Table of Contents


```
% Author(s): Andrew Patella
% Assignment title: Project 2
% Purpose: Model how changing initial parameters changes the trajectory of
% a bottle rocket
% Creation date: 11/06/2023
% Revisions: N/A

close all; clear; clc;
tic

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% This script calculates the trajectories for different initial
% parameters.

% Initially, it uses 8 points for plotting trajectories. This is because
% too many becomes cluttered on the plots.

% The second part calculates trajectories for n parameter values between
% initial and final values for a scatter plot. More points means more
% certainty in the prediction of optimal parameter value because it
% has more resolution so the approsimation is more accurate.

% The third part of this function is a process of refining initial
% conditions to get the trajectory as close as possible to 80 m.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

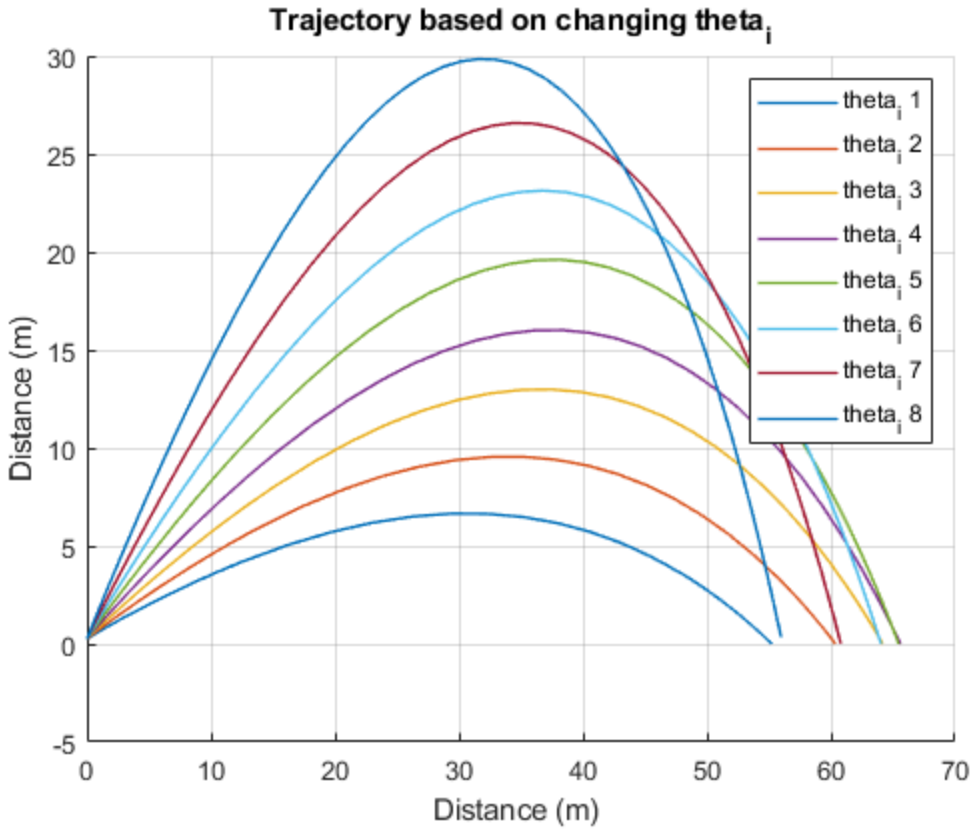
Constants and initial conditions

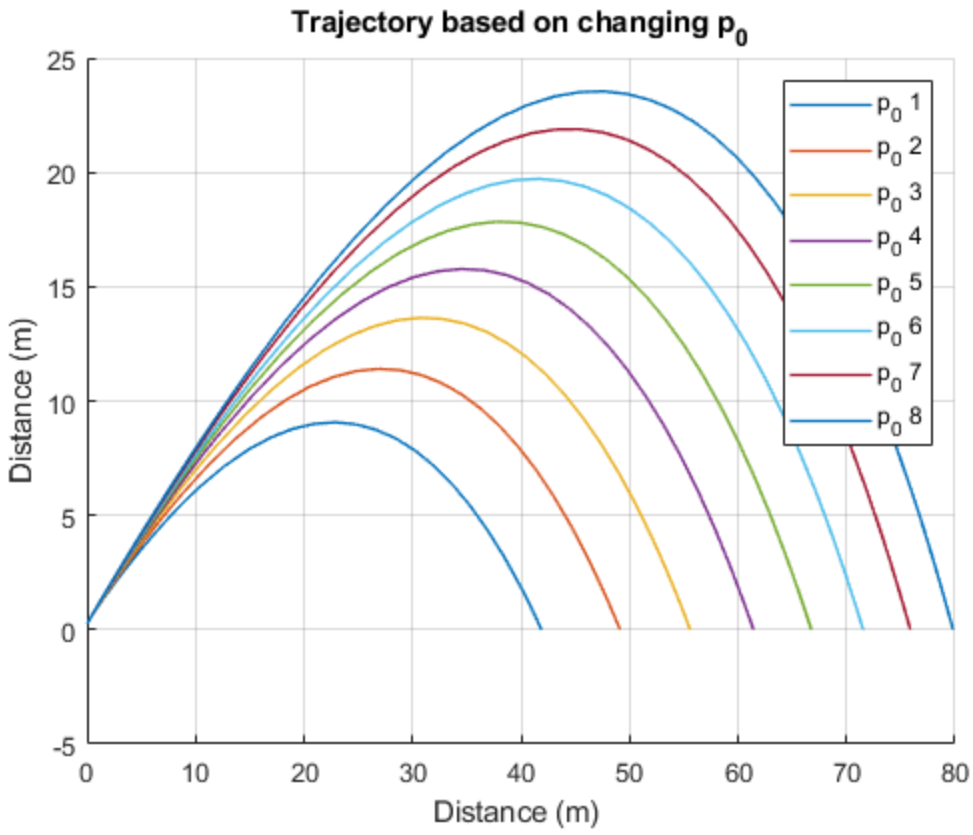
```
% Storing constants in a struct using the getConst() function
const = getConst();

%Integration time vector
int_time = [0 5];
```

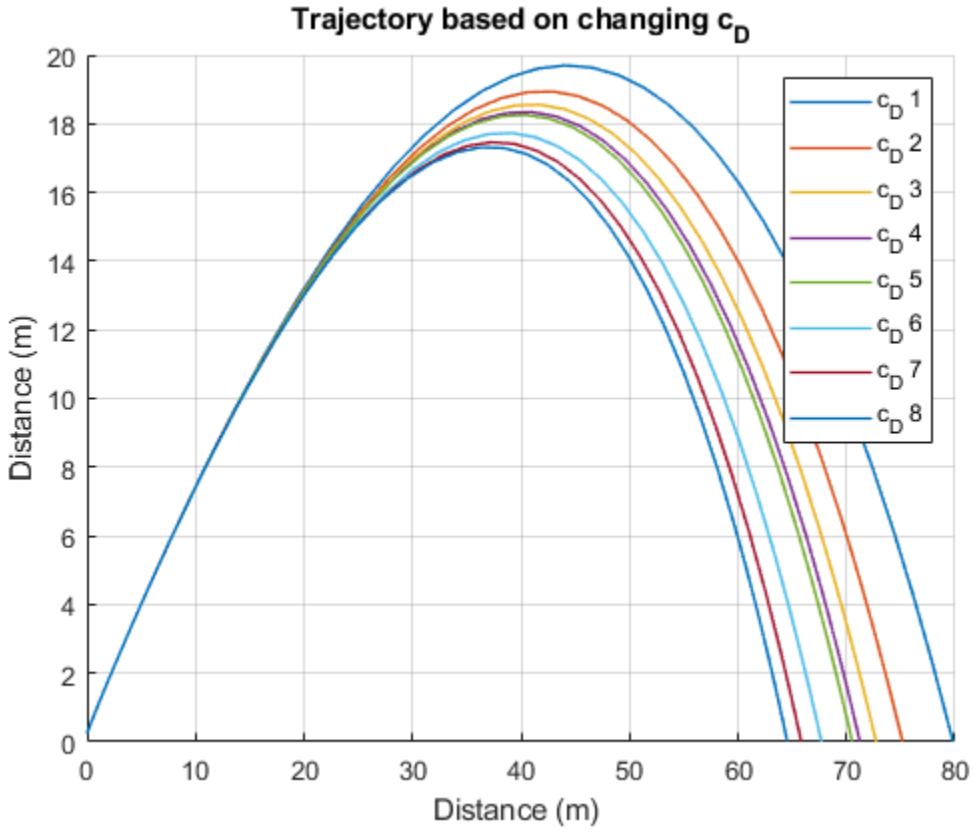
Change in performance due to change in parameters

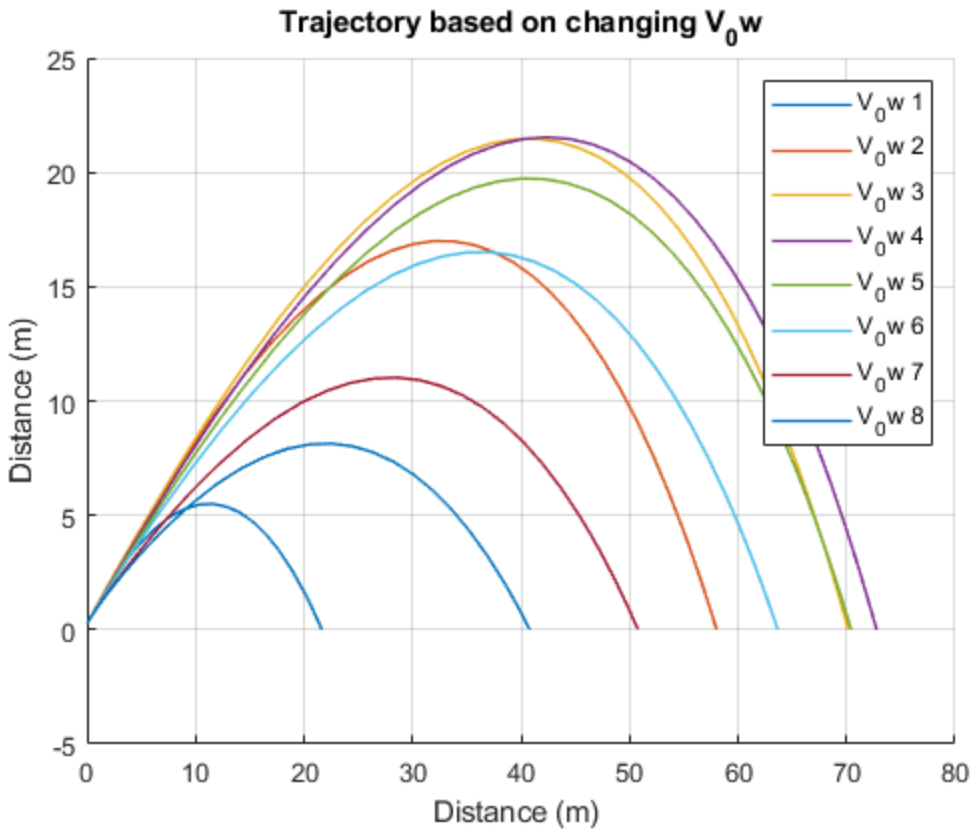
```
% Arrays of 8 values of each parameter to check
```

```
% Only 8 were chosen initially just to provide helpful plots that aren't
% too crowded.
theta_test = [25,30,35,40,45,50,55,60]; %Degrees
theta_test = theta_test*pi/180; %Radians

pressure_test = [45,50,55,60,65,70,75,80]; %psi
pressure_test = pressure_test*6894.76; %Pa

V_test = [0.001,0.1,0.2,0.3,.4,.5,.6,.65]; %Percent
V_test = const.V_b * V_test; %Volume

cd_test = [0.3,0.35,0.38,0.4,0.41,0.45,0.48,0.5];

%Using local varying_parameters function to create a cell of the states
%from state matrix func for each to see differences due to a new parameter
[~,state_theta] = varying_parameters("theta_i",theta_test,int_time);
[~,state_P] = varying_parameters("p_0",pressure_test,int_time);
[~,state_c] = varying_parameters("c_D",cd_test,int_time);
[~,stateV] = varying_parameters("V_0w",V_test,int_time);

%Plotting the trajectories with corresponding changed values
plotFun(state_theta,'theta_i');
plotFun(state_P,'p_0');
plotFun(state_c,'c_D');
plotFun(stateV,'V_0w');
```




Trajectory based on changing p_0
Distance (m)
Distance (m)
p_0 1
p_0 2
p_0 3
p_0 4
p_0 5
p_0 6
p_0 7
p_0 8


Trajectory based on changing c_D
Distance (m)
Distance (m)
c_D 1
c_D 2
c_D 3
c_D 4
c_D 5
c_D 6
c_D 7
c_D 8



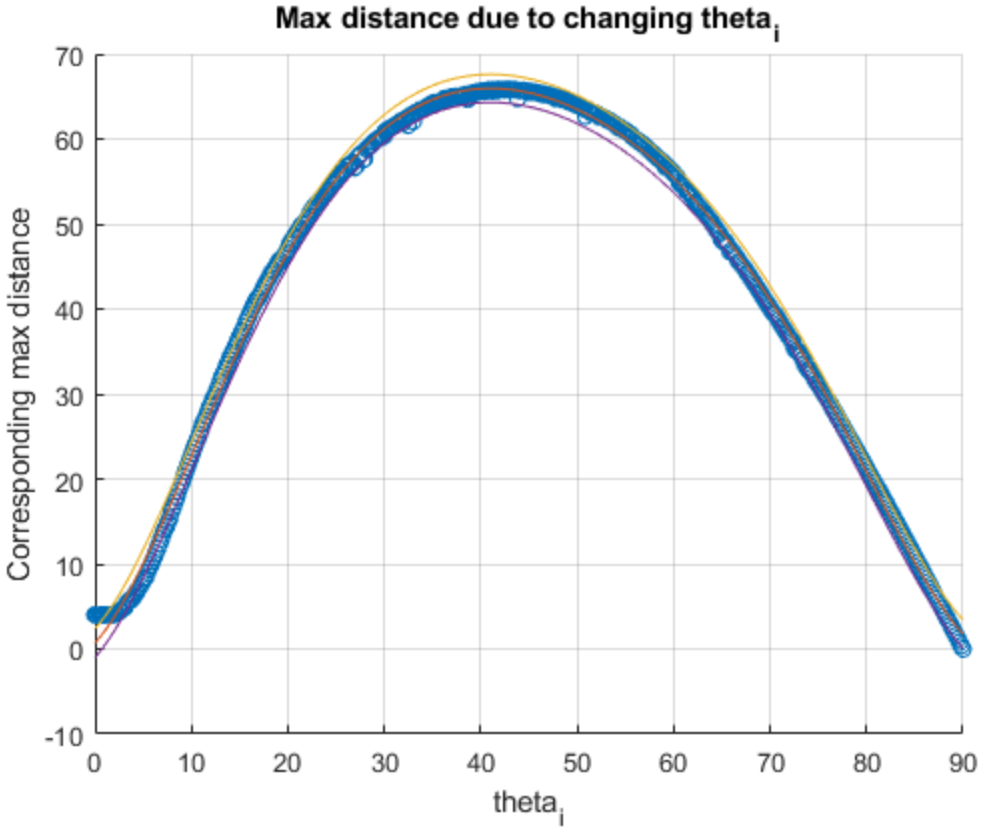
Scatter Plots for optimization

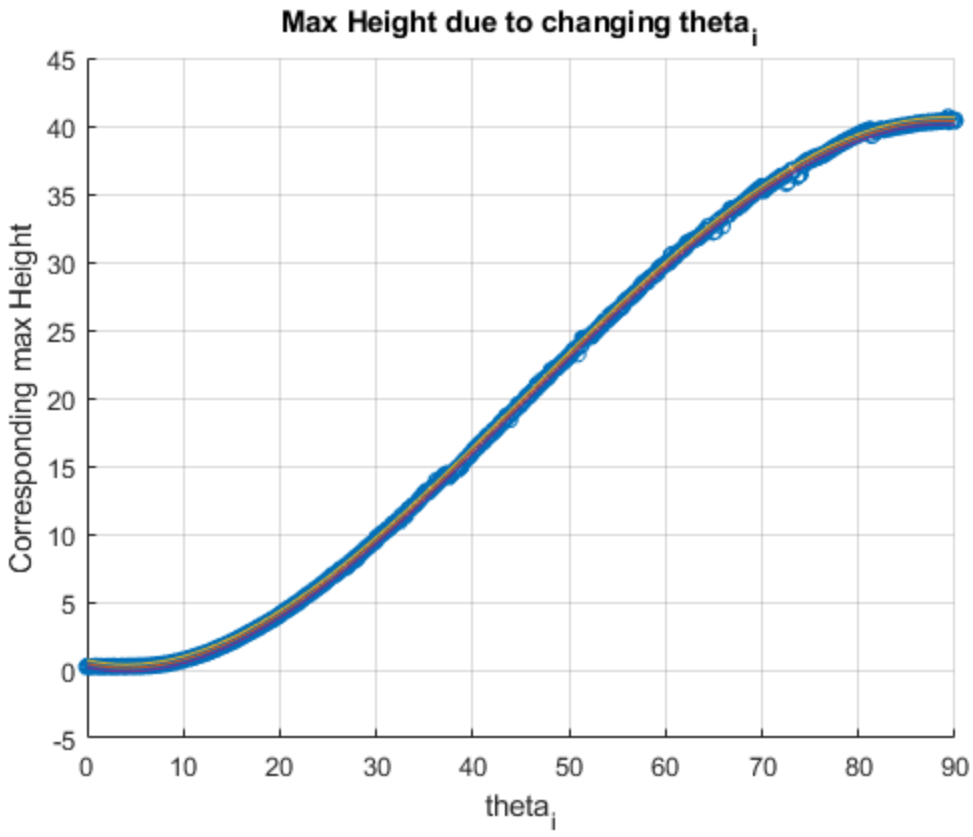
```
%Max and min initial pressures (MUST BE IN PSI)
pmax = 80;
pmin = 45;

%Max and min coefficients of drag (unitless)
cmax = 0.3;
cmin = 0.5;

%Max and min launch angle (MUST BE IN DEGREES)
thetamin = 0;
thetamax = 90;

%Max and min percents of water
Vmin = 0.001;
Vmax = 0.6;

% Number of values, times ode45 will run with different parameters
npts = 500;

% Long matrices between max and min values for more scatter plot (Accuracy)
p_long = linspace(pmin,pmax,npts);
p_long = p_long * 6894.76;
```

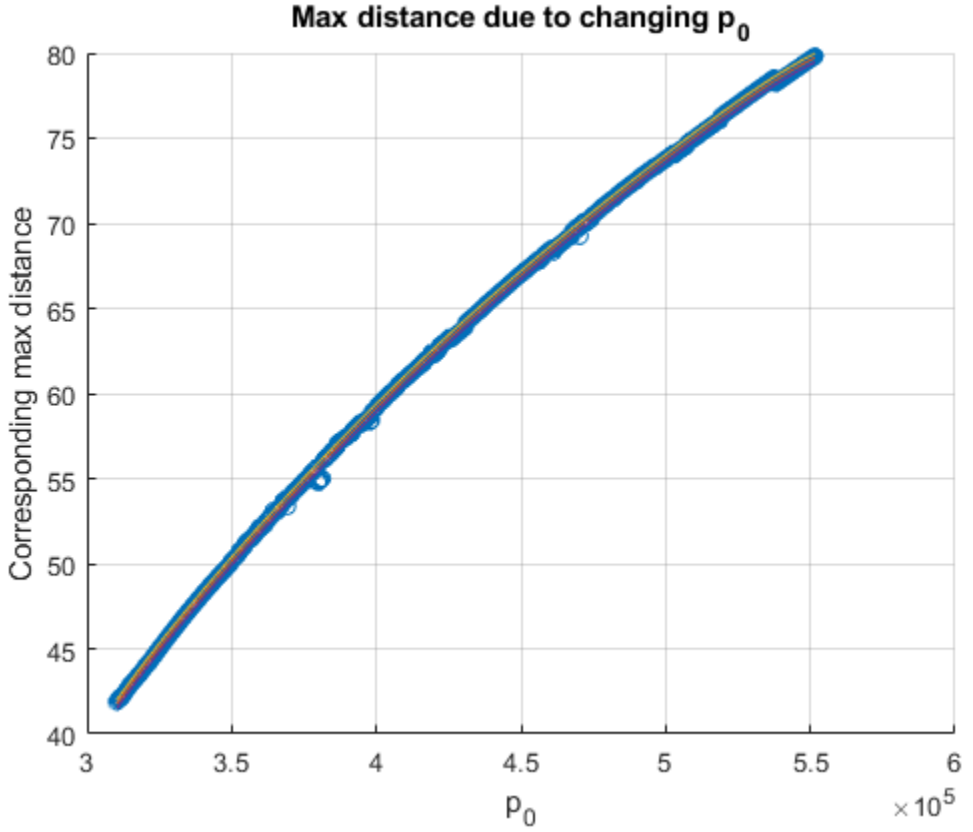
```
c_long = linspace(cmin,cmax,npts);

theta_longd = linspace(thetamin,thetamax,npts); % degrees
theta_long = theta_longd*pi/180; %radians

V_long_percent = linspace(Vmin,Vmax,npts);
V_long = V_long_percent*const.V_b;



%Doing calculations again with long matrices for input to plotScatter
[time_thetalong,state_thetalong] =
 varying_parameters("theta_i",theta_long,int_time);
[time_Plong,state_plong] = varying_parameters("p_0",p_long,int_time);
[time_clong,state_clong] = varying_parameters("c_D",c_long,int_time);
[time_Vlong,state_Vlong] = varying_parameters("V_0w",V_long,int_time);

% Plotting max distance with array of varied parameters for optimization
[max_distanceT,idealTheta] =
 plotScatter(theta_longd,state_thetalong,'theta_i');
[max_distanceP,idealP] = plotScatter(p_long,state_plong,'p_0');
[max_distanceC,idealC] = plotScatter(c_long,state_clong,'c_D');
[max_distanceV,idealV] = plotScatter(V_long_percent,state_Vlong,'V_0w');

%Reconverting from Pa to PSI
idealPpsi = idealP/6894.76;

%Printing optimal constraints for farthest flight (changing only one
%variable)
fprintf('Best theta for max distance is %2.2f degrees for %2.2f m
\n',idealTheta(1),max_distanceT(1));
fprintf('Best c_D for max distance is %2.2f for %2.2f m
\n',idealC(1),max_distanceC(1));
fprintf('Best P_0 for max distance is %2.2f psi for %2.2f m
\n',idealPpsi(1),max_distanceP(1));
fprintf('Best percent volume of water for max distance is %2.2f%% for %2.2f m
\n',100*idealV(1),max_distanceV(1));

disp(' ');

fprintf('Best theta for max height is %2.2f degrees for %2.2f m
\n',idealTheta(2),max_distanceT(2));
fprintf('Best c_D for max height is %2.2f for %2.2f m
\n',idealC(2),max_distanceC(2));
fprintf('Best P_0 for max height is %2.2f psi for %2.2f m
\n',idealPpsi(2),max_distanceP(2));
fprintf('Best percent volume of water for max height is %2.2f%% for %2.2f m
\n',100*idealV(2),max_distanceV(2));

disp(' ');
```

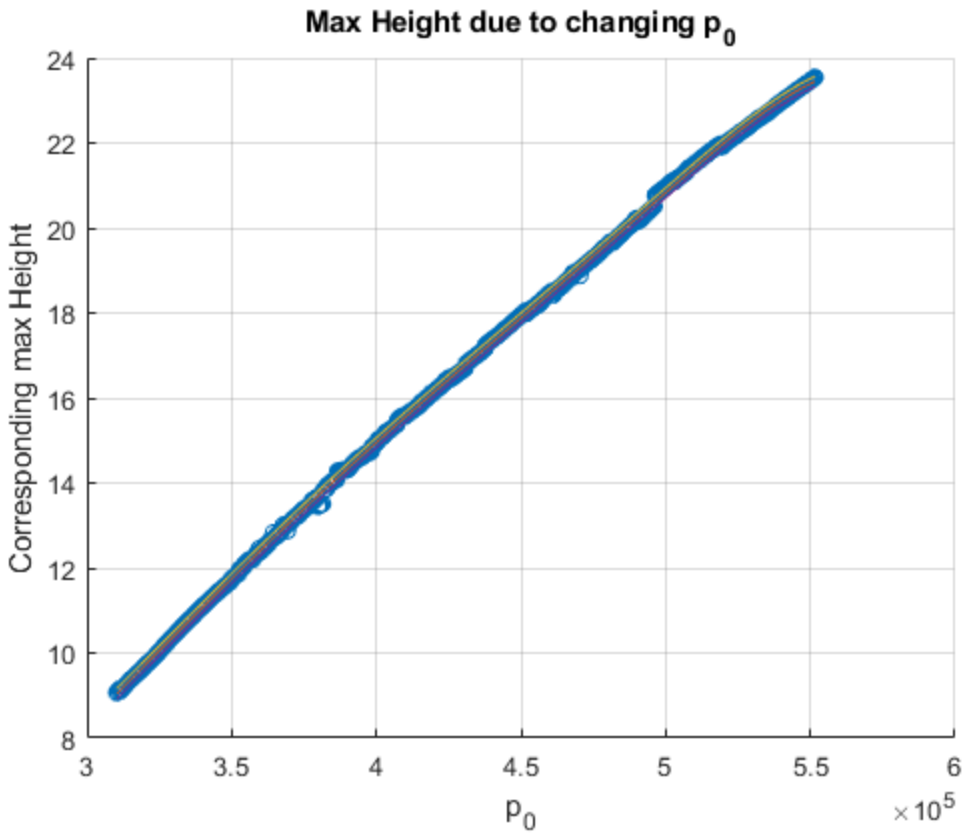
```
Best theta for max distance is 40.94 degrees for 65.95 m
Best c_D for max distance is 0.30 for 79.77 m
```

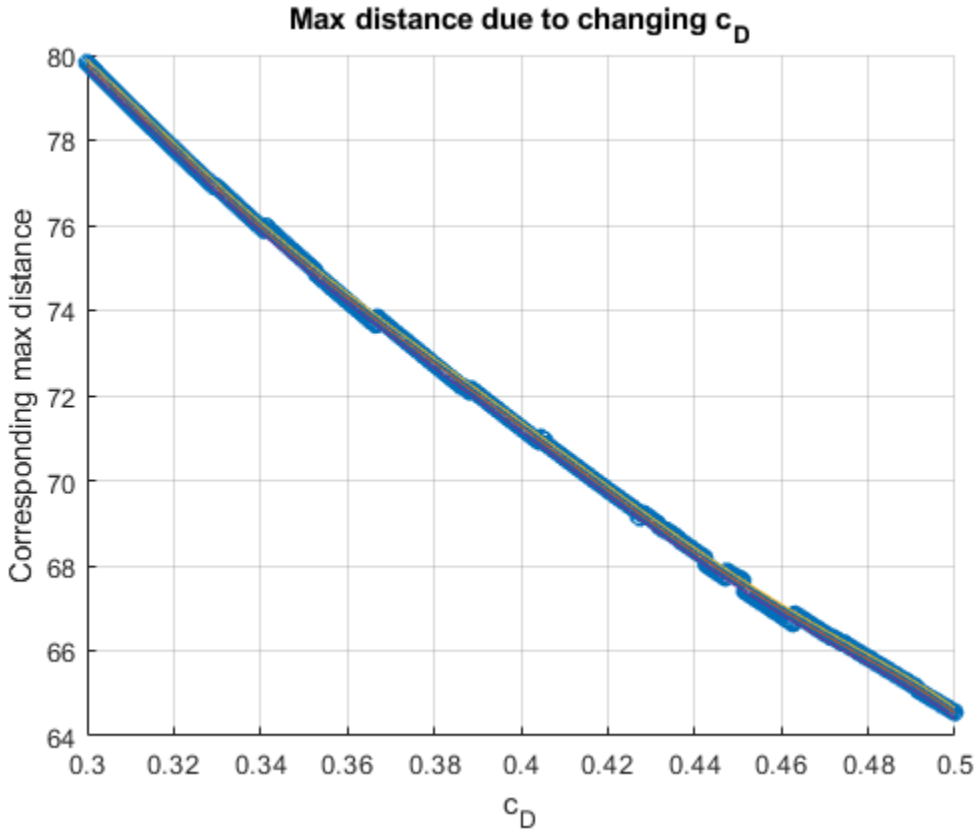
```
Best P_0 for max distance is 80.00 psi for 79.72 m
Best percent volume of water for max distance is 30.35% for 72.73 m

Best theta for max height is 88.92 degrees for 40.38 m
Best c_D for max height is 0.30 for 19.68 m
Best P_0 for max height is 80.00 psi for 23.45 m
Best percent volume of water for max height is 25.91% for 21.84 m
```

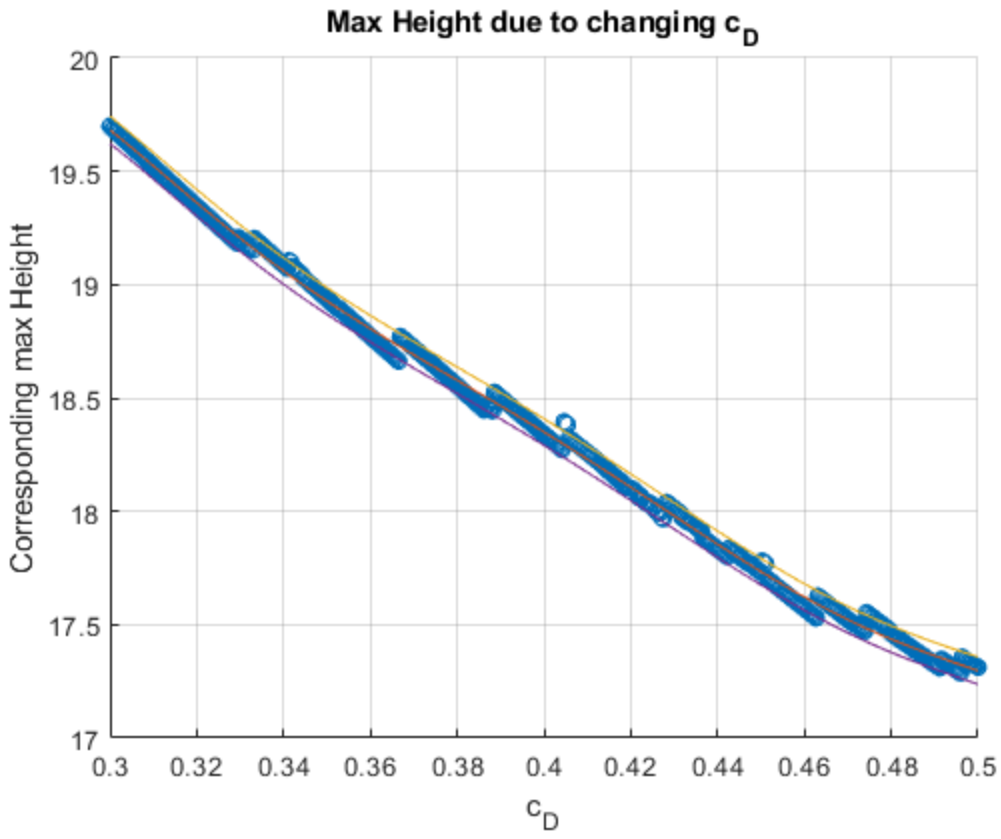


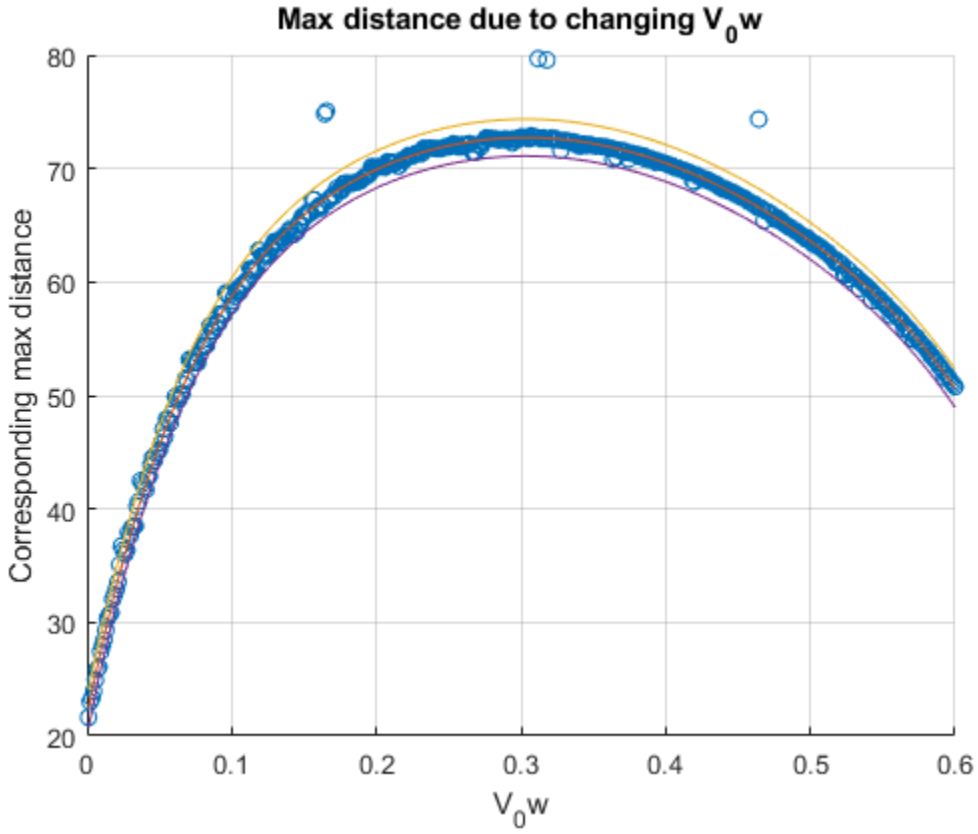

Max Height due to changing theta$_i$
Corresponding max Height
theta$_i$
45
40
35
30
25
20
15
10
5
0
-5
0
10
20
30
40
50
60
70
80
90

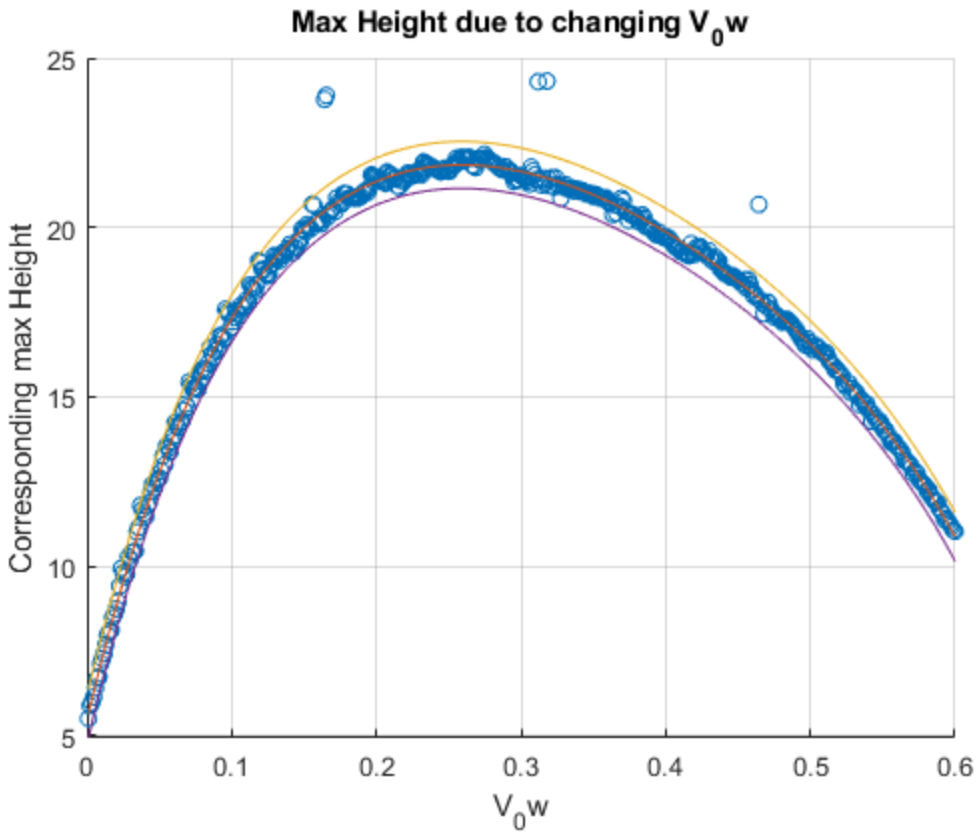

Max distance due to changing p$_0$
Corresponding max distance
p$_0$
$\times 10^5$
80
75
70
65
60
55
50
45
40
3
3.5
4
4.5
5
5.5
6


Max Height due to changing p_0
Corresponding max Height
24
22
20
18
16
14
12
10
8
3
3.5
4
4.5
5
5.5
6
p_0
$\times 10^5$


Max distance due to changing c_D
Corresponding max distance
80
78
76
74
72
70
68
66
64
0.3
0.32
0.34
0.36
0.38
0.4
0.42
0.44
0.46
0.48
0.5
c_D


Max Height due to changing c_D
Corresponding max Height
c_D
20
19.5
19
18.5
18
17.5
17
0.3
0.32
0.34
0.36
0.38
0.4
0.42
0.44
0.46
0.48
0.5

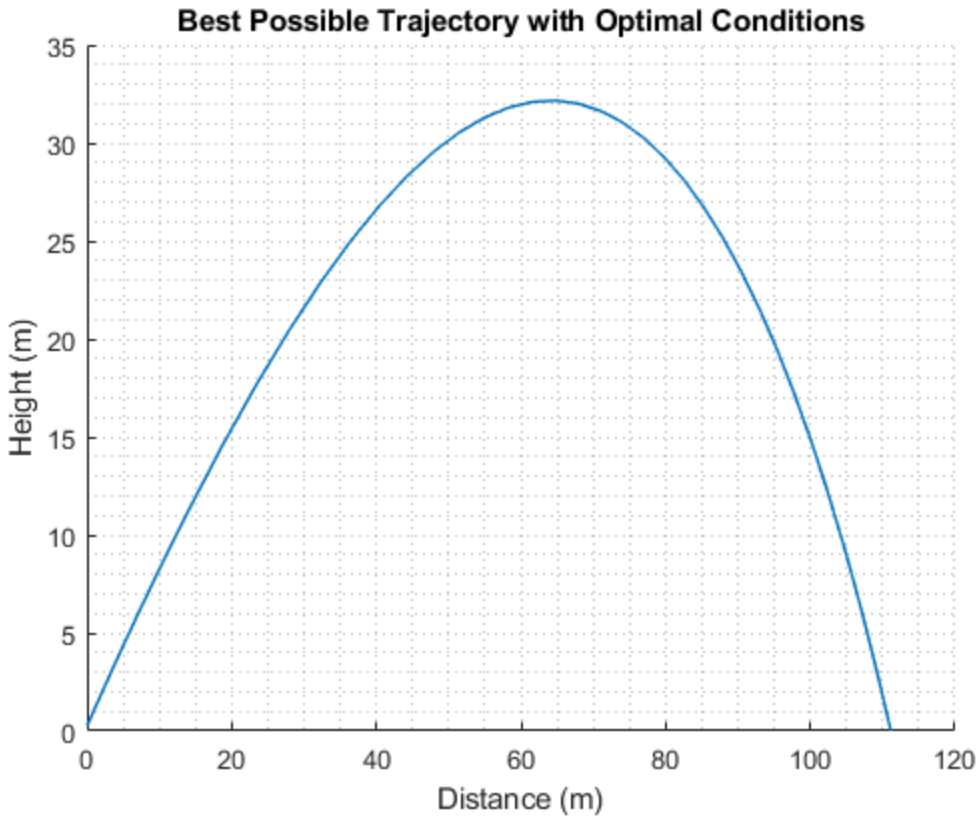

Max distance due to changing V_0w
Corresponding max distance
V_0w
80
70
60
50
40
30
20
0
0.1
0.2
0.3
0.4
0.5
0.6



Using all of the optimal conditions for maximum distance

```
%Using the update4Const function to calculate the trajectory with all 4
%best values
constBest =
 update4Const("theta_i",idealTheta(1)*(pi/180),"c_D",idealC(1),"p_0",idealP(1),"V_0w",idea

%Calculating initial conditions for ode45 (some don't change)
vx0 = constBest.v_0*cos(constBest.theta_i);
vz0 = constBest.v_0*sin(constBest.theta_i);

% Creating a column vector of initial condition
initial_conditions = [constBest.x_0 ; vx0 ; constBest.z_0 ; vz0 ;
 constBest.m_0tot ; constBest.V_0a ; constBest.m_0a];

%A new time span beacause when optimized the rocket takes more than 5
%seconds to fly
int_time2 = [0,6];

%new function handle with the best const struct
f = @(t,y)state_matrix_func(constBest,t,y);

%Calculating the flight using the best conditions for maximum flight
```

```
[best_t,best_state] = ode45(f,int_time2,initial_conditions);

%Plotting the best trajectory
figure();
hold on;
plot(best_state(:,1),best_state(:,3),'linewidth',1);
ylim([0,35]);
xlabel('Distance (m)');
ylabel('Height (m)');
grid minor;
title('Best Possible Trajectory with Optimal Conditions');
```



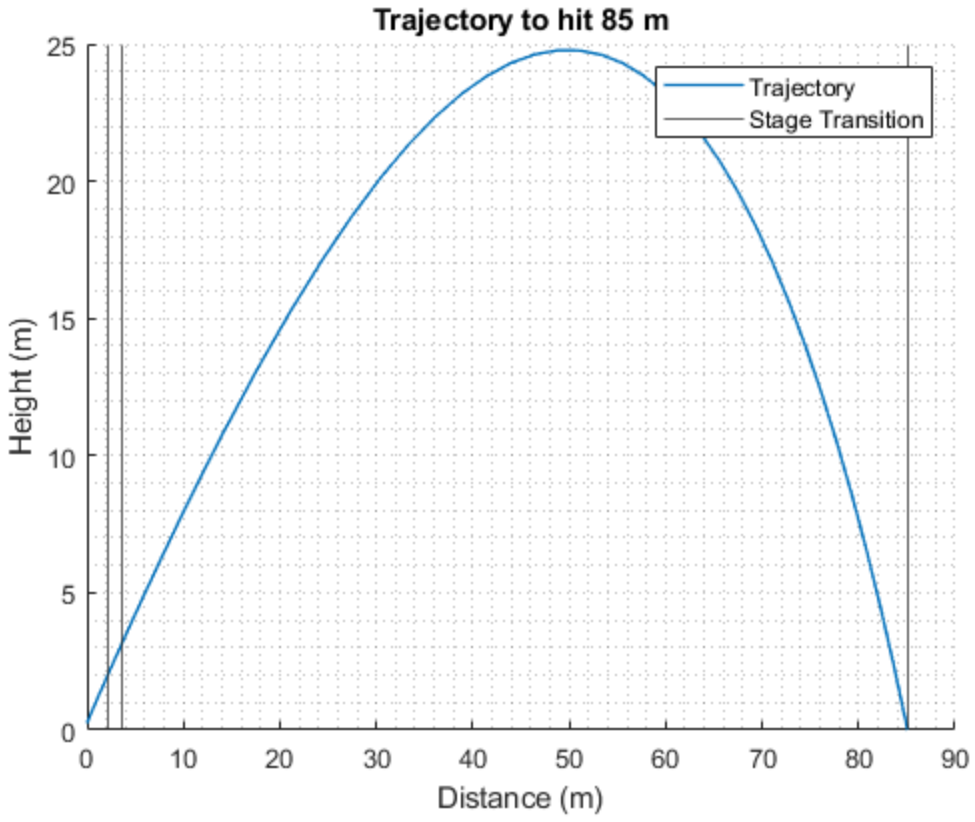
Hitting 85 m

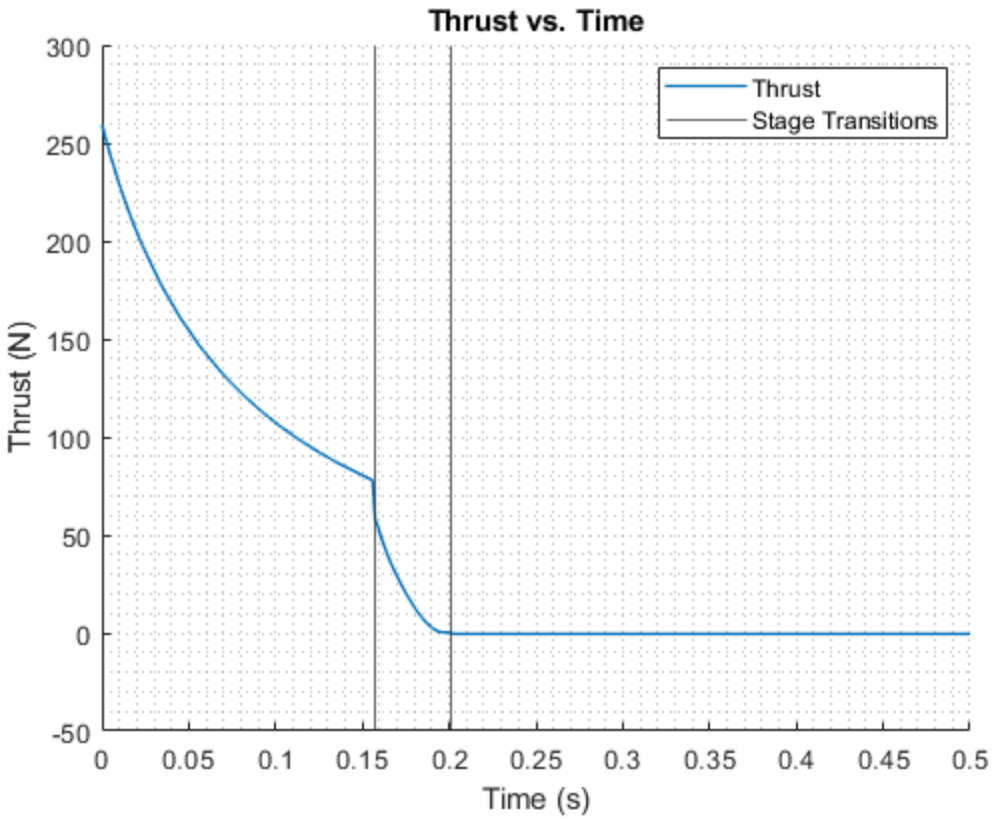
```
%Changeable Parameters to adjust to test how close to 80m is possible
target_psi = 80;
target_c = 0.4259;
target_psi = target_psi*6894.76;

%Using update 2 const to change 2 parameters in the const struct
constTarget = update2Const("p_0",target_psi,"c_D",target_c);

vx0 = constTarget.v_0*cos(constTarget.theta_i);
vz0 = constTarget.v_0*sin(constTarget.theta_i);
```

```
initial_conditions = [constTarget.x_0 ; vx0 ; constTarget.z_0 ; vz0 ;
 constTarget.m_0tot ; constTarget.V_0a ; constTarget.m_0a];

%Setting up a new function handle
f_target = @(t,state)state_matrix_func(constTarget,t,state);


%Calculating the flight using the best conditions for maximum flight
[target_t,target_state] = ode45(f_target,int_time,initial_conditions);

% Preallocating
thrust = zeros(length(target_t),1);
stage = zeros(length(target_t),1);

%Calculating the thrust and stage from the ode45 values
for i = 1:length(target_t)
    [~,thrust(i),stage(i)] =
 state_matrix_func(constTarget,target_t(i),target_state(i,:));
end


%Index where the change occurs
stage2 = find(stage==2,1);
stage3 = find(stage==3,1);
stage4 = find(stage==4,1);

% The transitions between stages in time
transition1_time = target_t(stage2);
transition2_time = target_t(stage3);
transition3_time = target_t(stage4);

%The transition between stages in x position
transition1_x = target_state(stage2,1);
transition2_x = target_state(stage3,1);
transition3_x = target_state(stage4,1);

%Plotting the best trajectory
figure();
hold on;
plot(target_state(:,1),target_state(:,3),'linewidth',1);
ylim([0,25]);
xlabel('Distance (m)');
ylabel('Height (m)');
grid minor;
xline(transition1_x);
xline(transition2_x);
xline(transition3_x);
title('Trajectory to hit 85 m');
legend('Trajectory','Stage Transition');


figure();
hold on;
plot(target_t,thrust,'Linewidth',1);
```

```
xline(transition1_time);
xline(transition2_time);
xline(transition3_time);
grid minor;
xlabel('Time (s)');
ylabel('Thrust (N)');
title('Thrust vs. Time');
xlim([0,0.5]);
legend('Thrust','Stage Transitions');

%Printing the landing distance to see if the given parameters provided the
%desired trajectory
fprintf('Distance traveled: %2.2f\n',max(target_state(:,1)));
fprintf('Peak thrust is %2.2f N\n',max(thrust));
fprintf('Max height: %2.2f m\n',max(target_state(:,3)));

runtime = toc;
```

```
Distance traveled: 85.08
Peak thrust is 259.44 N
Max height: 24.77 m
```





Published with MATLAB® R2023a