EEE4119F
Landing on an Unknown Planet

By Aadam Osman (OSMAAD003)

Table of Contents

Introduction

Given the lander description, this report aims to outline the engineering strategy followed in system identification; and thereafter control system design to achieve the following requirements under the stipulated scenarios:

- y_{final} with 0% overshoot
- y_{final} less than $10\ [m]$ (with respect to the geometric centre of the lander)
- $\left|v_{y_{final}}\right| < 2\ \left[\frac{m}{s}\right]$
- $\left|\theta_{final}\right| < 10\ [deg]$

Along with the following additional requirements:

- Minimizing fuel consumption based on the linear cost function: $\int F_1 + F_2\ dt$
- Minimizing horizontal landing velocity: $\left|V_{x\,final}\right| \approx 0\ \left[\frac{m}{s}\right]$
- Minimizing horizontal landing location: $\left|x_{final}\right| \approx 0\ [m]$

Where these requirements are subjected to the following scenarios (modes):

Mode	$y_{intial}[m]$	$\theta_{intial}\ [rad]$	$\dot{\theta}_{intial}\ [rad/s]$
1	1000 ± 5	0	0
2	1000 ± 50	± 0.5	0
3	1300 ± 200	$\pi \pm 0.5$	± 0.1

It must be highlighted that, all calculations, derivations and designs were all produced in MATLAB; and relevant codes are attached in the appendix with an inferred reference throughout this report.

System Identification

In the brief, the following information was provided:

- Dimension of the lander: $L_1 = 6\ [m]$ and $L_2 = 8\ [m]$
- Mass of the lander: $m = 4000\ [kg]$
- The centre of mass (offset by dL) is above the geometric centre

However, no information pertaining the gravitational acceleration, maximum thruster force (F_{max}), the offset dL and equation of motions subjected to the lander were provided. That being said, this section outlines the procedures followed in determining the above mentioned unknowns, using the given information provided; and is done so below:

Gravity

Given the simulation model, it was acknowledged that there were two means of calculating the acceleration due to gravity. The first uses the general equations of motion of an object subjected to free fall; from this, one can determine the gravitational acceleration given the initial and final velocity, the time of free fall (Δt) and the change in displacement (Δs). This method works, but a far less computational intensive approach, is by means of Newtonian Mechanics, specifically using the equation that governs Newton's Second law: $\sum_i F_i = ma$ – this approach was adopted and outlined below:

In any of the modes (1,2 or 3), set the thruster force outputs to zero – for the reason that the only force subjected to the lander is due to gravity. Given this, the Free Body Diagram (FBD) was developed, and is presented below:

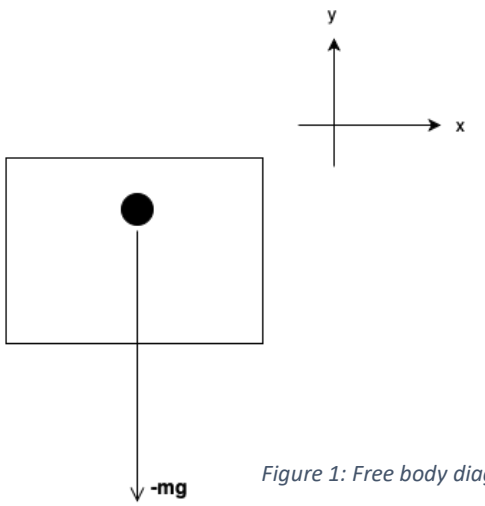


Figure 1: Free body diagram of lander in free fall

From the FBD, the relation of Newton's Second Law (in the y direction) for the given lander is as follows:

$$ma_y = \sum_i F_{y_i}$$
$$\rightarrow ma_y = -mg$$
$$\therefore g = -a_y = -\ddot{y}$$

Running the simulation in mode 1; the following is a plot of $\ddot{y}$:

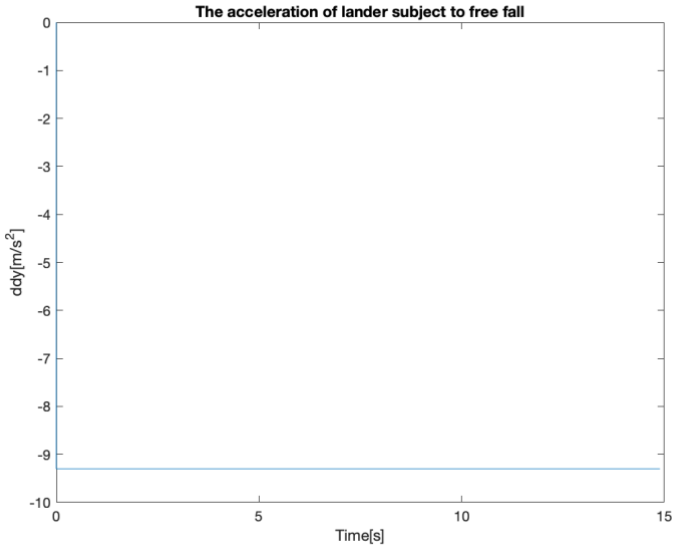


Figure 2: Acceleration of lander in free fall

From the relationship derived previously, the gravitational acceleration is quoted:

$$g = 9.3\ \left[\frac{m}{s^2}\right]$$

Maximum Force

The procedure of determining the maximum force that can be applied to the system, is fairly trivial. It involves applying an increasing thruster force[s] to the system, and determining the instance in which either the angular or linear acceleration saturates. In this report, F_{max} was evaluated when the angular acceleration saturates, and the procedure is outlined below:

1. Set one of the thruster to zero, and the other to a linear increasing force (in *Simulink* make use of the ramp block).
2. Note the instance where the angular acceleration saturates, use this to evaluate the force applied at that point: this is the F_{max} that can be applied by a single thruster.

A plot of the angular acceleration ($\ddot{\theta}$), and F_2 as a linearly increasing force, with $F_1 = 0\ [N]$, can be seen below:

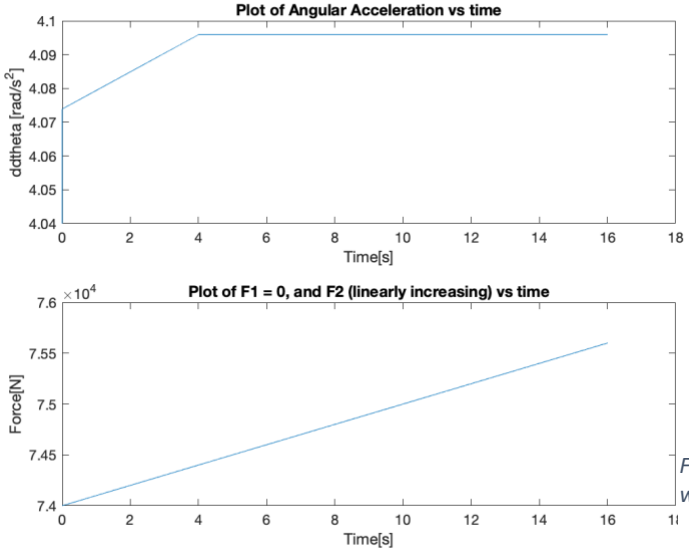


Figure 3: Angular Acceleration Saturation, along with Maximum Force that can be applied

From the plots above, it can be noted that $\ddot{\theta}$ saturates at $4.096\ [\frac{rad}{s^2}]$, when $F_2 \geq 74400$.
Therefore,

$$F_{max} \approx 74400\ [N]$$

And thus, the thruster forces are bounded by the following inequality,

$$0 \leq F_{1,2} \leq 74400$$

Equations of Motion (EOM)

The procedure of determining the equations of motion (that is vital to control system design) is done using Euler-Langrage Mechanics, and is derived below:

It must be noted that the following equations of motions are determined with the fact that dL was not known.

To begin, one has to define the generalised co-ordinates of the system; those being:

$$x - intertial\ axis\ to\ body's\ centre\ of\ mass$$
$$y - intertial\ axis\ to\ body's\ centre\ of\ mass$$
$$\theta - meausred\ relative\ to\ the\ inertial\ horizontal\ axis$$

Therefore, the generalised coordinate vector is quoted:

$$q = [x\ y\ \theta]^T$$

Now that the generalised coordinates have been established, the position vector (using the centre of mass as the origin of the body axis) was formulated:

$$r = \begin{bmatrix} x \\ y \end{bmatrix}$$

where,

$$\dot{r} = \frac{d}{dt}(r) = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}$$

Given the above, the Kinetic and Potential Energy of the system may be calculated as follows:

For Kinetic Energy, it is known that:

$$T_{total} = T_{translational} + T_{rotational}$$

The Translational Kinetic energy is governed by,

$$T_{translational} = \frac{1}{2} m\dot{r}^T\dot{r} = \frac{1}{2} m[\dot{x} \quad \dot{y}] \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{1}{2}\mathrm{m}(\dot{x}^2 + \dot{y}^2)$$
$$\text{where } \mathrm{m} = 4000\ [\mathrm{kg}]$$

$$\therefore \mathrm{T_{translational}} = 2000(\dot{x}^2 + \dot{y}^2)$$

And the rotational kinetic energy,

$$T_{rotational} = \frac{1}{2} I_{cm}\dot{\theta}^2$$

where I_{cm} was determined via parallel axis theorem, by a distance dL away from the geometric centre,

$$I_{cm} = I_0 + m(dL)^2$$

Given that the lander is reduced to a rectangular plane,

$$I_0 = \frac{1}{12}(L_1^2 + L_2^2)$$

Therefore,

$$I_{cm} = \frac{1}{12}(L_1^2 + L_2^2) + m(dL)^2 = 4000dL^2 + \frac{1 \times 10^5}{3}$$

From this, the total kinetic energy is,

$$T_{total} = 2000(\dot{x}^2 + \dot{y}^2) + \left(2000dL^2 + \frac{50000}{3}\right)\dot{\theta}^2$$

The potential energy of the system is simply governed by,

$$V_g = m[0\ g]\begin{bmatrix} x \\ y \end{bmatrix} = mgy = (4000)(9.3)y = 37200y$$

Now that both the Kinetic and Potential energy of the system have been defined, one can begin to formulate the Manipulator Equation:

$$M(q)\ddot{q} + C(q,\dot{q}) + G(q) = Q$$

The elements of the Mass, $M(q)$, Centrifugal/Coriolis, $C(q,\dot{q})$, and Gravity matrix, $G(q)$, were determined using their relationships, and are quoted below:

$$M_{i,j}(q) = \frac{\partial T_{total}}{\partial \dot{q}_i \partial \dot{q}_j}$$

$$\rightarrow M(q) = \begin{bmatrix} 4000 & 0 & 0 \\ 0 & 4000 & 0 \\ 0 & 0 & 4000dL^2 + \frac{100000}{3} \end{bmatrix}$$

$$C(q,\dot{q}) = \dot{M}\dot{q} - \frac{\partial T}{\partial q}$$

$$\rightarrow C(q,\dot{q}) = [0 \quad 0 \quad 0]^T$$

$$G(q) = \frac{\partial V}{\partial q}$$

$$\rightarrow G(q) = [0 \quad 37200 \quad 0]^T$$

Now that the manipulator equation has been derived. The next step, is to determine the Generalised Forces to the system, this was done using the relationship below:

$$Q_i = \sum_{j=1}^{m} F_j \cdot \frac{\partial r_j}{\partial q_i}$$

It must be noted that Generalized Forces are to be defined in the inertial frame, and thus transformation of the thrust forces and their relevant positions in the body frame to the inertial frame is required. The procedure followed is presented below:

Given the diagram of the lander, the position of F_1 and F_2 in the body frame are:

$$r_{F_1}^1 = [-3 \quad -dL - 4]^T$$
$$r_{F_2}^1 = [3 \quad -dL - 4]^T$$

The positions of the two thruster forces, respectively, in the inertial frames was determined by a rotation transform matrix; where the positions are quoted below:

$$r_{F_1}^0 = [x\ y]^T + R_1^0\begin{bmatrix} -3 \\ -dL - 4 \end{bmatrix} = \begin{bmatrix} x - 3\cos(\theta) + \sin(\theta)\,(dL + 4) \\ y - 3sin(\theta) - \cos(\theta)(dL + 4) \end{bmatrix}$$
$$r_{F_2}^0 = [x\ y]^T + R_1^0\begin{bmatrix} 3 \\ -dL - 4 \end{bmatrix} = \begin{bmatrix} x + 3\cos(\theta) + \sin(\theta)\,(dL + 4) \\ y + 3sin(\theta) - \cos(\theta)(dL + 4) \end{bmatrix}$$

Where,

$$R_1^0 = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

And the thruster forces in the inertial frame are quoted below (using the rotation matrix specified above):

$$F_1^0 = R_1^0\begin{bmatrix} 0 \\ F_1 \end{bmatrix} = \begin{bmatrix} -F_1\sin(\theta) \\ F_1\cos(\theta) \end{bmatrix}$$
$$F_2^0 = R_1^0\begin{bmatrix} 0 \\ F_2 \end{bmatrix} = \begin{bmatrix} -F_2\sin(\theta) \\ F_2\cos(\theta) \end{bmatrix}$$

Finally, the generalized forces can be determined and are quoted below:

$$Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_\theta \end{bmatrix} = \begin{bmatrix} -F_1 \sin(\theta) - F_2 \sin(\theta) \\ F_1 \cos(\theta) + F_2 \cos(\theta) \\ 3F_2 - 3F_1 \end{bmatrix}$$

It is known that,

$$M(q)\ddot{q} + C(q, \dot{q}) + G(q) = Q$$

And therefore the equation of motions are determined:

$$\ddot{q} = M(q)^{-1}\big(Q - C(q, \dot{q}) - G(q)\big)$$

$$\therefore \ddot{q} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} -\dfrac{\sin(\theta)\,(F_1 + F_2)}{4000} \\ \dfrac{F_1 \cos(\theta)}{4000} + \dfrac{F_2 \cos(\theta)}{4000} - \dfrac{93}{10} \\ -\dfrac{9F_1 - 9F_2}{4000(3dL^2 + 25)} \end{bmatrix}$$

Determining dL

From the EOM above, it can be seen that dL depends only on $\ddot{\theta}$, F_1 and F_2. Thus solving for dL, with respect to those states, gives:

$$dL = \frac{\pm\sqrt{30}\sqrt{\dfrac{9F_1 - 9F_2 + 100000\ddot{\theta}}{\ddot{\theta}}}}{600}$$

And as it can be seen from the simulation model, one has access to all of these states. Thus, using the above equation, dL was plotted over time with $F_1 = 0[N]$ and $F_2 = 1000\ [N]$:

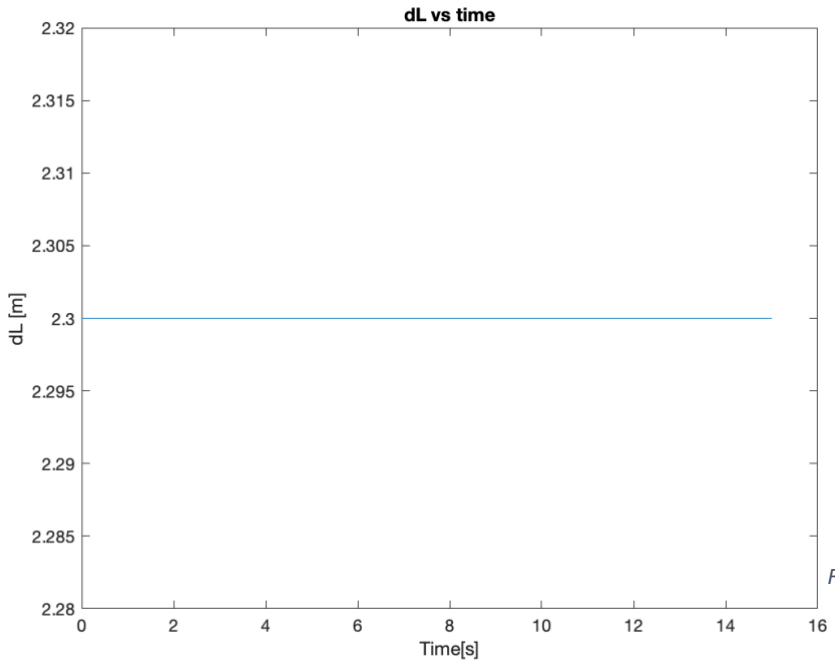


Figure 4: Determining dL

From the figure above, it is evident that $dL = 2.3\ [m]$.

Since dL is known, the final EOM was determined:

$$\ddot{q} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} -\dfrac{\sin(\theta)\,(F_1 + F_2)}{4000} \\ \dfrac{F_1\cos(\theta)}{4000} + \dfrac{F_2\cos(\theta)}{4000} - \dfrac{93}{10} \\ -\dfrac{9F_1 - 9F_2}{163480} \end{bmatrix}$$

Controller Design

Given the equation of motions derived previously, the non-linear state space model is stated below:

$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ \theta \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ -\dfrac{\sin(\theta)\,(F_1 + F_2)}{4000} \\ \dot{y} \\ \dfrac{F_1\cos(\theta)}{4000} + \dfrac{F_2\cos(\theta)}{4000} - \dfrac{93}{10} \\ \dot{\theta} \\ -\dfrac{9F_1 - 9F_2}{163480} \end{bmatrix}$$

Proposed Control Design Methods

Given the fact that the system is non-linear, one can propose the possibility of introducing the use of non-linear model predictive control (NMPC). This control method, has seen extraordinary results in controlling complex non-linear systems. However, in the eyes of this report the non-linear system derived are under the jurisdiction of soft non-linearities (due to the trigonometric dependencies), thus the need for such a high complex control algorithm is not required, although it may provide superior results. Thus, the second control option (and the one carried out throughout this report), is to first linearize the system about an operating point (with small Δ in the states and control action), and perform traditional linear control (that being a state-feedback control system). The pure reason, of using a state-feedback controller is simply because one has access to all states and the fact that one is dealing with a MIMO system – it is tremendously less computational expensive in formulating a MIMO state-feedback control system over a general PID controller.

Linear State-space formulation

In order to linearize the system, one must determine the equilibrium/operating point, such that non-linear system is stable about the given point, this was derived in the following manner:

Symbolically the condition needed to be satisfied is:

$$\dot{x} = f(x^*, u^*) = 0$$

where $\dot{x}$ represents the states.

From the EOM, the condition formulation (with $F_1, F_2 \neq 0$) is as follows:

$$\ddot{x} = -\frac{\sin(\theta)\,(F_1 + F_2)}{4000} = 0 \rightarrow \sin(\theta)\,(F_1 + F_2) = 0 \ldots (1)$$

$$\ddot{y} = \frac{F_1\cos(\theta)}{4000} + \frac{F_2\cos(\theta)}{4000} - \frac{93}{10} = 0 \rightarrow (F_1 + F_2) = \frac{mg}{\cos(\theta)} \ldots (2)$$

$$\ddot{\theta} = -\frac{9F_1 - 9F_2}{163480} = 0 \rightarrow F_1 = F_2 \ldots (3)$$

$$where\ m = 4000\ [kg], g = 9.3\ [\tfrac{m}{s^2}]$$

Given that $F_1, F_2 > 0$,

From (1) it is noted: $\theta = 0[rad]$

Using (1), from (2) it is noted: $(F_1 + F_2) = mg$

Using (2) and (3): $F_1, F_2 = \frac{mg}{2}\ [N]\ \ and\ \theta = 0\ [rad]$

Given the equilibrium point, the non-linear state space model may be linearized using a local linearization method around the operating point. This was done in *MATLAB*:

```
f_x_u = [dx;ddx;dy;ddy;dtheta;ddtheta];
A = jacobian(f_x_u,[x;dx;y;dy;theta;dtheta]);
B= jacobian(f_x_u,[F1;F2]);
lin_force = m*g/2;
A_lin = double(subs(A,[theta F1 F2],[0 lin_force lin_force]));
B_lin = double(subs(B,[theta F1 F2],[0 lin_force lin_force]));
```

Controller Design (via LQR)

The following design process was conducted to achieve $y, \dot{y}, \theta, x \; and \; \dot{x}$ control as well as reducing the control cost. The controller was designed using an LQR control algorithm, to assist in operating the dynamic system with minimum cost, hence the choice of using this control method. The procedure was done so under an observation based method, which involved analysing the results obtained in each mode, and weighting the Q [6x6] and R[2x2] cost matrices accordingly (required for the LQR control law algorithm). A summary of the tuning procedure followed is presented in the table below:

Mode	Iteration	Comments	Solution	Result
1	1.	Lander is not landing ($y > 100\ m$) with $\dot{y} < -12\frac{m}{s}$	Increase Q_y from 1 to 1E4	Lander lands, but vertical velocity to high when landing
	2.	Lander is in now landing, but $\dot{y} < -15\frac{m}{s}$	Increase $Q_{\dot{y}}$ from 1 to 4.3E5	Lander Lands perfectly, but time to land is 45 seconds
	3.	Lander lands perfectly, but the cost must be reduced by reducing the landing time.	Increase Q_y from 1E4 to 1.9E5. Increase $Q_{\dot{y}}$ from 4.3E5 to 1.71E6.	Lander Lands perfectly within 26 seconds, thus reducing cost.
2.	1.	Lander is not landing near $x = 0$, and $\dot{x}$ is too high when landing	Increase Q_x from 1 to 9.5E9 and $Q_{\dot{x}}$ from 1 to 7.5E9	Final $\lvert x \rvert \approx 0.02\ [m]$ and $\lvert \dot{x} \rvert = 0.01\ [\frac{m}{s}]$
	2.	Sometimes, the control action due to correcting $x \; and \; \dot{x}$ causes the system to rotate above 90 degrees – leading to instability.	Increase Q_θ from 1 to 1E5 and $Q_{\dot{\theta}}$from 1 to 1E13 – to reduce the aggressive rate that the angle changes.	Stability is now guaranteed, while achieving all of the above results above.
3	1.	When the system, is around 180 degrees, the controller used in the previous modes fails to correct the angle, and thus leading to an unstable system.	Design a phase controller, to correct the angle of the controller first then allow the 6 state controller to take over in a certain angle range. Therefore using LQR, design a $\theta \; and \; \dot{\theta}$ controller, with Q_θ and $Q_{\dot{\theta}}$ equal to 1E10 and 1E7 respectively, with a less aggressive control action achieved by setting $R_{F_1}$and $R_{F_2}$to 1000	System, is stable, and achieves all the above results accordingly in all three modes.

Tuning in mode 1 (additional notes)

Beginning with mode 1, with both cost matrices initially set to their respective Identity matrices: $Q = I_{[6\times6]} \; and \; R = I_{[2\times2]}$. It was found that under these cost conditions, the lander was not landing. Thus, Q_y was increased (to 1E4) – which provided a somewhat favourable result: the lander lands, but with a high velocity. Given this, the cost weighting $Q_{\dot{y}}$ was increased to 4.3E5 – which allowed the lander to land with a vertical within the requirement constraint, however it took the lander 45 seconds to achieve this – thus having a high cost function. To reduce the cost, meant reducing the time in which the lander lands, therefore Q_y was weighted to 1.9E5. But the lander was now landing with a high speed, thus $Q_{\dot{y}}$

was increased further to 1.71E6. Producing a favourable cost, vertical speed and landing height. Allowing further development of the controller, which was done in mode 2.

Tuning in mode 2 (additional notes)

In mode 2, the first desired attribute was x and $\dot{x}$ control (for additional marks). The problem faced, was that when the lander rotates to correct its angle, it moves further away from the origin. Therefore, the need to increase the dominance of x and $\dot{x}$ control is required. This was achieved by increasing Q_x to 9.5E9 and $Q_{\dot{x}}$ which provided extremely favourable results (outlined in the summary above). However, upon, increasing these weightings accordingly, it resulted in aggressive θ control which lead to unstable runs. To reduce this aggressiveness, the rate at which θ changes was weighted higher, hence $Q_{\dot{\theta}} = 1\text{E}13$; and Q_{θ} was increased to 1E5 to ensure that landing angle was minimized.

Tuning in mode 3 (additional notes)

Mode 3 proposed a bigger problem. One would assume that the previous controller designed would somewhat be able to control the system even if the lander starts upside down (i.e. $\theta_i = 180\ [deg]$). Unfortunately, it was not the case. This is because the system was linearized around an operating point $\theta_i = 0\ [deg]$. It is also not possible to linearized about $\theta_i = 180\ [deg]$, because in this case: $F_{1,2}\ must\ equal\ -\frac{mg}{2}$, which is not possible (due to the thruster constraint discussed previously). However, a way to overcome this, is to design two-phase control system. The first being, the one previously designed; and the second being an angle controller – such that it controls the orientation of the lander to a point where controller 1 will then initiate thereafter. This was carried out.

For controller 2:

Initially, $Q_{[2\times2]} = I_{[2\times2]}$ and $R_{[2\times2]} = I_{[2\times2]}$. From here, Q_{θ} and $Q_{\dot{\theta}}$ were increased to 1E10 and 1E7 respectively to provide the necessary angle correction. However, the control action was far too aggressive, allowing the lander's x position to divert outwards by 100 or sometimes 200 meters (while in the air). Thus to reduce this effect, the aggressiveness of the controller was reduced by increasing R_{F_1} and R_{F_2} to 1000.

Controller Implementation

The implementation of the controller was fairly simple. It was based on the following pseudo-code principle:

If abs(θ) > 70 degrees -> while(angle is not aprox. 0 degrees) -> correct angle [using controller 2]
else provide 6 state control [using controller 1]
if (control action<0) set control action to 0
if (control action> Fmax) set control action to Fmax

In the simulink model, a while loop was simulated using a memory block and a mode variable. The code derived is presented below:

```
function [F1, F2, modeout] =
fcn(x,dx,y,dy,th,dth,K, K_180,modein)
%account for landing gear
y = y-10;

%mode variable used to simulate while
loop
modeout = modein;

%th is greater than 73 degrees, begin
while loop process
if (abs(th)>1.3)
    modeout = 1;
end

%while loop process carried out
if(modeout==1)
    %controller 2
    u = -K_180*[th;dth];
    F1 = u(1);
    F2 =u(2);
    %condition to terminate while loop
    if(abs(th)<0.09)
        modeout = 0;
    end
```

```
else
    %else case controller 1
    u = -K*[x;dx;y;dy;th;dth];
    F1 = 4000*9.3/2 +u(1);
    F2 = 4000*9.3/2 +u(2);
    modeout =0;
end

%negative control action case
if(F1<0)
    F1=0;
end
if(F2<0)
    F2=0;
end

%saturation limits
if(F1>74400)
    F1=74400;
end
if(F2>74400)
    F2=74400;
end
end
```

Stability Assessment of Controller 1 for Phase Control System

It was previously noted that the control system designed and applied to the lander, to achieve the stipulated requirements, was of type phase. No other information has been stated to justify choice of the phase condition (*abs(θ) > 70 degrees)* in which Controller 2 would be active. The procedure that was used to determine the given condition was by analysing the ability of Controller 1 to stabilise the system for various angles ranging from 0 to 90 degrees; or in other words did the lander manage to land safely based on an initial angle criteria. The binary plot of stability vs angle is presented below (in mode 2):

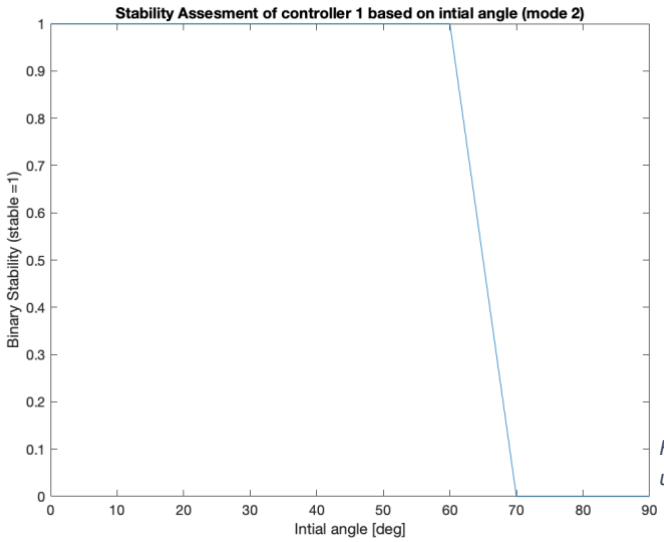


Figure 5: Stability Assessment of Controller 1, used determine phasing in the control system

Given the plot above, the phase condition choice stipulated above should now be apparent.

Results

Mode 1

From the plots presented in Appendix A, it can be seen that the lander touches the ground safely at 10 [m] with a vertical velocity ($|\dot{y}|$) of approximately 0.4 $\left[\frac{m}{s}\right]$, and worst cost estimate of 9.458E5 [Ns].

Mode 2

From the plots presented in Appendix B, it can be seen that the lander touches the ground safely at 10 [m] with a with a vertical velocity ($|\dot{y}|$) of approximately 0.4 $\left[\frac{m}{s}\right]$, an angle θ of $0.002\ [rad]$, a horizontal position (x) of approximately $0.015\ [m]$, horizontal velocity ($\dot{x}$) of $0.0063\ [\frac{m}{s}]$ and a worst cost estimate of 9.923E5 [Ns].

Mode 3

From the plots presented in Appendix C, it can be seen that the lander touches the ground safely at 10 [m] with a with a vertical velocity ($|\dot{y}|$) of approximately 0.4 $\left[\frac{m}{s}\right]$, an angle θ of $0.006\ [rad]$, a horizontal position (x) of approximately $0.015[m]$, horizontal velocity ($\dot{x}$) of $0.02\ [\frac{m}{s}]$ and a worst cost estimate of 1.042E6 [Ns].

The position plot of all three modes are presented below:

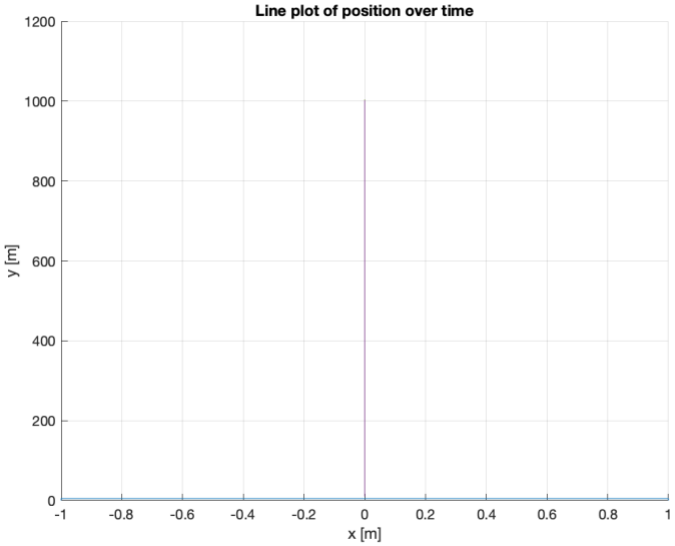


Figure 6: Line Plot in mode 1

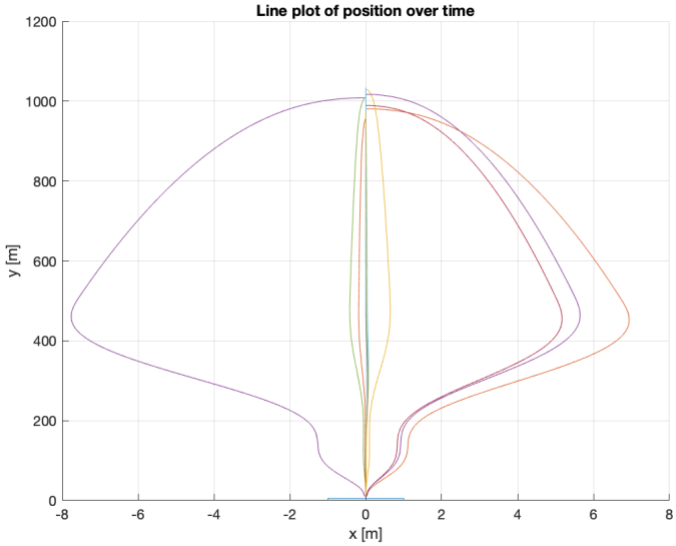


Figure 6: Line Plot in mode 2

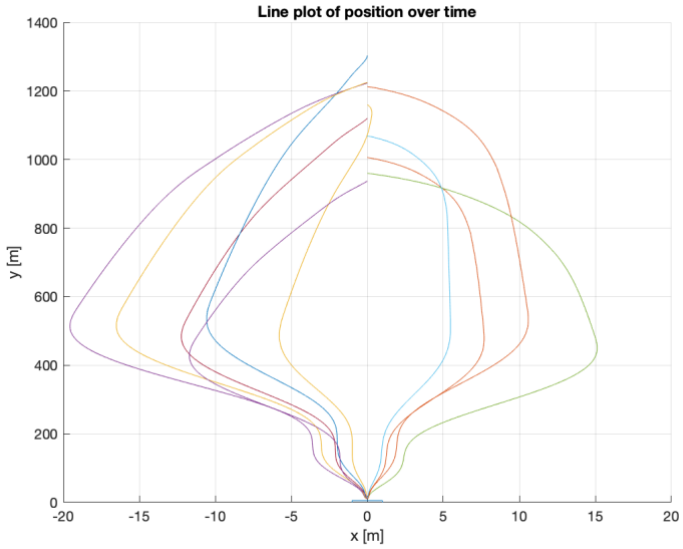
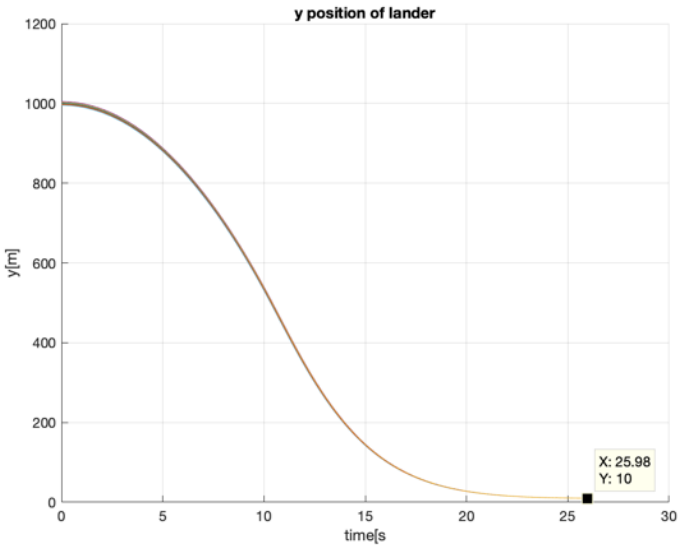
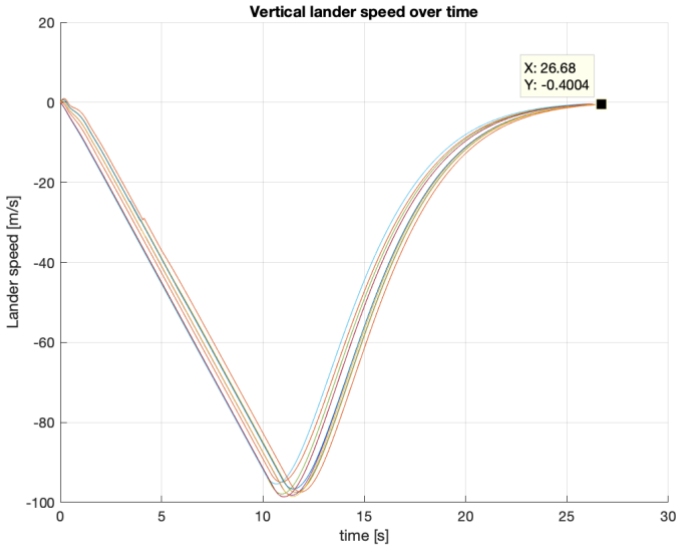


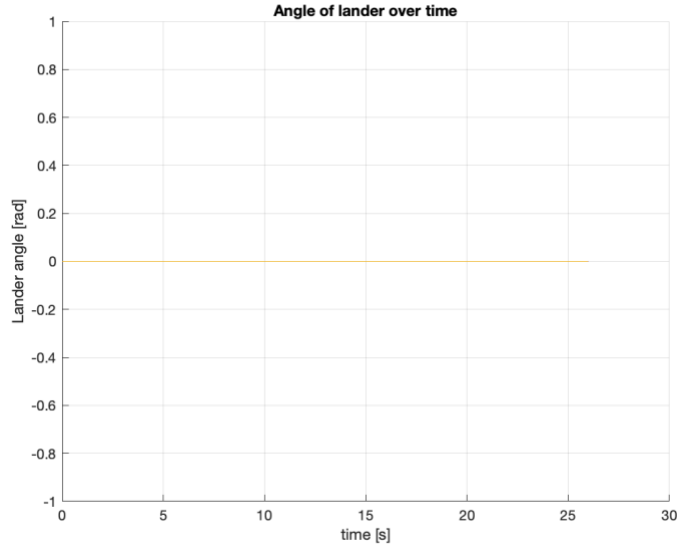
Figure 8: Line Plot in Mode 3

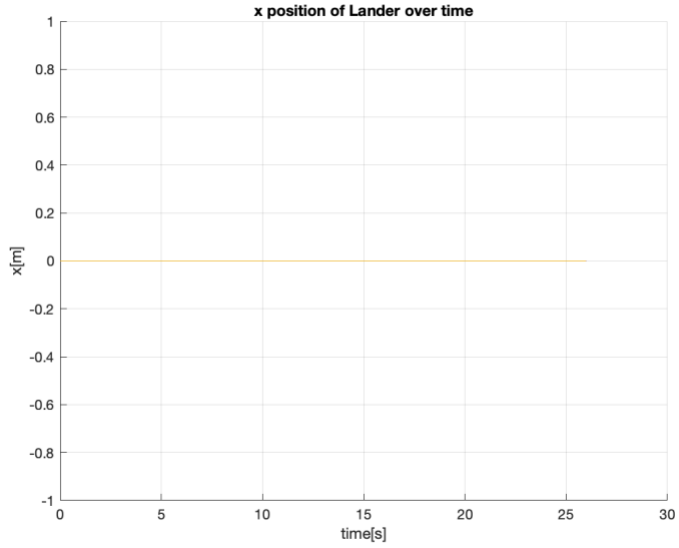
Appendix

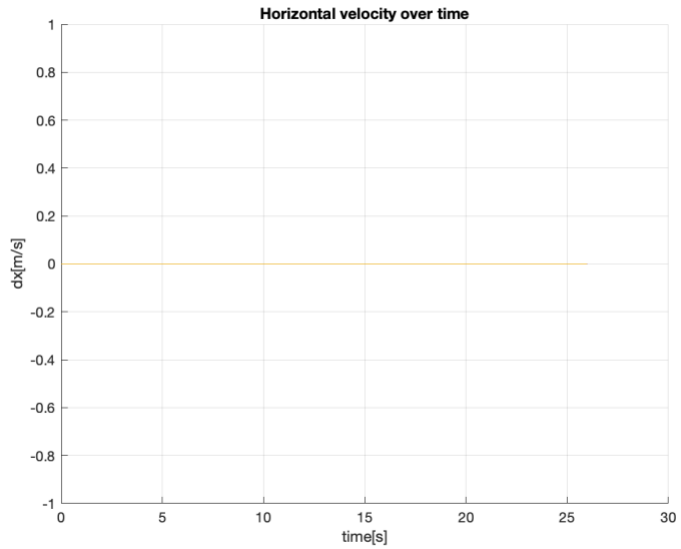
Appendix A: Mode 1 results

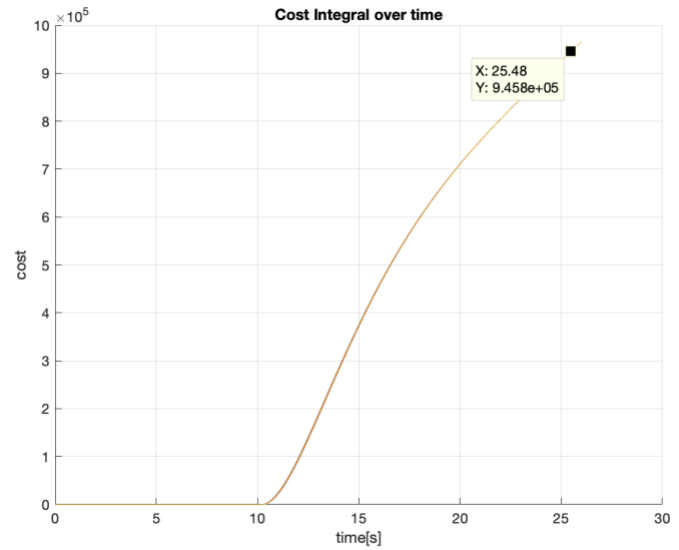




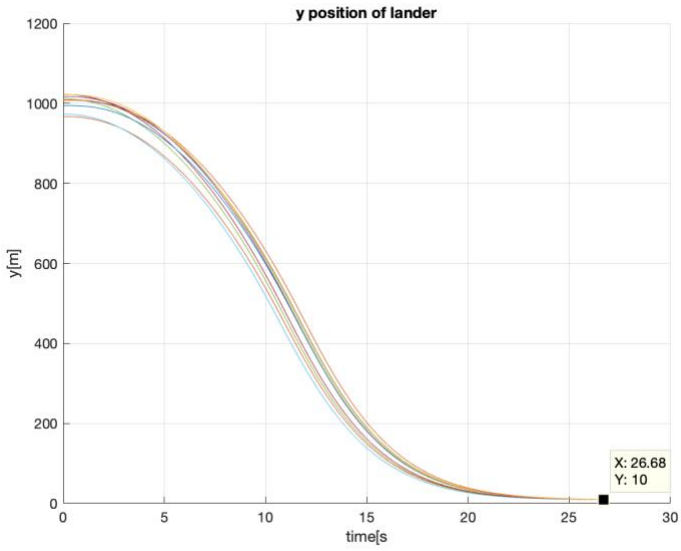


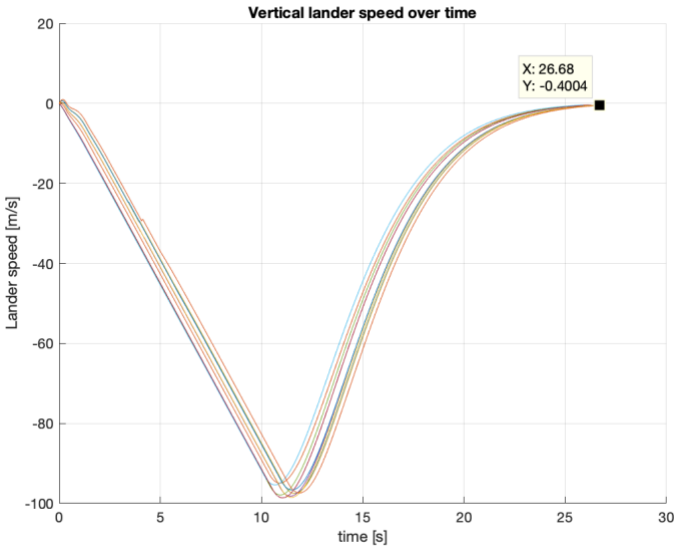


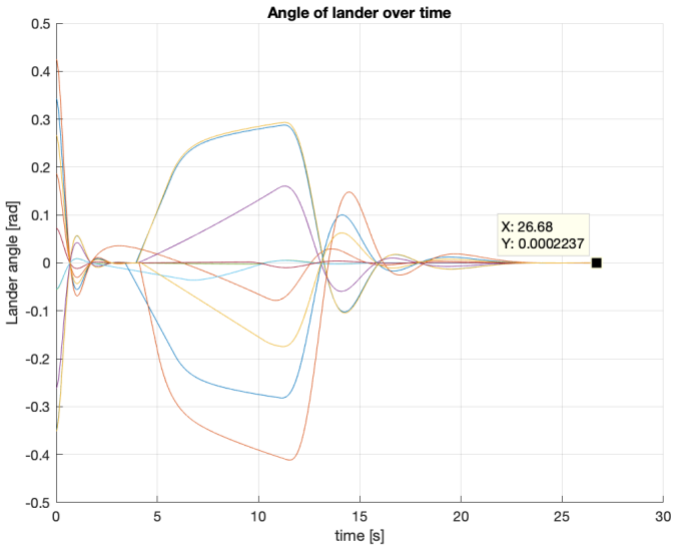


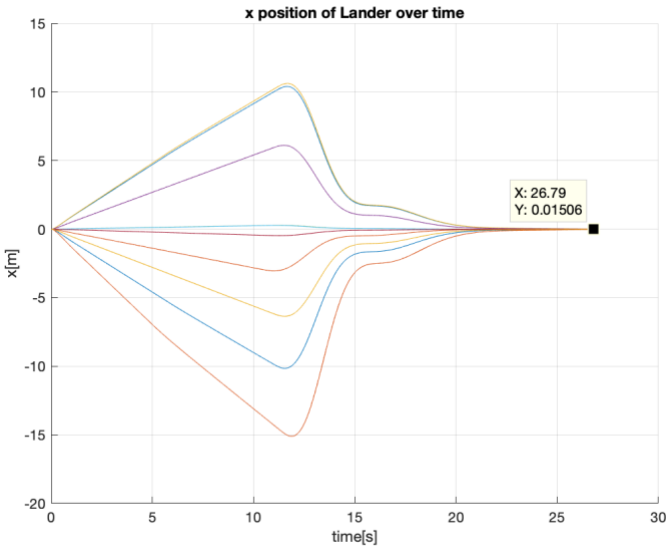


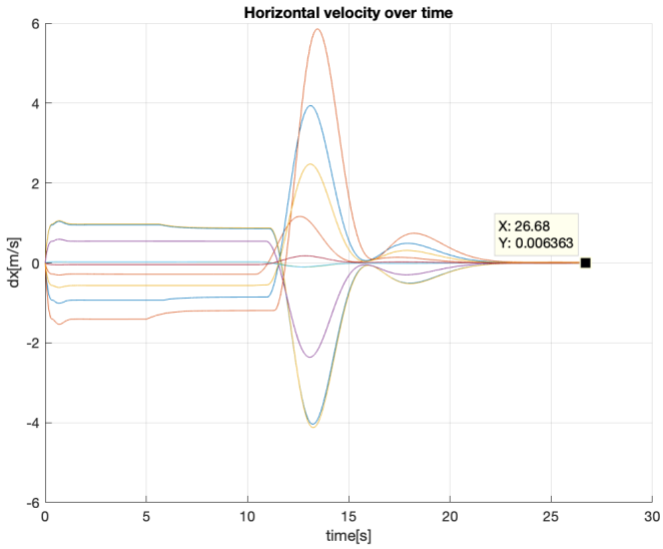
Appendix B: Mode 2 Results

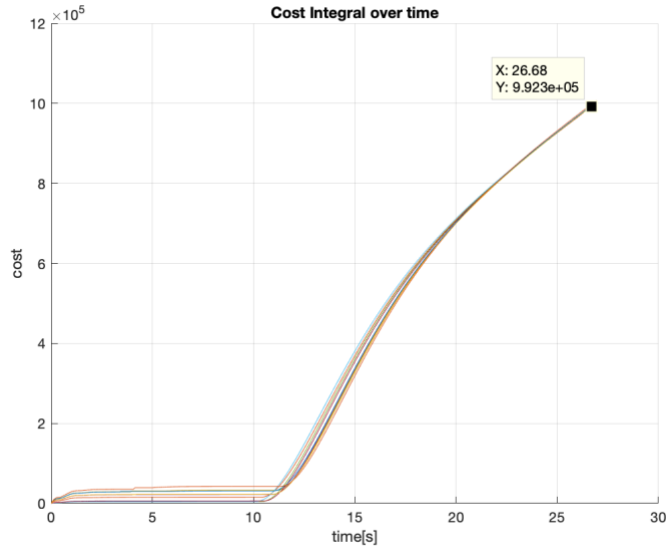




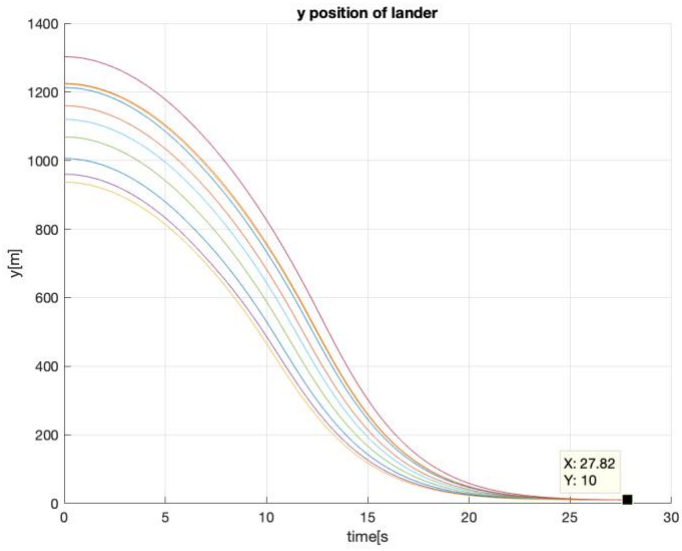


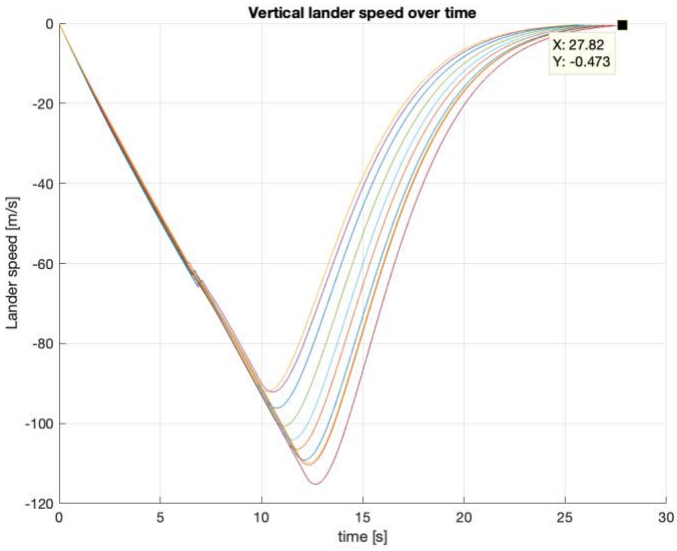


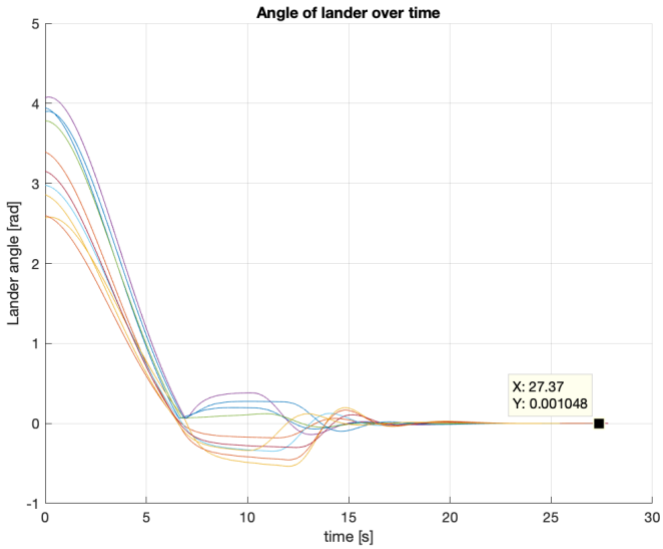


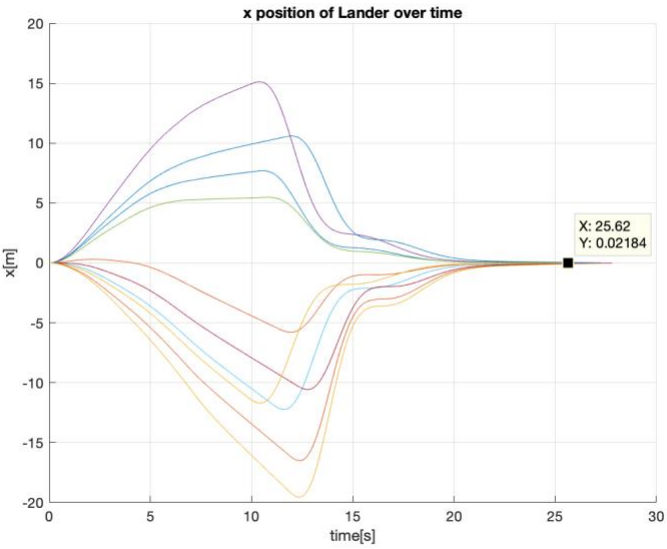


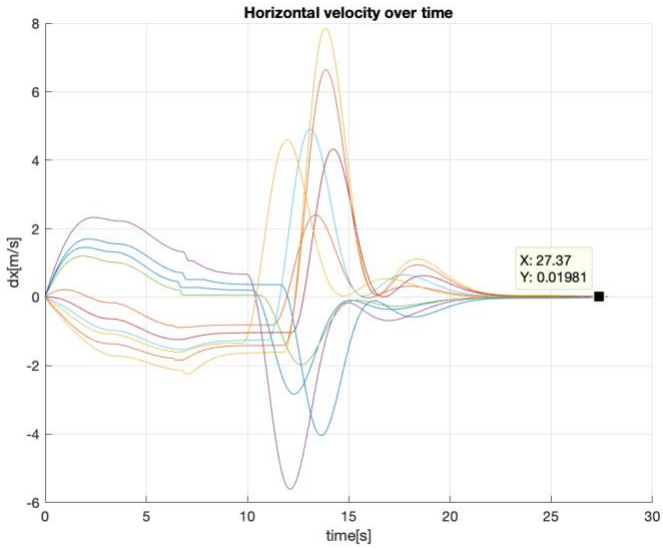
Appendix C: Mode 3 Results

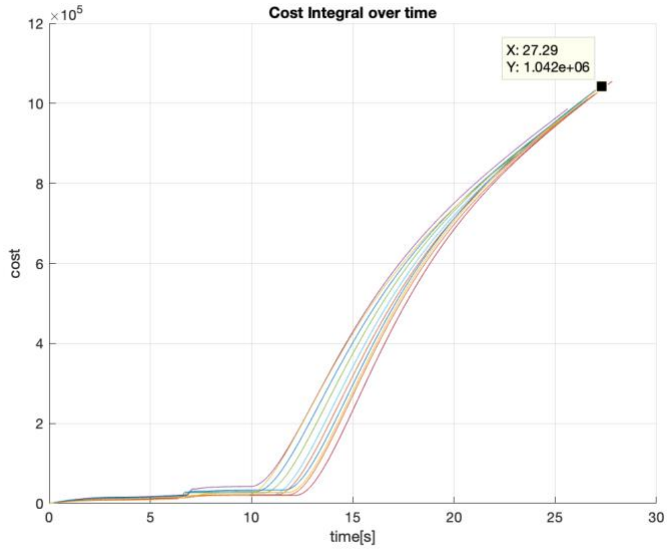












Appendix D: MATLAB Code (System Identification)

```
syms x y theta dx dy dtheta ddx ddy ddtheta 'real';
syms F1 F2 dL 'real' ;


g=9.3;
L1 = 6;
L2 = 8;
m = 4000;

q = [x;y;theta];
dq =[dx;dy;dtheta];
ddq =[ddx;ddy;ddtheta];


r = [x;y];
r_dot = jacobian(r,q)*dq;

I = (1/12)*m*(L1^2+L2^2) + m*(dL^2);

T_body = 0.5*m*transpose(r_dot)*r_dot + 0.5*I*dtheta^2;

U_body = r(2)*g*m;


for i = 1 : length(q)

    for j = 1: length(q)

        M(i,j) = diff(diff(T_body,dq(i)),dq(j));
    end
end

M = simplify(M);

dM = sym(zeros(length(M), length(M)));
for i = 1:length(M)
    for j = 1:length(M)
        dM(i,j) = jacobian(M(i,j),q)*dq;
    end
end

dM = simplify(dM);
C = dM * dq - jacobian(T_body, q)';

C = simplify(C);

G = sym(zeros(length(M),1));
for i = 1: length(q)
    G(i,1) = diff(U_body,q(i));
end
G = simplify(G);


Manip_eq = M*ddq+C+G;


R0_1 = [cos(theta) sin(theta);-sin(theta) cos(theta)];
rF2_1 = [L1/2;-1*dL-L2/2];
rF1_1 = [-L1/2;-1*dL-L2/2];

rF1_0 = [x;y] + transpose(R0_1)*rF1_1;
rF2_0 = [x;y]+ transpose(R0_1)*rF2_1;

F1_0 = transpose(R0_1)*[0;F1];
F2_0 = transpose(R0_1)*[0;F2];

Qx = transpose(F1_0)*(diff(rF1_0,x))+transpose(F2_0)*(diff(rF2_0,x));
Qy = transpose(F1_0)*(diff(rF1_0,y))+transpose(F2_0)*(diff(rF2_0,y));
Qtheta= transpose(F1_0)*(diff(rF1_0,theta))+transpose(F2_0)*(diff(rF2_0,theta));

ddq_ans = inv(M)*([Qx;Qy;Qtheta]-C-G);
ddq_ans = simplify(ddq_ans);
```

Appendix E: MATLAB (Controller Design)

```
%Mechatronics 2 Lander Controller Design
%Author: Aadam Osman (OSMAAD003)
%Date:20 April 2019

clear;
clc;

syms x y theta dx dy dtheta F1 F2;

%mass and gravity
g=9.3;
m=4000;

%from EOM code
ddx = -(sin(theta)*(F1+F2))/4000;
ddy =  (F1+F2)*(cos(theta))/4000 - 93/10;
ddtheta = (9*(F2-F1))/163480;

%non-linear state space model
f_x_u = [dx;ddx;dy;ddy;dtheta;ddtheta];

%linearize non-linear system
A = jacobian(f_x_u,[x;dx;y;dy;theta;dtheta]);
B= jacobian(f_x_u,[F1;F2]);

%assign intial condition theta = 0, F1 = mg/2 and F2=mg/2
lin_force = m*g/2;
A_lin = double(subs(A,[theta F1 F2],[0 lin_force lin_force]));
B_lin = double(subs(B,[theta F1 F2],[0 lin_force lin_force]));

%definition of cost matrices for controller 1
Q= zeros(6,6);
R =zeros(2,2);

%Assigning relevant cost weightings
Q(1,1) = 9.5E9;%x
Q(2,2) = 7.5E9;%dx
Q(3,3) = 1.9E5;%y
Q(4,4) = 1.71E6;%dy
Q(5,5) = 1E5;%th
Q(6,6) = 1E13;%dth

R(1,1) = 1; %F1
R(2,2) = R(1,1); %F2

%controller design via LQR
K = lqr (A_lin,B_lin,Q,R);


%controller 2
%non-linear state space model
f_x_u = [dtheta;ddtheta];

%linearise system about operating point theta = 0, F1 = 0 and F2 =0
A = jacobian(f_x_u,[theta;dtheta]);
B= jacobian(f_x_u,[F1;F2]);

A_lin = double(subs(A,[theta F1 F2],[0 0 0]));
B_lin = double(subs(B,[theta F1 F2],[0 0 0]));

%defining cost matrices
Q = zeros(2,2);
R= zeros(2,2);

%assigning the associated weightings
Q(1,1) = 1E10;%theta
Q(2,2) = 1E7;%dtheta

R(1,1) = 1000;%F1
R(2,2) = R(1,1);%F2
```

```
K_180 = lqr(A_lin,B_lin,Q,R);
```