Table of Contents

ASEN 2012 Project 2 - Individual Portion

```
%_______________________________________________________________________
%   By: Ian Faber
%   SID: 108577813
%   Started: 11/12/21, 15:00
%   Finished: 11/14/21, 17:46
%
%   Runs a bottle rocket simulation subject to a set of initial
 parameters
%   through 3 different phases of flight: water thrust, air thrust,
 and
%   ballistic flight. Utilizes ODE45 and a custom EOM function, plots
%   relevant parameters to the rocket's flight
%
%   Somehow completed in just over 2 days!!!!!!!!!!!
%_______________________________________________________________________
```

Setup

```
% Housekeeping
clc; clear; close all;

% Get all constants from the const structure
const = getConst();

% Difference of Bottle and initial water volumes
VAirInit = const.Vbottle - const.VWaterInit;

% Need absolute pressure of air, also convert psi to Pa
PAirInit = (const.PGageInit+const.PAmb)*6894.76;

% Calculate rho w/ Ideal Gas EOS
rhoAirInit = (PAirInit)/(const.R*const.TAirInit);

% Calculate initial masses
mAirInit = rhoAirInit*VAirInit;
mWaterInit = const.rhoWater*const.VWaterInit;
```

```
mRocketInit = const.mBottle + mAirInit + mWaterInit;

% Calculate initial x and z velocities
vx0 = const.vInit*cosd(const.thetaInit);
vz0 = const.vInit*sind(const.thetaInit);

% Format the initial conditions vector, and by extension variables to
% integrate
X0 = [const.xInit; const.zInit; vx0; vz0; mRocketInit; mAirInit;
 VAirInit];

% Define events worthy of stopping integration
options = odeset('Events',@phase);
```

Simulation

```
% Integrate! Solves for the trajectory of the rocket by integrating
 the
% variables in X0 over tspan according to the derivative information
% contained in rocketEOM. Also stops integration according to
 "options," a
% predefined set of stopping conditions
[time, state] = ode45(@(t,state)rocketEOM(t,state,const), const.tspan,
 X0, options);

% Extract intermediate variables from rocketEOM for debugging,
 particularly
% weight, drag, thrust, and air pressure. Found this approach on the
 MATLAB
% forums.
[~,gravCell, dragCell, thrustCell, PairCell] =
 cellfun(@(t,state)rocketEOM(t,state.',const), num2cell(time),
 num2cell(state,2), 'uni', 0);

%Allocate space for intermediate variables
gravity = zeros(length(time),1);
drag = zeros(length(time),1);
thrust = zeros(length(time),1);
Pair = zeros(length(time),1);

% Extract intermediate variables from their cells
for i = 1:length(time)
    gravity(i) = norm(gravCell{i});
    drag(i) = norm(dragCell{i});
    thrust(i) = norm(thrustCell{i});
    Pair(i) = norm(PairCell{i});
end
```

Extraction

```
% Extract variables of interest
rocketX = state(:,1);
rocketZ = state(:,2);
```

```
rocketVx = state(:,3);
rocketVz = state(:,4);
rocketM = state(:,5);
rocketMair = state(:,6);
rocketV = state(:,7);

% Find maximum values of interest
maxRange = max(rocketX)
maxHeight = max(rocketZ)
maxVx = max(rocketVx)
maxVy = max(rocketVz)
maxThrust = max(thrust)
```

```
maxRange =

   60.2658


maxHeight =

   17.2294


maxVx =

   25.6057


maxVy =

   20.5830


maxThrust =

  191.0459
```

Plotting

```
% Plot the trajectory and variables of interest for the bottle
 rocket's
% flight!
f = figure();
f.Position = [100 100 740 740];


% Trajectory
subplot(5,2,1)
hold on;
title("Bottle Rocket Full Trajectory");
color_line3d(time, rocketX, rocketZ, zeros(1,length(time)));
```

```
xlim([0, 80]);
ylim([0, 30]);
xlabel("Range (m)");
ylabel("Height (m)");
hold off;

% X velocity
subplot(5,2,2)
hold on;
title("Bottle Rocket X-velocity");
plot(time, rocketVx);
xlim([0, 4]);
ylim([0, 30]);
xlabel("Time (sec)");
ylabel("X-velocity (m/s)");
hold off;

% Z velocity
subplot(5,2,3)
hold on;
title("Bottle Rocket Z-velocity");
plot(time, rocketVz);
xlim([0, 4])
ylim([-20, 25]);
xlabel("Time (sec)");
ylabel("Z-velocity (m/s)");
hold off;

% Air volume
subplot(5,2,4)
hold on;
title("Bottle Rocket Air volume");
plot(time, rocketV);
xlim([0, 0.25]);
ylim([1e-3, 2e-3]);
xlabel("Time (sec)");
ylabel("Air volume (m^3)");
hold off;

% Rocket mass
subplot(5,2,5)
hold on;
title("Bottle Rocket Mass");
plot(time, rocketM);
xlabel("Time (sec)");
ylabel("Rocket mass (kg)");
hold off;

% Air mass
subplot(5,2,6)
hold on;
title("Bottle Rocket Air Mass");
plot(time, rocketMair);
xlabel("Time (sec)");
```

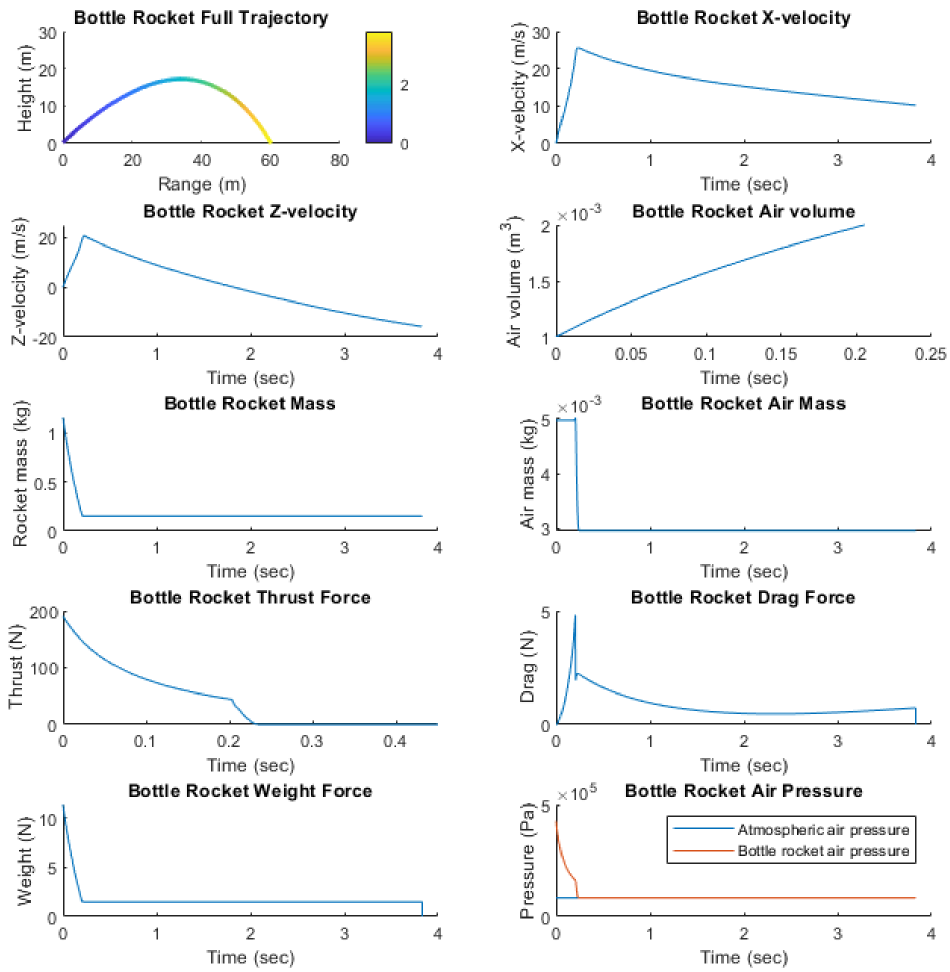
```
ylabel("Air mass (kg)");
hold off;

% Thrust
subplot(5,2,7)
hold on;
title("Bottle Rocket Thrust Force");
plot(time, thrust);
xlim([0, 0.45])
ylim([0, 200])
xlabel("Time (sec)");
ylabel("Thrust (N)");
hold off;

% Drag
subplot(5,2,8)
hold on;
title("Bottle Rocket Drag Force");
plot(time, drag);
xlabel("Time (sec)");
ylabel("Drag (N)");
hold off;

% Weight
subplot(5,2,9)
hold on;
title("Bottle Rocket Weight Force");
plot(time, gravity);
xlabel("Time (sec)");
ylabel("Weight (N)");
hold off;

% Air and ambient pressure
subplot(5,2,10)
hold on;
title("Bottle Rocket Air Pressure");
plot(time, const.PAmb*6894.76*ones(length(time),1));
plot(time, Pair);
xlabel("Time (sec)");
ylabel("Pressure (Pa)");
legend("Atmospheric air pressure", "Bottle rocket air pressure")
hold off;
```



rocketEOM Function

```
function [dX, fGrav, fDrag, fThrust, Pair] = rocketEOM(t,X,const)
% Function that defines the equations of motion and rates of change of
% various variables important for the flight of a water/air propelled
% bottle rocket.
%   Inputs: Time vector, t, state vector, X, formatted as
%   [x;z;vx;vz;m;mAir;Vair], constant structure, const
%
%   Outputs: Rates of change, dX, formatted as
%   [vx;vz;ax;az;dmdt;dmAdt;dVdt], intermediate weight force variable,
%   fGrav, intermediate drag variable, fDrag, intermediate thrust
 variable,
%   fThrust, intermediate air pressure variable, Pair
%

    % Extract current state variables
```

```
    x = X(1);
    z = X(2);
    vx = X(3);
    vz = X(4);
    m = X(5);
    mAir = X(6);
    Vair = X(7);

    % Do a sanity check on mass, it should never be below the mass of
the
    % bottle itself
    if(m < const.mBottle)
       m = const.mBottle + mAir;
    end

    % Calculate areas of various parts of the bottle
    Abottle = pi*(const.dBottle/200)^2; % Convert diameters from cm to
m
    Athroat = pi*(const.dThroat/200)^2; % Convert diameters from cm to
m

    % Define a velocity vector for heading calculations
    v = [vx; vz];

    % Calculate the ambient pressure in Pa
    PAmb = const.PAmb*6894.76; % Convert form psi to Pa

    % Define initial values of various state variables
    PAirInit = (const.PGageInit+const.PAmb)*6894.76;
    rhoAirInit = (PAirInit)/(const.R*const.TAirInit);
    VAirInit = const.Vbottle - const.VWaterInit;
    mAirInit = rhoAirInit*VAirInit;

    % Calculate currrent air density for state determination
    rhoAir = mAir/Vair;

    % State determination for the rocket heading state machine:
    %
    %   "ONSTAND" if the rocket has not travelled the length of the
launch
    %   stand, the heading will be fixed at the angle of the launch
stand
    %
    %   "FREEFLIGHT" if the rocket has travelled the length of the
launch
    %   stand, heading will be free to rotate as the rocket flies
through
    %   the air
    %
    if(norm([x (z-const.zInit)]) < const.lStand)
        headingState = "ONSTAND";
    else
        headingState = "FREEFLIGHT";
    end
```

```
    % State determination for the rocket flight phase state machine:
    %
    %   "WATERTHRUST" if there is still water in the rocket, i.e. the
    %   volume of air has not completely become the volume of the
bottle
    %
    %   "AIRTHRUST" if there is no more water in the rocket, yet the
air is
    %   still pressurized to above ambient pressure
    %
    %   "BALLISTIC" if the air in the bottle is no longer pressurized
and
    %   the density of air and ambient match
    %
    %   "GROUND" if the rocket's z coordinate aligns with ground
level,
    %   which stops the flight and simulation
    %
    if Vair < const.Vbottle && rhoAir > const.rhoAmb && z > 0
        flightState = "WATERTHRUST";
    elseif Vair >= const.Vbottle && rhoAir > const.rhoAmb && z > 0
        flightState = "AIRTHRUST";
        if Vair > const.Vbottle
           Vair = const.Vbottle;
        end
    elseif Vair >= const.Vbottle && rhoAir <= const.rhoAmb && z > 0
        flightState = "BALLISTIC";
        if rhoAir < const.rhoAmb
           rhoAir = const.rhoAmb;
        end
    else
        flightState = "GROUND";
    end

    % Rocket heading state machine
    switch headingState
        case "ONSTAND"
            % Heading fixed at "thetaInit"
            h = [cosd(const.thetaInit); sind(const.thetaInit)];
        case "FREEFLIGHT"
            % Heading based on velocity
            h = v/norm(v);
        otherwise
            % In case something funky happens ;)
            h = [0; 0];
    end

    % Rocket flight phase state machine
    switch flightState
        case "WATERTHRUST"
            % Calculate weight and drag forces
            fGrav = [0; -m*const.g];
            fDrag = -h*(0.5*const.Cdrag*Abottle*rhoAir*norm(v)^2);
```

```
            % Calculate air pressure according to equation 3
            Pair = PAirInit*(VAirInit/Vair)^const.gamma;

            % Calculate rocket mass rate of change with equation 10
            dmdt = -const.Cdis*Athroat*sqrt(2*const.rhoWater*(Pair-
PAmb));

            % Air mass is constant
            dmAdt = 0;

            % Calculate air volume rate of change with equation 9
            dVdt = const.Cdis*Athroat*sqrt((2/
const.rhoWater)*(PAirInit*((VAirInit/Vair)^const.gamma)-
PAmb)); %const.Cdis*Athroat*sqrt((2*(Pair-PAmb)/const.rhoWater));

            % Calculate thrust force with equation 8
            fThrust = h*(2*const.Cdis*Athroat*(Pair-PAmb));

        case "AIRTHRUST"
            %Calculate weight and drag forces
            fGrav = [0; -m*const.g];
            fDrag = -
h*(0.5*const.Cdrag*Abottle*const.rhoAmb*norm(v)^2);

            %Calculate air pressure according to equations 13 and 14
            Pair = PAirInit*((mAir*VAirInit)/
(mAirInit*Vair))^const.gamma;

            % Sanity check, air pressure should never go below ambient
            % pressure
            if(Pair < PAmb)
               Pair = PAmb;
            end

            % Calculate air temperature with equation 15-2
            Tair = Pair/(rhoAir*const.R);

            % Calculate critical temperature to characterize flow
            % characteristics of the air out of the bottle
            Pcrit = Pair*((2/(const.gamma + 1))^(const.gamma/
(const.gamma - 1)));

            % Critical pressure is greater than ambient, choked flow
            if(Pcrit > PAmb)
                % Calculate air exit characteristics
                Me = 1; % Mach number (1 if choked)
                Te = (2/(const.gamma+1))*Tair; % Exit temperature, eq.
 18-1

                Pe = Pcrit; % Exit pressure, eq. 18-3
                rhoE = Pe/(const.R*Te); % Exit density, eq. 18-2
            else % Critical pressure is at most ambient, unchoked flow
                % Calculate exit characteristics
```

```
                % Mach number, eq. 19 rearranged
                Me = sqrt( (2/(const.gamma-1)) * ( ((Pair/
PAmb)^((const.gamma-1)/const.gamma)) - 1 ) );

                % Exit temperature, eq. 20-1 rearranged
                Te = Tair/(1 + ((const.gamma - 1)/2)*Me^2);

                Pe = PAmb; % Exit pressure, eq. 20-3
                rhoE = PAmb/(const.R*Te); % Exit density, eq. 20-2
            end

            % Calculate air exit velocity using equation 21
            Ve = Me*sqrt(const.gamma*const.R*Te);

            % Calculate air mass rate of change with equation 23
            dmAdt = -const.Cdis*rhoE*Athroat*Ve;

            %Calculate rocket mass rate of change with equation 24
            dmdt = -const.Cdis*rhoE*Athroat*Ve;

            % Volume of air is constant
            dVdt = 0;

            % Calculate thrust force with equation 22
            fThrust = h*(-dmAdt*Ve + Athroat*(PAmb-Pe));

        case "BALLISTIC"
            % Air pressure is ambient
            Pair = PAmb;

            % Only forces acting on the rocket are gravity and drag
            fGrav = [0; -m*const.g];
            fDrag = -
h*(0.5*const.Cdrag*Abottle*const.rhoAmb*(norm(v)^2));
            fThrust = [0; 0];

            % All other state variables constant
            dmdt = 0;
            dmAdt = 0;
            dVdt = 0;

        otherwise
            % Air pressure is ambient, rocket has stopped moving
            Pair = PAmb;
            vx = 0;
            vz = 0;

            % No forces acting on the rocket, "fGrav" assumed to
 include
            % normal force from the ground
            fGrav = [0; 0];
            fDrag = [0; 0];
            fThrust = [0; 0];
```

```
            % All other state variables constant
            dmdt = 0;
            dmAdt = 0;
            dVdt = 0;

    end

    % Calculate the net force on the rocket with equation 1, modified
 for
    % sign convention
    fNet = fThrust + fDrag + fGrav;

    % Calculate x and z accelerations
    ax = fNet(1) / m;
    az = fNet(2) / m;

    % Assign rates of change
    dX = [vx; vz; ax; az; dmdt; dmAdt; dVdt];

    % Debugging/rocket monitoring stream
    %fprintf("Time: %.3f, Location: [%.3f, %.3f], Velocity: [%.3f,
 %.3f], Thrust: [%.3f, %.3f], Heading state: %s, Heading: [%.3f,
 %.3f], mass; %f kg, flight state: %s\n", t, x, z, vx, vz, fThrust(1),
 fThrust(2), headingState, h(1), h(2), m, flightState);

end
```

getConst function

```
function const = getConst()
% Defines a constant structure with values outlined in the project
 document
%   Inputs: None
%
%   Outputs: Constant structure, const
%

const.g = 9.81; % Acceleration from gravity, m/s^2

const.Cdis = 0.8; % Discharge coefficient

const.rhoAmb = 0.961; % Ambient air density, kg/m^3

const.Vbottle = 0.002; % Empty bottle volume, m^3

const.PAmb = 12.1; % Atmospheric pressure, psi

const.gamma = 1.4; % Air specific heat ratio

const.rhoWater = 1000; % Water density, kg/m^3

const.dThroat = 2.1; % Rocket throat diameter, cm
```

```
const.dBottle = 10.5; % Bottle diameter, cm

const.R = 287; % Gas constant of air, J/kg*K

const.mBottle = 0.15; % Empty bottle mass, kg

const.Cdrag = 0.5; % Drag coefficient

const.PGageInit = 50; % Initial bottle air gage pressure, psi

const.VWaterInit = 0.001; % Initial bottle water volume, m^3

const.TAirInit = 300; % Initial air temperature, K

const.vInit = 0; % Initial rocket velocity, m/s

const.thetaInit = 45; % Initial angle of rocket, deg

const.xInit = 0; % Initial horizontal distance, m

const.zInit = 0.25; % Initial vertical height, m

const.lStand = 0.5; % Length of the test stand

const.tspan = [0 5]; % Time span of integration [start stop], sec

end
```

phase function

```
function [value, isterminal, direction] = phase(t,X)
% Define events of interest to ODE45, specifically integration
 termination
% events
%   Inputs: Time vector, t, and state vector, X, formatted as
%   [x;z;vx;vz;m;mAir;Vair]
%
%   Outputs: Value to watch, value, whether the value will terminate
%   integration, isterminal, and what direction to watch the value
 change,
%   direction
%

%Extract current height
z = X(2);

value = z; % which variable to use (hint, this indicates when the z-
coordinate hits the GROUND level, not sea level)
isterminal = 1; % terminate integration? (0 or 1)
direction = -1; % test when the value first goes negative (-1) or
 positive (+1)
end
```

Color_line3d function (Taken from ASEN1320, commented out due to weird PDF publishing issues)

```
% function h = color_line3d(c, x, y, z)
% % color_line3 plots a 3-D "line" with c-data as color
% %
% %       color_line3d(c, x, y)
% %
% %  in:  x      x-data
% %       y      y-data
% %       z      z-data
% %       c      coloring
% %
%
% h = surface(...
%   'XData',[x(:) x(:)],...
%   'YData',[y(:) y(:)],...
%   'ZData',[z(:) z(:)],...
%   'CData',[c(:) c(:)],...
%   'FaceColor','interp',...
%   'EdgeColor','interp',...
%   'Marker','none', ...
%   'LineWidth',2);
%
% colorbar;
%
% end
```

Published with MATLAB® R2021a