Jack Kirschler & Tony Tonkovich - ME 6900 - 3/27/24

Project 3 – ODE45

Intro

Our Team was tasked with implementing the previous project's Simulink model into a MATLAB script using the ODE45 function. First, we made a simple projectile motion example using a ball with an initial vertical velocity. Next, we added our rocket model into the script to launch vertically until apogee. Finally, we implemented the turning equation to put our rocket on an orbital trajectory. After completing tuning the script, we compared the ODE45 model to the Simulink model.

Problem 1

To begin the simple projectile motion problem, we defined the initial conditions of the ball which consisted of a height of 0 m, and a velocity of 200 m/s. Then we defined an arbitrary time vector that started at 0 seconds till 100 seconds.

After defining initial conditions, the function was made that ODE45 would be solving which had the rate of height change and velocity change (acceleration of gravity). This was then fed to the ODE45 function with the time vector and the initial conditions to provide a solution to problem 1 which can be seen in Figure 1 of the appendix.

Problem 2

The same basic usage applied to problem 2 as it did in problem 1; define any initial conditions for the simulation and define all differential equations (time derivatives) of each condition within the ODE45 function. Starting off, the initial conditions for height, velocity, and mass were defined, their values were 0 m, 0 m/s, and 1167.2 kg respectively. The time derivative of height is the same as in problem 1 and is simply defined as the resultant velocity. However, for the derivative of velocity, there were some additions made from problem 1. Acceleration due to gravity remained the same in this problem, but we also needed to account for thrust and drag forces, this yields a derivative shown in equation 1, where T is thrust, Fd is drag force, m is the mass, and g is gravity.

$$\frac{dv}{dt} = \frac{T+Fd}{m} - g \qquad (1)$$

The final derivative that was defined was the change in mass. This was taken from our mass loss from motor burn during our initial OpenRocket flight simulation and is defined as -1.154 kg/s. With these initial conditions and time derivatives defined, there was only one final thing to implement before this model was ready to run; a condition to set the derivative of mass and thrust force equal to zero if the simulation time was equal to or greater than the burn time, which simply modeled the stopping of motor burn.

Problem 2 also required the use of three custom MATLAB functions that were derived from blocks that were used in our Simulink model. The first one was our atmosphere model that produced the air density and speed of sound at a given altitude, this function interpolates from input data that

we gave it. The second function also uses the same interpolation method to find a coefficient of drag at a given altitude, the data used to populate this function was taken from our OpenRocket simulation. Finally, we used a function to calculate drag, while this could have been done within the main script, we already had the function written so it made our code much cleaner having it as a function that we could call.

Finally, with all modeling complete, we could run the problem 2 script, a burn time of 130 s and average thrust of 52,000 N was used. We plotted altitude and velocity vs. time, the rocket's body forces vs. time, and the rocket mass vs. time, which all can be found in the appendix of this report. The given burn time and thrust values guided our rocket to an apogee of about 600 km and max velocity of 3.7 km/s. All plots produced looked as we expected.

Problem 3

Problem 3 built off problem 2 utilizing the same initial conditions with an additional initial horizontal position of 0 meter and an initial theta condition of 88 degrees. The function utilizes the same Cd interpolation function, atmosphere function, and drag function as problem 2.

The addition of a 2^{nd} dimension required the change of position to be equal to the velocity times the sin or cos of the rocket's angle. The acceleration is the same as problem 2 just with an added sin theta multiplied by gravities acceleration due to the flat earth assumption. Finally, an angular velocity equation was added to change the angle of the rocket as it launched, shown in equation 2.

After the function was made, it can be run by ODE45 which required parameter tuning to achieve the required turn and altitude. The results can be seen in the appendix under Figures 5, 6, 7, and 8.

Conclusion

This project provided us with yet another way of simulating rocket flight and built on the skills that we developed in the previous 2 projects. Simulating rocket flight by scripting in MATLAB and using ODE45 was the probably the most straight forward method used thus far as it consisted of defining initial conditions and the time derivatives for each. However, there still is some refining to do, as results from problem 3 slightly differed from that of our Simulink model. The average thrust used was 50 kN instead of 53 kN, burn time was 130s instead of 150s, and the constant used in the turning equation was 40.1 as opposed to 12.35 in Simulink. We came across lots of small time-step issues while tuning, which would produce singularities in our ODE45 solver, the process of tuning in Simulink was much more consistent.

Appendix

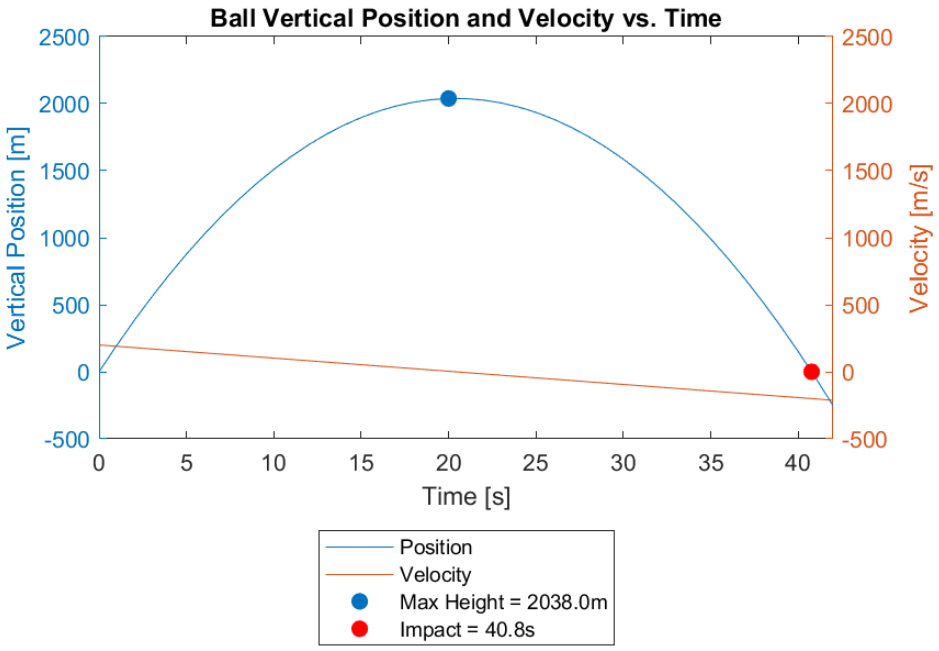

Figure 1: P1 velocity and position vs time

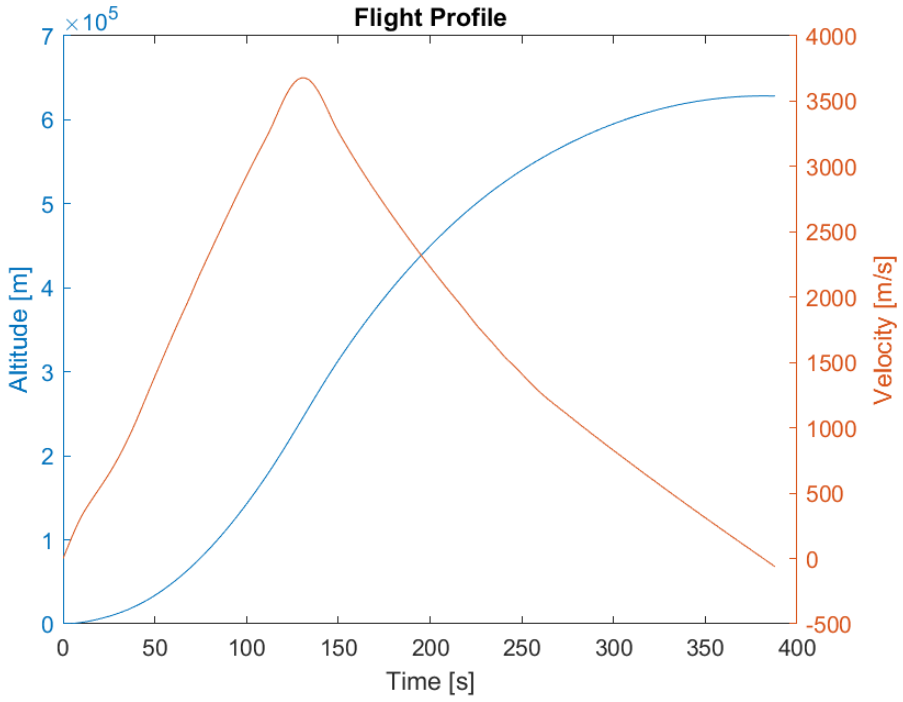

Figure 2: P2 Flight Profile

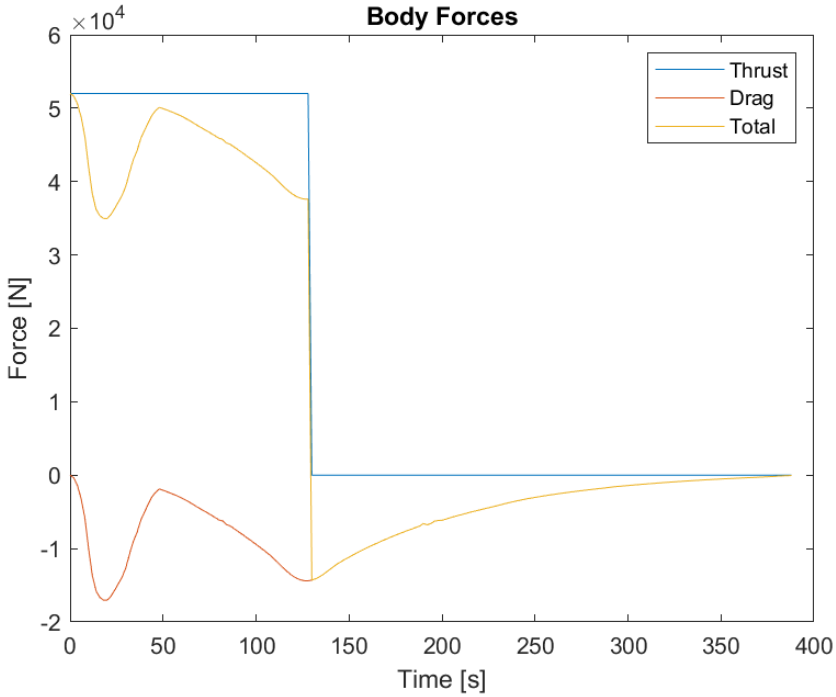

Figure 3: P2 Rocket Body Forces

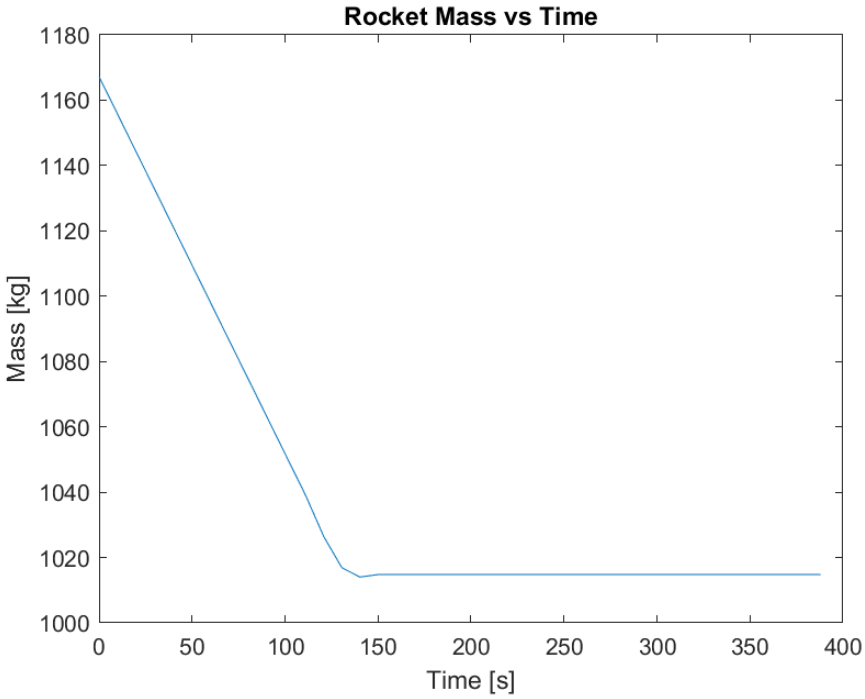

Figure 4: P2 Rocket Mass

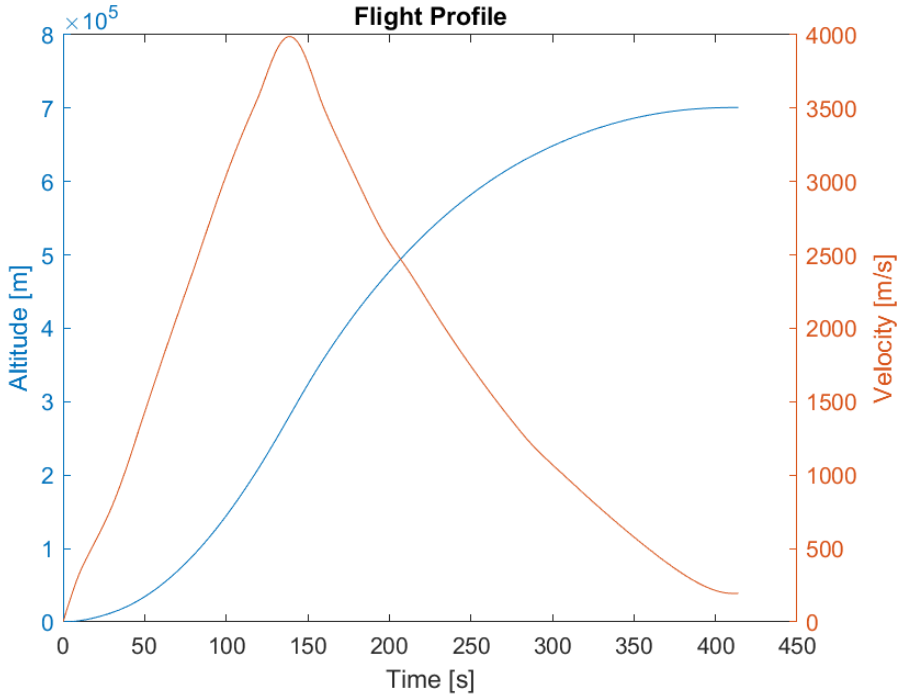

Figure 5: P3 Flight profile

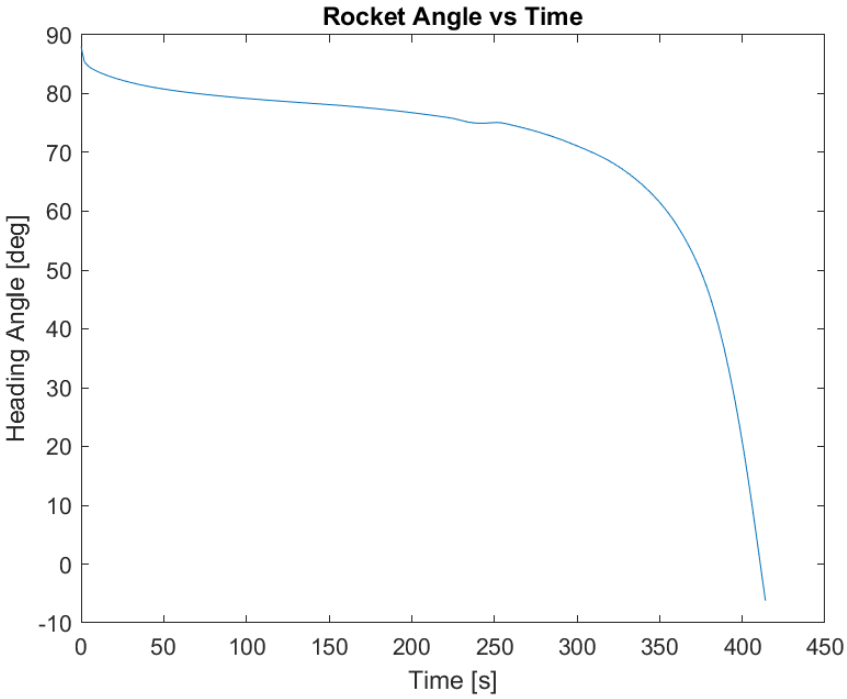

Figure 6: P3 Rocket Angle

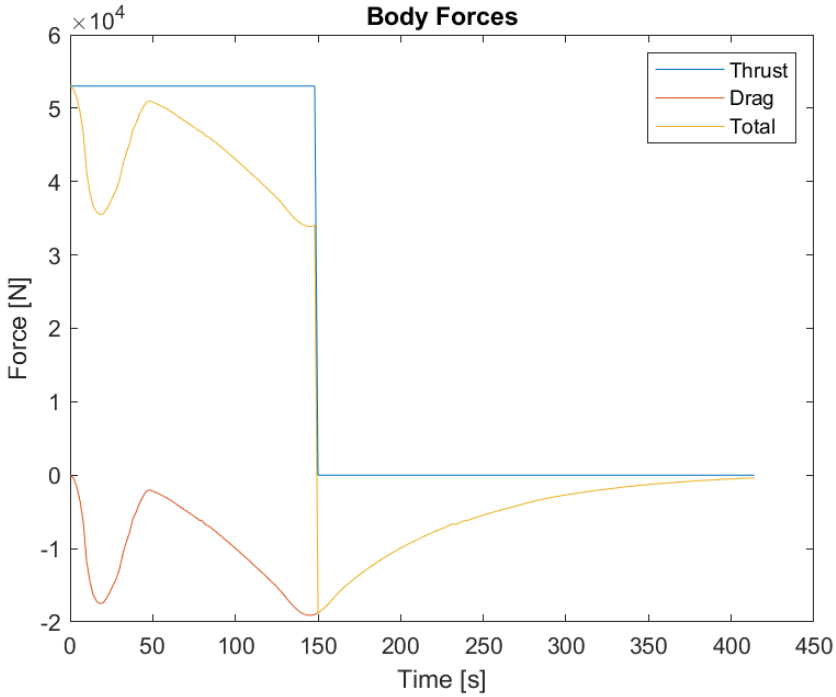

Figure 7: P3 Body Forces

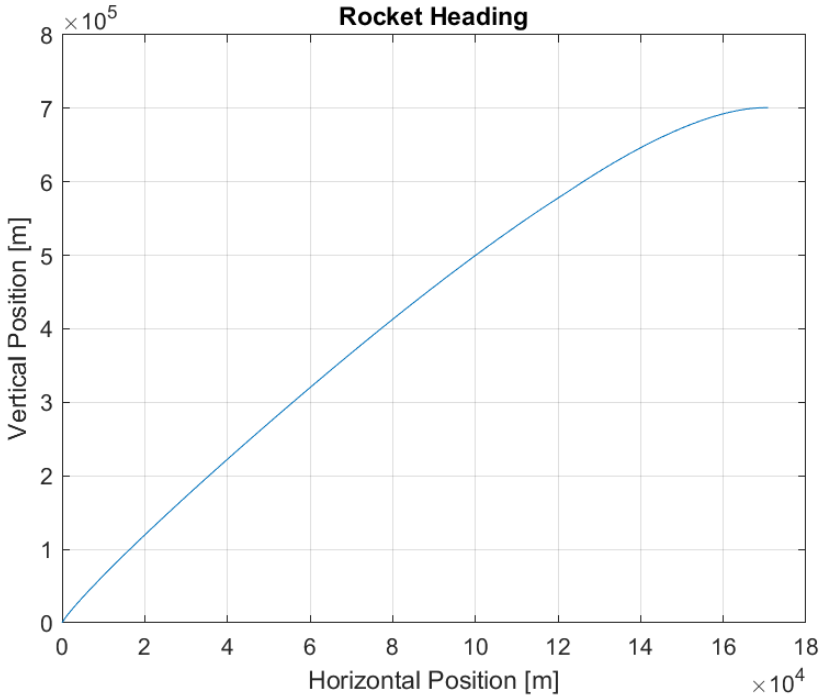

Figure 8: P3 Heading

Source Code

```
% ME 6900 Project 3 - Problem 1
clear, clc, close all

h0  = 0;
v0  = 200;
tf  = 42;

[t,x] = ode45(@ode, [0:1:tf], [h0, v0]);

hMax   = max(x(:,1));
tZeroH = roots([-4.905 200 0]);

figure;
yyaxis left
plot(t, x(:,1))
ylabel('Vertical Position [m]')
yyaxis right
plot(t, x(:,2))
ylabel('Velocity [m/s]')
hold on
plot(t(hMax == x(:,1)), hMax,'.', 'color', "#0072BD", 'MarkerSize', 25)
hold on
plot(tZeroH(2), 0, 'r.', 'MarkerSize', 25)
xlim([0,42]);
xlabel('Time [s]')
title('Ball Vertical Position and Velocity vs. Time')
legend({'Position','Velocity',sprintf('Max Height = %.1fm', hMax),...
        sprintf('Impact = %.1fs', tZeroH(2))}, 'Location', 'southoutside')
saveas(gcf, 'Figures/(P1)ball.png');

fprintf('Time till impact: %s [s]',tZeroH(2))

figure(2)
xlim([0,43])
ylim([0,2200])
xlabel('Time [s]')
ylabel('Vertical Position [m]')
hold on
pause(10)
for i=1:1:length([0:1:tf])
    pause(0.5)
    scatter(t(i), x(i,1), 'r')

end

function dxdt = ode(t,x)
    dx = x(2);
    dv = -9.81;
    dxdt = [dx; dv];
end
```

```
% ME 6900 Project 3 - Problem 2
clear, clc, close all

% Constants
tBurn      = 130;    %(s)
tSim       = 388;
mdot       = -1.154;  %(kg/s)

% Initial Conditions
v0         = 0;      % Initial velocity (m/s)
h0         = 0;      % Initial altitude (m)
m0         = 1167.2; % Initial mass (kg)

% ODE Solver
[t, x] = ode45(@(t,x) flight(t, x, mdot, tBurn), [0:2:tSim], [h0; v0; m0;]);

for i = 1:length(t)
    [~, fd(i), rho(i), cd(i), c(i), T(i)] = ...
    flight(t(i), x(i,:), mdot, tBurn);
end

fd = fd';
rho = rho';
cd = cd';
c = c';
T = T';
alt = x(:,1);
vel = x(:,2);
m   = x(:,3);
mach = vel./c;


figure;
yyaxis left
plot(t, alt)
ylabel('Altitude [m]')
yyaxis right
plot(t, vel)
ylabel('Velocity [m/s]');
xlabel('Time [s]')
title('Flight Profile')
saveas(gcf, 'Figures/P2FlightProfile.png');

figure;
plot(t, T, t, fd, t, T+fd);
ylabel('Force [N]');
xlabel('Time [s]')
title('Body Forces')
legend('Thrust','Drag','Total')
saveas(gcf, 'Figures/P2BodyForces.png');
```

```
figure;
plot(t, m);
ylabel('Mass [kg]');
xlabel('Time [s]');
title('Rocket Mass vs Time');
saveas(gcf, 'Figures/P2Mass.png');

figure;
plot(t, x(:,2));
ylabel('Acceleration [m/s^2]');
xlabel('Time [s]');
title('Acceleration vs Time');
saveas(gcf, 'Figures/P2accel.png');

figure(1)
xlim([0,388])
ylim([0,700000])
xlabel('Time [s]')
ylabel('Vertical Position [m]')
hold on
pause(10)
for i=1:1:length([0:2:tSim])
    pause(0.1)
    scatter(t(i), x(i,1), 'r')

end

function [dxdt, fd, rho, cd, c, T] = flight(t, x, mdot, tBurn)

    T = 52000;
    A = 0.5;

    h = x(1);
    v = x(2);
    m = x(3);

    [rho, c] = atmosmodel(h);
    cd = cd_interp(v,c);
    fd = drag(rho,v,cd,A);

    if t >= tBurn
        dm = 0;
        T = 0;
    else
        dm = mdot;
    end

    dh = x(2);
    dv = (T + fd) / m - 9.81;
    dxdt = [dh; dv; dm;];

end
```

```
% ME 6900 Project 3 - Problem 3
clear, clc, close all

% Constants
tBurn      = 150;    %(s)
tSim       = 415;
mdot       = -1.154;  %(kg/s)

% Initial Conditions
v0         = 1;      % Initial velocity (m/s)
h0         = 0;      % Initial altitude (m)
x0         = 0;      % Initial hor. pos.(m)
m0         = 1167.2; % Initial mass (kg)
th0        = 88;     % Initial body angle (deg)

% ODE Solver
[t, x] = ode45(@(t,x) flight(t, x, mdot, tBurn), [0:2:tSim], [h0; v0; m0; x0;
th0;]);

for i = 1:length(t)
    [~, fd(i), rho(i), cd(i), c(i), T(i)]= flight(t(i), x(i,:), mdot,tBurn);
end

fd   = fd'; rho  = rho'; cd   = cd'; c    = c'; T    = T';
alt  = x(:,1); vel  = x(:,2); m    = x(:,3); hor  = x(:,4);
th   = x(:,5); mach = vel./c;

figure;
yyaxis left
plot(t, alt)
ylabel('Altitude [m]')
yyaxis right
plot(t, vel)
ylabel('Velocity [m/s]');
xlabel('Time [s]')
title('Flight Profile')
saveas(gcf, 'Figures/P3FlightProfile.png');


figure;
plot(hor, alt)
xlabel('Horizontal Position [m]')
ylabel('Vertical Position [m]')
grid on
title('Rocket Heading')
saveas(gcf, 'Figures/P3heading.png');

figure;
plot(t, th)
xlabel('Time [s]')
ylabel('Heading Angle [deg]')
title('Rocket Angle vs Time')
saveas(gcf, 'Figures/P3Angle.png');
```

```
figure;
plot(t, T, t, fd, t, T+fd);
ylabel('Force [N]');
xlabel('Time [s]')
title('Body Forces')
legend('Thrust','Drag','Total')
saveas(gcf, 'Figures/P3BodyForces.png');

figure;
plot(t, m)
xlabel('Time [s]')
ylabel('Mass [kg]')
title('Rocket Mass vs Time')
saveas(gcf, 'Figures/P3mass.png');

figure(1)
xlim([0,180000])
ylim([0,800000])
xlabel('Horizontal Position [m]')
ylabel('Vertical Position [m]')
hold on
pause(10)
for i=1:1:length([0:2:tSim])
    pause(0.1)
    scatter(hor(i), alt(i,1), 'r')

end

function [dxdt, fd, rho, cd, c, T] = flight(t, x, mdot, tBurn)

    T = 53000;
    A = 0.5;
    g = 9.81;
    m_cst = 40.1;
    h = x(1);
    v = x(2);
    m = x(3);
    th = x(5);

    [rho, c] = atmosmodel(h);
    cd = cd_interp(v,c);
    fd = drag(rho,v,cd,A);

    if t >= tBurn
        dm = 0;
        T = 0;
    else
        dm = mdot;
    end
    dh = v*sind(th);
    dx = v*cosd(th);
    dv = (T + fd) / m - g*sind(th);
    dth = m_cst*(-(g*cosd(th))/v);
    dxdt = [dh; dv; dm; dx; dth;];
end
```

Helper Functions

```
function [cdcurrent] = cd_interp(v,c)

    % Sim Data
    mach = load('Sim_Data\mach2.mat').mach;
    cd = load('Sim_Data\cd2.mat').cd;

    machcurrent = v/c;
    cdcurrent = interp1(mach, cd, machcurrent, 'nearest','extrap');


end
```

```
function [density, c] = atmosmodel(alt)

    c_data = load('Atmosphere_Model/c.mat').alt;
    rho_data = load('Atmosphere_Model/rho.mat').alt;
    alt_data = load('Atmosphere_Model/altitude.mat').alt;
    density = interp1(alt_data, rho_data, alt, 'linear');
    c = interp1(alt_data, c_data, alt, 'linear');

    density(alt < min(alt_data)) = rho_data(1);
    density(alt > max(alt_data)) = rho_data(end);
    c(alt < min(alt_data)) = c_data(1);
    c(alt > max(alt_data)) = c_data(end);

end
```

```
function fd = drag(rho,v,Cd,A)

    if v < 0
        fd = (1/2)*rho*v^2*Cd*A;
    else
        fd = (-1/2)*rho*v^2*Cd*A;
    end
end
```