %% ASEN 2012 Project 2 (Water Bottle Rocket)
2
3 % Authors
4 %
5 % 1) Cole MacPherson
6 % 2) Ankrit Upreti
7 %
8 % https://www.timeanddate.com/weather/usa/boulder/historic?month=3&year=2020
9
10 %% Housekeeping
11
12 clc;
13 clear;
14 close all;
15
16 %% Declare Global Variables and Constants/Verification Constants
17
18 % Gloabl Variables Declaration
19
20 global g discharge_coeff rho_air_amb volume_bottle p_amb gamma rho_water ...
21     diameter_throat area_throat area_bottle diameter_bottle gas_const_air ...
22     mass_bottle cd p_gage_0 volume_water_0 temp_air_0 vel_0 theta_0 x_0 y_0 ...
23     z_0 vel0_x vel0_y vel0_z l_stand mass_air_0 mass_rocket_0 heading_0 ...
24     volume_air_0 p_abs_0 tspan windVelG windVelA;
25
26 % Constants
27
28 g = 9.81; % acceleration due to gravity [m/s^2]
29 discharge_coeff = 0.8; % discharge coefficient
30 rho_air_amb = 1.17358; % ambient air density [kg/m^3] GOLD ROCKET -> 1.17358 [kg/m^3] TA ...
    BASELINE -> 1.23478
31 volume_bottle = 0.002; % volume of the empty bottle [m^3]
32 p_amb = 99640; % atmospheric pressure [pa] GOLD ROCKET -> 99640 [pa] TA BASELINE -> 101592
33 gamma = 1.4; % ratio of specific heats for heat
34 rho_water = 1000; % density of water [kg/m^2]
35 diameter_throat = 2.1 / 100; % diameter of throat [m]
36 diameter_bottle = 10.5 / 100; % diameter of bottle [m]
37 area_throat = (1/4) * pi * (diameter_throat)^2; % area of throat [m^2]
38 area_bottle = (1/4) * pi * (diameter_bottle)^2; % area of bottle [m^2]
39 gas_const_air = 287; % gas constant of air [J/(kg*K)]
40 mass_bottle = 0.126; % mass of the empty bottle w/ cone and fins [kg]
41 cd = 0.327; % coefficient of drag
42 l_stand = 0.5; % length of test stand [m]
43 tspan = [0, 5]; % integration time span [s]
44
45 % Max Distance Constants
46
47 p_gage_0 = 40.5 * 6894.75729; % initial gage pressure of air in bottle [pa] MAXIMUM ...
    DISTANCE -> 4.136854374e+05 -> 60 psi // GOLD ROCKET -> 40.5 psi
48 p_abs_0 = p_gage_0 + p_amb; % absolute pressure of air inside the bottle [pa]
49 volume_water_0 = 0.0005833; % initial volume of water inside bottle [m^3]
50 volume_air_0 = volume_bottle - volume_water_0; % initial volume of air inside the bottle [...
    m^3]
51 temp_air_0 = 295.15; % initial temperature of air [K] GOLD ROCKET -> 295.15 TA BASELINE -> ...
    283.15
52 vel_0 = 0; % initial velocity of bottle rocket [m/s]
53 vel0_x = 0; % initial velocity of bottle rocket in the x direction [m/s]
54 vel0_y = 0; % initial velocity of bottle rocket in the y direction [m/s]
55 vel0_z = 0; % initial velocity of bottle rocket in the z direction [m/s]
56 theta_0 = 45; % initial angle of bottle rocket [degrees] MAXIMUM DISTANCE -> 40
57 x_0 = 0; % initial downrange distance [m]
58 y_0 = 0; % initial crossrange distance [m]
```

```

59 z_0 = 0.25; % initial verital height [m]
60 windTheta = 0;
61 windSPDG = 0; % wind velocity [m/s]
62 windSPDA = 0;
63 windDir = [cosd(windTheta) sind(windTheta) 0]; % wind velocity direction [m/s]
64 windVelG = [windDir(1)*windSPDG, windDir(2)*windSPDG, windDir(3)*windSPDG];
65 windVelA = [windDir(1)*windSPDA, windDir(2)*windSPDA, windDir(3)*windSPDA];
66 heading_0 = [cosd(theta_0), 0, sind(theta_0)]; % initial heading of bottle rocket [x,y]
67 mass_air_0 = (p_abs_0 * volume_air_0) / (gas_const_air * temp_air_0); % initial mass of air...
    [kg]
68 mass_rocket_0 = mass_bottle + (rho_water * volume_water_0) + mass_air_0; % total mass of ...
    the bottle rocket [kg]
69
70
71 %% Calculations
72
73 initial_conditions = [x_0, y_0, z_0, vel0_x, vel0_y, vel0_z, mass_rocket_0, mass_air_0, ...
    volume_air_0]; % initial conditions
74
75 [time, data] = ode45('ode45func2',tspan,initial_conditions);
76
77 %[thrust, phase1_end_i, phase2_end_i] = thrustvecfunc2(time,data);
78
79 %% Plot trajectory of bottle rocket
80 i = 1;
81 while data(i,3) >= 0
82     i = i+1;
83 end
84
85 max_distance_x = (data(i-1,1)+data(i,1)) / 2;
86 max_distance_y = (data(i-1,2)+data(i,2)) / 2;
87 max_distance_z = 0;
88 max_distances = [max_distance_x,max_distance_y,max_distance_z];
89 distanceFromLaunch = sqrt(max_distance_x^2+max_distance_y^2);
90
91 x_limit = max_distances(1)*1.05;
92 y_limit = max_distances(2)*1.05;
93 z_limit = max(data(:,3))*1.05;
94
95 j = 1;
96 while data(j,3) < max(data(:,3))
97     j = j+1;
98     max_height = [data(j,1) data(j,2) data(j,3)];
99 end
100
101 figure(1)
102 plot3(x_0,y_0,z_0,'or',max_distances(1),max_distances(2), ...
103     max_distances(3),'*r',max_height(1),max_height(2),max_height(3),'+r', ...
104     data(1:i,1),data(1:i,2),data(1:i,3),'b','LineWidth',2);
105 xlim([x_0 x_limit]);
106 if y_limit > 0
107     ylim([y_0 y_limit]);
108 elseif y_limit < 0
109     ylim([y_limit y_0]);
110 end
111 zlim([z_0-0.3 z_limit]);
112 title('Rocket Trajectory');
113 xlabel('Downrange Distance, [m]');
114 ylabel('Crossrange Distance, [m]');
115 zlabel('Vertical Distance, [m]');
116 legend(['Launch Stand Location'], ['Maximum Downrange Distance, ' num2str(max_distances(1)) ...
    ' [m]'], ['Maximum Height, ' num2str(max_height(3)) ' [m]'], 'location','northoutside');
117 grid on;
118
119 %% Montecarlo Error Ellipses
120
121 xyCoorSim = zeros(100,2);
122

```

```

123 for i = 1:101
124
125     rho_air_amb = 1.17358 + (0.1*rand - 0.05); % ambient air density [kg/m^3] GOLD ROCKET ...
        -> 1.17358 [kg/m^3] TA BASELINE -> 1.23478
126     p_amb = 99640 + (1000*rand - 500); % atmospheric pressure [pa] GOLD ROCKET -> 99640 [pa...
        ] TA BASELINE -> 101592
127     cd = 0.327 + 0.004*randn; % coefficient of drag
128     diameter_throat = (2.1 / 100) + 0.0005*randn; % diameter of throat [m]
129     diameter_bottle = (10.5 / 100) + 0.0005*randn; % diameter of bottle [m]
130     area_throat = (1/4) * pi * (diameter_throat)^2; % area of throat [m^2]
131     area_bottle = (1/4) * pi * (diameter_bottle)^2; % area of bottle [m^2]
132
133     p_gage_0 = (40.5 + (1*rand - 0.5)) * 6894.75729; % initial gage pressure of air in ...
        bottle [pa] MAXIMUM DISTANCE -> 4.136854374e+05 -> 60 psi // GOLD ROCKET -> 40.5 ...
        psi
134     p_abs_0 = p_gage_0 + p_amb; % absolute pressure of air inside the bottle [pa]
135     volume_water_0 = 0.0005833 + (0.0000005*randn); % initial volume of water inside ...
        bottle [m^3] MAXIMUM DISTANCE -> 5.25e-04 [m^3]
136     volume_air_0 = volume_bottle - volume_water_0; % initial volume of air inside the ...
        bottle [m^3]
137     temp_air_0 = 295.15 + (2*rand - 1); % initial temperature of air [K] GOLD ROCKET -> ...
        295.15 TA BASELINE -> 283.15
138     theta_0 = 45 + randn; % initial angle of bottle rocket [degrees] MAXIMUM DISTANCE -> ...
        40
139     windTheta = 360*rand;
140     windSPDG = 10*rand; % wind ground velocity [m/s]
141     windSPDA = 10*rand; % wind aloft velocity [m/s]
142     windDir = [cosd(windTheta) sind(windTheta) 0]; % wind velocity direction [m/s]
143     windVelG = [windDir(1)*windSPDG, windDir(2)*windSPDG, windDir(3)*windSPDG];
144     windVelA = [windDir(1)*windSPDA, windDir(2)*windSPDA, windDir(3)*windSPDA];
145     heading_0 = [cosd(theta_0), 0, sind(theta_0)]; % initial heading of bottle rocket [x,y]
146     mass_air_0 = (p_abs_0 * volume_air_0) / (gas_const_air * temp_air_0); % initial mass of...
        air [kg]
147     mass_rocket_0 = mass_bottle + (rho_water * volume_water_0) + mass_air_0; % total mass ...
        of the bottle rocket [kg]
148
149
150     initial_conditions = [x_0, y_0, z_0, vel0_x, vel0_y, vel0_z, mass_rocket_0, mass_air_0,...
        volume_air_0]; % initial conditions
151
152     [timeSim, dataSim] = ode45('ode45func2',tspan,initial_conditions);
153
154     j = 1;
155     while dataSim(j,3) >= 0
156         j = j+1;
157     end
158     j = j-1;
159
160     xyCoorSim(i,:) = [(dataSim(j,1)+dataSim(j+1,1))/2, (dataSim(j,2)+dataSim(j+1,2))/2];
161
162 end
163
164 error_ellipses(xyCoorSim,0);
165
166 %% Montecarlo Error Ellipses GOLD ROCKET
167
168 xyCoorSim_g = zeros(100,2);
169
170 for i = 1:101
171
172     rho_air_amb = 1.17358 + (0.1*rand - 0.05); % ambient air density [kg/m^3] GOLD ROCKET ...
        -> 1.17358 [kg/m^3] TA BASELINE -> 1.23478
173     p_amb = 99640 + (1000*rand - 500); % atmospheric pressure [pa] GOLD ROCKET -> 99640 [pa...
        ] TA BASELINE -> 101592
174     cd = 0.327 + 0.004*randn; % coefficient of drag
175     diameter_throat = (2.1 / 100) + 0.0005*randn; % diameter of throat [m]
176     diameter_bottle = (10.5 / 100) + 0.0005*randn; % diameter of bottle [m]
177     area_throat = (1/4) * pi * (diameter_throat)^2; % area of throat [m^2]

```

```

178 area_bottle = (1/4) * pi * (diameter_bottle)^2; % area of bottle [m^2]
179
180 p_gage_0 = (40.5 + (1*rand - 0.5)) * 6894.75729; % initial gage pressure of air in ...
    bottle [pa] MAXIMUM DISTANCE -> 4.136854374e+05 -> 60 psi // GOLD ROCKET -> 40.5 ...
    psi
181 p_abs_0 = p_gage_0 + p_amb; % absolute pressure of air inside the bottle [pa]
182 volume_water_0 = 0.000993 + (0.0000005*randn); % initial volume of water inside bottle...
    [m^3] MAXIMUM DISTANCE -> 5.25e-04 [m^3]
183 volume_air_0 = volume_bottle - volume_water_0; % initial volume of air inside the ...
    bottle [m^3]
184 temp_air_0 = 295.15 + (2*rand - 1); % initial temperature of air [K] GOLD ROCKET -> ...
    295.15 TA BASELINE -> 283.15
185 theta_0 = 45 + randn; % initial angle of bottle rocket [degrees] MAXIMUM DISTANCE -> ...
    40
186 windTheta = (136+(22.5*rand-11.25)) + 1.5*randn;
187 windSPDG = 0 + (2*0.44704*rand-0.44704); % wind ground velocity [m/s]
188 windSPDA = 0.44704 + (2*0.44704*rand-0.44704); % wind aloft velocity [m/s]
189 windDir = [cosd(windTheta) sind(windTheta) 0]; % wind velocity direction [m/s]
190 windVelG = [windDir(1)*windSPDG, windDir(2)*windSPDG, windDir(3)*windSPDG];
191 windVelA = [windDir(1)*windSPDA, windDir(2)*windSPDA, windDir(3)*windSPDA];
192 heading_0 = [cosd(theta_0), 0, sind(theta_0)]; % initial heading of bottle rocket [x,y]
193 mass_air_0 = (p_abs_0 * volume_air_0) / (gas_const_air * temp_air_0); % initial mass of...
    air [kg]
194 mass_rocket_0 = mass_bottle + (rho_water * volume_water_0) + mass_air_0; % total mass ...
    of the bottle rocket [kg]
195
196
197 initial_conditions = [x_0, y_0, z_0, vel0_x, vel0_y, vel0_z, mass_rocket_0, mass_air_0,...
    volume_air_0]; % initial conditions
198
199 [timeSim_g, dataSim_g] = ode45('ode45func2',tspan,initial_conditions);
200
201 j = 1;
202 while dataSim_g(j,3) > 0
203     j = j+1;
204 end
205 j = j-1;
206
207 xyCoorSim_g(i,:) = [(dataSim_g(j,1)+dataSim_g(j+1,1))/2, (dataSim_g(j,2)+dataSim_g(j...
    +1,2))/2];
208
209 end
210
211 error_ellipses(xyCoorSim_g,1);

```



## ode45 function

```

1 % Differential Equations for the bottle rocket
2
3 function datafunc = ode45func2(t,conditions_0)
4
5 %% Global Variables
6 global g discharge_coeff rho_air_amb volume_bottle p_amb gamma rho_water ...
7     area_throat area_bottle gas_const_air cd x_0 z_0 l_stand mass_air_0 ...
8     heading_0 volume_air_0 p_abs_0 windVelG windVelA;
9
10 %% Initial Conditions
11 x = conditions_0(1); % x position
12 y = conditions_0(2); % y position
13 z = conditions_0(3); % z position
14 vel_x = conditions_0(4); % Velocity downrange direction
15 vel_y = conditions_0(5); % Velocity crossrange direction
16 vel_z = conditions_0(6); % Velocity vertical direction
17 mass_rocket = conditions_0(7); % Mass of the bottle rocket
18 mass_air = conditions_0(8); % mass of air in the bottle rocket
19 volume_air = conditions_0(9); % volume of air inside the bottle rocket
20
21 %% Calculations
22 windVel = (windVelG + windVelA) / 2;
23 if sqrt((x - x_0)^2 + (z - z_0)^2) > l_stand
24     vel_rel = [vel_x-windVel(1), vel_y-windVel(2), vel_z-windVel(3)];
25     vel_rel_tot = sqrt((vel_x + windVel(1))^2 + (vel_y + windVel(2))^2 + (vel_z + ...
26         windVel(3))^2);
27 else
28     vel_rel = [vel_x, vel_y, vel_z];
29     vel_rel_tot = sqrt((vel_x)^2 + (vel_y)^2 + (vel_z)^2);
30 end
31 drag = (rho_air_amb / 2) * (vel_rel_tot)^2 * cd * area_bottle; % find drag using equation 2...
32     [N]
33 if volume_air >= volume_bottle % if the volume of the air is greater than or equal to ...
34     the volume of the bottle (signifies the end of the water thrust stage)
35
36     p_water_thrust_end = p_abs_0 * ...
37         ((volume_air_0 / volume_bottle)^(gamma)); % pressure at the end of the water ...
38         thrust stage, using equation 13
39
40     p_air_thrust_stage = p_water_thrust_end * ...
41         ((mass_air / mass_air_0)^(gamma)); % pressure of air during air thrust stage, ...
42         using equation 14
43
44     rho_air_thrust_stage = mass_air / volume_bottle; % denstiy of air at ...
45     p_air_thrust_stage, using equation 15
46     temp_air_thrust_stage = p_air_thrust_stage / ...
47         (rho_air_thrust_stage * gas_const_air); % temperature of air at ...
48     p_air_thrust_stage, using equation 15
49 end
50
51 if volume_air < volume_bottle % if the volume of air is less than the volume of the ...
52     bottle (signifies the bottle rocket is still in the water thrust stage)
53
54     p_water_thrust_stage = p_abs_0 * (volume_air_0 / volume_air)^(gamma); % pressure in...
55     the bottle, using equation 3
56
57     thrust = 2 * discharge_coeff * area_throat * ...
58         (p_water_thrust_stage - p_amb); % thrust of the bottle rocket [N], using ...
59     equation 5
60
61     dvolumeair_dt = discharge_coeff * area_throat * sqrt((2 / rho_water) * ...

```

```

55     (p_abs_0 * ((volume_air_0 / volume_air)^(gamma)) - p_amb)); % change in volume ...
56     over change in time, using equation 9
57     dmasstot_dt = - discharge_coeff * rho_water * area_throat * ...
58     sqrt((2 * (p_water_thrust_stage - p_amb)) / rho_water); % change in the total ...
59     mass of the bottle rocket over change in time, using equation 10
60     dmassair_dt = 0; % change in mass of air over change in time, using equation 24 (...
61     zero for the water thrust stage)
62 elseif p_air_thrust_stage > p_amb % if p_air_thrust_stage is greater than p_amb (...
63     signifies the bottle rocket is in the air thrust phase)
64     p_critical = p_air_thrust_stage * ...
65     ((2 / (gamma + 1))^(gamma / (gamma - 1))); % critical pressure in the bottle, ...
66     using equation 16
67     if p_critical > p_amb % if p_critical is greater than p_amb (checking for choked ...
68     flow)
69         mach_exit = 1; % definition of choked flow
70
71         temp_exit = (2 / (gamma + 1)) * temp_air_thrust_stage; % exit temperature, using ...
72         equation 18
73         p_exit = p_critical; % exit pressure equals critical pressure if at choked flow,...
74         using equation 18
75         rho_exit = p_exit / (gas_const_air * temp_exit); % exit density, using equation ...
76         18
77         vel_exit = sqrt(gamma * gas_const_air * temp_exit); % exit velocity, using ...
78         equation 17
79     else % if not choked flow
80         mach_exit = sqrt((2 / (gamma - 1)) * ...
81         (((p_air_thrust_stage / p_amb)^((gamma - 1) / gamma)) - 1)); % exit mach ...
82         number, using equation 19
83         temp_exit = temp_air_thrust_stage / (1 + ((gamma - 1) / 2) * ...
84         (mach_exit)^(2)); % exit temperature, using equation 20
85         p_exit = p_amb; % exit pressure equals ambient pressure, using equation 20
86         rho_exit = p_exit / (gas_const_air * temp_exit); % exit density, using equation...
87         20
88         vel_exit = mach_exit * sqrt(gamma * gas_const_air * temp_exit); % exit velocity...
89         , using equation 21
90     end
91     dmassair_dt = -1 * ...
92     (discharge_coeff * rho_exit * area_throat * vel_exit); % change in mass of the ...
93     air in the rocket over change in time, using equation 23
94     thrust = (-1 * dmassair_dt * vel_exit) + (p_amb - p_exit) * ...
95     area_throat; % thrust of the rocket, using equation 22
96     dmasstot_dt = dmassair_dt; % change in total mass of the bottle rocket over time, ...
97     using equation 24
98     dvolumeair_dt = 0; % change in volme of air over change in time (zero because the ...
99     voulme of air no longer changes since the bottle rocket is in air thrust stage)
100 else % if the bottle rocket is in stage 3 flight, no thrust
101
102     thrust = 0; % no thrust in stage 3
103     dvolumeair_dt = 0; % volume of air is not changing in stage 3
104     dmasstot_dt = 0; % mass of bottle rocket is not changing in stage 3
105     dmassair_dt = 0; % mass of air is not changing in stage 3
106

```

```

107     end
108
109     % x, y, and z components of velocity
110     dx_dt = vel_x;
111     dy_dt = vel_y;
112     dz_dt = vel_z;
113
114     if sqrt((x - x_0)^2 + (z - z_0)^2) < l_stand % if the bottle rocket hasn't left the ...
        test stand
115
116         % initial headings, component wise
117         x_height = heading_0(1);
118         y_height = heading_0(2);
119         z_height = heading_0(3);
120
121     else % if the bottle rocket has left the test stand
122
123         % heading, component wise
124         x_height = vel_rel(1) / abs(vel_rel_tot);
125         y_height = vel_rel(2) / abs(vel_rel_tot);
126         z_height = vel_rel(3) / abs(vel_rel_tot);
127
128     end
129
130     % acceleration in each component
131     dvelx_dt = (thrust - drag) * (x_height / mass_rocket);
132     dvely_dt = (thrust - drag) * (y_height / mass_rocket);
133     dvelz_dt = (thrust - drag) * (z_height / mass_rocket) - g;
134
135     % differentials for ode45
136     datafunc(1) = dx_dt;
137     datafunc(2) = dy_dt;
138     datafunc(3) = dz_dt;
139     datafunc(4) = dvelx_dt;
140     datafunc(5) = dvely_dt;
141     datafunc(6) = dvelz_dt;
142     datafunc(7) = dmasstot_dt;
143     datafunc(8) = dmassair_dt;
144     datafunc(9) = dvolumeair_dt;
145     datafunc = datafunc'; % get equationResults into a column vec for ode45
146
147 end

```

## Sensitivity Analysis

```

1 %% Housekeeping
2
3 clc;
4 clear;
5 close all;
6
7 %% Constants
8
9 % Gloabl Variables Declaration
10
11 global g discharge_coeff rho_air_amb volume_bottle p_amb gamma rho_water ...
12     diameter_throat area_throat area_bottle diameter_bottle gas_const_air ...
13     mass_bottle cd p_gage_0 volume_water_0 temp_air_0 vel_0 theta_0 x_0 y_0 ...
14     z_0 vel0_x vel0_y vel0_z l_stand mass_air_0 mass_rocket_0 heading_0 ...
15     volume_air_0 p_abs_0 tspan windVelG windVelA;
16
17 % Constants
18
19 g = 9.81; % acceleration due to gravity [m/s^2]
20 discharge_coeff = 0.8; % discharge coefficient
21 rho_air_amb = 1.17358; % ambient air density [kg/m^3] GOLD ROCKET -> 1.17358 [kg/m^3] TA ...
    BASELINE -> 1.23478
22 volume_bottle = 0.002; % volume of the empty bottle [m^3]
23 p_amb = 99640; % atmospheric pressure [pa] GOLD ROCKET -> 99640 [pa] TA BASELINE -> 101592
24 gamma = 1.4; % ratio of specific heats for heat
25 rho_water = 1000; % density of water [kg/m^3]
26 diameter_throat = 2.1 / 100; % diameter of throat [m]
27 diameter_bottle = 10.5 / 100; % diameter of bottle [m]
28 area_throat = (1/4) * pi * (diameter_throat)^2; % area of throat [m^2]
29 area_bottle = (1/4) * pi * (diameter_bottle)^2; % area of bottle [m^2]
30 gas_const_air = 287; % gas constant of air [J/(kg*K)]
31 mass_bottle = 0.126; % mass of the empty bottle w/ cone and fins [kg]
32 cd = 0.327; % coefficient of drag
33 l_stand = 0.5; % langth of test stand [m]
34 tspan = [0, 5]; % integration time span [s]
35
36 % Max Distance Constants
37
38 p_gage_0 = 40.5 * 6894.75729; % initial gage pressure of air in bottle [pa] MAXIMUM ...
    DISTANCE -> 4.136854374e+05 -> 60 psi // GOLD ROCKET -> 40.5 psi
39 p_abs_0 = p_gage_0 + p_amb; % absolute pressure of air inside the bottle [pa]
40 volume_water_0 = 0.000993; % intitial volume of water inside bottle [m^3] MAXIMUM DISTANCE ...
    -> 5.25e-04 [m^3]
41 volume_air_0 = volume_bottle - volume_water_0; % intitial volume of air inside the bottle [...
    m^3]
42 temp_air_0 = 295.15; % initial temperature of air [K] GOLD ROCKET -> 295.15 TA BASELINE -> ...
    283.15
43 vel_0 = 0; % initial velocity of bottle rocket [m/s]
44 vel0_x = 0; % initial velocity of bottle rocket in the x direction [m/s]
45 vel0_y = 0; % initial velocity of bottle rocket in the y direction [m/s]
46 vel0_z = 0; % initial velocity of bottle rocket in the z direction [m/s]
47 theta_0 = 45; % intitial angle of bottle rocket [degrees] MAXIMUM DISTANCE -> 40
48 x_0 = 0; % initial downrange distance [m]
49 y_0 = 0; % initial crossrange distance [m]
50 z_0 = 0.25; % initial verital height [m]
51 windTheta = 0;
52 windSPDG = 0; % wind velocity [m/s]
53 windSPDA = 0;
54 windDir = [cosd(windTheta) sind(windTheta) 0]; % wind velocity direction [m/s]
55 windVelG = [windDir(1)*windSPDG, windDir(2)*windSPDG, windDir(3)*windSPDG];
56 windVelA = [windDir(1)*windSPDA, windDir(2)*windSPDA, windDir(3)*windSPDA];
57 heading_0 = [cosd(theta_0), 0, sind(theta_0)]; % initial heading of bottle rocket [x,y]
58 mass_air_0 = (p_abs_0 * volume_air_0) / (gas_const_air * temp_air_0); % initial mass of air...
    [kg]

```

```

59 mass_rocket_0 = mass_bottle + (rho_water * volume_water_0) + mass_air_0; % total mass of ...
    the bottle rocket [kg]
60
61 initial_conditions = [x_0, y_0, z_0, vel0_x, vel0_y, vel0_z, mass_rocket_0, mass_air_0, ...
    volume_air_0]; % initial conditions
62
63 figNum = 1;
64
65 %% Coefficient of Drag
66 cd_values = .2:.005:.5;
67 cd_x_dist = zeros(length(cd_values),1);
68 max_cd_dist_coords = [0 0];
69 j = 1;
70 for i = .2:.005:.5
71     cd = i; % coefficient of drag
72
73     [~, data_cd] = ode45('ode45func2',tspan,initial_conditions);
74
75     k = 1;
76     while data_cd(k,3) ≥ 0
77         k = k+1;
78     end
79
80     cd_x_dist(j) = (data_cd(k,1)+data_cd(k-1,1)) / 2;
81     j = j + 1;
82
83     if cd_x_dist(j-1) > max_cd_dist_coords(2)
84         max_cd_dist_coords(2) = cd_x_dist(j-1);
85         max_cd_dist_coords(1) = i;
86     end
87
88 end
89
90 cd = 0.327; % coefficient of drag
91 [~, data_cd] = ode45('ode45func2',tspan,initial_conditions);
92
93 k = 1;
94 while data_cd(k,3) ≥ 0
95     k = k+1;
96 end
97 ta_cd_dist = [0.327, (data_cd(k,1)+data_cd(k-1,1)) / 2];
98
99 figure(figNum)
100 p1 = plot(cd_values,cd_x_dist,'o','linewidth',1.25);
101 hold on
102 p2 = plot(max_cd_dist_coords(1),max_cd_dist_coords(2),'og','linewidth',1.75);
103 p3 = plot(ta_cd_dist(1),ta_cd_dist(2),'or','linewidth',1.75);
104 %xline(0.327,'--k','Gold Rocket Value','LabelHorizontalAlignment','left','...
    LabelVerticalAlignment','bottom','linewidth',1.25);
105 title('Sensitivity Analysis of the Coefficient of Drag');
106 xlabel('Coefficient of Drag');
107 ylabel('Downrange Distance, [m]');
108 legend([p2,p3],{'Max Distance = ' num2str(max_cd_dist_coords(2)) ' [m], coefficient of ...
    drag = ' num2str(max_cd_dist_coords(1))'},{'TA: Max Distance = ' num2str(ta_cd_dist(2)) ...
    ' [m], coefficient of drag = ' num2str(ta_cd_dist(1))}),'location','southwest');
109 hold off
110 figNum = figNum + 1;
111
112 %% Mass of Propellant
113 mass_values = 0.1*0.002:11/600000:0.6*0.002;
114 mass_values = mass_values.*rho_water;
115 mass_x_dist = zeros(length(mass_values),1);
116 max_dist_mass = [0 0];
117 j = 1;
118 for i = 0.1*0.002:11/600000:0.6*0.002
119
120     p_gage_0 = 40.5 * 6894.75729; % initial gage pressure of air in bottle [pa] MAXIMUM ...
        DISTANCE -> 4.136854374e+05 -> 60 psi // GOLD ROCKET -> 40.5 psi

```

```

121 p_abs_0 = p_gage_0 + p_amb; % absolute pressure of air inside the bottle [pa]
122 volume_water_0 = i; % initial volume of water inside bottle [m^3] MAXIMUM DISTANCE -> ...
    5.25e-04 [m^3]
123 volume_air_0 = volume_bottle - volume_water_0; % initial volume of air inside the ...
    bottle [m^3]
124 temp_air_0 = 295.15; % initial temperature of air [K] GOLD ROCKET -> 295.15 TA BASELINE...
    -> 283.15
125 vel_0 = 0; % initial velocity of bottle rocket [m/s]
126 vel0_x = 0; % initial velocity of bottle rocket in the x direction [m/s]
127 vel0_y = 0; % initial velocity of bottle rocket in the y direction [m/s]
128 vel0_z = 0; % initial velocity of bottle rocket in the z direction [m/s]
129 theta_0 = 45; % initial angle of bottle rocket [degrees] MAXIMUM DISTANCE -> 40
130 x_0 = 0; % initial downrange distance [m]
131 y_0 = 0; % initial crossrange distance [m]
132 z_0 = 0.25; % initial vertical height [m]
133 windTheta = 0;
134 windSPDG = 0; % wind velocity [m/s]
135 windSPDA = 0;
136 windDir = [cosd(windTheta) sind(windTheta) 0]; % wind velocity direction [m/s]
137 windVelG = [windDir(1)*windSPDG, windDir(2)*windSPDG, windDir(3)*windSPDG];
138 windVelA = [windDir(1)*windSPDA, windDir(2)*windSPDA, windDir(3)*windSPDA];
139 heading_0 = [cosd(theta_0), 0, sind(theta_0)]; % initial heading of bottle rocket [x,y]
140 mass_air_0 = (p_abs_0 * volume_air_0) / (gas_const_air * temp_air_0); % initial mass of...
    air [kg]
141 mass_rocket_0 = mass_bottle + (rho_water * volume_water_0) + mass_air_0; % total mass ...
    of the bottle rocket [kg]
142
143 initial_conditions = [x_0, y_0, z_0, vel0_x, vel0_y, vel0_z, mass_rocket_0, mass_air_0,...
    volume_air_0]; % initial conditions
144
145 [~, data_mass] = ode45('ode45func2',tspan,initial_conditions);
146
147 k = 1;
148 while data_mass(k,3) > 0
149     k = k+1;
150 end
151
152 mass_x_dist(j) = (data_mass(k,1)+data_mass(k-1,1)) / 2;
153 j = j + 1;
154
155 if mass_x_dist(j-1) > max_dist_mass(2) && j-1 ~= 27
156     max_dist_mass(2) = mass_x_dist(j-1);
157     max_dist_mass(1) = i;
158 end
159
160 end
161
162 p_gage_0 = 40.5 * 6894.75729; % initial gage pressure of air in bottle [pa] MAXIMUM ...
    DISTANCE -> 4.136854374e+05 -> 60 psi // GOLD ROCKET -> 40.5 psi
163 p_abs_0 = p_gage_0 + p_amb; % absolute pressure of air inside the bottle [pa]
164 volume_water_0 = 0.000993; % initial volume of water inside bottle [m^3] MAXIMUM ...
    DISTANCE -> 5.25e-04 [m^3]
165 volume_air_0 = volume_bottle - volume_water_0; % initial volume of air inside the ...
    bottle [m^3]
166 temp_air_0 = 295.15; % initial temperature of air [K] GOLD ROCKET -> 295.15 TA BASELINE...
    -> 283.15
167 vel_0 = 0; % initial velocity of bottle rocket [m/s]
168 vel0_x = 0; % initial velocity of bottle rocket in the x direction [m/s]
169 vel0_y = 0; % initial velocity of bottle rocket in the y direction [m/s]
170 vel0_z = 0; % initial velocity of bottle rocket in the z direction [m/s]
171 theta_0 = 45; % initial angle of bottle rocket [degrees] MAXIMUM DISTANCE -> 40
172 x_0 = 0; % initial downrange distance [m]
173 y_0 = 0; % initial crossrange distance [m]
174 z_0 = 0.25; % initial vertical height [m]
175 windTheta = 0;
176 windSPDG = 0; % wind velocity [m/s]
177 windSPDA = 0;
178 windDir = [cosd(windTheta) sind(windTheta) 0]; % wind velocity direction [m/s]

```

```

179     windVelG = [windDir(1)*windSPDG, windDir(2)*windSPDG, windDir(3)*windSPDG];
180     windVelA = [windDir(1)*windSPDA, windDir(2)*windSPDA, windDir(3)*windSPDA];
181     heading_0 = [cosd(theta_0),0, sind(theta_0)]; % initial heading of bottle rocket [x,y]
182     mass_air_0 = (p_abs_0 * volume_air_0) / (gas_const_air * temp_air_0); % initial mass of...
        air [kg]
183     mass_rocket_0 = mass_bottle + (rho_water * volume_water_0) + mass_air_0; % total mass ...
        of the bottle rocket [kg]
184
185     initial_conditions = [x_0, y_0, z_0, vel0_x, vel0_y, vel0_z, mass_rocket_0, mass_air_0,...
        volume_air_0]; % initial conditions
186
187     [~, data_mass] = ode45('ode45func2',tspan,initial_conditions);
188
189     k = 1;
190     while data_mass(k,3) >= 0
191         k = k+1;
192     end
193     ta_mass_dist = [0.000993*1000, (data_mass(k,1)+data_mass(k-1,1)) / 2];
194
195     figure(figNum)
196     p4 = plot(mass_values,mass_x_dist,'o','linewidth',1.25);
197     hold on
198     p5 = plot(max_dist_mass(1)*1000,max_dist_mass(2),'og','linewidth',1.75);
199     p6 = plot(ta_mass_dist(1),ta_mass_dist(2),'or','linewidth',1.75);
200     %xline(993/1000,'--k','Gold Rocket Value','LabelHorizontalAlignment','left','linewidth...
        ',1.25);
201     title('Sensitivity Analysis of the Mass of Propellant');
202     xlabel('Mass of Propellant, [kg]');
203     ylabel('Downrange Distance, [m]');
204     legend([p5,p6],{'Max Distance = ' num2str(max_dist_mass(2)) ' [m], mass of water = ' (...
        num2str(max_dist_mass(1)*1000)) ' [kg]'},['TA: Max Distance = ' num2str(ta_mass_dist(2)...
        ) ' [m], mass of water = ' (num2str(ta_mass_dist(1))) ' [kg]']});
205     hold off
206     figNum = figNum + 1;
207
208     %% Water Density
209     p_gage_0 = 40.5 * 6894.75729; % initial gage pressure of air in bottle [pa] MAXIMUM ...
        DISTANCE -> 4.136854374e+05 -> 60 psi // GOLD ROCKET -> 40.5 psi
210     p_abs_0 = p_gage_0 + p_amb; % absolute pressure of air inside the bottle [pa]
211     volume_water_0 = 0.000993; % initial volume of water inside bottle [m^3] MAXIMUM ...
        DISTANCE -> 5.25e-04 [m^3]
212     volume_air_0 = volume_bottle - volume_water_0; % initial volume of air inside the ...
        bottle [m^3]
213     temp_air_0 = 295.15; % initial temperature of air [K] GOLD ROCKET -> 295.15 TA BASELINE...
        -> 283.15
214     vel_0 = 0; % initial velocity of bottle rocket [m/s]
215     vel0_x = 0; % initial velocity of bottle rocket in the x direction [m/s]
216     vel0_y = 0; % initial velocity of bottle rocket in the y direction [m/s]
217     vel0_z = 0; % initial velocity of bottle rocket in the z direction [m/s]
218     theta_0 = 45; % initial angle of bottle rocket [degrees] MAXIMUM DISTANCE -> 40
219     x_0 = 0; % initial downrange distance [m]
220     y_0 = 0; % initial crossrange distance [m]
221     z_0 = 0.25; % initial vertical height [m]
222     windTheta = 0;
223     windSPDG = 0; % wind velocity [m/s]
224     windSPDA = 0;
225     windDir = [cosd(windTheta) sind(windTheta) 0]; % wind velocity direction [m/s]
226     windVelG = [windDir(1)*windSPDG, windDir(2)*windSPDG, windDir(3)*windSPDG];
227     windVelA = [windDir(1)*windSPDA, windDir(2)*windSPDA, windDir(3)*windSPDA];
228     heading_0 = [cosd(theta_0),0, sind(theta_0)]; % initial heading of bottle rocket [x,y]
229     mass_air_0 = (p_abs_0 * volume_air_0) / (gas_const_air * temp_air_0); % initial mass of...
        air [kg]
230     mass_rocket_0 = mass_bottle + (rho_water * volume_water_0) + mass_air_0; % total mass ...
        of the bottle rocket [kg]
231
232
233     density_values = 1000:1:1060;
234     density_x_dist = zeros(length(density_values),1);

```

```

235 max_dist_density = [0 0];
236 j = 1;
237 for i = 1000:1:1060
238
239     rho_water = i; % density of water [kg/m^3]
240
241     p_gage_0 = 40.5 * 6894.75729; % initial gage pressure of air in bottle [pa] MAXIMUM ...
        DISTANCE -> 4.136854374e+05 -> 60 psi // GOLD ROCKET -> 40.5 psi
242     p_abs_0 = p_gage_0 + p_amb; % absolute pressure of air inside the bottle [pa]
243     volume_water_0 = 0.000993; % initial volume of water inside bottle [m^3] MAXIMUM ...
        DISTANCE -> 5.25e-04 [m^3]
244     volume_air_0 = volume_bottle - volume_water_0; % initial volume of air inside the ...
        bottle [m^3]
245     temp_air_0 = 295.15; % initial temperature of air [K] GOLD ROCKET -> 295.15 TA BASELINE...
        -> 283.15
246     vel_0 = 0; % initial velocity of bottle rocket [m/s]
247     vel0_x = 0; % initial velocity of bottle rocket in the x direction [m/s]
248     vel0_y = 0; % initial velocity of bottle rocket in the y direction [m/s]
249     vel0_z = 0; % initial velocity of bottle rocket in the z direction [m/s]
250     theta_0 = 45; % initial angle of bottle rocket [degrees] MAXIMUM DISTANCE -> 40
251     x_0 = 0; % initial downrange distance [m]
252     y_0 = 0; % initial crossrange distance [m]
253     z_0 = 0.25; % initial vertical height [m]
254     windTheta = 0;
255     windSPDG = 0; % wind velocity [m/s]
256     windSPDA = 0;
257     windDir = [cosd(windTheta) sind(windTheta) 0]; % wind velocity direction [m/s]
258     windVelG = [windDir(1)*windSPDG, windDir(2)*windSPDG, windDir(3)*windSPDG];
259     windVelA = [windDir(1)*windSPDA, windDir(2)*windSPDA, windDir(3)*windSPDA];
260     heading_0 = [cosd(theta_0), 0, sind(theta_0)]; % initial heading of bottle rocket [x,y]
261     mass_air_0 = (p_abs_0 * volume_air_0) / (gas_const_air * temp_air_0); % initial mass of...
        air [kg]
262     mass_rocket_0 = mass_bottle + (rho_water * volume_water_0) + mass_air_0; % total mass ...
        of the bottle rocket [kg]
263
264     initial_conditions = [x_0, y_0, z_0, vel0_x, vel0_y, vel0_z, mass_rocket_0, mass_air_0,...
        volume_air_0]; % initial conditions
265
266     [~, data_density] = ode45('ode45func2',tspan,initial_conditions);
267
268     k = 1;
269     while data_density(k,3) > 0
270         k = k+1;
271     end
272
273     density_x_dist(j) = (data_density(k,1)+data_density(k-1,1)) / 2;
274     j = j + 1;
275
276     if density_x_dist(j-1) > max_dist_density(2) && j-1 ≠ 16
277         max_dist_density(2) = density_x_dist(j-1);
278         max_dist_density(1) = i;
279     end
280
281     if i == 1000
282         ta_max_density = [i, density_x_dist(j-1)];
283     end
284
285 end
286
287 figure(figNum)
288 p7 = plot(density_values,density_x_dist,'o','linewidth',1.25);
289 hold on
290 p8 = plot(max_dist_density(1),max_dist_density(2),'og','linewidth',1.75);
291 p9 = plot(ta_max_density(1),ta_max_density(2),'or','linewidth',1.75);
292 %xline(1000,'--k','Gold Rocket Value','LabelHorizontalAlignment','left','...
        LabelVerticalAlignment','bottom','linewidth',1.25);
293 title('Sensitivity Analysis of the Density of Propellant');
294 xlabel('Density of Propellant, [kg/m^3]');

```



```

295 ylabel('Downrange Distance, [m]');
296 legend([p8,p9],[ 'Max Distance = ' num2str(max_dist_density(2)) ' [m], density of ...
    propellant = ' (num2str(max_dist_density(1))) ' [kg/m^3]'], ['TA: Max Distance = ' ...
    num2str(ta_max_density(2)) ' [m], density of propellant = ' (num2str(ta_max_density(1)))...
    ) ' [kg/m^3]'],'location','southwest');
297 hold off
298 figNum = figNum + 1;
299
300 %% Launch Pad Angle
301 rho_water = 1000; % density of water [kg/m^3]
302 p_gage_0 = 40.5 * 6894.75729; % initial gage pressure of air in bottle [pa] MAXIMUM ...
    DISTANCE -> 4.136854374e+05 -> 60 psi // GOLD ROCKET -> 40.5 psi
303 p_abs_0 = p_gage_0 + p_amb; % absolute pressure of air inside the bottle [pa]
304 volume_water_0 = 0.000993; % initial volume of water inside bottle [m^3] MAXIMUM ...
    DISTANCE -> 5.25e-04 [m^3]
305 volume_air_0 = volume_bottle - volume_water_0; % initial volume of air inside the ...
    bottle [m^3]
306 temp_air_0 = 295.15; % initial temperature of air [K] GOLD ROCKET -> 295.15 TA BASELINE...
    -> 283.15
307 vel_0 = 0; % initial velocity of bottle rocket [m/s]
308 vel0_x = 0; % initial velocity of bottle rocket in the x direction [m/s]
309 vel0_y = 0; % initial velocity of bottle rocket in the y direction [m/s]
310 vel0_z = 0; % initial velocity of bottle rocket in the z direction [m/s]
311 theta_0 = 45; % initial angle of bottle rocket [degrees] MAXIMUM DISTANCE -> 40
312 x_0 = 0; % initial downrange distance [m]
313 y_0 = 0; % initial crossrange distance [m]
314 z_0 = 0.25; % initial vertical height [m]
315 windTheta = 0;
316 windSPDG = 0; % wind velocity [m/s]
317 windSPDA = 0;
318 windDir = [cosd(windTheta) sind(windTheta) 0]; % wind velocity direction [m/s]
319 windVelG = [windDir(1)*windSPDG, windDir(2)*windSPDG, windDir(3)*windSPDG];
320 windVelA = [windDir(1)*windSPDA, windDir(2)*windSPDA, windDir(3)*windSPDA];
321 heading_0 = [cosd(theta_0),0, sind(theta_0)]; % initial heading of bottle rocket [x,y]
322 mass_air_0 = (p_abs_0 * volume_air_0) / (gas_const_air * temp_air_0); % initial mass of...
    air [kg]
323 mass_rocket_0 = mass_bottle + (rho_water * volume_water_0) + mass_air_0; % total mass ...
    of the bottle rocket [kg]
324
325 LA_values = 1:1:89;
326 LA_x_dist = zeros(length(LA_values),1);
327 tspan = [0,10];
328 max_dist_LA = [0 0];
329 j = 1;
330 for i = 1:1:89
331
332     theta_0 = i; % initial angle of bottle rocket [degrees] MAXIMUM DISTANCE -> 40
333     heading_0 = [cosd(theta_0),0, sind(theta_0)]; % initial heading of bottle rocket [x,y]
334
335     initial_conditions = [x_0, y_0, z_0, vel0_x, vel0_y, vel0_z, mass_rocket_0, mass_air_0,...
        volume_air_0]; % initial conditions
336
337     [~, data_LA] = ode45('ode45func2',tspan,initial_conditions);
338
339     k = 1;
340     while data_LA(k,3) > 0
341         k = k+1;
342     end
343
344     LA_x_dist(j) = (data_LA(k,1)+data_LA(k-1,1)) / 2;
345     j = j + 1;
346
347     if LA_x_dist(j-1) > max_dist_LA(2) && j-1 ~= 42
348         max_dist_LA(2) = LA_x_dist(j-1);
349         max_dist_LA(1) = i;
350     end
351
352     if i == 45

```

```

353         ta_max_LA = [i, LA_x_dist(j-1)];
354     end
355
356 end
357
358 figure(figNum)
359 p10 = plot(LA_values,LA_x_dist,'o','linewidth',1.25);
360 hold on
361 p11 = plot(max_dist_LA(1),max_dist_LA(2),'og','linewidth',1.75);
362 p12 = plot(ta_max_LA(1),ta_max_LA(2),'or','linewidth',1.75);
363 %xline(45,'--k','Gold Rocket Value','LabelHorizontalAlignment','left','...
    LabelVerticalAlignment','bottom','linewidth',1.25);
364 title('Sensitivity Analysis of the Launch Angle');
365 xlabel('Launch Angle, [deg]');
366 ylabel('Downrange Distance, [m]');
367 legend([p11,p12],{'Max Distance = ' num2str(51.0911) ' [m], launch angle = ' num2str(...
    max_dist_LA(1)) ' [deg]'},{'TA: Max Distance = ' num2str(51.0911) ' [m], launch angle =...
    ' num2str(ta_max_LA(1)) ' [deg]'}},'location','north');
368 hold off
369 figNum = figNum + 1;

```

## Error Ellipses

```
1 function [] = error_ellipses(data,ta)
2
3     %% Replace this section of code with your real data
4     % Simulate and plot 100 data points, (YOU SHOULD USE REAL DATA HERE!)
5     x = data(:,1);
6     y = data(:,2);
7
8     %%
9
10    % Calculate covariance matrix
11    P = cov(x,y);
12    mean_x = mean(x);
13    mean_y = mean(y);
14    figure;
15    plot(mean_x,mean_y,'y.','markersize',24)
16    axis equal;
17    grid on;
18    xlabel('x, Downrange Distance [m]');
19    ylabel('y, Crossrange Distance [m]');
20    title('Landing Points with Error Ellipses');
21    hold on;
22    plot(x,y,'k.','markersize',6)
23    if ta == 1
24        plot(40.5,-8.61,'m.','markersize',24)
25    end
26
27    % Calculate the define the error ellipses
28    n=100; % Number of points around ellipse
29    p=0:pi/n:2*pi; % angles around a circle
30
31    [eigvec,eigval] = eig(P); % Compute eigen-stuff
32    xy_vect = [cos(p'),sin(p')] * sqrt(eigval) * eigvec; % Transformation
33    x_vect = xy_vect(:,1);
34    y_vect = xy_vect(:,2);
35
36    % Plot the error ellipses overlaid on the same figure
37    plot(1*x_vect+mean_x, 1*y_vect+mean_y, 'b', 'Linewidth', 1.5)
38    plot(2*x_vect+mean_x, 2*y_vect+mean_y, 'g', 'Linewidth', 1.5)
39    plot(3*x_vect+mean_x, 3*y_vect+mean_y, 'r', 'Linewidth', 1.5)
40    if ta == 1
41        legend('Center','Landing Spots','Actual Landing Spot','1\sigma','2\sigma','3\sigma'...
42            , 'location','southeast');
43    else
44        legend('Center','Landing Spots','1\sigma','2\sigma','3\sigma');
45        semimajor = max(3*x_vect);
46        semiminor = max(3*y_vect);
47        area = pi*semimajor*semiminor;
48        fprintf(['The semimajor axis is ' num2str(semimajor) ' [m]. The semiminor axis is '...
49            num2str(semiminor) ' [m]. The area is ' num2str(area) ' [m^2]. \n']);
50    end
51 end
```