University of Colorado - Boulder

ASEN 2004 - Vehicle Design and Performance

Lab 3

Water Bottle Rocket Final Executive Summary

Authors:
Cole MacPherson[a]

[a]SID: 108521329

Instructors:
Torin Clark
Aaron Johnson
John Mah

April 29 2020

College of Engineering & Applied Science
UNIVERSITY OF COLORADO BOULDER

Contents

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^i_{air}	=	initial air pressure in the bottle
T	=	thrust
T_{air}	=	temperature of the air
v	=	volume of air in the bottle
v^i_{air}	=	initial volume of air in the bottle
V_e	=	exit velocity
ρ_{air}	=	density of the air
ρ_e	=	exit density
ρ_w	=	density of water
θ	=	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).

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

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

$$\dot{m} = -\mathrm{c_d}\mathrm{A_t}\sqrt{2\rho_\mathrm{w}\left(p - p_a\right)} \tag{2}$$

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

$$T = 2c_d A_t\left(p - p_a\right) \tag{3}$$

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)

$$\dot{m} = -c_\mathrm{d}\rho_e \mathrm{A_t}\mathrm{V_e} \tag{4}$$

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.

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

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 who's influence on the flight of the rocket was known ahead of the test. Variable 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 form the TA baseline rocket.

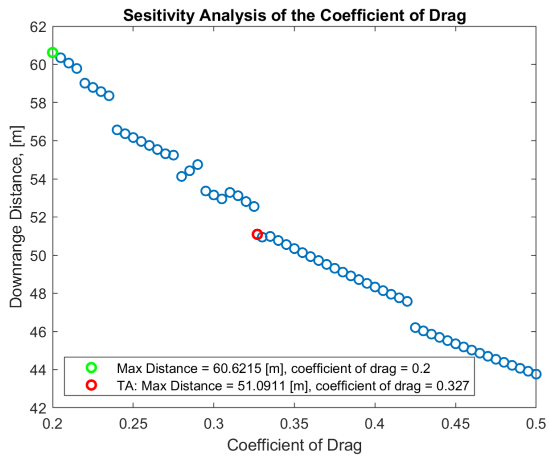

Fig. 1 Sensitivity Analysis of the Coefficient of Drag

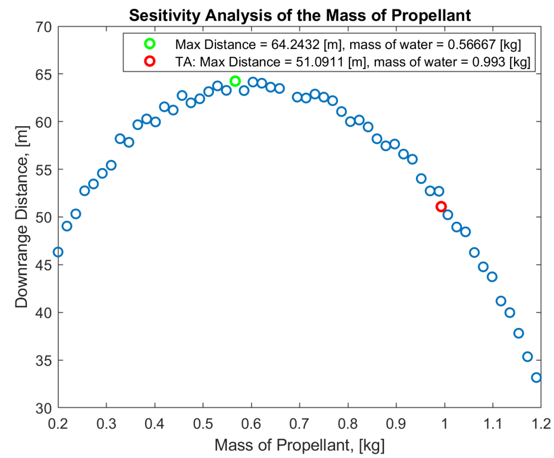

Fig. 2 Sensitivity Analysis of the Propellant Mass

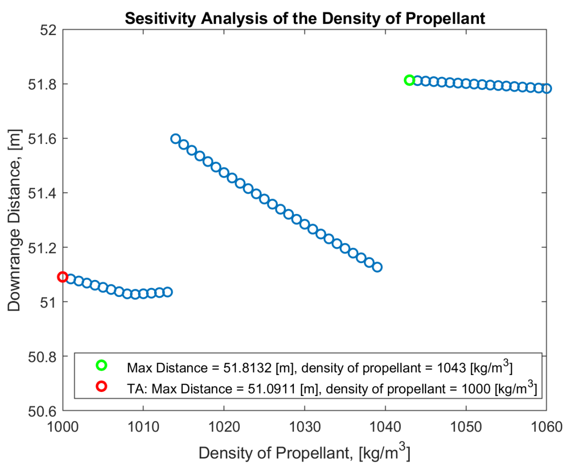

Fig. 3 Sensitivity Analysis of the Propellant Density

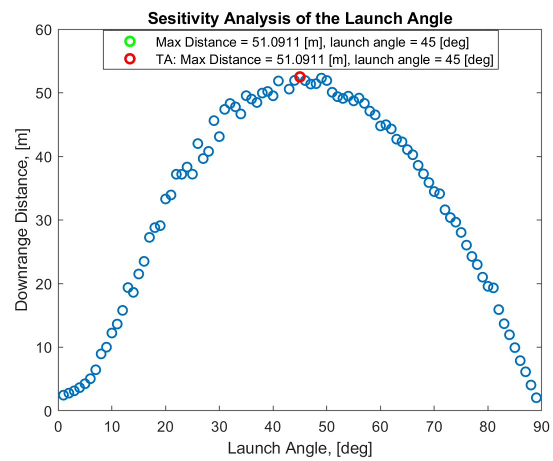

Fig. 4 Sensitivity Analysis of the Launch Angle

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

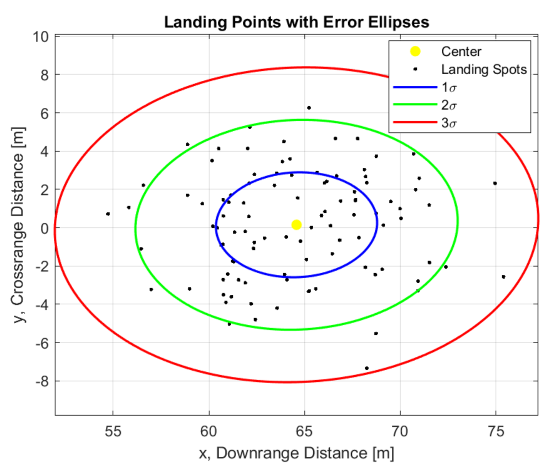

Fig. 5 Error Ellipses for Designed Rocket

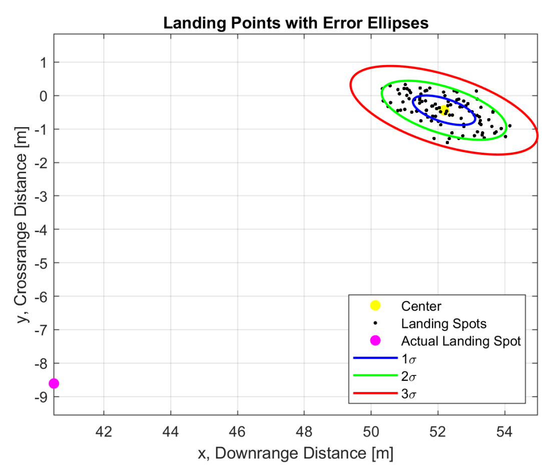

Fig. 6 Error Ellipses for TA Gold Rocket

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 the 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

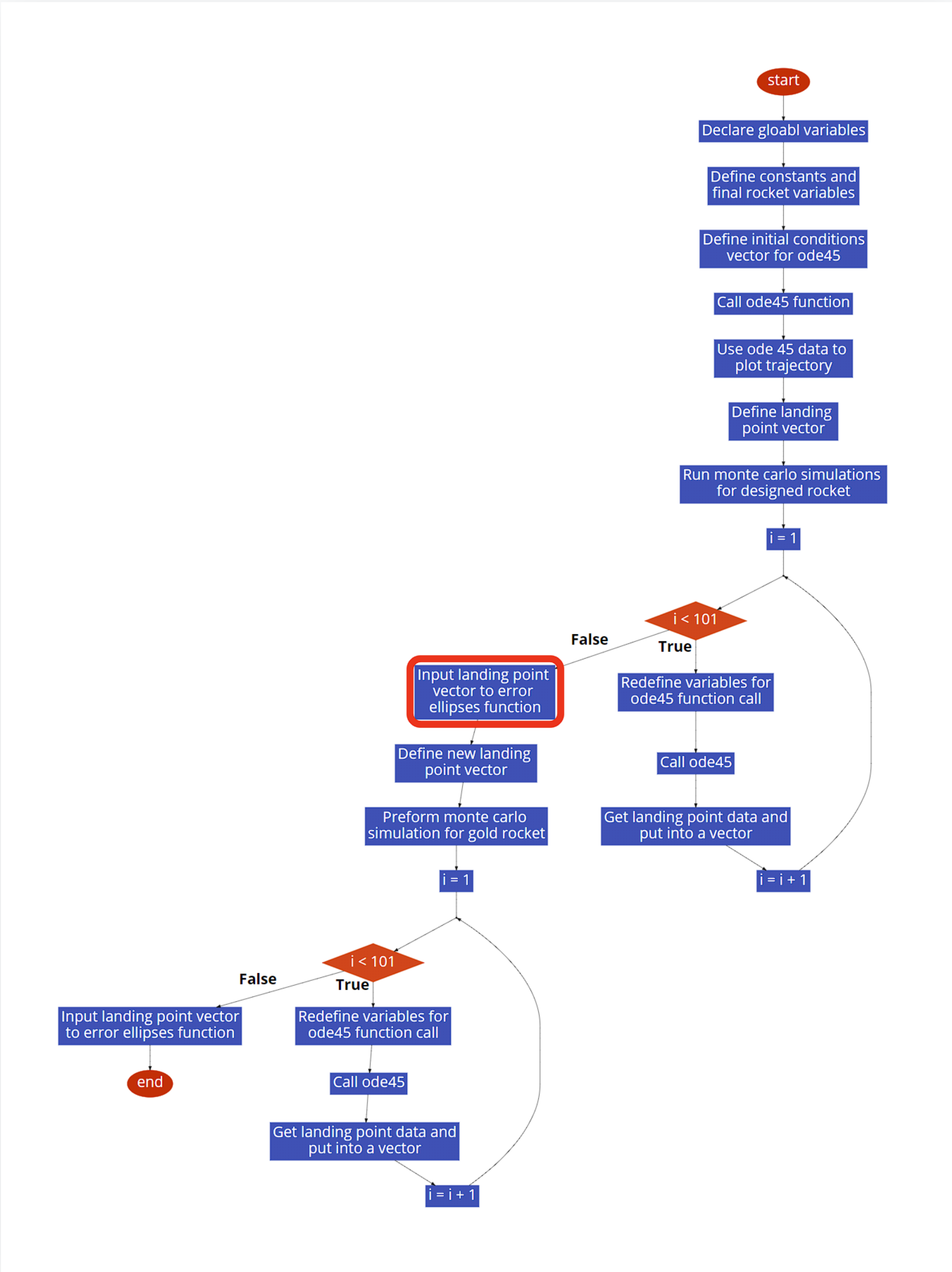

Fig. 7 Flow Chart for the Main Script

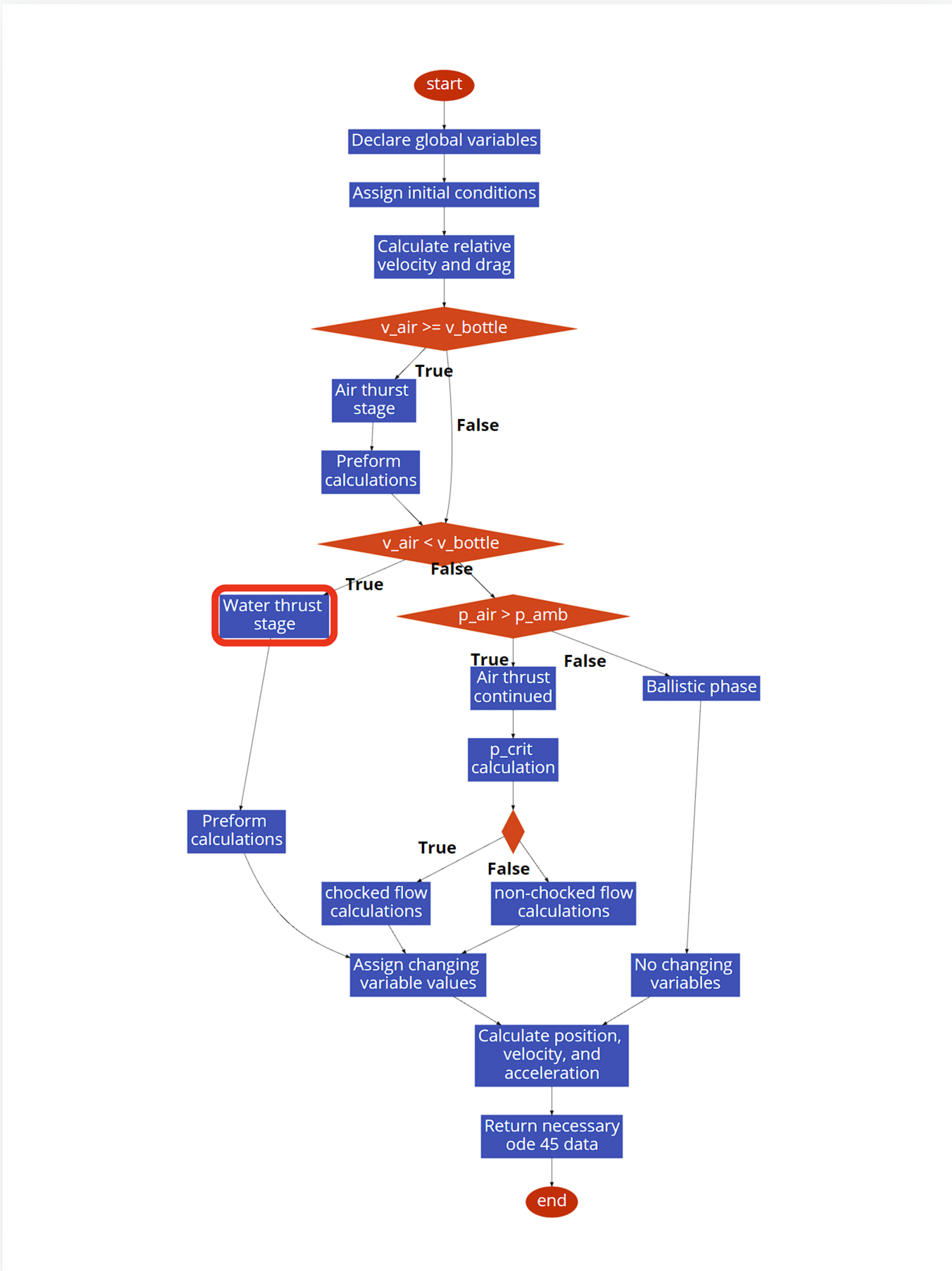

Fig. 8 Flow Chart for the ode45 Function

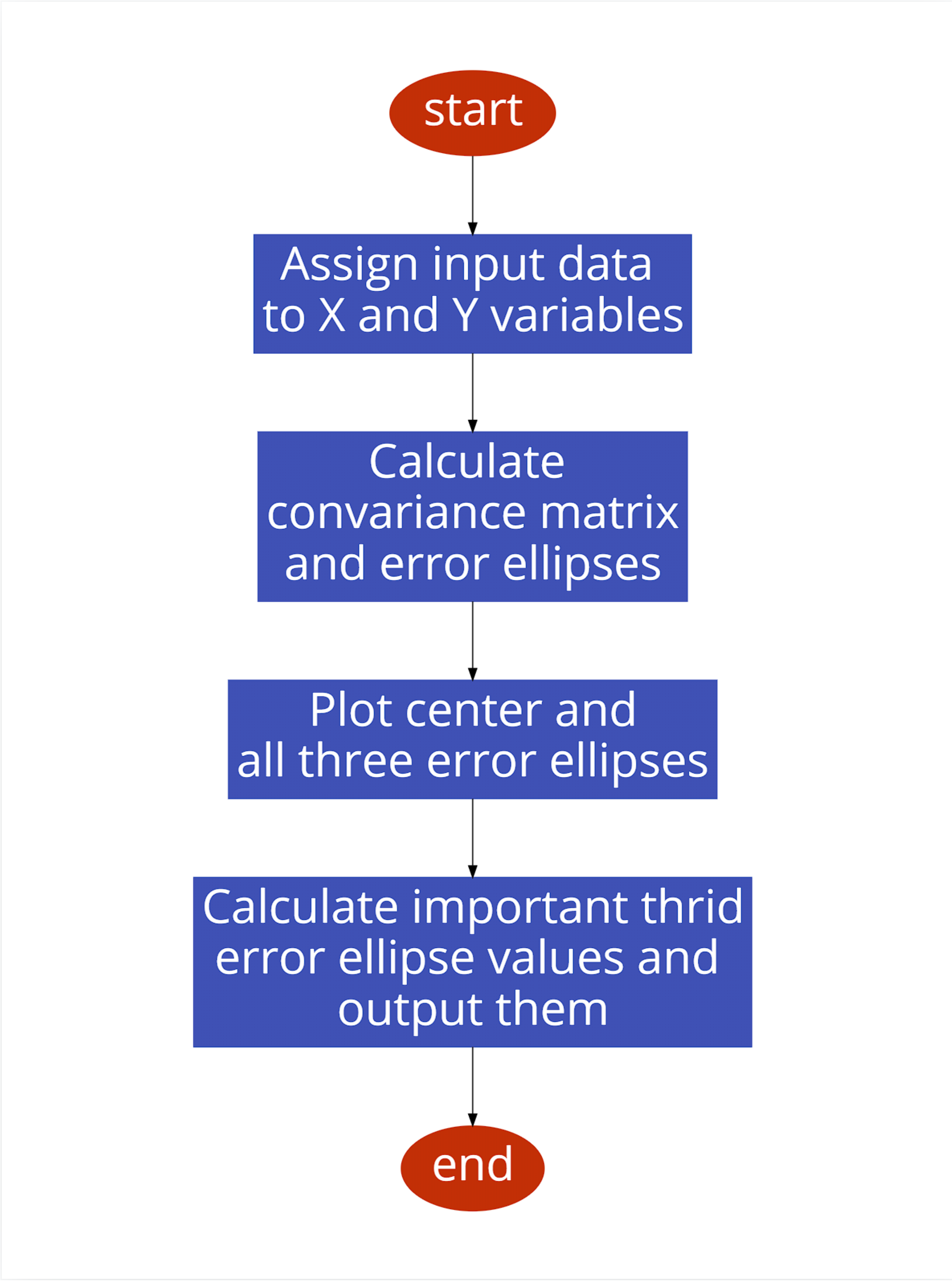

Fig. 9 Flow Chart for the Error Ellipse Function

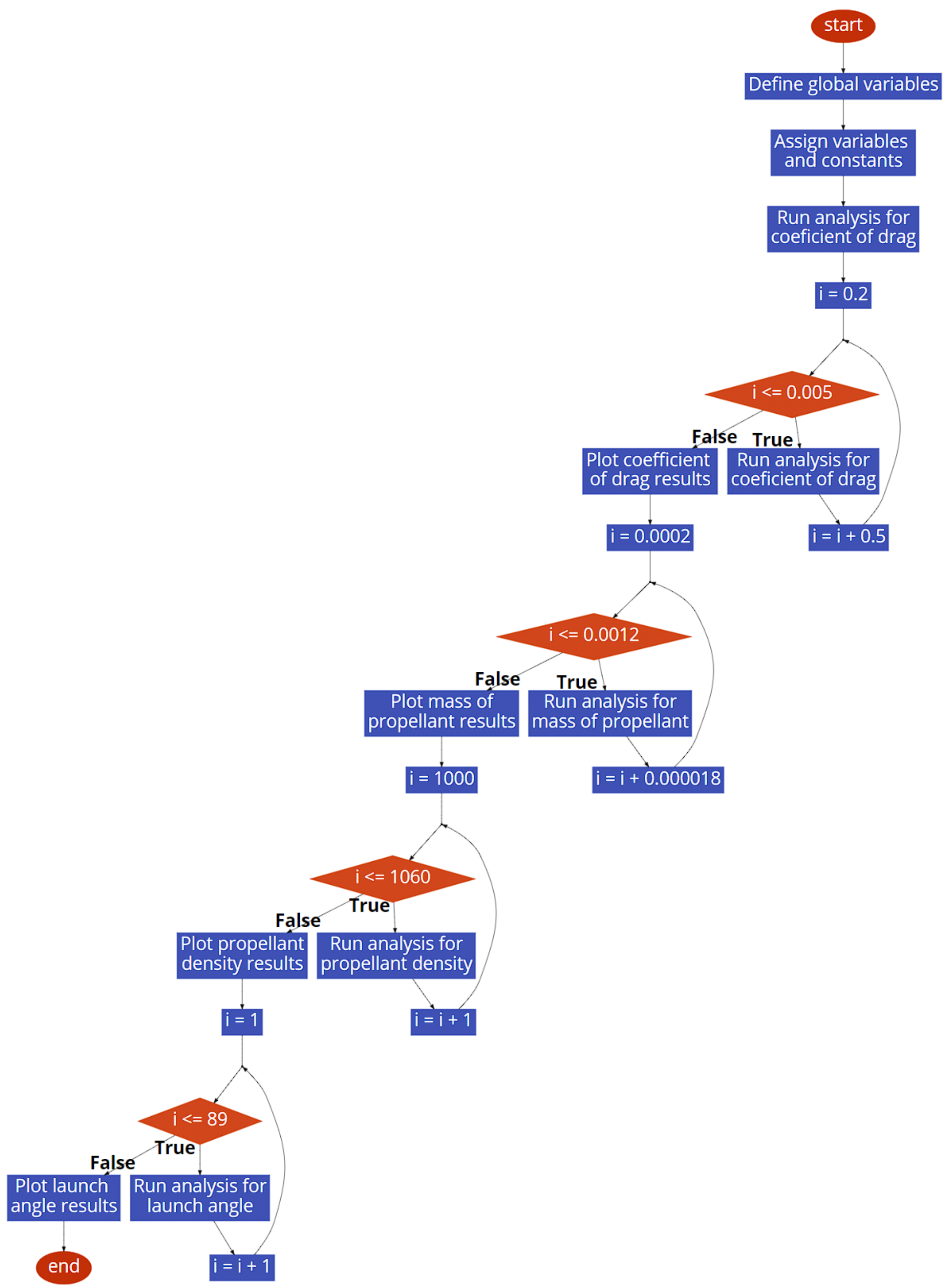

Fig. 10 Flow Chart for the Sensitivity Analysis Function

Appendix B: MATLAB Code

Main Script

```
%% ASEN 2012 Project 2 (Water Bottle Rocket)

% Authors
%
% 1) Cole MacPherson
% 2) Ankrit Uprety
%
% https://www.timeanddate.com/weather/usa/boulder/historic?month=3&year=2020

%% Housekeeping

clc;
clear;
close all;

%% Declare Global Variables and Constants/Verification Constants

% Gloabl Variables Declaration

global g discharge_coeff rho_air_amb volume_bottle p_amb gamma rho_water ...
    diameter_throat area_throat area_bottle diameter_bottle gas_const_air ...
    mass_bottle cd p_gage_0 volume_water_0 temp_air_0 vel_0 theta_0 x_0 y_0 ...
    z_0 vel0_x vel0_y vel0_z l_stand mass_air_0 mass_rocket_0 heading_0 ...
    volume_air_0 p_abs_0 tspan windVelG windVelA;

% Constants

g = 9.81; % acceleration due to gravity [m/s^2]
discharge_coeff = 0.8; % discharge coefficient
rho_air_amb = 1.17358; % ambient air density [kg/m^3] GOLD ROCKET -> 1.17358 [kg/m^3] TA ...
    BASELINE -> 1.23478
volume_bottle = 0.002; % volume of the empty bottle [m^3]
p_amb = 99640; % atmospheric pressure [pa] GOLD ROCKET -> 99640 [pa] TA BASELINE -> 101592
gamma = 1.4; % ratio of specific heats for heat
rho_water = 1000; % density of water [kg/m^2]
diameter_throat = 2.1 / 100; % diameter of throat [m]
diameter_bottle = 10.5 / 100; % diameter of bottle [m]
area_throat = (1/4) * pi * (diameter_throat)^2; % area of throat [m^2]
area_bottle = (1/4) * pi * (diameter_bottle)^2; % area of bottle [m^2]
gas_const_air = 287; % gas constant of air [J/(kg*K)]
mass_bottle = 0.126; % mass of the empty bottle w/ cone and fins [kg]
cd = 0.327; % coefficient of drag
l_stand = 0.5; % langth of test stand [m]
tspan = [0, 5]; % integration time span [s]

% Max Distance Constants

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
p_abs_0 = p_gage_0 + p_amb; % absolutee pressure of air inside the bottle [pa]
volume_water_0 = 0.0005833; % intitial volume of water inside bottle [m^3]
volume_air_0 = volume_bottle - volume_water_0; % intitial volume of air inside the bottle [...
    m^3]
temp_air_0 = 295.15; % initial temperature of air [K] GOLD ROCKET -> 295.15 TA BASELINE -> ...
    283.15
vel_0 = 0; % initial velocity of bottle rocket [m/s]
vel0_x = 0; % initial velocity of bottle rocket in the x direction [m/s]
vel0_y = 0; % initial velocity of bottle rocket in the y direction [m/s]
vel0_z = 0; % initial velocity of bottle rocket in the z direction [m/s]
theta_0 = 45; % intitial angle of bottle rocket [degrees] MAXIMUM DISTANCE -> 40
x_0 = 0; % initial downrange distance [m]
y_0 = 0; % initial crossrange distance [m]
```

```
z_0 = 0.25; % initial verital height [m]
windTheta = 0;
windSPDG = 0; % wind velocity [m/s]
windSPDA = 0;
windDir = [cosd(windTheta) sind(windTheta) 0]; % wind velocity direction [m/s]
windVelG = [windDir(1)*windSPDG, windDir(2)*windSPDG, windDir(3)*windSPDG];
windVelA = [windDir(1)*windSPDA, windDir(2)*windSPDA, windDir(3)*windSPDA];
heading_0 = [cosd(theta_0),0, sind(theta_0)]; % initial heading of bottle rocket [x,y]
mass_air_0 = (p_abs_0 * volume_air_0) / (gas_const_air * temp_air_0); % initial mass of air...
     [kg]
mass_rocket_0 = mass_bottle + (rho_water * volume_water_0) + mass_air_0; % total mass of ...
    the bottle rocket [kg]


%% Calculations

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

[time, data] = ode45('ode45func2',tspan,initial_conditions);

%[thrust, phase1_end_i, phase2_end_i] =  thrustvecfunc2(time,data);

%% Plot trajectory of bottle rocket
i = 1;
while data(i,3) ≥ 0
    i = i+1;
end

max_distance_x = (data(i-1,1)+data(i,1)) / 2;
max_distance_y = (data(i-1,2)+data(i,2)) / 2;
max_distance_z = 0;
max_distances = [max_distance_x,max_distance_y,max_distance_z];
distanceFromLaunch = sqrt(max_distance_x^2+max_distance_y^2);

x_limit = max_distances(1)*1.05;
y_limit = max_distances(2)*1.05;
z_limit = max(data(:,3))*1.05;

j = 1;
while data(j,3) ≠ max(data(:,3))
    j = j+1;
    max_height = [data(j,1) data(j,2) data(j,3)];
end

figure(1)
plot3(x_0,y_0,z_0,'or',max_distances(1),max_distances(2), ...
    max_distances(3),'*r',max_height(1),max_height(2),max_height(3),'+r', ...
    data(1:i,1),data(1:i,2),data(1:i,3),'b','LineWidth',2);
xlim([x_0 x_limit]);
if y_limit > 0
    ylim([y_0 y_limit]);
elseif y_limit < 0
    ylim([y_limit y_0]);
end
zlim([z_0-0.3 z_limit]);
title('Rocket Trajectory');
xlabel('Downrange Distance, [m]');
ylabel('Crossrange Distance, [m]');
zlabel('Vertical Distance, [m]');
legend(['Launch Stand Location'],['Maximum Downrange Distance, ' num2str(max_distances(1)) ...
    ' [m]'],['Maximum Height, ' num2str(max_height(3)) ' [m]'],'location','northoutside');
grid on;

%% Montecarlo Error Ellipses

xyCoorSim = zeros(100,2);

```

```
for i = 1:101

    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
    p_amb = 99640 + (1000*rand - 500); % atmospheric pressure [pa] GOLD ROCKET -> 99640 [pa...
        ] TA BASELINE -> 101592
    cd = 0.327 + 0.004*randn; % coefficient of drag
    diameter_throat = (2.1 / 100) + 0.0005*randn; % diameter of throat [m]
    diameter_bottle = (10.5 / 100) + 0.0005*randn; % diameter of bottle [m]
    area_throat = (1/4) * pi * (diameter_throat)^2; % area of throat [m^2]
    area_bottle = (1/4) * pi * (diameter_bottle)^2; % area of bottle [m^2]

    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
    p_abs_0 = p_gage_0 + p_amb; % absolutee pressure of air inside the bottle [pa]
    volume_water_0 = 0.0005833 + (0.0000005*randn); % intitial volume of water inside ...
        bottle [m^3] MAXIMUM DISTANCE -> 5.25e-04 [m^3]
    volume_air_0 = volume_bottle - volume_water_0; % intitial volume of air inside the ...
        bottle [m^3]
    temp_air_0 = 295.15 + (2*rand - 1); % initial temperature of air [K] GOLD ROCKET -> ...
        295.15 TA BASELINE -> 283.15
    theta_0 = 45 + randn; % intitial angle of bottle rocket [degrees] MAXIMUM DISTANCE -> ...
        40
    windTheta = 360*rand;
    windSPDG = 10*rand; % wind ground velocity [m/s]
    windSPDA = 10*rand; % wind aloft velocity [m/s]
    windDir = [cosd(windTheta) sind(windTheta) 0]; % wind velocity direction [m/s]
    windVelG = [windDir(1)*windSPDG, windDir(2)*windSPDG, windDir(3)*windSPDG];
    windVelA = [windDir(1)*windSPDA, windDir(2)*windSPDA, windDir(3)*windSPDA];
    heading_0 = [cosd(theta_0),0, sind(theta_0)]; % initial heading of bottle rocket [x,y]
    mass_air_0 = (p_abs_0 * volume_air_0) / (gas_const_air * temp_air_0); % initial mass of...
         air [kg]
    mass_rocket_0 = mass_bottle + (rho_water * volume_water_0) + mass_air_0; % total mass ...
        of the bottle rocket [kg]


    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

    [timeSim, dataSim] = ode45('ode45func2',tspan,initial_conditions);

    j = 1;
    while dataSim(j,3) ≥ 0
        j = j+1;
    end
    j = j-1;

    xyCoorSim(i,:) = [(dataSim(j,1)+dataSim(j+1,1))/2, (dataSim(j,2)+dataSim(j+1,2))/2];

end

error_ellipses(xyCoorSim,0);

%% Montecarlo Error Ellipses GOLD ROCKET

xyCoorSim_g = zeros(100,2);

for i = 1:101

    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
    p_amb = 99640 + (1000*rand - 500); % atmospheric pressure [pa] GOLD ROCKET -> 99640 [pa...
        ] TA BASELINE -> 101592
    cd = 0.327 + 0.004*randn; % coefficient of drag
    diameter_throat = (2.1 / 100) + 0.0005*randn; % diameter of throat [m]
    diameter_bottle = (10.5 / 100) + 0.0005*randn; % diameter of bottle [m]
    area_throat = (1/4) * pi * (diameter_throat)^2; % area of throat [m^2]
```

```
    area_bottle = (1/4) * pi * (diameter_bottle)^2; % area of bottle [m^2]

    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
    p_abs_0 = p_gage_0 + p_amb; % absolutee pressure of air inside the bottle [pa]
    volume_water_0 = 0.000993 + (0.0000005*randn); % intitial volume of water inside bottle...
         [m^3] MAXIMUM DISTANCE -> 5.25e-04 [m^3]
    volume_air_0 = volume_bottle - volume_water_0; % intitial volume of air inside the ...
        bottle [m^3]
    temp_air_0 = 295.15 + (2*rand - 1); % initial temperature of air [K] GOLD ROCKET -> ...
        295.15 TA BASELINE -> 283.15
    theta_0 = 45 + randn; % intitial angle of bottle rocket [degrees] MAXIMUM DISTANCE -> ...
        40
    windTheta = (136+(22.5*rand-11.25)) + 1.5*randn;
    windSPDG = 0 + (2*0.44704*rand-0.44704); % wind ground velocity [m/s]
    windSPDA = 0.44704 + (2*0.44704*rand-0.44704); % wind aloft velocity [m/s]
    windDir = [cosd(windTheta) sind(windTheta) 0]; % wind velocity direction [m/s]
    windVelG = [windDir(1)*windSPDG, windDir(2)*windSPDG, windDir(3)*windSPDG];
    windVelA = [windDir(1)*windSPDA, windDir(2)*windSPDA, windDir(3)*windSPDA];
    heading_0 = [cosd(theta_0),0, sind(theta_0)]; % initial heading of bottle rocket [x,y]
    mass_air_0 = (p_abs_0 * volume_air_0) / (gas_const_air * temp_air_0); % initial mass of...
         air [kg]
    mass_rocket_0 = mass_bottle + (rho_water * volume_water_0) + mass_air_0; % total mass ...
        of the bottle rocket [kg]


    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

    [timeSim_g, dataSim_g] = ode45('ode45func2',tspan,initial_conditions);

    j = 1;
    while dataSim_g(j,3) ≥ 0
        j = j+1;
    end
    j = j-1;

    xyCoorSim_g(i,:) = [(dataSim_g(j,1)+dataSim_g(j+1,1))/2, (dataSim_g(j,2)+dataSim_g(j...
        +1,2))/2];

end

error_ellipses(xyCoorSim_g,1);
```

ode45 function

```
% Differential Equations for the bottle rocket

function datafunc = ode45func2(t,conditions_0)

%% Global Variables
global g discharge_coeff rho_air_amb volume_bottle p_amb gamma rho_water ...
    area_throat area_bottle gas_const_air cd x_0 z_0 l_stand mass_air_0 ...
    heading_0 volume_air_0 p_abs_0 windVelG windVelA;

%% Initial Conditions
x = conditions_0(1); % x position
y = conditions_0(2); % y position
z = conditions_0(3); % z position
vel_x = conditions_0(4); % Velocity downrange direction
vel_y = conditions_0(5); % Velocity crossrange direction
vel_z = conditions_0(6); % Velocity vertical direction
mass_rocket = conditions_0(7); % Mass of the bottle rocket
mass_air = conditions_0(8); % mass of air in the bottle rocket
volume_air = conditions_0(9); % volume of air inside the bottle rocket

%% Calculations
windVel = (windVelG + windVelA) / 2;
if sqrt((x - x_0)^(2) + (z - z_0)^(2)) > l_stand
        vel_rel = [vel_x-windVel(1), vel_y-windVel(2), vel_z-windVel(3)];
        vel_rel_tot = sqrt((vel_x + windVel(1))^2 + (vel_y + windVel(2))^2 + (vel_z + ...
            windVel(3))^2);
else
    vel_rel = [vel_x, vel_y, vel_z];
    vel_rel_tot = sqrt((vel_x)^2 + (vel_y)^2 + (vel_z)^2);
end

drag = (rho_air_amb / 2) * (vel_rel_tot)^2 * cd * area_bottle; % find drag using equation 2...
     [N]

    if volume_air ≥ volume_bottle % if the volume of the air is greater than or equal to ...
        the volume of the bottle (signifies the end of the water thrust stage)

        p_water_thrust_end = p_abs_0 * ...
            ((volume_air_0 / volume_bottle)^(gamma)); % pressure at the end of the water ...
                thrust stage, using equation 13

        p_air_thrust_stage = p_water_thrust_end * ...
            ((mass_air / mass_air_0)^(gamma)); % pressure of air during air thrust stage, ...
                using equation 14

        rho_air_thrust_stage = mass_air / volume_bottle; % denstiy of air at ...
            p_air_thrust_stage, using equation 15
        temp_air_thrust_stage = p_air_thrust_stage / ...
            (rho_air_thrust_stage * gas_const_air); % temperature of air at ...
                p_air_thrust_stage, using equation 15

    end

    if volume_air < volume_bottle % if the volume of air is less than the volume of the ...
        bottle (signifies the bottle rocket is still in the water thrust stage)

        p_water_thrust_stage = p_abs_0 * (volume_air_0 / volume_air)^(gamma); % pressure in...
             the bottle, using equation 3

        thrust = 2 * discharge_coeff * area_throat * ...
            (p_water_thrust_stage - p_amb); % thrust of the bottle rocket [N], using ...
                equation 5

        dvolumeair_dt = discharge_coeff * area_throat * sqrt((2 / rho_water) * ...
```

```
            (p_abs_0 * ((volume_air_0 / volume_air)^(gamma)) - p_amb)); % change in volume ...
                over change in time, using equation 9

        dmasstot_dt = - discharge_coeff * rho_water * area_throat * ...
            sqrt((2 * (p_water_thrust_stage - p_amb)) / rho_water); % change in the total ...
                mass of the bottle rocket over change in time, using equation 10

        dmassair_dt = 0; % change in mass of air over change in time, using equation 24 (...
            zero for the water thrust stage)

    elseif p_air_thrust_stage > p_amb % if p_air_thrust_stage is greater than p_amb (...
        signifies the bottle rocket is in the air thrust phase)

        p_critical = p_air_thrust_stage * ...
            ((2 / (gamma + 1))^(gamma / (gamma - 1))); % critical pressure in the bottle, ...
                using equation 16

        if p_critical > p_amb % if p_critical is greater than p_amb (checking for chocked ...
            flow)

          mach_exit = 1; % definition of chocked flow

          temp_exit = (2 / (gamma + 1)) * temp_air_thrust_stage; % exit temperature, using ...
              equation 18
          p_exit = p_critical; % exit pressure equals critical pressure if at chocked flow,...
               using equation 18
          rho_exit = p_exit / (gas_const_air * temp_exit); % exit density, using equation ...
              18

          vel_exit = sqrt(gamma * gas_const_air * temp_exit); % exit velocity, using ...
              equation 17

        else % if not chocked flow

            mach_exit = sqrt((2 / (gamma - 1)) * ...
                (((p_air_thrust_stage / p_amb)^((gamma - 1) / gamma)) - 1)); % exit mach ...
                    number, using equation 19
            temp_exit = temp_air_thrust_stage / (1 + ((gamma - 1) / 2) * ...
                (mach_exit)^(2)); % exit temperature, using equation 20
            p_exit = p_amb; % exit pressure equals ambient pressure, using equation 20
            rho_exit = p_exit / (gas_const_air * temp_exit); % exit density, using equation...
                 20

            vel_exit = mach_exit * sqrt(gamma * gas_const_air * temp_exit); % exit velocity...
                , using equation 21

        end

        dmassair_dt = -1 * ...
            (discharge_coeff * rho_exit * area_throat * vel_exit); % change in mass of the ...
                air in the rocket over change in time, using equation 23

        thrust = (-1 * dmassair_dt * vel_exit) + (p_amb - p_exit) * ...
            area_throat; % thrust of the rocket, using equation 22

        dmasstot_dt = dmassair_dt; % change in total mass of the bottle rocket over time, ...
            using equation 24

        dvolumeair_dt = 0; % change in volme of air over change in time (zero because the ...
            voulme of air no longer changes since the bottle rocket is in air thrust stage)

    else % if the bottle rocket is in stage 3 flight, no thrust

        thrust = 0; % no thrust in stage 3
        dvolumeair_dt = 0; % volume of air is not changing in stage 3
        dmasstot_dt = 0; % mass of bottle rocket is not changing in stage 3
        dmassair_dt = 0; % mass of air is not changing in stage 3

```

```
    end

    % x, y, and z components of velocity
    dx_dt = vel_x;
    dy_dt = vel_y;
    dz_dt = vel_z;

    if sqrt((x - x_0)^(2) + (z - z_0)^(2)) < l_stand % if the bottle rocket hasn't lef the ...
        test stand

        % initial headings, component wise
        x_height = heading_0(1);
        y_height = heading_0(2);
        z_height = heading_0(3);

    else % if the bottle rocket has left the test stand

        % heading, component wise
        x_height = vel_rel(1) / abs(vel_rel_tot);
        y_height = vel_rel(2) / abs(vel_rel_tot);
        z_height = vel_rel(3) / abs(vel_rel_tot);

    end

    % acceleration in each component
    dvelx_dt = (thrust - drag) * (x_height / mass_rocket);
    dvely_dt = (thrust - drag) * (y_height / mass_rocket);
    dvelz_dt = (thrust - drag) * (z_height / mass_rocket) - g;

    % differentials for ode45
    datafunc(1) = dx_dt;
    datafunc(2) = dy_dt;
    datafunc(3) = dz_dt;
    datafunc(4) = dvelx_dt;
    datafunc(5) = dvely_dt;
    datafunc(6) = dvelz_dt;
    datafunc(7) = dmasstot_dt;
    datafunc(8) = dmassair_dt;
    datafunc(9) = dvolumeair_dt;
    datafunc = datafunc'; % get equationResults into a column vec for ode45

end
```

Sensitivity Analysis

```
%% Housekeeping

clc;
clear;
close all;

%% Constants

% Gloabl Variables Declaration

global g discharge_coeff rho_air_amb volume_bottle p_amb gamma rho_water ...
    diameter_throat area_throat area_bottle diameter_bottle gas_const_air ...
    mass_bottle cd p_gage_0 volume_water_0 temp_air_0 vel_0 theta_0 x_0 y_0 ...
    z_0 vel0_x vel0_y vel0_z l_stand mass_air_0 mass_rocket_0 heading_0 ...
    volume_air_0 p_abs_0 tspan windVelG windVelA;

% Constants

g = 9.81; % acceleration due to gravity [m/s^2]
discharge_coeff = 0.8; % discharge coefficient
rho_air_amb = 1.17358; % ambient air density [kg/m^3] GOLD ROCKET -> 1.17358 [kg/m^3] TA ...
    BASELINE -> 1.23478
volume_bottle = 0.002; % volume of the empty bottle [m^3]
p_amb = 99640; % atmospheric pressure [pa] GOLD ROCKET -> 99640 [pa] TA BASELINE -> 101592
gamma = 1.4; % ratio of specific heats for heat
rho_water = 1000; % density of water [kg/m^3]
diameter_throat = 2.1 / 100; % diameter of throat [m]
diameter_bottle = 10.5 / 100; % diameter of bottle [m]
area_throat = (1/4) * pi * (diameter_throat)^2; % area of throat [m^2]
area_bottle = (1/4) * pi * (diameter_bottle)^2; % area of bottle [m^2]
gas_const_air = 287; % gas constant of air [J/(kg*K)]
mass_bottle = 0.126; % mass of the empty bottle w/ cone and fins [kg]
cd = 0.327; % coefficient of drag
l_stand = 0.5; % langth of test stand [m]
tspan = [0, 5]; % integration time span [s]

% Max Distance Constants

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
p_abs_0 = p_gage_0 + p_amb; % absolutee pressure of air inside the bottle [pa]
volume_water_0 = 0.000993; % intitial volume of water inside bottle [m^3] MAXIMUM DISTANCE ...
    -> 5.25e-04 [m^3]
volume_air_0 = volume_bottle - volume_water_0; % intitial volume of air inside the bottle [...
    m^3]
temp_air_0 = 295.15; % initial temperature of air [K] GOLD ROCKET -> 295.15 TA BASELINE -> ...
    283.15
vel_0 = 0; % initial velocity of bottle rocket [m/s]
vel0_x = 0; % initial velocity of bottle rocket in the x direction [m/s]
vel0_y = 0; % initial velocity of bottle rocket in the y direction [m/s]
vel0_z = 0; % initial velocity of bottle rocket in the z direction [m/s]
theta_0 = 45; % intitial angle of bottle rocket [degrees] MAXIMUM DISTANCE -> 40
x_0 = 0; % initial downrange distance [m]
y_0 = 0; % initial crossrange distance [m]
z_0 = 0.25; % initial verital height [m]
windTheta = 0;
windSPDG = 0; % wind velocity [m/s]
windSPDA = 0;
windDir = [cosd(windTheta) sind(windTheta) 0]; % wind velocity direction [m/s]
windVelG = [windDir(1)*windSPDG, windDir(2)*windSPDG, windDir(3)*windSPDG];
windVelA = [windDir(1)*windSPDA, windDir(2)*windSPDA, windDir(3)*windSPDA];
heading_0 = [cosd(theta_0),0, sind(theta_0)]; % initial heading of bottle rocket [x,y]
mass_air_0 = (p_abs_0 * volume_air_0) / (gas_const_air * temp_air_0); % initial mass of air...
     [kg]
```

```
mass_rocket_0 = mass_bottle + (rho_water * volume_water_0) + mass_air_0; % total mass of ...
    the bottle rocket [kg]

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

figNum = 1;

%% Coefficient of Drag
cd_values = .2:.005:.5;
cd_x_dist = zeros(length(cd_values),1);
max_cd_dist_coords = [0 0];
j = 1;
for i = .2:.005:.5
    cd = i; % coefficient of drag

    [¬, data_cd] = ode45('ode45func2',tspan,initial_conditions);

    k = 1;
    while data_cd(k,3) ≥ 0
        k = k+1;
    end

    cd_x_dist(j) = (data_cd(k,1)+data_cd(k-1,1)) / 2;
    j = j + 1;

    if cd_x_dist(j-1) > max_cd_dist_coords(2)
        max_cd_dist_coords(2) = cd_x_dist(j-1);
        max_cd_dist_coords(1) = i;
    end

end

cd = 0.327; % coefficient of drag
[¬, data_cd] = ode45('ode45func2',tspan,initial_conditions);

k = 1;
while data_cd(k,3) ≥ 0
    k = k+1;
end
ta_cd_dist = [0.327, (data_cd(k,1)+data_cd(k-1,1)) / 2];

figure(figNum)
p1 = plot(cd_values,cd_x_dist,'o','linewidth',1.25);
hold on
p2 = plot(max_cd_dist_coords(1),max_cd_dist_coords(2),'og','linewidth',1.75);
p3 = plot(ta_cd_dist(1),ta_cd_dist(2),'or','linewidth',1.75);
%xline(0.327,'--k','Gold Rocket Value','LabelHorizontalAlignment','left','...
    LabelVerticalAlignment','bottom','linewidth',1.25);
title('Sesitivity Analysis of the Coefficient of Drag');
xlabel('Coefficient of Drag');
ylabel('Downrange Distance, [m]');
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');
hold off
figNum = figNum + 1;

%% Mass of Propellent
mass_values = 0.1*0.002:11/600000:0.6*0.002;
mass_values = mass_values.*rho_water;
mass_x_dist = zeros(length(mass_values),1);
max_dist_mass = [0 0];
j = 1;
for i = 0.1*0.002:11/600000:0.6*0.002

    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
```

```
    p_abs_0 = p_gage_0 + p_amb; % absolutee pressure of air inside the bottle [pa]
    volume_water_0 = i; % intitial volume of water inside bottle [m^3] MAXIMUM DISTANCE -> ...
        5.25e-04 [m^3]
    volume_air_0 = volume_bottle - volume_water_0; % intitial volume of air inside the ...
        bottle [m^3]
    temp_air_0 = 295.15; % initial temperature of air [K] GOLD ROCKET -> 295.15 TA BASELINE...
         -> 283.15
    vel_0 = 0; % initial velocity of bottle rocket [m/s]
    vel0_x = 0; % initial velocity of bottle rocket in the x direction [m/s]
    vel0_y = 0; % initial velocity of bottle rocket in the y direction [m/s]
    vel0_z = 0; % initial velocity of bottle rocket in the z direction [m/s]
    theta_0 = 45; % intitial angle of bottle rocket [degrees] MAXIMUM DISTANCE -> 40
    x_0 = 0; % initial downrange distance [m]
    y_0 = 0; % initial crossrange distance [m]
    z_0 = 0.25; % initial verital height [m]
    windTheta = 0;
    windSPDG = 0; % wind velocity [m/s]
    windSPDA = 0;
    windDir = [cosd(windTheta) sind(windTheta) 0]; % wind velocity direction [m/s]
    windVelG = [windDir(1)*windSPDG, windDir(2)*windSPDG, windDir(3)*windSPDG];
    windVelA = [windDir(1)*windSPDA, windDir(2)*windSPDA, windDir(3)*windSPDA];
    heading_0 = [cosd(theta_0),0, sind(theta_0)]; % initial heading of bottle rocket [x,y]
    mass_air_0 = (p_abs_0 * volume_air_0) / (gas_const_air * temp_air_0); % initial mass of...
         air [kg]
    mass_rocket_0 = mass_bottle + (rho_water * volume_water_0) + mass_air_0; % total mass ...
        of the bottle rocket [kg]

    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

    [¬, data_mass] = ode45('ode45func2',tspan,initial_conditions);

    k = 1;
    while data_mass(k,3) ≥ 0
        k = k+1;
    end

    mass_x_dist(j) = (data_mass(k,1)+data_mass(k-1,1)) / 2;
    j = j + 1;

    if mass_x_dist(j-1) > max_dist_mass(2) && j-1 ≠ 27
        max_dist_mass(2) = mass_x_dist(j-1);
        max_dist_mass(1) = i;
    end

end

    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
    p_abs_0 = p_gage_0 + p_amb; % absolutee pressure of air inside the bottle [pa]
    volume_water_0 = 0.000993; % intitial volume of water inside bottle [m^3] MAXIMUM ...
        DISTANCE -> 5.25e-04 [m^3]
    volume_air_0 = volume_bottle - volume_water_0; % intitial volume of air inside the ...
        bottle [m^3]
    temp_air_0 = 295.15; % initial temperature of air [K] GOLD ROCKET -> 295.15 TA BASELINE...
         -> 283.15
    vel_0 = 0; % initial velocity of bottle rocket [m/s]
    vel0_x = 0; % initial velocity of bottle rocket in the x direction [m/s]
    vel0_y = 0; % initial velocity of bottle rocket in the y direction [m/s]
    vel0_z = 0; % initial velocity of bottle rocket in the z direction [m/s]
    theta_0 = 45; % intitial angle of bottle rocket [degrees] MAXIMUM DISTANCE -> 40
    x_0 = 0; % initial downrange distance [m]
    y_0 = 0; % initial crossrange distance [m]
    z_0 = 0.25; % initial verital height [m]
    windTheta = 0;
    windSPDG = 0; % wind velocity [m/s]
    windSPDA = 0;
    windDir = [cosd(windTheta) sind(windTheta) 0]; % wind velocity direction [m/s]
```

```
    windVelG = [windDir(1)*windSPDG, windDir(2)*windSPDG, windDir(3)*windSPDG];
    windVelA = [windDir(1)*windSPDA, windDir(2)*windSPDA, windDir(3)*windSPDA];
    heading_0 = [cosd(theta_0),0, sind(theta_0)]; % initial heading of bottle rocket [x,y]
    mass_air_0 = (p_abs_0 * volume_air_0) / (gas_const_air * temp_air_0); % initial mass of...
         air [kg]
    mass_rocket_0 = mass_bottle + (rho_water * volume_water_0) + mass_air_0; % total mass ...
        of the bottle rocket [kg]

    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

    [¬, data_mass] = ode45('ode45func2',tspan,initial_conditions);

k = 1;
while data_mass(k,3) ≥ 0
    k = k+1;
end
ta_mass_dist = [0.000993*1000, (data_mass(k,1)+data_mass(k-1,1)) / 2];

figure(figNum)
p4 = plot(mass_values,mass_x_dist,'o','linewidth',1.25);
hold on
p5 = plot(max_dist_mass(1)*1000,max_dist_mass(2),'og','linewidth',1.75);
p6 = plot(ta_mass_dist(1),ta_mass_dist(2),'or','linewidth',1.75);
%xline(993/1000,'--k','Gold Rocket Value','LabelHorizontalAlignment','left','linewidth...
    ',1.25);
title('Sesitivity Analysis of the Mass of Propellant');
xlabel('Mass of Propellant, [kg]');
ylabel('Downrange Distance, [m]');
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]']});
hold off
figNum = figNum + 1;

%% Water Density
    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
    p_abs_0 = p_gage_0 + p_amb; % absolutee pressure of air inside the bottle [pa]
    volume_water_0 = 0.000993; % intitial volume of water inside bottle [m^3] MAXIMUM ...
        DISTANCE -> 5.25e-04 [m^3]
    volume_air_0 = volume_bottle - volume_water_0; % intitial volume of air inside the ...
        bottle [m^3]
    temp_air_0 = 295.15; % initial temperature of air [K] GOLD ROCKET -> 295.15 TA BASELINE...
         -> 283.15
    vel_0 = 0; % initial velocity of bottle rocket [m/s]
    vel0_x = 0; % initial velocity of bottle rocket in the x direction [m/s]
    vel0_y = 0; % initial velocity of bottle rocket in the y direction [m/s]
    vel0_z = 0; % initial velocity of bottle rocket in the z direction [m/s]
    theta_0 = 45; % intitial angle of bottle rocket [degrees] MAXIMUM DISTANCE -> 40
    x_0 = 0; % initial downrange distance [m]
    y_0 = 0; % initial crossrange distance [m]
    z_0 = 0.25; % initial verital height [m]
    windTheta = 0;
    windSPDG = 0; % wind velocity [m/s]
    windSPDA = 0;
    windDir = [cosd(windTheta) sind(windTheta) 0]; % wind velocity direction [m/s]
    windVelG = [windDir(1)*windSPDG, windDir(2)*windSPDG, windDir(3)*windSPDG];
    windVelA = [windDir(1)*windSPDA, windDir(2)*windSPDA, windDir(3)*windSPDA];
    heading_0 = [cosd(theta_0),0, sind(theta_0)]; % initial heading of bottle rocket [x,y]
    mass_air_0 = (p_abs_0 * volume_air_0) / (gas_const_air * temp_air_0); % initial mass of...
         air [kg]
    mass_rocket_0 = mass_bottle + (rho_water * volume_water_0) + mass_air_0; % total mass ...
        of the bottle rocket [kg]


density_values = 1000:1:1060;
density_x_dist = zeros(length(density_values),1);
```

```
max_dist_density = [0 0];
j = 1;
for i = 1000:1:1060

    rho_water = i; % density of water [kg/m^3]

    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
    p_abs_0 = p_gage_0 + p_amb; % absolutee pressure of air inside the bottle [pa]
    volume_water_0 = 0.000993; % intitial volume of water inside bottle [m^3] MAXIMUM ...
        DISTANCE -> 5.25e-04 [m^3]
    volume_air_0 = volume_bottle - volume_water_0; % intitial volume of air inside the ...
        bottle [m^3]
    temp_air_0 = 295.15; % initial temperature of air [K] GOLD ROCKET -> 295.15 TA BASELINE...
         -> 283.15
    vel_0 = 0; % initial velocity of bottle rocket [m/s]
    vel0_x = 0; % initial velocity of bottle rocket in the x direction [m/s]
    vel0_y = 0; % initial velocity of bottle rocket in the y direction [m/s]
    vel0_z = 0; % initial velocity of bottle rocket in the z direction [m/s]
    theta_0 = 45; % intitial angle of bottle rocket [degrees] MAXIMUM DISTANCE -> 40
    x_0 = 0; % initial downrange distance [m]
    y_0 = 0; % initial crossrange distance [m]
    z_0 = 0.25; % initial verital height [m]
    windTheta = 0;
    windSPDG = 0; % wind velocity [m/s]
    windSPDA = 0;
    windDir = [cosd(windTheta) sind(windTheta) 0]; % wind velocity direction [m/s]
    windVelG = [windDir(1)*windSPDG, windDir(2)*windSPDG, windDir(3)*windSPDG];
    windVelA = [windDir(1)*windSPDA, windDir(2)*windSPDA, windDir(3)*windSPDA];
    heading_0 = [cosd(theta_0),0, sind(theta_0)]; % initial heading of bottle rocket [x,y]
    mass_air_0 = (p_abs_0 * volume_air_0) / (gas_const_air * temp_air_0); % initial mass of...
         air [kg]
    mass_rocket_0 = mass_bottle + (rho_water * volume_water_0) + mass_air_0; % total mass ...
        of the bottle rocket [kg]

    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

    [¬, data_density] = ode45('ode45func2',tspan,initial_conditions);

    k = 1;
    while data_density(k,3) ≥ 0
        k = k+1;
    end

    density_x_dist(j) = (data_density(k,1)+data_density(k-1,1)) / 2;
    j = j + 1;

    if density_x_dist(j-1) > max_dist_density(2) && j-1 ≠ 16
        max_dist_density(2) = density_x_dist(j-1);
        max_dist_density(1) = i;
    end

    if i == 1000
        ta_max_density = [i, density_x_dist(j-1)];
    end

end

figure(figNum)
p7 = plot(density_values,density_x_dist,'o','linewidth',1.25);
hold on
p8 = plot(max_dist_density(1),max_dist_density(2),'og','linewidth',1.75);
p9 = plot(ta_max_density(1),ta_max_density(2),'or','linewidth',1.75);
%xline(1000,'--k','Gold Rocket Value','LabelHorizontalAlignment','left','...
    LabelVerticalAlignment','bottom','linewidth',1.25);
title('Sesitivity Analysis of the Density of Propellant');
xlabel('Density of Propellant, [kg/m^3]');
```

```
ylabel('Downrange Distance, [m]');
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');
hold off
figNum = figNum + 1;

%% Launch Pad Angle
    rho_water = 1000; % density of water [kg/m^3]
    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
    p_abs_0 = p_gage_0 + p_amb; % absolutee pressure of air inside the bottle [pa]
    volume_water_0 = 0.000993; % intitial volume of water inside bottle [m^3] MAXIMUM ...
        DISTANCE -> 5.25e-04 [m^3]
    volume_air_0 = volume_bottle - volume_water_0; % intitial volume of air inside the ...
        bottle [m^3]
    temp_air_0 = 295.15; % initial temperature of air [K] GOLD ROCKET -> 295.15 TA BASELINE...
         -> 283.15
    vel_0 = 0; % initial velocity of bottle rocket [m/s]
    vel0_x = 0; % initial velocity of bottle rocket in the x direction [m/s]
    vel0_y = 0; % initial velocity of bottle rocket in the y direction [m/s]
    vel0_z = 0; % initial velocity of bottle rocket in the z direction [m/s]
    theta_0 = 45; % intitial angle of bottle rocket [degrees] MAXIMUM DISTANCE -> 40
    x_0 = 0; % initial downrange distance [m]
    y_0 = 0; % initial crossrange distance [m]
    z_0 = 0.25; % initial verital height [m]
    windTheta = 0;
    windSPDG = 0; % wind velocity [m/s]
    windSPDA = 0;
    windDir = [cosd(windTheta) sind(windTheta) 0]; % wind velocity direction [m/s]
    windVelG = [windDir(1)*windSPDG, windDir(2)*windSPDG, windDir(3)*windSPDG];
    windVelA = [windDir(1)*windSPDA, windDir(2)*windSPDA, windDir(3)*windSPDA];
    heading_0 = [cosd(theta_0),0, sind(theta_0)]; % initial heading of bottle rocket [x,y]
    mass_air_0 = (p_abs_0 * volume_air_0) / (gas_const_air * temp_air_0); % initial mass of...
         air [kg]
    mass_rocket_0 = mass_bottle + (rho_water * volume_water_0) + mass_air_0; % total mass ...
        of the bottle rocket [kg]

LA_values = 1:1:89;
LA_x_dist = zeros(length(LA_values),1);
tspan = [0,10];
max_dist_LA = [0 0];
j = 1;
for i = 1:1:89

    theta_0 = i; % intitial angle of bottle rocket [degrees] MAXIMUM DISTANCE -> 40
    heading_0 = [cosd(theta_0),0, sind(theta_0)]; % initial heading of bottle rocket [x,y]

    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

    [¬, data_LA] = ode45('ode45func2',tspan,initial_conditions);

    k = 1;
    while data_LA(k,3) ≥ 0
        k = k+1;
    end

    LA_x_dist(j) = (data_LA(k,1)+data_LA(k-1,1)) / 2;
    j = j + 1;

    if LA_x_dist(j-1) > max_dist_LA(2) && j-1 ≠ 42
        max_dist_LA(2) = LA_x_dist(j-1);
        max_dist_LA(1) = i;
    end

    if i == 45
```

```
        ta_max_LA = [i, LA_x_dist(j-1)];
    end

end

figure(figNum)
p10 = plot(LA_values,LA_x_dist,'o','linewidth',1.25);
hold on
p11 = plot(max_dist_LA(1),max_dist_LA(2),'og','linewidth',1.75);
p12 = plot(ta_max_LA(1),ta_max_LA(2),'or','linewidth',1.75);
%xline(45,'--k','Gold Rocket Value','LabelHorizontalAlignment','left','...
    LabelVerticalAlignment','bottom','linewidth',1.25);
title('Sesitivity Analysis of the Launch Angle');
xlabel('Launch Angle, [deg]');
ylabel('Downrange Distance, [m]');
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');
hold off
figNum = figNum + 1;
```

Error Ellipses

```matlab
function [] = error_ellipses(data,ta)

    %% Replace this section of code with your real data
    % Simulate and plot 100 data points, (YOU SHOULD USE REAL DATA HERE!)
    x = data(:,1);
    y = data(:,2);

    %%

    % Calculate covariance matrix
    P = cov(x,y);
    mean_x = mean(x);
    mean_y = mean(y);
    figure;
    plot(mean_x,mean_y,'y.','markersize',24)
    axis equal;
    grid on;
    xlabel('x, Downrange Distance [m]');
    ylabel('y, Crossrange Distance [m]');
    title('Landing Points with Error Ellipses');
    hold on;
    plot(x,y,'k.','markersize',6)
    if ta == 1
        plot(40.5,-8.61,'m.','markersize',24)
    end

    % Calculate the define the error ellipses
    n=100; % Number of points around ellipse
    p=0:pi/n:2*pi; % angles around a circle

    [eigvec,eigval] = eig(P); % Compute eigen-stuff
    xy_vect = [cos(p'),sin(p')] * sqrt(eigval) * eigvec'; % Transformation
    x_vect = xy_vect(:,1);
    y_vect = xy_vect(:,2);

    % Plot the error ellipses overlaid on the same figure
    plot(1*x_vect+mean_x, 1*y_vect+mean_y, 'b', 'Linewidth', 1.5)
    plot(2*x_vect+mean_x, 2*y_vect+mean_y, 'g', 'Linewidth', 1.5)
    plot(3*x_vect+mean_x, 3*y_vect+mean_y, 'r', 'Linewidth', 1.5)
    if ta == 1
        legend('Center','Landing Spots','Actual Landing Spot','1\sigma','2\sigma','3\sigma'...
            ,'location','southeast');
    else
        legend('Center','Landing Spots','1\sigma','2\sigma','3\sigma');
        semimajor = max(3*x_vect);
        semiminor = max(3*y_vect);
        area = pi*semimajor*semiminor;
        fprintf(['The semimajor axis is ' num2str(semimajor) ' [m]. The semiminor axis is '...
             num2str(semiminor) ' [m]. The area is ' num2str(area) ' [m^2]. \n']);
    end

end
```