Table of Contents

ASEN 2012 Project 2: Bottle Rocket Modeling

```
%Zak Reichenbach
%This code was developed to use ode45 and numerical integration on 7
%properties( Xposition, Zposition, VelocityX, VelocityZ, Mass of the
%Rocket, Mass of the Air, Volume of Air) to model the trajectory of a
 rocket
% using Newton's laws of motion. Solving for theta outside of the
%function call as reccomended by the project designers.
%Student ID: 109050187
%
%Started:11/11/2020
%Last Edited:12/5/2020
```

House keeping

```
clc;clear;
```

Constants

```
Values =
 num2cell([9.81;0.8;0.961;0.002;12.1;1.4;1000;2.1;10.5;287;0.15;0.5;50;0.001;300;0
Names
 =["g","Cd","RoeAmb","VolBot","PAmb","Gamma","RoeWat",'Dt','Db','R','MBot','CD','P
C = cell2struct(Values,Names,1);  %Create Struct of Constants

C.Pgage = C.Pgage*6894.76;      %Convert to pascals
C.PAmb = C.PAmb *6894.76;       %Convert to pascals

P0 = C.Pgage + C.PAmb;          %Pabs = Pgage + Patm

At = pi/4*((C.Dt*10^-2))^2;             %m^2 Area of throat
Ab = pi/4*((C.Db*10^-2))^2;             %m^2 Area if bottle

VolAir0 = C.VolBot - C.VolWat0; %m^3 Initial Volume of Air
```

```
tspan = [0 5];                   %Span of whole test

%Find Initial Mass Values

M0_Water = C.RoeWat*(C.VolWat0);           %(12)
M0_Air = (P0*VolAir0)/(C.R*C.Tair0);       %(12)
M0 = C.MBot + M0_Water + M0_Air;           %(12)

C.Theta = deg2rad(C.Theta);          %Theta ---> Radians
%Initial State vectors
Properties0 = [C.Z0 C.X0 C.V0 C.V0 M0 M0_Air VolAir0];
```

Test Case Changed Properties

```
%Alone makes the rocket shoot to 80ft, max pressure in a 2-liter is
 896318
%N/m^2, Results - Test Case: Idealized and Easiest if you wanna be
 careful.
%      P0 = 610940;
%Angle needed to fire rocket to 80ft at max pressure
%     P0 = 689476;
%     C.Theta = deg2rad(31.4);
%The closest you can get by just changing the angle.
%     C.Theta = deg2rad(48);
%Large angles break the code,

%The initial volume of water changes need to be made in the struct
 (Setting
%the intial volume to 0.0011 makes it under shoot 3 meters and the
 more the volume increases the shorter it goes until the function
 breaks
%makeing the inital volume 0.0009 makes it GO FURTHER by .7 meters, I
%guess, but by 0.0008 it starts to fall short again.

%The Coefficient of drage changes also need to be made in the struct,
 the
%original experimentation was done with a coefficient of drag value of
 0.5.
%The minimum coefficient of drag of 0.3 by itself as a change sends
 the
%rocket an addtional 8.8 meters.

% %Now, to make the most efficient test case possible, assuming that
 CD=0.5
%     C.Theta = deg2rad(48);
%     P0 = 610940;
% %Using the ideal launch angle, using just enough pressure to get the
 rocket
% %to 80m.

%Just one last case to optimize for everything but the air pressure.
%      C.Theta = deg2rad(50);
%      C.CD = 0.3;
```

```
%      VolAir0 = 0.0011046;     %  works like C.VolWat0 = 0.0009
 without needing to be changed in the struct;
%
```

ODE CALCULATION

```
terminalCond = odeset('Events', @hitGround);

[t,Properties] = ode45(@(t,s) Rocket(t,s,C,At,Ab,P0,VolAir0,M0_Air),
 tspan, Properties0, terminalCond);
```

Further Calculations for finding theta, thrust, air pressure, and making our graphs

```
%Pulling out things that change in ODE45
z = Properties(:,1);
x = Properties(:,2);
V_Z = Properties(:,3);
V_X = Properties(:,4);
M_Tot = Properties(:,5);
M_Air = Properties(:,6);
Vol = Properties(:,7);

 %Find legth of ODE45 output
 n = length(Properties);

%Theta calculations
theta = (atan(z./x));

%Find Max Z and X distances
 [maxHeight,idx] = max(z);
 disp(maxHeight);
 [maxDistance,idx] = max(x);
 disp(maxDistance);

 %Finding the Change in phases



for i = 1:n

%Drag in X and Z direction
D_X = 0.5*C.RoeAmb*C.CD*Ab*V_X.^2;
D_Z = 0.5*C.RoeAmb*C.CD*Ab*V_Z.^2;
```

Phase 1

```
if(Vol(i) < C.VolBot)
       AirPressure(i) = P0*(VolAir0/Vol(i)).^C.Gamma;      %(3)

       Ve = sqrt((2/C.RoeWat)*(AirPressure(i) - C.PAmb));  %(7)
```

```
        Thrust(i) = 2*C.Cd * At * (AirPressure(i) - C.PAmb);

        roe_air(i) = M_Air(i)/Vol(i);

    else
        %Pressure of air when all water is gone
        p_end = P0*(VolAir0/C.VolBot)^C.Gamma;             %(13)
        %Temperature of air when all water is gone
        t_end = C.Tair0*(VolAir0/C.VolBot)^(C.Gamma-1);    %(13)
        %Pressure at end
         AirPressure(i) = p_end*(M_Air(i)/M0_Air)^C.Gamma;     %(14)

    end
```

Phase 2

```
if(Vol(i) >= C.VolBot)&&(AirPressure(i)>C.PAmb)
        %Air Pressure for pressurized air
        AirPressure(i) = p_end*(M_Air(i)/M0_Air)^C.Gamma;     %(14)
        %Change in vol
        dv_dt = 0;                                       %No more
 volume Change
        %Critical Pressure Calculation
        p_crit = AirPressure(i)*(2/(C.Gamma+1))^(C.Gamma/
(C.Gamma-1)); %(16)
        %Air Density Calculation
        roe_air(i) = M_Air(i)/Vol(i);                        %(15)
        %Air Temperature Calculation
        T(i) = AirPressure(i)/(roe_air(i)*C.R);                  %(15)

        %Flow
        if(p_crit > C.PAmb) %Choked
            %Exit Temp
            T_e(i) = (2/(C.Gamma+1))*T(i);                       %(18)
            %Exit velocity
            V_e(i) = sqrt(C.Gamma*C.R*T_e(i));                   %(17)
            %Exit Density
            roe_e = p_crit/(C.R*T_e(i));                      %(18)
            %Mass rate of change air
            M_dot_Air(i) = -C.Cd*roe_e*At*V_e(i);                %(23)
            %Mass rate of change total (The only changes are in air)
            M_dot = M_dot_Air(i);
            %Exit Pressure
            p_e = p_crit;
            %Thrust
            Thrust(i) = -(M_dot_Air(i)*V_e(i))+(C.PAmb-p_e)*At;

        elseif (p_crit <= C.PAmb) %Non-choked
            %Exit Mach Number
            M_e = sqrt((((AirPressure(i)/C.PAmb)^((C.Gamma-1)/
C.Gamma))-1)/((C.Gamma-1)/2));
            %Air Pressure for non-chocked flow
```

```
            AirPressure(i) =
 C.PAmb*(1+((C.Gamma-1)/2)*M_e^2)^(C.Gamma/(C.Gamma-1)); %(19)
            %Exit Temperature
            T_e(i) = T(i)/(1+(((C.Gamma-1)/2)*(M_e^2)));     %(20)
            %Exit Velocity
            V_e(i) = M_e*sqrt(C.Gamma*C.R*T_e(i));           %(21)
            %Exit Density
            roe_e = C.PAmb/(C.R*T_e(i));                  %(20)
            %Mass rate of change air
            M_dot_Air(i) = (-C.Cd)*roe_e*At*V_e(i);          %(23)
            %Mass rate of change total
            M_dot = M_dot_Air(i);                         %(24)
            p_e = C.PAmb;

            Thrust(i) = -(M_dot_Air(i)*V_e(i))+(C.PAmb-p_e)*At;
        end
end
```

Phase 3

```
   if(Vol(i)>= C.VolBot) && (AirPressure(i) <= C.PAmb)
        %Empty rocket
        Thrust(i) = 0;
        M_dot = 0;

    end

end

%Transpose Thrust and AirPressure vector to make it the same as the
 others
Thrust = Thrust';
AirPressure = AirPressure';

%Find the index where the phase changes.
 for i = 1:n
   if Vol(i) == Vol(i+1)
    EndPhase1 = i;
    break
   end

 end
Phase1Time = t(EndPhase1) ;
Phase1Location = x(EndPhase1);

 for i = 1:n
   if(AirPressure(i) <= C.PAmb)
   EndPhase2 = i;
   break
   end
 end
 Phase2Time = t(EndPhase2);
 Phase2Location = x(EndPhase2);
```

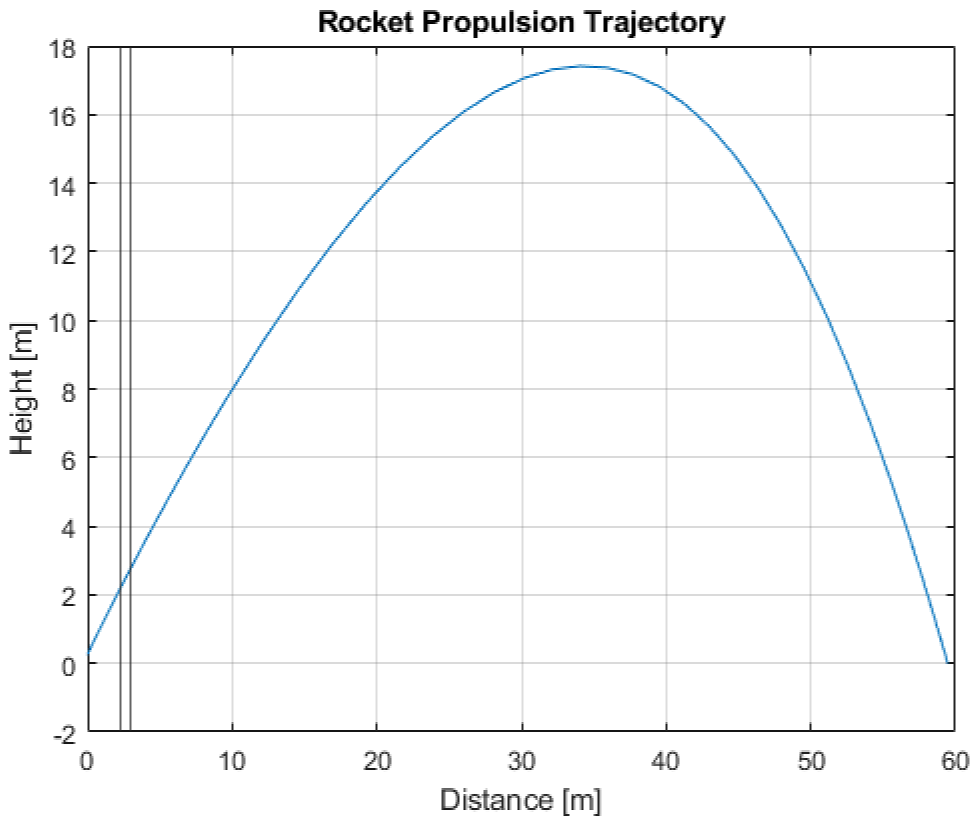
17.4165

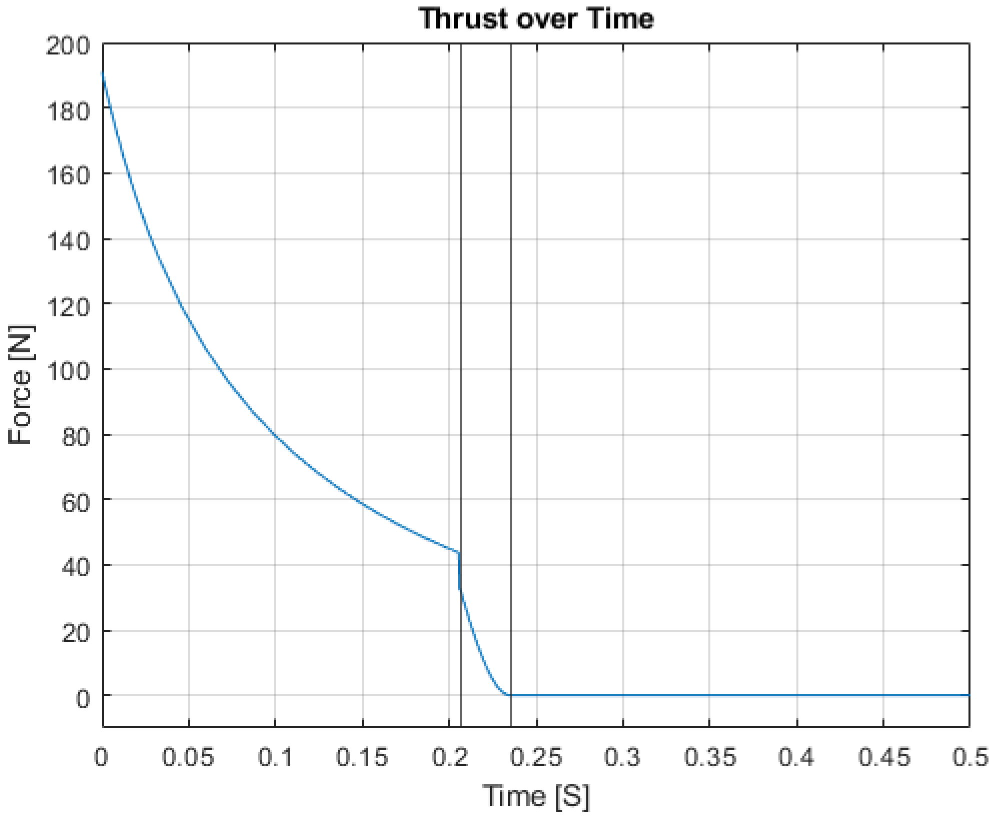
59.4692

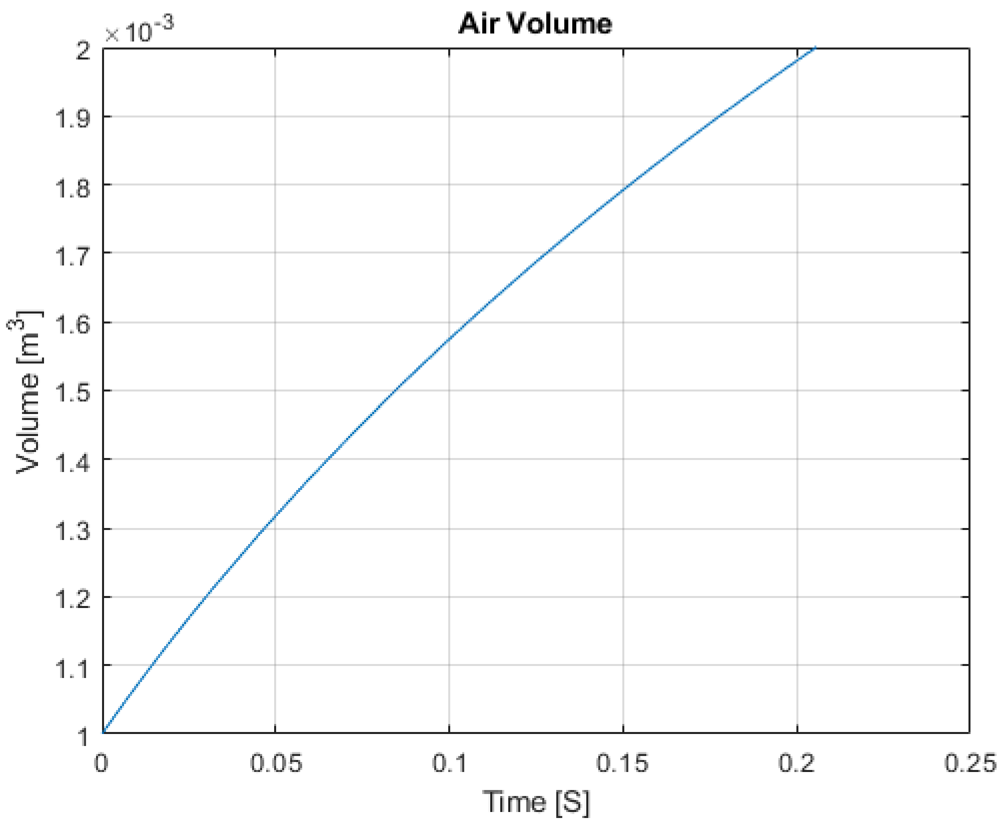
Plot

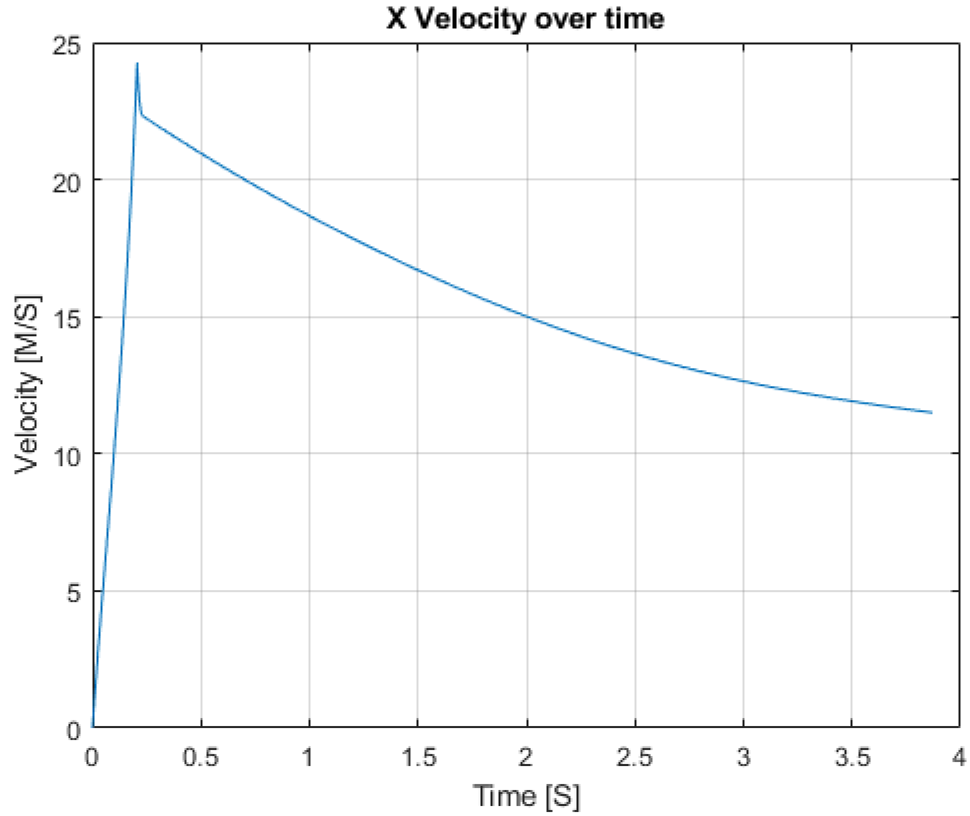
```
figure(1)
plot(x,z);
grid on
hold on
title('Rocket Propulsion Trajectory')
xlabel('Distance [m]')
ylabel('Height [m]')
% ylim([0 20])
xline(Phase1Location);
xline(Phase2Location);
hold off

figure(2)
plot(t,Thrust);
grid on
hold on
title('Thrust over Time')
xlabel('Time [S]')
ylabel('Force [N]')
ylim([-10 200])
xlim([0 0.5])
xline(Phase1Time);
xline(Phase2Time);
hold off

figure(3)
plot(t,Vol);
grid on
title('Air Volume')
xlabel('Time [S]')
ylabel('Volume [m^3]')
xlim ([0 0.25])
ylim ([1*10^-3 2*10^-3])

figure(4)
plot(t,V_X)
grid on
title('X Velocity over time')
xlabel('Time [S]')
ylabel('Velocity [M/S]')
xlim ([0 4])

figure(5)
plot(t,V_Z)
grid on
title('Z Velocity over time')
```

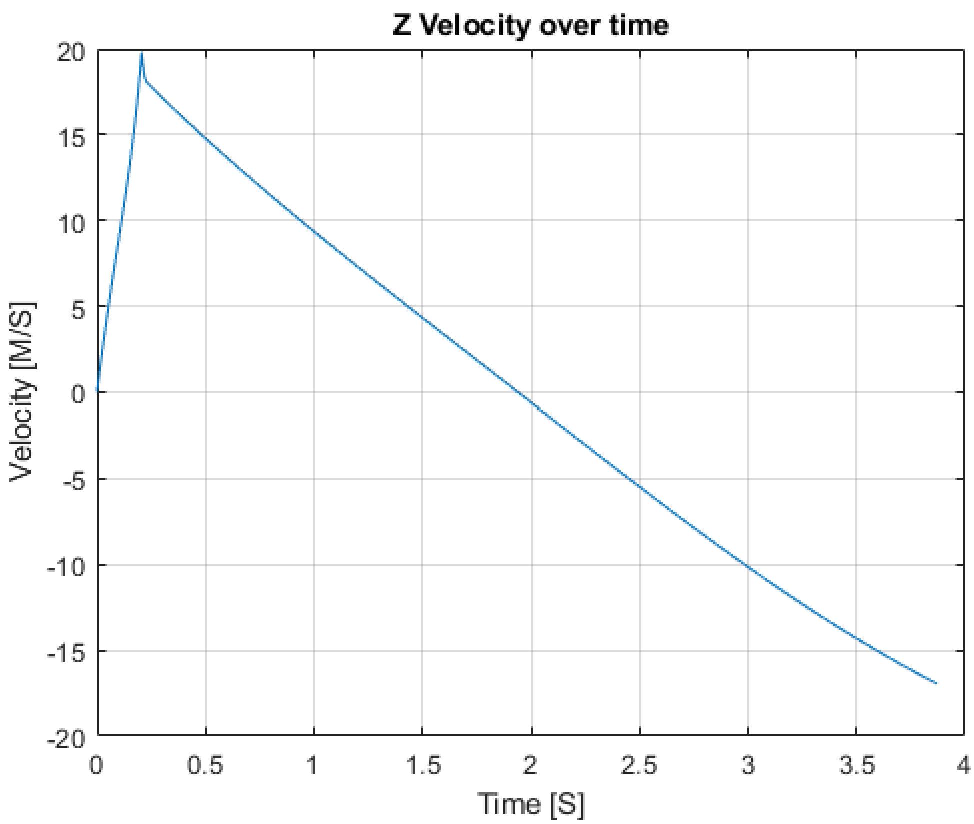
```
xlabel('Time [S]')
ylabel('Velocity [M/S]')
xlim([0 4])
```




Thrust over Time
Force [N]
Time [S]
200
180
160
140
120
100
80
60
40
20
0
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5


Air Volume
×10^-3
Volume [m^3]
Time [S]
2
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1
0
0.05
0.1
0.15
0.2
0.25


X Velocity over time
Velocity [M/S]
Time [S]
25
20
15
10
5
0
0
0.5
1
1.5
2
2.5
3
3.5
4


Z Velocity over time
Velocity [M/S]
Time [S]
20
15
10
5
0
-5
-10
-15
-20
0
0.5
1
1.5
2
2.5
3
3.5
4

Functions

```
 function state =
 Rocket(t,state_i,C,AreaThroat,AreaBottle,P0,Vol0,M0_Air)


 %State Vector
 state = zeros(7,1);
 Z = state_i(1);                         %Z position [m]
 X = state_i(2);                         %X position [m]
 V_Z = state_i(3);                       %Velocity [m/s]
 V_X = state_i(4);                       %Velocity [m/s]
 M_Tot = state_i(5);                     %Mass of Rocket [kg]
 M_Air = state_i(6);                     %Mass of Air    [kg]
 VolAir = state_i(7);                    %Volume of Air  [m^3]

%Heading Calculations
Velocity = sqrt(V_Z^2+V_X^2);            %[m/s]
%Heading position
if (sqrt((X-C.X0)^2+(Z-C.Z0)^2) < C.L)   %Launch angle 45 deg
    hz = sin(C.Theta);
    hx = cos(C.Theta);
else
    hz = V_Z/Velocity;                   %sin(theta)
    hx = V_X/Velocity;                   %cos(theta)
end

q_z = (1/2)*C.RoeAmb*V_Z^2;              %Dynamic Pressure
q_x = (1/2)*C.RoeAmb*V_X^2;
Drag_Z = q_z*C.CD*AreaBottle;            %Drag
Drag_X = q_x*C.CD*AreaBottle;
%%%%%%%%%CALCULATE THETA_DOT HERE

    %Water Propulsion
    if(VolAir < C.VolBot)
       %Change in Vol
       dv_dt = C.Cd * AreaThroat * sqrt((2/C.RoeWat) * (P0 * (Vol0/
VolAir)^C.Gamma - C.PAmb));    %(9)
       %Change in pressure
       AirPressure = P0 * (Vol0/VolAir)^C.Gamma;
                            %(3)
       %Change in thrust
       Thrust = 2 * C.Cd * AreaThroat * (AirPressure - C.PAmb);
                            %(8)
       %Mass rate of change of total mass
       M_dot = (-C.Cd) * AreaThroat * sqrt((2 * (AirPressure -
 C.PAmb)) * C.RoeWat);             %(10)
       %Mass rate of change of air
       M_dot_Air = 0;
       %Acceleration in X and Z
       Sum_FX = (Thrust*hx - Drag_X*hx);
       Sum_FZ = (Thrust*hz - Drag_Z*hz - (M_Tot*C.g));
       Accel_X = Sum_FX/M_Tot;
```

```
        Accel_Z = Sum_FZ/M_Tot;
    else %End of phase 1 conditions for phase 2
        %Pressure of air when all water is gone
        p_end = P0 * (Vol0/C.VolBot)^C.Gamma;             %(13)
        %Temperature of air when all water is gone
        t_end = C.Tair0 * (Vol0/C.VolBot)^(C.Gamma-1);    %(13)
        %Pressure at end
        AirPressure = p_end*(M_Air/M0_Air)^C.Gamma;       %(14)
    end

    %Pressurized Air
    if(VolAir >= C.VolBot)&&(AirPressure>C.PAmb)
        %Starting air Pressure for pressurized air phase
        AirPressure = p_end*(M_Air/M0_Air)^C.Gamma;     %(14)
        %Change in vol
        dv_dt = 0;                                      %No more
 volume Change
        %Critical Pressure Calculation
        p_crit = AirPressure*(2/(C.Gamma+1))^(C.Gamma/
(C.Gamma-1)); %(16)
        %Air Density Calculation
        roe_air = M_Air/VolAir;                       %(15)
        %Air Temperature Calculation
        T = AirPressure/(roe_air*C.R);                  %(15)

        %Flow
        if(p_crit > C.PAmb) %Choked
            %Exit Temp
            T_e = (2/(C.Gamma+1))*T;                    %(18)
            %Exit velocity
            V_e = sqrt(C.Gamma*C.R*T_e);                %(17)
            %Exit Density
            roe_e = p_crit/(C.R*T_e);                   %(18)
            %Mass rate of change air
            M_dot_Air = -C.Cd*roe_e*AreaThroat*V_e;     %(23)
            %Mass rate of change total (The only changes are in air)
            M_dot = M_dot_Air;
            %Exit Pressure
            p_e = p_crit;
            %Calculate thrust
            Thrust = M_dot_Air *V_e +(AreaThroat*(C.PAmb-p_e));
            %Calculate the sum of the forces on the rocket with
 respect to
            %the heading in the X and Z direction
            Sum_FX = Thrust*hx - Drag_X*hx;
            Sum_FZ = Thrust*hz - Drag_Z*hz - M_Tot*C.g;
            %Calculate the X and Y accelerations
            Accel_X = Sum_FX /M_Tot;
            Accel_Z = Sum_FZ /M_Tot;

        elseif (p_crit <= C.PAmb) %Non-choked
            %Exit Mach Number
            M_e = sqrt((((AirPressure/C.PAmb)^((C.Gamma-1)/
C.Gamma))-1)/((C.Gamma-1)/2));
```

```
            %Air Pressure for non-chocked flow
            AirPressure = C.PAmb*(1+((C.Gamma-1)/2)*M_e^2)^(C.Gamma/
(C.Gamma-1)); %(19)  DONT NEED IT BUT ITS THERE I GUESS
            %Exit Temperature
            T_e = T/(1+(((C.Gamma-1)/2)*(M_e^2)));      %(20)
            %Exit Velocity
            V_e = M_e*sqrt(C.Gamma*C.R*T_e);            %(21)
            %Exit Density
            roe_e = C.PAmb/(C.R*T_e);                   %(20)
            %Mass rate of change air
            M_dot_Air = (-C.Cd)*roe_e*AreaThroat*V_e;   %(23)
            %Mass rate of change total
            M_dot = M_dot_Air;                          %(24)
            p_e = C.PAmb;
            %Calculate thrust
            Thrust = M_dot_Air *V_e +(AreaThroat*(C.PAmb-p_e));
            %Calculate the sum of the forces on the rocket with
 respect to
            %the heading in the X and Z direction
            Sum_FX = Thrust*hx - Drag_X*hx;
            Sum_FZ = Thrust*hz - Drag_Z*hz - M_Tot*C.g;
            %Calculate the X and Y accelerations
            Accel_X = Sum_FX /M_Tot;
            Accel_Z = Sum_FZ /M_Tot;
        end

    end
    %Ballistic Phase
    if(VolAir>= C.VolBot) && (AirPressure <= C.PAmb)
        %Empty rocket
        Thrust = 0;
        M_dot = 0;
        M_dot_Air = 0;
        dv_dt = 0;
            %Calculate the sum of the forces on the rocket with
 respect to
            %the heading in the X and Z direction
            Sum_FX = Thrust*hx - Drag_X*hx;
            Sum_FZ = Thrust*hz - Drag_Z*hz - M_Tot*C.g;
            %Calculate the X and Y accelerations
            Accel_X = Sum_FX /M_Tot;
            Accel_Z = Sum_FZ /M_Tot;

    end

%Outputing the derivative of things on top of our state vectors
state(1) = V_Z;             %[m/s]
state(2) = V_X;             %[m/s]
state(3) = Accel_Z;         %[m/s^2]
state(4) = Accel_X;         %[m/s^2]
state(5) = M_dot;           %[kg/s]
state(6) = M_dot_Air;       %[kg/s]
state(7) = dv_dt;           %[m^3/s]
```

```
end


%This stops the ODE45 call when the rocket would hit the ground
function [value, isterminal, direction] = hitGround(t,state)
value  =(state(1) <= 0);
isterminal = 1;
direction =  0;
end
```

Published with MATLAB® R2019a