An Approach to Optimizing the Mars Ascent Vehicle

GE413 Final Report

Danny Lohan
and
Jason McDonald

September 10, 2014

Contents

Abstract

This report explores the effect of plant design in minimizing control effort. The plant of interest is the Mars Ascent Vehicle, part of the planned Mars Sample Return mission, and an evaluation of concurrent robustness of design and control system is presented. The angles of the rocket thrusters are treated as design variables in order to explore the benefits of different plant designs in case of system malfunctions. A linear quadratic regulator is used to minimize control effort. Designs were observed to converge towards the base thruster design. Studies on robustness did not result in conclusive data. The scale irregularity in state errors resulted in a poorly optimized control system. The results of several simulations are presented along with their implications.

1 Introduction

1.1 Background and Motivation

In recent years, much attention has been given to the further exploration of Mars. Rovers have landed and explore the surface while orbiters collect images and information and send it back to Earth. The information acquired using orbiters and rovers is valuable, but it would be of great worth to be able to take samples of Martian atmosphere and soil and return the samples to Earth for further study and testing. A mission has been proposed that would collect samples from Mars and return to Earth by using a multi-craft approach. Essential in that plan is the development of a Mars Ascent Vehicle (MAV) that would be able to transfer samples from a rover on the Martian surface to an orbiter, which would then bring the samples back to Earth.[1] [2] [3] [4] This report will discuss the design of the controls of the MAV so that this mission can be carried out successfully.

Inherent in a mission with so many moving parts is the chance that something can go wrong. With a control system that relies on thrusters that release compressed air, common modes of failure include thrusters that do not fire when they should and thrusters that fire when they should not. These modes of failure will be referred to as "off" failures and "on" failures, respectively. This project focuses on the design of the control system of the MAV that will overcome these failure modes. Of keen interest is an investigation into how altering the physical orientation of the thrusters can affect the robustness of the controls of the MAV.

Work has been done in developing a control system for one proposed construction of the MAV [5], but this project will explore the design space of the MAV to determine if a change in the physical states of the system will increase the robustness of the vehicle. This project will serve as a proof of concept in the application of a do-design approach to optimizing the design of the MAV.

1.2 Problem Formulation

In order to facilitate a solution, the problem will be divided into two optimization problems, an outer-loop and an inner-loop problem. The outer-loop problem will focus on determining the locations of the thrusters on the rocket. The inner-loop problem will determine the optimal control of the rocket to move from one orientation to another.

1.2.1 System Specifications

The following list details the values that were accepted as constant throughout the problem.

- Mass of MAV and propellant: $m = 73.1\ kg$ [6]
- Moment of inertia of MAV: $I = 33\ kg/m^2$ [6]
- Maximum Thrust for each rocket: $F_{Max} = 6318.9\ N$ [6]
- Mass of Mars: $M = 639{\times}10^{21}\ kg$
- Radius of Mars: $R = 3390\ km$
- Universal gravitational constant: $G = 6.67{\times}10^{-11}\ m^3/(kg * s^2)$

The gravitational constant is used to calculate the force on the MAV due to gravity. Because of the heights at which the MAV will operate, the force of gravity will change over time. The universal law of gravity is given by Equation 1

$$F_{gravity} = \frac{GMm}{(R+y)^2} \tag{1}$$

in which y is the height of the MAV above the surface of Mars, $F_{gravity}$ is the force of gravity, and all other variables are defined as above.

Also important to the understanding of the problem is to know the initial and final states of the MAV. These can be seen in Equation 2.

$$X_{Init} = [x, \dot{x}, y, \dot{y}, \theta, \dot{\theta}]_{Init} = [0, 0, 200000, 1.5 \times 10^3, 0, 0] \tag{2a}$$

$$X_{Final} = [x, \dot{x}, y, \dot{y}, \theta, \dot{\theta}]_{Final} = [0, 0, 500000, 0, 0, 0] \tag{2b}$$

The initial values of y and $\dot{y}$ are set at the point that the second stage of the rocket begins to control the MAV.

1.2.2 Design Variables

In the outer-loop problem the orientation of the thrusters will be determined. Figure 1 displays the coordinate system in which the thrusters will be translated and/or rotated.

In the inner loop problem, the design variables will be the activity of each thruster. These will be continuous variables that range from 0 percent thrust to 100 percent thrust.

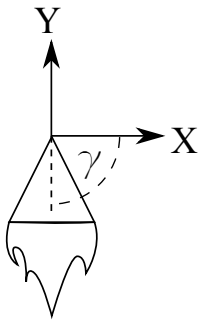

Figure 1: Thruster Orientation

1.2.3 Constraints

The inner loop problem constraints will focus on the physical design of the rocket. The rocket will maintain a fixed geometry. The location of the thrusters will be limited to the boundaries of the vehicle body. In addition, thrusters will only be placed on the left and right boundaries, seeing as how the top and bottom of the rocket must be left open for connections and other necessities.

Thrusters will be considered directional, in that they only propel the rocket in their respective axial direction. As a result, thrusters must be directed either parallel to or away from the body. Rocket thrusters will also not be coupled, meaning the left and right side thrusters will not necessarily be mirrored.

The inner-loop constraints will be modifying the actions of the thrusters. The rocket will be required to travel from point A to point B in a fixed time frame. The rocket will also be moving to start and must progress to a predefined position to be considered a success.

1.2.4 Objective Function

Considering these constraints, there are numerous possibilities for optimization. One thought is to determine which rocket will complete the allotted task by consuming the least amount of fuel, Equation 3.

$$F = \sum_{i}^{jets} f_i t_i \tag{3}$$

Where F is the fuel consumption, f is the thrust of jet i, and t is the time the thruster was active. Additionally, if a design fails to successfully reach the desired position, the objective function receives a large penalty to ensure that only designs that successfully navigate to the desired position are considered fit designs. The penalty factor is added as a weighted sum of squares of the states of the MAV as shown in Equation 4

$$\begin{aligned} \underset{X,f,t}{\text{minimize}} \quad & \phi = \sum_{j}^{time} \sum_{i}^{jets} (f_i * \delta t)_j + X_f^T Q_e X_f \\ \text{subject to} \quad & t_{final} \leq 200 \\ & 0 \leq |f_i| \leq 6318.9 \end{aligned} \tag{4}$$

in which Q_e is a positive-definite, diagonal matrix that holds the weights for each state of the MAV.

2 System Model

2.1 Equations of Motion

The equations of motion will be derived from the four thruster rocket shown in Figure 2. Four unique thrust forces: A,B,C,and D will be applied at some angles, theta, psi, gamma and xi in order propel the rocket forward.

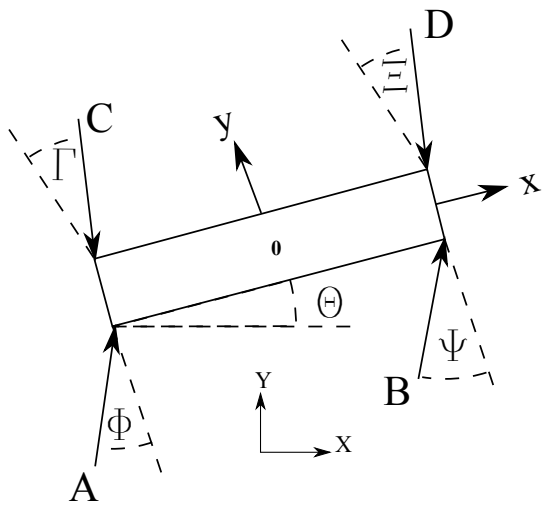

Figure 2: Simplified Rocket

Summing the forces in both the X an Y direction, the following equation results.

$$\Sigma F = MA = \begin{pmatrix} F_x \\ F_y \end{pmatrix} = \begin{pmatrix} Asin(\phi) + Bsin(\psi) - Csin(\Gamma) - Dsin(\Xi) \\ Acos(\phi) + Csin(\Gamma) - Bcos(\psi) - Dsin(\Xi) \end{pmatrix} \tag{5}$$

Re-arranging components, the acceleration can be solved for in terms of known quantities.

$$A = \frac{1}{m} \begin{pmatrix} Asin(\phi) + Bsin(\psi) - Csin(\Gamma) - Dsin(\Xi) \\ Acos(\phi) + Csin(\Gamma) - Bcos(\psi) - Dsin(\Xi) \end{pmatrix} \tag{6}$$

Integrating the acceleration function in terms of time will result in the velocity equation that follows.

$$V = V_0 + \frac{1}{m}\begin{pmatrix} Asin(\phi) + Bsin(\psi) - Csin(\Gamma) - Dsin(\Xi) \\ Acos(\phi) + Csin(\Gamma) - Bcos(\psi) - Dsin(\Xi) \end{pmatrix} t \tag{7}$$

Integrating once more, the continuous position function is resultant.

$$P_{local} = \begin{pmatrix} P_x \\ P_y \end{pmatrix} = P_0 + Vt + \frac{1}{2}At^2 \tag{8}$$

Granted that the rocket will be experiencing disturbances and therefore not traveling in a straight line, states relating to rotations are also of interest. Summing the moments about the center of the rocket will result in the following equation.

$$\Sigma M_o = -\frac{1}{2}Asin(\phi) + \frac{1}{2}Csin(\Gamma) + \frac{1}{2}Bcos(\psi) - \frac{1}{2}Dsin(\Xi) = I_{zz}\alpha \tag{9}$$

The equations can be rearranged to solve for angular acceleration alpha.

$$\alpha = \frac{1}{I_{zz}}\left(-\frac{1}{2}Asin(\phi) + \frac{1}{2}Csin(\Gamma) + \frac{1}{2}Bcos(\psi) - \frac{1}{2}Dsin(\Xi)\right) \tag{10}$$

Integrating the angular acceleration will result in the angular velocity equation shown below.

$$\dot{\Theta} = \frac{1}{I_{zz}}\left(-\frac{1}{2}Asin(\phi) + \frac{1}{2}Csin(\Gamma) + \frac{1}{2}Bcos(\psi) - \frac{1}{2}Dsin(\Xi)\right) t + \dot{\Theta}_0 \tag{11}$$

Integrating the angular velocity equation will reveal the desired rotation state. The continuous form follows.

$$\Theta = \frac{1}{2I_{zz}}\alpha t^2 + \dot{\theta}_0 t + \theta_0 \tag{12}$$

Since the model will be required to handle multiple different trajectories and types of error, it is discretized in order to handle changes in control force that are not predicted.

$$P_{k+1} = P_k + \Delta P_k \tag{13}$$

In order to determine the location of the rocket that does not travel in a straight line, information about rotation was used to assemble a transformation matrix. To ensure that proper velocities and positions are being calculated the rotation matrix is enforced in the following two equations

$$V = V_0 + \frac{1}{m}RAt \tag{14}$$

$$P_{local} = Vt + \frac{1}{2}RAt^2 \tag{15}$$

Global positions can be calculated by summing the local positions at each time step.

$$P_{global} = \Sigma P_{local} \tag{16}$$

At this point, the location of the rocket can be calculated by an accuracy dependent on time step size.

2.2 Control System

In order to develop a control system that could be used confidently to test different MAV thruster layouts, the equations of motion were rewritten into a state-space form that was convenient to work with. The dynamics of the MAV were assumed to be:

$$\begin{aligned} m\ddot{x} &= -(f_R + f_L)sin\theta \\ m\ddot{y} &= (f_R + f_L)cos\theta - mg \\ I_{zz}\ddot{\theta} &= w(f_R - f_L) \end{aligned} \tag{17}$$

where I is the moment of inertia, m is the mass, x and y are position, theta is the MAV rotation, g is the force of gravity, andf_R and f_L are the right and left forces from the thrusters, respectively. Supposing that it is desired for the MAV to approach a certain position $(\bar{x}, \bar{y})$, the following were defined:

$$\begin{aligned} T &= f_R + f_L \\ M &= w(f_R - f_L) \end{aligned} \tag{18}$$

Then assuming θ is either known or can be estimated,

$$\begin{aligned} T &= \frac{mg + u_1}{cos\theta} \\ M &= u_2 \end{aligned} \tag{19}$$

for some u_1 and u_2. The dynamics can be rewritten

$$\begin{aligned} m\ddot{x} &= -(mg + u_1)tan\theta \\ m\ddot{y} &= u_1 \\ I_{zz}\ddot{\theta} &= u_2. \end{aligned} \tag{20}$$

This assumes small θ. Letting the error be defined as

$$\begin{aligned} e_1 &= x - \bar{x} \\ e_2 &= y - \bar{y} \\ e_3 &= \theta \end{aligned} \tag{21}$$

and assuming that $(\bar{x}, \bar{y})$ is constant, the error dynamics can be described by

$$\begin{aligned} m\ddot{e_1} &= -(mg + u_1)tan(e_3) \\ m\ddot{e_2} &= u_1 \\ I_{zz}\ddot{e_3} &= u_2. \end{aligned} \tag{22}$$

Again, assuming small θ, the first of the above equations can be approximated by $m\ddot{e_1} \approx -mge_3$.

The system can then be described by the state-space model

$$\dot{E} = AE + Bu \tag{23}$$

where $E = [e_1 e_2 e_3 \dot{e_1} \dot{e_2} \dot{e_3}]^T$ and u is the control effort from the thrusters on each side of the MAV and

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -g & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{24}$$

and

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1/m & 0 \\ 0 & 0 \\ 0 & 1/I_{zz} \end{bmatrix}. \tag{25}$$

Now that the system was represented as a state-space model, a linear quadratic regulator was used to determine the gains that should be applied to the system for optimal system performance. A linear quadratic regulator has a quadratic cost function defined as

$$J = \frac{1}{2}x^T(t_f)F(t_f)x(t_f) + \int_{t_i}^{t_f}(x^TQx + u^TRu)dt \tag{26}$$

and it minimizes the control cost defined by $u = -Kx$, where K is given by $K = R^{-1}B^TP$, where P is the solution to the Riccati differential equation. These gains were applied to the states and controlled the system.

3 Optimization Results

For this study patternsearch is used to optimize results. Granted constraints were implicitly enforced, fminunc was attempted and resulted with no exploration of design space. Genetic algorithms were also tested and a significant increase in resultant convergence time was observed. Results based on genetic algorithms did not better those of the patternsearch algorithm. Patternsearch was implemented to find the optimal thruster angles for each thruster between the values of 0 and 90 degrees.

3.1 Model Verification

In order to verify the use of the model in optimization, a base case was developed. The rocket was to travel between the initial and final states with no disturbances experienced. Intuitively it is clear that a rocket design with the thrusters pointed downwards perpendicular to the ground plane would perform the best by the objective criteria. The simulation resulted in an objective value of 1.54×10^7 and thruster orientations as expected. The trajectory of the rocket is shown in Figure 3.

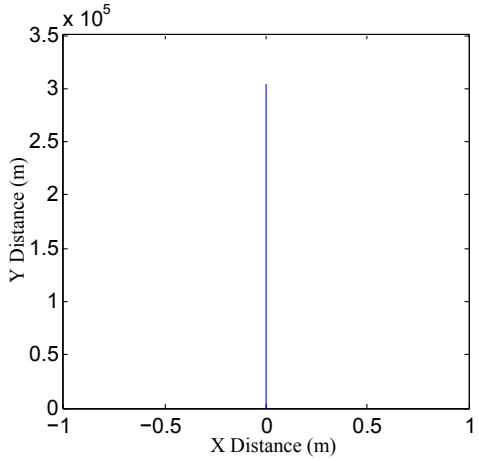

Figure 3: Base Case

With a verified optimization setup, the designs can be confidently tested in an optimization with different cases of failure. For the purpose of this study, one thruster will shut down for a time step to cause a disturbance in the trajectory.

3.2 Back Right Thruster Failure

In this simulation, the right thruster is blocked for one time step. Immediately, the rocket can be seen to veer off to the right. Control force is then seen to be applied as the rocket begins to veer to the right. A general trend of swaying to either side of the final state can be seen. Though there appears to be a large offset in the position of the rocket, it must be noted that it does not move more than 30 meters in the X direction for every 1000 meters in the Y direction.

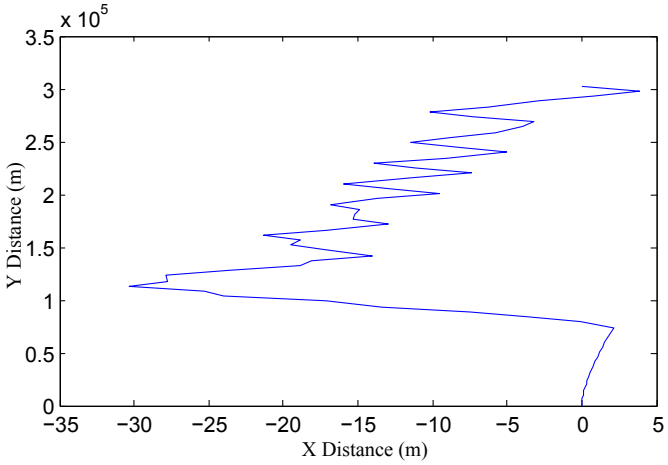

Figure 4: Impulse Disturbance Case

The following plot demonstrates this as the axis are set equal to each other. In the grand scope of the mission, the rocket barely drifts off course by the action of the malfunction.

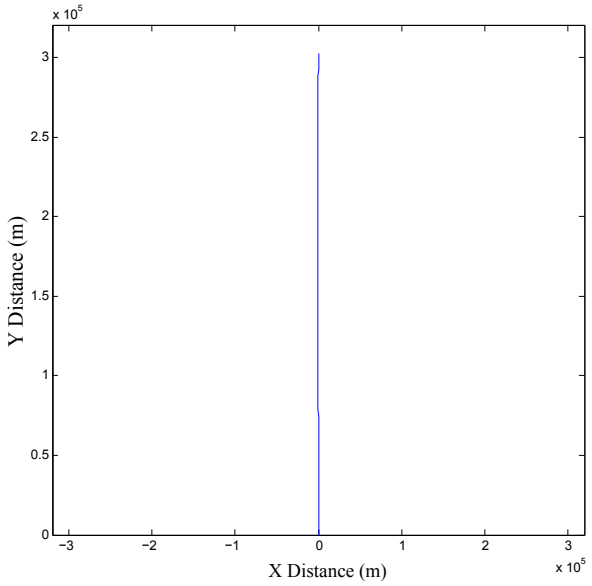

Figure 5: Impulse Disturbance Case

In this optimization, the plant design converged to the base design. To better

observe the accuracy of the control system and plant design, the final states can be observed. The rocket reached the target within excellent horizontal states, but had issues satisfying the vertical states. Further manipulation of the control system should resolve this issue. The final states of error resultant from this design are given in the following table.

Table 1: Back Right Thruster Failure Error States

States	x	$\dot{x}$	y	$\dot{y}$	θ	$\dot{\theta}$
Units	(m)	(m/s)	(m)	(m/s)	(rad)	(rad/s)
Final Error States	0.02	-1.8	2639	4857	1.42	93.27

3.3 Back Left Thruster Failure

In this simulation, the left thruster is turned off for a time step. As soon as the disturbance is felt, the rocket veers off to the left. As the last simulation revealed a tendency to sway, this design showed similiar characteristics.

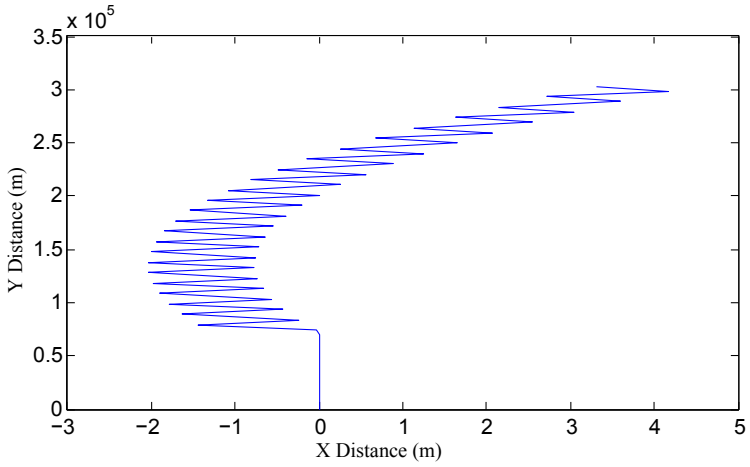

Figure 6: Impulse Disturbance Case

With similiar results, this plant design converged near the base design. Noting both this design and the prior design, there seems be a tendency for the designs to perform better near the base design.In this simulation the rocket was further offset but retained relatively good state values. The vertical states were not as desirable, but granted the rocket is entering an orbital distance, this may not be as severe an issue. The final states of error resultant from this design are given in the following table.

Table 2: Back Left Thruster Failure Error States

States	x	$\dot{x}$	y	$\dot{y}$	θ	$\dot{\theta}$
Units	(m)	(m/s)	(m)	(m/s)	(rad)	(rad/s)
Final Error States	3.3	0.37	3179	4870	0.89	-95.74

3.4 Additional Studies

Given that the optimal designs for each failure types did not provide sufficient results, further studies were conducted on each design to evaluate the effects of plant design on the objective value.

Table 3: Complete Design Evaluation

Failure Mode	No Failure	Right Failure	Left Failure
Standard Design	1.54×10^7	1.11×10^7	1.11×10^7
Back Right Failure Optimized	1.54×10^7	7.83×10^7	6.7×10^7
Back Right Failure Optimized	3.28×10^7	1.1×10^7	1.1×10^7

Based on the results seen from the standard design, it can be concluded that the control system needs further tuning. The rocket performed better when disturbances were experienced than when not. This is counter-intuitive as additional thrust will not be required to steer back on target in the test case.

Also interesitng to note was in the case of the right thruster failure, with its optimal design it result in the best overall objective value. This outperforms the standard case when no disturbance is felt. Though trends initially show superior behavior near the base designs, there may be situations in which the base design is ousted by another specialized design.

4 Conclusion

From the study conducted here, it can be seen that the base plant design outperformed another plant design tested by the patternsearch algorithm. Results are not strongly supported by experimental evidence. The base case rocket design in the test case scenario with no malfunctions should result in the optimal fuel consumption. In these simulations, this was not the case. Given the extreme ill conditioning with respect to error states, optimizing LQR gain values has proven difficult. Further work will include the implementation of direct transcription to better reject disturbance and result in an optimal trajectory. Hopes are that these simulations will allow for a further exploration of design robustness. With a more robust control system, the effectiveness of such a control system to reject different types of error is also of interest.

5 Appendix A: Code

```
function [Objective] = lqrTrajectory()
%% Initialize Variables
    %Planetary Parameters
    G = 6.67E-11;
    massMars = 639E21;

    %Trajectory Parameters
    startSpot = 3390000+200;
    currentState = [0;0;0;1.5E3;0;0];
    finalState = [0;0;300000;0;0;0];
    ThrustArray = [0,6318.9,0,6318.9];

    %Rocket Parameters
    mass = 73.1; %kg
    I_zz = 33;
    maxThrust = 6318.9;
    minThrust = -6318.9;
    Thrust = [0,6318.9,0,6318.9];
%   ThrusterAngle = [0.0463676452636719,8.01086425781250e-05,0,0];%Left
optimized
%   ThrusterAngle =[0.000274658203125000,0,0,0]; %Right optimized
    ThrusterAngle = [0 0 0 0];
    % ThrusterAngle = angle;

    %Test parameters
    maxDuration = 200;
    dt = 1;
    AngVel = 0;
    Theta = 0;
    Vel = [0, 1.5E3]';
    Pos_local = 0;
    Rotation = zeros(2,2);
    P_global = [0;0];
    Fuel_Consumption = 0;
    V_global = [0; 0];

    %Control System Pameters
    Qe=diag([1000 1000 1 0 1 0]);
    q = [1 1 .01 1 1 1000000000];

 %% Begin Testing
    for k = 1:(1/dt)*maxDuration-1
        gravity = G*massMars/(startSpot + P_global(2,k))^2;
        gravityForce = mass*gravity;

        AngVel = AngVel + 1/(I_zz)*(0.5*(Thrust(4)*cos(ThrusterAngle(4)))-
0.5*(Thrust(3)*cos(ThrusterAngle(3)))- 0.5*(Thrust(2)*cos(ThrusterAngle(2)))+
0.5*(Thrust(1)*cos(ThrusterAngle(1)))); %#ok<*SAGROW>

        Theta(k+1) = Theta(k) + AngVel*dt+
1/(2*I_zz)*(0.5*(Thrust(4)*cos(ThrusterAngle(4)))-
0.5*(Thrust(3)*cos(ThrusterAngle(3)))- 0.5*(Thrust(2)*cos(ThrusterAngle(2)))+
0.5*(Thrust(1)*cos(ThrusterAngle(1))))*dt^2; %#ok<*SAGROW>

        Vel(:,k+1) = Vel(:,k) +
(1/mass)*Rotation*[(Thrust(4)*sin(ThrusterAngle(4)) +
```

```
Thrust(2)*sin(ThrusterAngle(2))- Thrust(3)*sin(ThrusterAngle(3)) -
Thrust(1)*sin(ThrusterAngle(1)))*dt, (Thrust(4)*cos(ThrusterAngle(4)) +
Thrust(2)*cos(ThrusterAngle(2))- Thrust(3)*cos(ThrusterAngle(3)) -
Thrust(1)*cos(ThrusterAngle(1)))*dt]';

        Pos_local = Vel(:,k+1)*dt +
(1/mass)*Rotation*[0.5*(Thrust(4)*sin(ThrusterAngle(4)) +
Thrust(2)*sin(ThrusterAngle(2))- Thrust(3)*sin(ThrusterAngle(3)) -
Thrust(1)*sin(ThrusterAngle(1)))*dt^2, 0.5*(Thrust(4)*cos(ThrusterAngle(4)) +
Thrust(2)*cos(ThrusterAngle(2))- Thrust(3)*cos(ThrusterAngle(3)) -
Thrust(1)*cos(ThrusterAngle(1))-mass*gravity)*dt^2]';

        Rotation = [cos(Theta(k)), sin(Theta(k)); -
sin(Theta(k)),cos(Theta(k))];
        V_global(:,k+1) =  Vel(:,k+1);%Rotation * Vel(k+1,:)';
        P_global(:,k+1) = P_global(:,k) + Pos_local;%Rotation * Pos_local';

        %x = [e_x, e_xdot, e_y, e_ydot, e_theta, e_thetadot]
        A=[0,1,0,0,0,0;...
           0,0,0,0,-gravity,0;...
           0,0,0,1,0,0;...
           0,0,0,0,0,0;...
           0,0,0,0,0,1;...
           0,0,0,0,0,0];

        B=[0,0;...
            0,0;...
            0,0;...
            1/mass,0;...
            0,0;...
            0,1/I_zz];

        %Qc = q; use when optimizing for gains
        %LQR for system
        Qc=diag(q);
        Rc=diag([1e-4,1e-4]);
        K=lqr(A,B,Qc,Rc);

        %Current Angle Update
        Theta(k+1) = mod(Theta(k+1),(2*pi));

        %Determines position based on applied gain values
        currentState(:,k+1) =
[P_global(1,k+1);V_global(1,k+1);P_global(2,k+1);V_global(2,k+1);Theta(k+1);A
ngVel];

        %This portion will determine error current error to final states
         u=-K*(currentState(:,k+1)-finalState);

        %Double back to solve for thrust from U values
        Thrust(2)=.5*(mass*gravity+u(1)-u(2));
        Thrust(4)=Thrust(2)+u(2);

        %Enforce constraints on thrust
        Thrust(1) =frontThrust(Thrust(2), maxThrust, minThrust);
```

```
        Thrust(2) =backThrust(Thrust(2), maxThrust, minThrust);
        Thrust(3) =frontThrust(Thrust(4), maxThrust, minThrust);
        Thrust(4) =backThrust(Thrust(4), maxThrust, minThrust);

        %Initialize a disturbance
        if k == 20
            Thrust(4) = 0;
        end

        %Store Thrust/Error values for debugging
        ThrustArray(k,:) = Thrust;
        Errors(:,k+1) = (currentState(:,k+1)-finalState);
        %Error values for final state
        ErrorValue = (currentState(:,k+1)-finalState) ;

        % Sum up fuel use per time step
         Fuel_Consumption = Fuel_Consumption + abs(Thrust(1))*dt+
abs(Thrust(2))*dt+ abs(Thrust(3))*dt+ abs(Thrust(4))*dt;

         %Cut out of similuation once end state is reached within tolerance
        if currentState(3,k+1) > 300000
            finalTime = k*dt;
            Errors = (currentState(:,k+1)-finalState);
            break;
        end
    end

    %Modified objective function to include state penalty
%     Objective = Fuel_Consumption;
    Objective = Fuel_Consumption+ ErrorValue'*Qe*ErrorValue;

    %     Plot specified Data
    plot(P_global(1,:),P_global(2,:),'-');%Trajectory

% Plot current states
% subplot(6,1,1)
% plot(1:k,currentState(1,1:k))%X
% subplot(6,1,2)
% plot(1:k,currentState(2,1:k))%X_velocity
% subplot(6,1,3)
% plot(1:k,currentState(3,1:k))%Y
% subplot(6,1,4)
% plot(1:k,currentState(4,1:k))%Y_velocity
% subplot(6,1,5)
% plot(1:k,currentState(5,1:k))%Theta
% subplot(6,1,6)
% plot(1:k,currentState(6,1:k))%AngVel


%Set bounds on the max and minimum thrust values
function f = frontThrust(f,fmax,fmin)
if (f>0)
    f = 0;
elseif (f<fmin)
    f = fmax;
```

```
end

function f = backThrust(f,fmax,fmin)
if (f>fmax)
    f = fmax;
elseif (f<0)
    f = 0;
end
```

6 Appendix B: Additional Plots

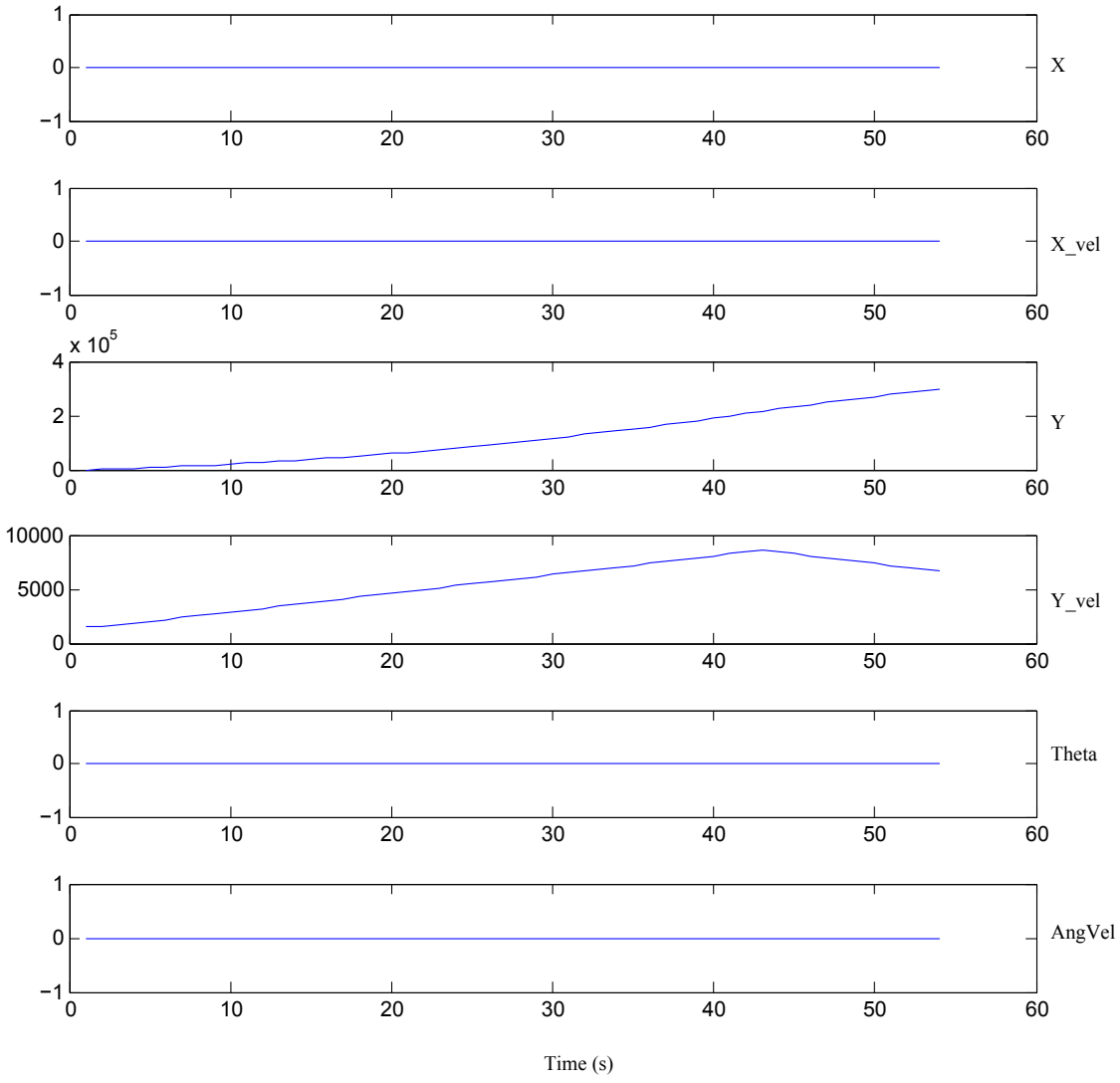

Figure 7: States of Base Case

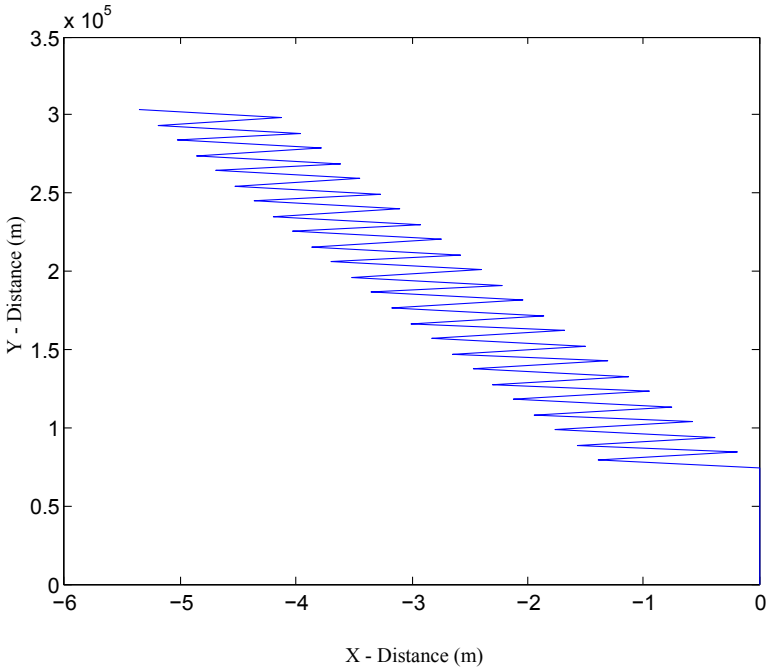

Figure 8: Trajectory of Base Case with disturbance

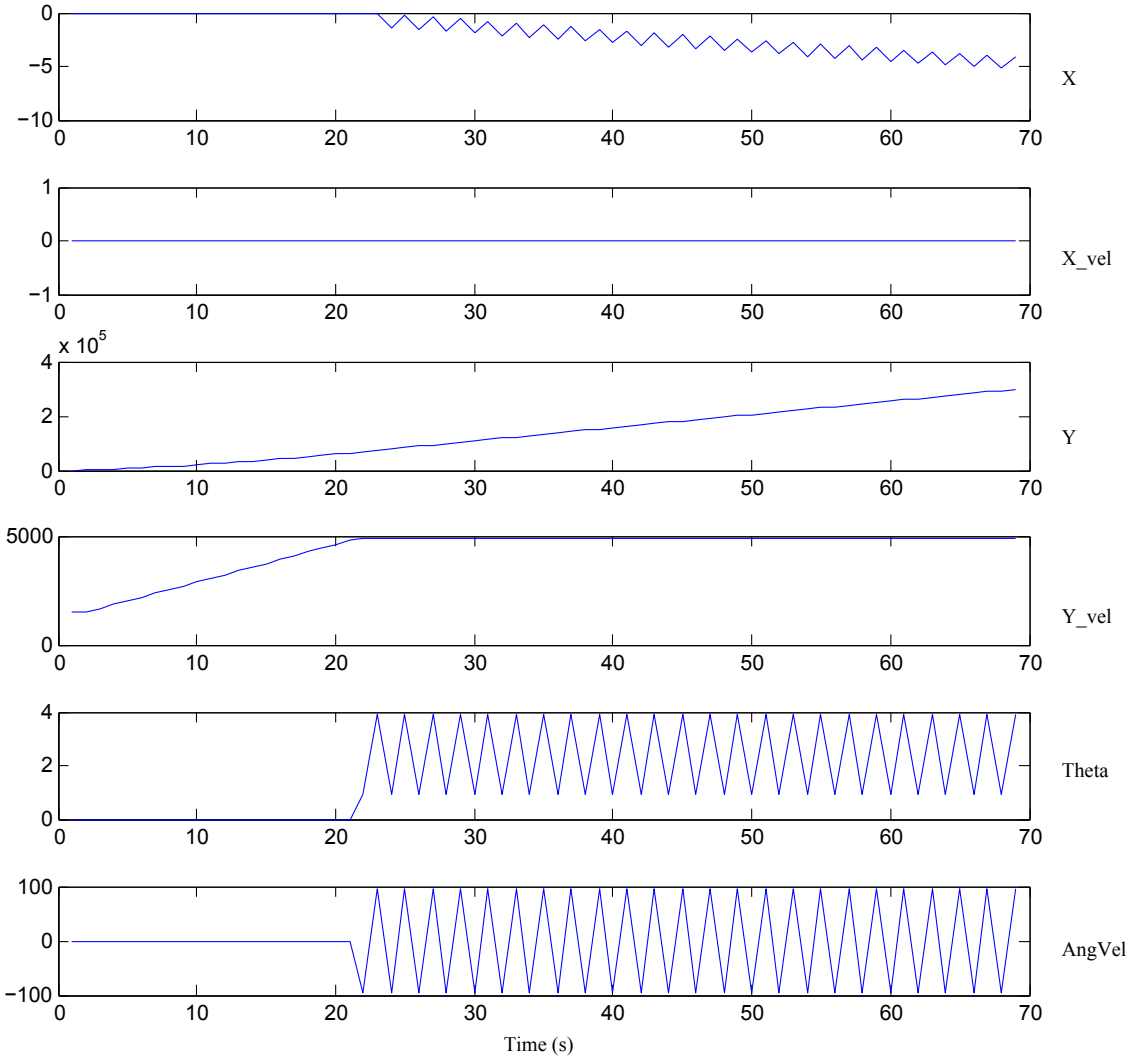

Figure 9: States of Base Case with disturbance

References

[1] David Stephenson. "Mars Ascent Vehicle - Concept Development". In: *38th Joint Propulsion Conference and Exhibit*. 2002. DOI: 10.2514/6.2002-4318.

[2] David D. Stephenson Harvey J. Willenberg. "Mars Ascent Vehicle Key Elements of a Mars Sample Return Mission". In: *Aerospace Conference*. IEEE. 2006. DOI: 10.1109/AERO.2006.1655737.

[3] Mark A. Trinidad Edward Zabrensky Anita Sengupta. "Mars Ascent Vehicle system studies and baseline conceptual design". In: *Aerospace Conference*. IEEE. 2012. DOI: 10.1109/AERO.2012.6187297.

[4] P.N. Desai R.D. Braun W.C. Engelund F.M. Cheatwood J.A. Kangas. "Mars Ascent Vehicle Flight Analysis". In: *7th AIAA/ASME Joint Thermophysics and Heat Transfer Conference*. 1998. DOI: 10.2514/6.1998-2850.

[5] Richard C. Shreffler. "Failure Detection, Isolation and Mitigation for a Spacecraft Solid Fuel Reaction Control System". MA thesis. Massachusetts Institute of Technology, 2012.

[6] Ian J. Dux Joseph A. Huwaldt R. Steve McKamey John W. Dankanich. *Mars Ascent Vehicle Gross Lift-off Mass Sensitivities for Robotic Mars Sample Return*. Tech. rep. NASA, 2011.