Institut Superior de l'Aéronautique et de l'Espace

MAE 602: Control of Dynamic Systems and Implementation

SUPAERO's Quadrotor Lab Control Project Report

Submitted on:
11 May 2022

Authors:
Akash SHARMA, Marco Wijaya

Contents

Introduction

The goal of this study is to design and test different controllers for different phases of the flight of a quadrotor simulator drone. The controllers are designed using MATLAB/Simulink and compared to the simulated behavior in the Bebop virtual environment. After the requirements were fulfilled, the controllers were ready to be tested on the real drone in the facility.
Overall, the control laws have been designed for vertical as well as lateral flight. The objective will be to have the drone change its altitude followed by performing trajectory control before it ultimately lands again.

The simulation environment has various aspects of the flight already programmed which allows us to focus on just designing the control laws. These include taking off and landing. There are several safety protocols in place as well which prevent the drone from crashing or getting damaged in any way during the course of the tests.

The quadrotor system was designed to fly in some different phases such as: take-off, hovering, flying, and landing with settings as the following:

- Take-off

  The quadrotor rises from 0 m height and is stable at the targeted altitude of 1.5m. This happens within 0-5s.
- Hovering

  During the hovering phase, the quadrotor is supposed to maintain its vertical position at 1.5m unless another signal is given.
- Flying

  This is the phase of the flight where the control laws are tested
- Landing

  The landing phase makes the quadrotor descent to the ground with a target reference velocity of 0m/s.

Vertical Control

In this vertical flight phase, the quadrotor drone is controlled to achieve a certain altitude through a **step input of 0.5m** once the quadrotor has successfully taken off to an altitude of 1.5m. The input for this flight is the reference altitude (Z_{ref}) and the output is the model vertical velocity (V_z).

The performance targets for the control law design are as follows:

- No static error
- Time response less than 1s (time to stabilize within the 5% bandwidth of target altitude)
- Overshoot less than 20%
- Phase margin more than 45 degrees

The following equations have been used to relate the given performance requirements of time response and overshoot to manipulatable variables, namely, cutoff frequency and damping ratio.

Overshoot: $D = 100 \exp\left(-\frac{\pi\sigma}{\sqrt{1-\sigma^2}}\right)$ Time response: $t_r^{5\%} = \frac{3}{\sigma\omega}$

With these formulas and taking the performance requirements, D = 20% and $t_r^{5\%} = 1s$, we get the limits $\boldsymbol{\sigma > 0.45}$ **and** $\boldsymbol{\omega < 6.66\, rad/s}$.

In order to successfully control a system, it is important to correctly approximate the dynamics of the system before you go ahead and choose an appropriate controller. To do this, first, a simple model is chosen to understand its performance against the Bebop simulator. Then, the model is upgraded to include higher order dynamics.
Three controllers have been designed to be incorporated with these two models. The combination of the models and controllers and also the results are as follows.

2.1. Pure Proportional (P) controller with 1st order system

A simple 1st order model consisting of a pure integrator was chosen as the first iteration for the test. To control the vertical speed, the controller was tuned knowing the following governing equation for the model:

$$V_z = \dot{z}$$

The Simulink model in Figure 1 shows the visualization of the comparison that is made between the model and Bebop simulation.

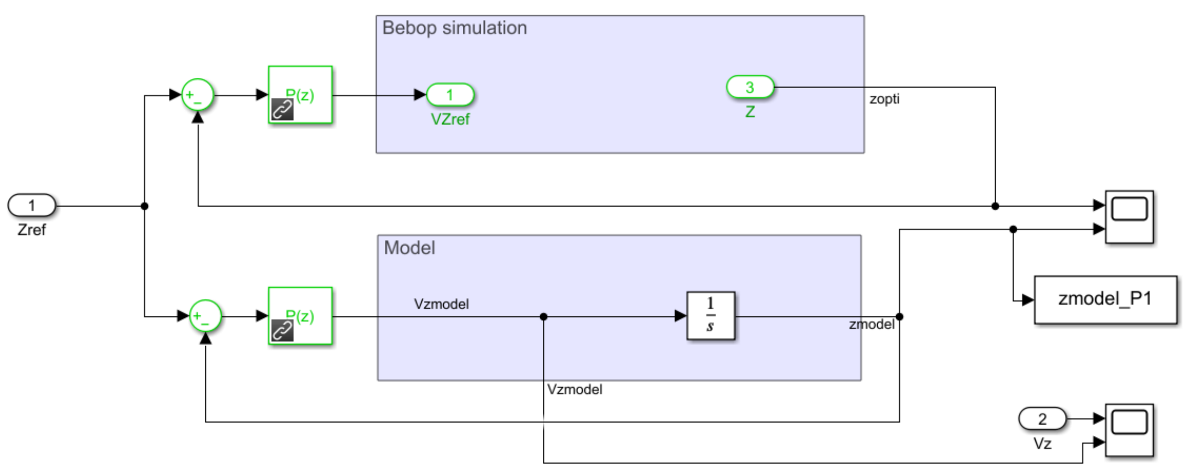

Figure 1: Block diagram for 1st order system with pure proportional controller

After the take-off phase, a step of **0.2m at 15s** in altitude was introduced to study the behavior of the control system. The dynamics of the simulation, the model as well as the conducted experiment of the Bebop drone are graphed with the area of interest in Figure 2.

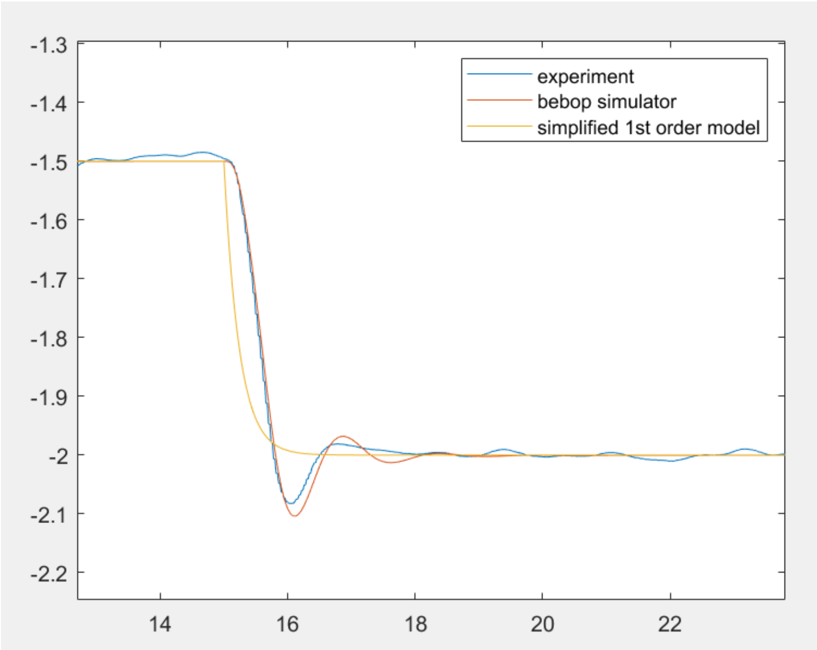

Figure 2: Altitude (m) vs time (s) for P-controller with 1st order system

It is evident from this graph that the 1st order model offers only a poor approximation of the dynamic behavior. Tuning the gain to the desired characteristics has resulted in poor agreement with the Opti track simulator. Since, the Opti track simulator produces an overshoot, a higher order model is required. The performance requirements of the model are met, though, as we see the steady state error goes to 0 and the system is stable enough as seen from the experimental data.

The performance metrics are:
K = 4
Time response = 0.7s
Overshoot = 0%
Phase Margin = 90°

2.2. Pure Proportional (P) controller with 2nd order system

Based on the previous result, it was decided to upgrade the model. In the previous result, only the governing equation which defined the relationship between altitude and velocity was modeled. However, it is also required to identify a dynamic between the target and the current vertical velocity of the drone. This is modeled by a first order system as follows making the full system a 2nd order system.

$$\frac{V_z}{V_{zref}} = \frac{1}{1 + \tau s}$$

After some trial and error where we tried matching the model with the Opti track response, the ideal value of $\boldsymbol{\tau}$ **= 0.33** was found. The new Simulink model and the altitude step response are shown below.

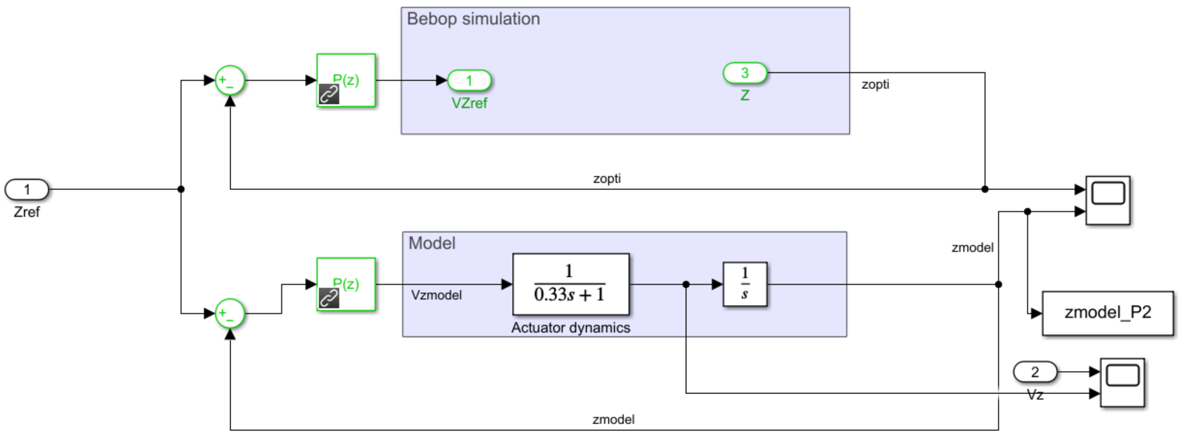

Figure 3: Simulink Model with higher order dynamics included

We have the open loop transfer function:

$$OL = \frac{K}{\tau s^2 + s}$$

And the closed loop transfer function:

$$CL = \frac{OL}{1 + OL} = \frac{\frac{K}{\tau}}{s^2 + \frac{s}{\tau} + \frac{K}{\tau}}$$

Comparing this with the characteristic equations, $\omega^2 = K/\tau$

$\boldsymbol{\sigma = 0.75}$ was chosen which satisfies the limit $\sigma > 0.45$.

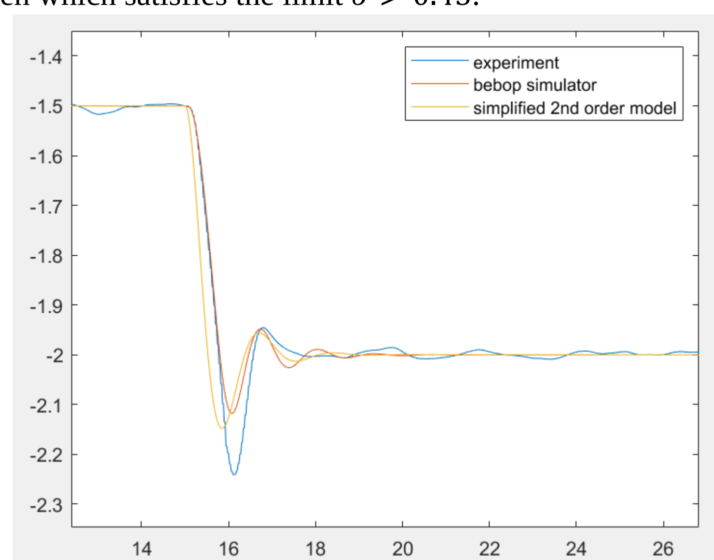

Figure 4: Altitude(m) vs Time(s) for P-controller with 2nd order system

This graph shows a good agreement between the 2nd order model, simulated quadrotor, and the quadrotor experiment. Particularly, it displays the overshoot which could not be detected by the 1st order model. **However, it does not satisfy the performance constraints defined**. The time response particularly could not be brought down to below 1 second even after allowing the response to overshoot close to the limit. The phase margin is not ideal as well.

The performance metrics are:
K = 5.2071
Time response = 1.98s
Overshoot = 7.27%
Phase Margin = 41.4°

For these reasons, it was decided to switch to a PD controller in which the derivative gain would control the overshoot while the proportional gain can maintain a swift time response.

2.3. Proportional Derivative (PD) controller with 2nd order system

With this system, a similar method of comparing the characteristic equation to the model was implemented to approximate the appropriate gain values. The associated data and the graph is as shown.

We have the open loop transfer function:

$$OL = \frac{P + Ds}{\tau s^2 + s}$$

And the closed loop transfer function:

$$CL = \frac{OL}{1 + OL} = \frac{P + Ds}{\tau s^2 + (D + 1)s + P} = \frac{\frac{P + Ds}{\tau}}{s^2 + \frac{(D + 1)}{\tau}s + \frac{P}{\tau}}$$

Comparing this with the characteristic equations, we obtain the relations:

$$P = \omega^2 \tau \qquad\qquad D = 2\sigma\omega\tau - 1$$

$\boldsymbol{\sigma = 0.7}$ was chosen which satisfies the limit $\sigma > 0.45$.

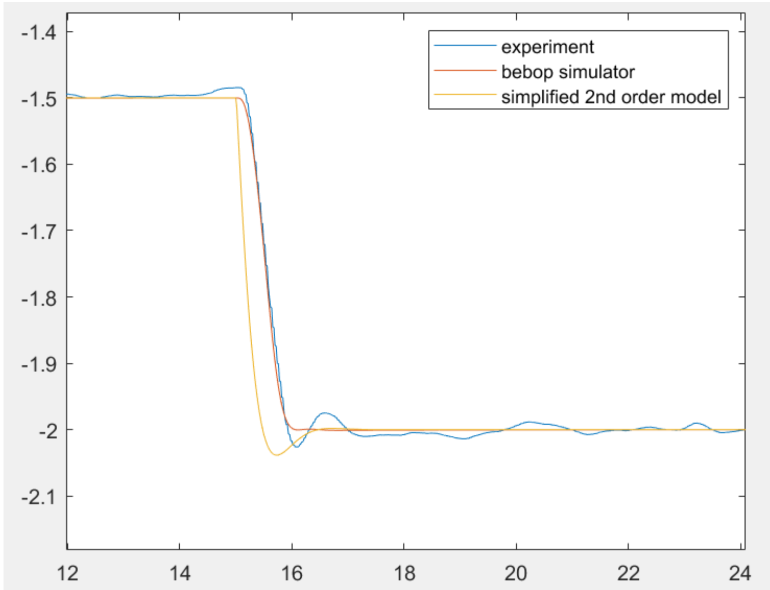

Figure 5: Altitude(m) vs Time(s) for PD-controller with 2nd order system

The performance metrics are:
$K_p = 6.0612$
$K_d = 0.98$
Time response = 0.95s
Overshoot = 1.9%
Phase Margin = 70°
Considering the performance requirements that are to be satisfied, the model and controller work to give a good enough agreement with the simulator while satisfying the performance metrics.

2.4. Proportional Derivative and Integral (PID) controller with 2nd order system

Finally, a PID controller was introduced for the 2nd order system which was tuned by trial and error. The result of altitude variation is graphed as follows.

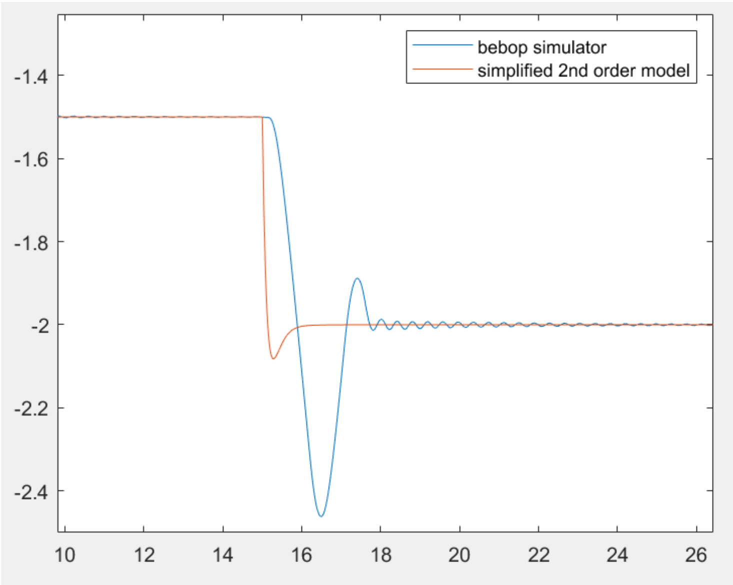

Figure 6: Altitude(m) vs Time(s) for PID-controller with 2nd order system

The performance metrics are:
K_p = 25.23
K_i = 37.39
K_d = 3.806
Time response = 1.75s
overshoot = 21.5%
Phase Margin = 73.6°

The performance seems to have worsened as compared to the one with the PD controller. The purpose of adding an additional integral gain is to force the steady state error to 0. However, as can be seen in Figure(5), the steady state error already goes to 0, hence the integral gain is redundant here and, in fact, ends up interfering in the dynamics of the system.

Lateral Control

To test the lateral control performance, the quadrotor drone is flown in both x and y directions. A trajectory of a square of **0.2m** is chosen for this test. A state feedback controller is implemented to be able to tune position as well as velocity gains. Since the dynamics of both x and y directions are the same, the same controller is used for both. The full system has to satisfy the following requirements:

- No static error
- 5% - time response less than 2s
- Rising time = 0.6s (time needed for the quadrotor model to travel from 10% to 90% of the target altitude)
- Overshoot less than 15%

The governing equations for lateral flight in the x and y direction are:

$$g\theta = \ddot{y} \qquad\qquad g\varphi = -\ddot{x}$$

In state space feedback, the control law with a pre-filter is as follows:

$$u = -Kx$$

Where, x is the state vector.

Like the methodology followed in vertical control, the limits of the damping ratio σ and cut-off frequency ω have been calculated with the given performance requirements D = 15% and $t_r^{5\%} = 2s$. These are as follows: $\sigma > 0.51$ & $\omega < 2.9\, rad/s$

The state space model corresponding to the governing equations is as follows:

$$\dot{x} = Ax + Bu$$

where, $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ g \end{bmatrix}$

The feedback gains are calculated by placing the desired closed loop poles. This is done by using "place" function on MATLAB.
The state space feedback model has been designed as below:

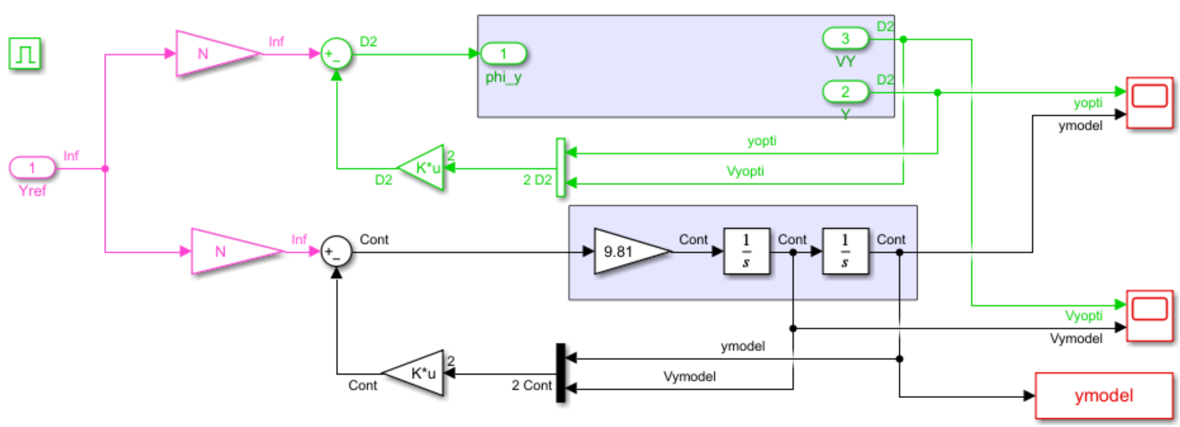

Figure 7: Simulink Model for modelling the y-direction

An additional gain of -1 was added in the x direction to consider the default reference system.
After some trial and error, values of $\sigma = 0.7$ & $\omega = 2.5$ were deemed to be appropriate as the resulting performance matched the one with the Opti track. These values satisfy the above-mentioned limits as well.
The metric of rise time has been defined as the time taken by the response to go from 10% to 90% of the target value. Since the target is 0.2m, these are 0.02m and 0.18m respectively.

After calculating the gains through this method, the results of lateral X and Y variations were graphed as follows.

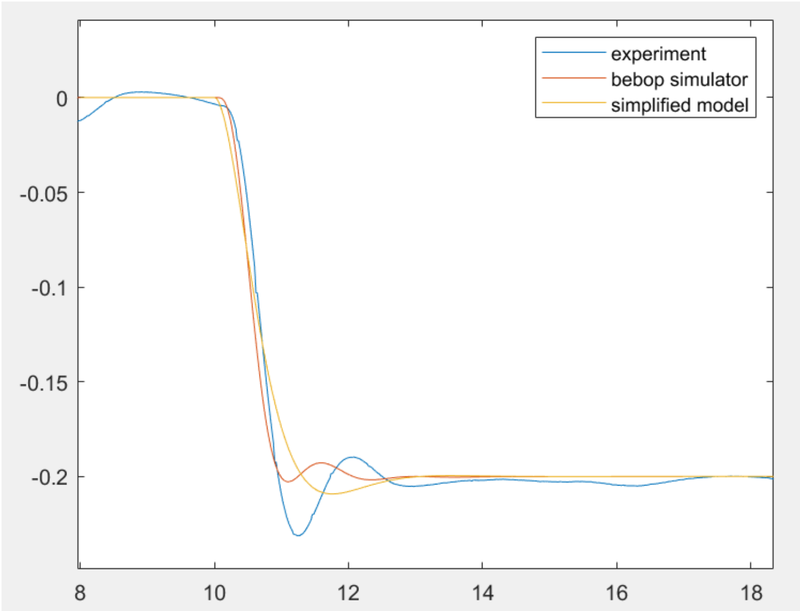

Figure 8: X distance (m) vs Time (s) for state feedback controller

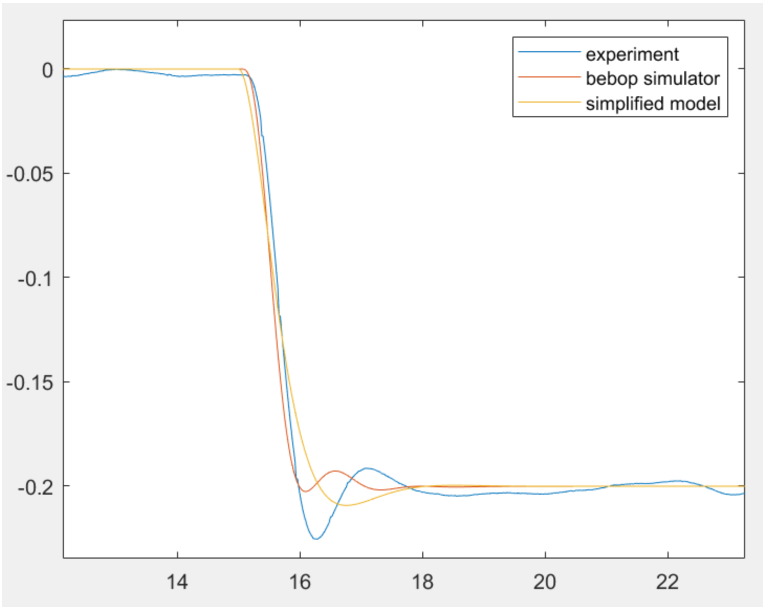

Figure 9: Y distance (m) vs Time (s) for state feedback controller

The performance metrics are:
K1 = 0.6371
K2 = 0.3568
N = 0.6371
Time response = 1.17s
Overshoot = 5%
Rise Time = 0.85s

We can see from these values that all requirements except the rise time are met. The challenge while selecting the right cutoff frequency and damping ratio, was to meet these requirements as well as have the model and the Opti track agree. However, since the rise time is still 0.2s slow, it has been deduced that the model is adequate to mimic the desired behavior and, thus, needs to be improved. Like in the case of vertical control, the actuator dynamics must be considered where the reference angle (θr) and the actual angle (θ) are related through a first order dynamic:

$$\frac{\theta_z}{\theta_{zref}} = \frac{1}{1 + \tau s}$$

Once this is done, one can see a better approximation of the higher order dynamics of lateral control with better performance.

Conclusion

This project deals with iterative attempts at approximating and controller the performance of a Bebop drone using MATLAB/Simulink. This task has been broken down into two phases of flight – vertical and lateral flight. The vertical flight has been modelled using combinations of 1st and 2nd order systems coupled with P,PD and PID controllers to test their performance. In the end, a 2nd order system controlled using a PD controller has yielded the best results.
The lateral control has been modelled through a double integrator simple model with a state feedback controller that has yielded almost satisfactory results.