Table of Contents


```
% Author(s): Vishnu Duriseti
% Assignment title: Project 2
% Purpose: Modeling the trajectory of a bottle rocket
% Creation date: 11/30
% Revisions: idk bro
```


```
% Author(s): Vishnu Duriseti
% Assignment title: Project 2
% Purpose: Modeling the trajectory of a bottle rocket
% Creation date: 11/30
% Revisions: idk bro
```


Housekeeping

```
clear all; clc; close all;
```

calling all this stuff and using ode45

```
c = getConst();
vol_air_initial = c.vol_empty_bottle - c.vol_water_initial;
m_air_initial = (c.p_0 * vol_air_initial) / (c.BIG_R_BIG_R_BIG_R * c.temp_0);
m_water_initial = c.vol_water_initial*c.rho_water;
m_r_initial = m_air_initial + m_water_initial;

% in order:x, v_x, z,   v_z,    m_r,        vol_air,          m_air
kenLand = [0,  0, c.z_0, 0, m_r_initial, vol_air_initial, m_air_initial];
tspan = [0 5];

%options = odeset('RelTol',1e-8,'AbsTol',1e-10, "Events", @finale, "MaxStep",
 1e-4);
[t, y] = ode45(@(t, y) plsprayforme(t, y, c, tspan), tspan, kenLand); % idk
 why ode45 is acting up bro, the t vector is all ove rthe pplace

% extracting thrust
thrust = zeros(size(y));
for i = 1:length(y)
    [~, thrust] = plsprayforme(t, y(i, :), c);
end
```

phase 1
phase 1

phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1

phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1

phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1

phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1

phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1

```
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
phase 1
```

plotting wooooooooo

```
data = load('project2verification.mat');

figure(); hold on
plot(y(:, 1), y(:, 3), 'linewidth', 1.5)
plot(data.verification.distance, data.verification.height, '--',"LineWidth",
 1.5)
xlabel("X Distance [m]")
ylabel("Y Distance [m]")
%plot(t, y(:, 6))
grid on;
grid minor;

figure(); hold on
plot(t, thrust, 'linewidth', 1.5)
plot(data.verification.time, data.verification.thrust,"--","LineWidth",1.5)
xlabel("Time [s]")
ylabel("Thrust [N]")
%plot(t, y(:, 6))
grid on;
grid minor;
```

```
Warning: Imaginary parts of complex X and/or Y arguments ignored.
Warning: Imaginary parts of complex X and/or Y arguments ignored.
```

Function Library Function Library Function Library Function Library

```
%Yahoo!!!!!!!!!!

%bc for some reason ode is literally tweaking (ik i can't use this, hopefully
 it's just for testing)
%function [value, isterminal, direction] = finale(t, sumsum)
%       value = sumsum(2) <= 0;
%       isterminal = 1;
%       direction = 0;
%end
```

```
function constantinople = getConst()
    constantinople.g = 9.807; % m/s^2
    constantinople.c_dis = 0.8;
    constantinople.rho_air = 0.961; % kg/m^3
    constantinople.vol_empty_bottle = 0.002; % m^3
    constantinople.p_atm = 83426.56; % Pa
    constantinople.gamma = 1.4;
    constantinople.rho_water = 1000; % kg/m^3
    constantinople.throat_goat = 0.021; % m
    constantinople.bottle_diam = 0.105; % m
    constantinople.BIG_R_BIG_R_BIG_R = 287; % J/(kg*K)
    constantinople.m_b = 0.15; % kg
    constantinople.c_DDDD = 0.48;
    constantinople.p_0 = 358527.379; % initial guage pressure
    constantinople.vol_water_initial = 0.00095; % m^3
    constantinople.temp_0 = 300; % K
    constantinople.vel_0 = 0;
    constantinople.theta_0 = 42; % degrees --> initial angle of rocket
    constantinople.z_0 = 0.25; % m
    constantinople.testStandLEEEEENGTH = 0.5; % m
    constantinople.vol_air_initial = constantinople.vol_empty_bottle -
 constantinople.vol_water_initial;

    constantinople.throat_area51 = pi * (constantinople.throat_goat/2)^2;
    constantinople.bottle_area51 = pi * (constantinople.bottle_diam/2)^2;
end

function [mojodojocasahouse, F_thrust] = plsprayforme(t, ouroboros, C,
 tspan) % t for ode45, C for constants, ouroboros for state vector conditions
    %disp(['time (s): ', num2str(t)])
    x_i = ouroboros(1);
    v_xi = ouroboros(2);
    z_i = ouroboros(3);
    v_zi = ouroboros(4);
    m_ri = ouroboros(5);
    vol_ai = ouroboros(6);
    m_ai = ouroboros(7);

    % condition checker for phase 2
    p_end = C.p_0 * (C.vol_air_initial/vol_ai)^C.gamma;
    pressure_2 = (m_ai/(C.vol_air_initial * C.rho_air))^C.gamma * p_end;

    F_thrust = 0;
    vel_e = 0;
    dm_r = 0;
    dvol_air = 0;
    dm_air = 0;

    % Calculate heading based on velocity direction

    if x_i < 0.5 * cosd(C.theta_0) % || (v_xi == 0 && v_zi == 0)
        % If initial velocity is zero, set heading to a default value
        h_x = cosd(C.theta_0); h_y = sind(C.theta_0);
```

```
        heading = [h_x, h_y]';
    else
        % Calculate heading based on previous iteration's velocity
        heading = [v_xi, v_zi] / norm([v_xi, v_zi]');
    end

    % calculating drag
    q = 0.5 * C.rho_air * (v_xi^2+v_zi^2);
    Drag = q * C.c_DDDD * C.bottle_area51;
    Drag_orientation = heading * Drag;

    % for now pretend like I know how to differentiate between the
    % different phases of flight

    % 1) Watah phase
    if abs(C.vol_empty_bottle - vol_ai) >= 0.000001 %t < 0.1825
        pressure_1 = C.p_0 * (C.vol_air_initial/vol_ai)^C.gamma; % OMG WE GOT
 PRESSURE
        % WOOOOOOOOOOOOOOO
        vel_e = sqrt((2*(pressure_1 - C.p_atm))/C.rho_water);
        dm_water = (-C.c_dis * C.throat_area51 * (pressure_1 - C.p_atm)) /
 vel_e;
        F_thrust = dm_water * vel_e; % easier than before lol
        dm_air = 0;
        dvol_air = C.c_dis * C.throat_area51 * vel_e;
        %disp(['phase 1'])
    % 2) no Watah phase but its aih presshah
    elseif (pressure_2 - C.p_atm > 8000) && (abs(C.vol_empty_bottle - vol_ai)
 <= 0.000001) % arbitrary tolerance value %t > 0.1825 && t < 0.22 %
        % tolerance values of 0.08 atm for pressure, and 0.000001 m^3 for
        % volume (experimental values that I am tweaking based on how the code
 behaves)

        rho = m_ai/C.vol_empty_bottle;
        T = pressure_2 / (rho * C.BIG_R_BIG_R_BIG_R); % woah its the ideal gas
 law!
        p_star = pressure_2 * (2/(C.gamma+1))^(C.gamma / (C.gamma + 1));
        rho_e = 0; % initializing to use outside of loops

        if p_star > C.p_atm
            %daslkdjlaksjd
            p_e = p_star;
            T_e = (2*T)/(C.gamma+1);
            rho_e = p_e/(C.BIG_R_BIG_R_BIG_R * T_e);
            vel_e = sqrt(C.gamma * C.BIG_R_BIG_R_BIG_R * T_e);
        elseif p_star < C.p_atm
            % sum else
            p_e = C.p_atm;
            M_e = sqrt((2/(C.gamma-1))*((pressure_2/C.p_atm)^((C.gamma-1)/
C.gamma)-1)); % I derived this goddamn eq by hand
            T_e = (1 + 2/(C.gamma * M_e^2)) * T;
            vel_e = M_e * sqrt(C.gamma * C.BIG_R_BIG_R_BIG_R * T_e);
        end
        dm_air = -C.c_dis * rho_e * C.throat_area51 * vel_e;
```

```
        dm_r = dm_air;
        dvol_air = 0;
        F_thrust = -dm_air * vel_e + (pressure_2 - C.p_atm) * C.throat_area51;

        %disp(['phase 2 '])

    % 3) Ballistic Phase
    else % pressure_2 - C.p_atm <= 8000 && z_i > 0 %t > 0.22 || vel_z not
 right atm
        % tolerance based on one set for phase 2 - has to match ofc or code
        % will break

        F_thrust = 0;
        dm_r = 0;
        dvol_air = 0;
        dm_air = 0;

        %disp(['phase 3-'])
    end
    thrust_vect = F_thrust * heading;
    grrrrrra = [0, -C.g];
    a_x = (thrust_vect(1) - Drag_orientation(1))/m_ri + grrrrrra(1); % mg/m =
 g
    a_z = (thrust_vect(2) - Drag_orientation(2))/m_ri + grrrrrra(2);
    mojodojocasahouse = [v_xi, a_x, v_zi, a_z, dm_r, dvol_air, dm_air]';


    % creating state vector --> ode45 will convert mojodojocasahouse to
    % pepe which then becomes the condition ouroboros which feeds back into
    % plsprayforme (hence the name ouroboros)
    % pepe = [x, v_x, z, v_z, m_r, vol_air, m_air]; % not being used at all,
 just helps me to see what ode45(mdcs) will equal

end
```

phase 1

Published with MATLAB® R2022a