7-2021

Fault Tolerant Deep Reinforcement Learning for Aerospace Applications

Christoph Elias Aoun
Embry-Riddle Aeronautical University, aounc@my.erau.edu

FAULT TOLERANT DEEP REINFORCEMENT LEARNING FOR AEROSPACE APPLICATIONS

By

Christoph Elias Aoun

A Thesis Submitted to the Faculty of Embry-Riddle Aeronautical University

In Partial Fulfillment of the Requirements for the Degree of

Master of Science in Aerospace Engineering

July 2021

Embry-Riddle Aeronautical University

Daytona Beach, Florida

FAULT TOLERANT DEEP REINFORCEMENT LEARNING FOR AEROSPACE APPLICATIONS

By

Christoph Elias Aoun

This Thesis was prepared under the direction of the candidate's Thesis Committee Chair, Dr. Hever Moncayo, Department of Aerospace Engineering, and has been approved by the members of the Thesis Committee. It was submitted to the Office of the Senior Vice President for Academic Affairs and Provost, and was accepted in the partial fulfillment of the requirements for the Degree of Master of Science in Aerospace Engineering.

THESIS COMMITTEE

Hever Moncayo Digitally signed by Hever Moncayo Date: 2021.07.29 16:19:40 -05'00'

Richard J. Prazenica Digitally signed by Richard J. Prazenica Date: 2021.07.29 20:19:38 -04'00'

Chairman, Dr. Hever Moncayo

Member, Dr. Richard Prazenica

K. Merve Dogan Digitally signed by K. Merve Dogan Date: 2021.07.30 00:04:26 -04'00'

Member, Dr. Kadriye Merve Dogan

Daewon Kim Digitally signed by Daewon Kim Date: 2021.07.30 12:55:42 -04'00'

Graduate Program Coordinator,
Dr. Daewon Kim

Date

Maj Dean Mirmirani Digitally signed by Maj Dean Mirmirani DN: cn=Maj Dean Mirmirani, o=Embry Riddle Aeronautical University, ou=Dean - College of Engineering, email=mirmioram@erau.edu, c=US Date: 2021.07.30 14:23:47 -04'00'

Dean of the College of Engineering,
Dr. Maj Mirmirani

Date

Maj Dean Mirmirani Digitally signed by Maj Dean Mirmirani DN: cn=Maj Dean Mirmirani, o=Embry Riddle Aeronautical University, ou=Dean - College of Engineering, email=mirmioram@erau.edu, c=US Date: 2021.07.30 14:24:34 -04'00'

Associate Provost of Academic Support,
Dr. Christopher Grant

Date

ACKNOWLEDGEMENTS

This section is dedicated to everyone who has stood by me especially throughout the several problems which I've been through during these past two years. Primarily, I would like to thank my father, mother, and brother for being there for me even when they are continents away from me. I wouldn't have been here without their encouragement and moral support all throughout these times even when they were going through worse situations than I can withstand. Thank you for being the rock on which I am building my palace.

A shout-out also to all my friends all around the world found in all the countries and from different nationalities. You have been my support system that I relied on along all the times of the day. A big thanks to my roommates for being there for me and supporting me for the past two years. Moreover, the new friendships and family which I've built at Embry-Riddle has helped me establish my skills not only academically, but also socially and mentally.

This all would not be possible without the generous support by the Department of State under the Fulbright Scholarship Program. I would not have even come to this institution without their trust in me and my vision and capabilities.

Finally, a big thank you to Dr. Hever Moncayo for his supervision, guidance, trust, and tolerance. His help in all the various levels of my academic journey at the Advanced Dynamics and Controls lab at Embry-Riddle was phenomenal. He was always ready to step in and help me at any point and trusted me by welcoming me from day one into his lab. Thank you for the opportunity and I hope I was up to the responsibility.

ABSTRACT

With the growing use of Unmanned Aerial Systems, a new need has risen for intelligent algorithms that not only stabilize or control the system, but rather would also include various factors such as optimality, robustness, adaptability, tracking, decision making, and many more. In this thesis, a deep-learning-based control system is designed with fault-tolerant and disturbance rejection capabilities and applied to a high-order nonlinear dynamic system. The approach uses a Reinforcement Learning architecture that combines concepts from optimal control, robust control, and game theory to create an optimally adaptive control for disturbance rejection. Additionally, a cascaded Observer-based Kalman Filter is formulated for estimating adverse inputs to the system. Numerical simulations are presented using different nonlinear model dynamics and scenarios. The Deep Reinforcement Learning and Observer architecture is demonstrated to be a promising control system alternative for fault tolerant applications.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

LIST OF TABLES

ABBREVIATIONS

ADCL	Advanced Dynamics and Control Laboratory
LQR	Linear Quadratic Regulator
PID	Proportional-Integral-Derivative
DL	Deep Learning
DRL	Deep Reinforcement Learning
ARE	Algebraic Ricatti Equation
HJB	Hamilton-Jacobi-Bellman
HJI	Hamilton-Jacobi-Isaacs
IMU	Inertial Measurement Unit
EKF	Extended Kalman Filter
UKF	Unscented Kalman Filter
LQG	Linear Quadratic Gaussian
SMC	Sliding Mode Controller
FL	Feedback Linearizer
NLDI	Nonlinear Dynamics Inversion
MPC	Model Predictive Controller
NST	Nested Saturation Technique
BC	Backstepping Controller
FLB	Fuzzy Logic Based
ANN	Artifical Neural Network
RL	Rienforcement Learning
IL	Iterative Learning
MB	Memory Based
BEL	Brain Emotional Learning
I-PID	Intelligent PID
GA	Genetic Algorithm

GAN	Generative Adversarial Neural Networks
LQG	Linear Quadratic Gaussian
LQE	Linear Quadratic Estimator
EID	Equivalent input disturbance
ESO	Extended state observer
GPIO	Generalized proportional integral observer
UDE	Uncertainty and disturbance estimator
UIO	Unknown input observer
DOB	Disturbance Observer
NDOB	Nonlinear Disturbance Observer

1. Introduction

In recent years, there has been an increase in the use of Unmanned Aerial Vehicles (UAV) for different types of missions. However, one of them main challenges has become to build controls that can reject various disturbances especially when dealing with new sometimes unknown environments. This is clearly portrayed in Figure (1.1) which shows the Mars Ingenuity Rover, which is dealing with an environment unknown to man.

Figure 1.1 Ingenuity Rover On Mars (Aeronautics & Administration, 2021).

When dealing with unpredictable environments, a robust, optimal, and adaptive system should be established in order to control the vehicle. One of the main difficulties in aerospace systems is the identification and quantification of disturbances that can occur during vehicle operation. However, with recent discoveries in artificial intelligence various algorithms have been proposed to address issues of robustness, adaptability, and optimality applied specifically to disturbance rejection.

Different branches of learning algorithms have developed, such as Machine Learning (ML), Deep Learning (DL), Reinforcement Learning (RL) and many others, the trend and its applications have become versatile in all of its formulations and combinations. With the arrival of such artificial intelligence concepts and the basis of Neural Networks, as well as high processing computers, a new domain has opened for human mental mimicry. This helped individuals formulate the mental functions of a human into an algorithm which can help solve issues in real life. This important Neural Network concept is portrayed in Figures

(1.2) and (1.3). As such, this thesis describes a deep learning method which includes aspects of optimal control algorithms along with optimization concepts from Neural Networks.

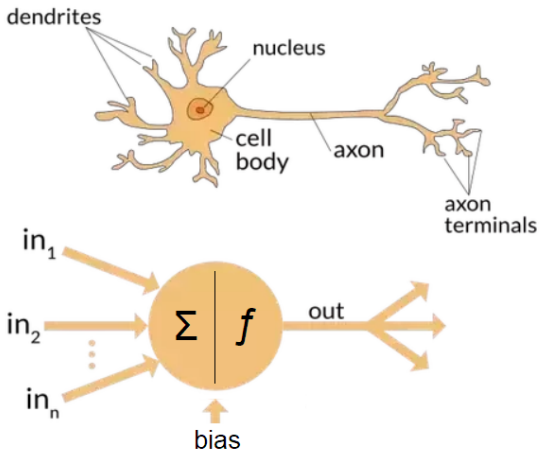


Figure 1.2 Comparison Between a Neuron as an equation and in the Brain (Nagyfi, 2018).

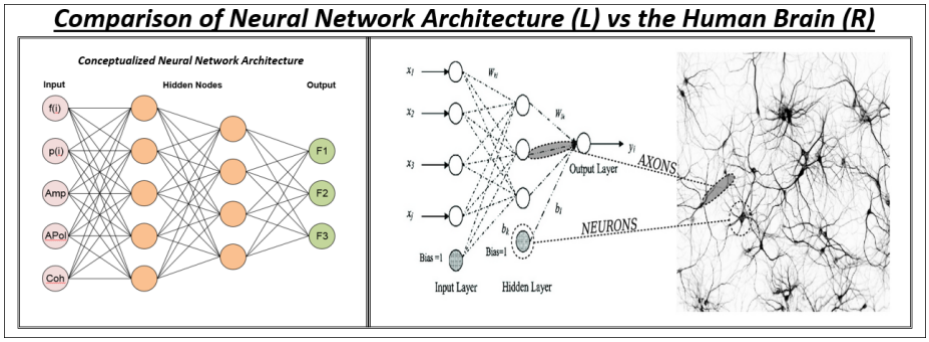


Figure 1.3 Comparison Between a Computer and Brain Neural network (Nagyfi, 2018).

1.1. Thesis Objectives

The main purpose of this research is to develop a Deep Learning algorithm which is capable of controlling a high order system with online learning characteristics along with disturbance rejection. The observability aspect of the problem adds upon the complexity of the proposed system as it will incorporate cascaded Kalman filters which would be used for total system observability. To achieve this goal, several elements must be identified and are summarized as follows:

- A study of different kinds of errors for UAV's and their dynamics

- A survey of all various control methods for UAV's
- Different methods of disturbance identification
- Different types of Artificial Intelligence algorithms

In this document, each point will be addressed by a distinguished chapter with further explanations and elaborations which will culminate with the proposed algorithm and the advances established from these proposals.

1.2. Thesis Outline

A generalized background and context of the problem is provided in Chapter 2, which includes the various surveys. This includes the errors to UAV systems, UAV control systems, and Artificial Intelligence algorithms.

Chapter 3 presents an Online Deep Learning algorithm that is proposed alongside a cascaded Kalman filter for state, dynamics, and disturbance estimation for full system observability and fault tolerance. This proposed system includes concepts from optimal control, robust control, game theory, and estimation.

Chapter 4 includes results of the generalized algorithm along with its variations and comparisons to other methods such as optimal control. Numerical simulations are performed using various platforms and dynamics including a state space, the DJI Phantom 2 quadrotor, and the Crazyflie 2.0 quadrotor.

Chapter 5 delves into main conclusion of the results and discusses the various options for elaborating upon the proposed method in future work.

2. Background

In this chapter, a various survey of different topics will be discussed including quadrotor dynamics, various faults that might be present for the systems, and different control methods. Moreover, a review of different estimation systems are studied as well as revewing the vast library of Artificial Intelligence methods and algorithms.

2.1. Quadrotor Dynamics

To describe high order system dynamics, such as a quadrotor, a generalized coordinate transformation should be established, and a set of aerodynamics concept shall be provided.In order to start with localizing the dynamics of the equation the forces governing the system should be transformed from the inertial frame (world frame) to the body frame, and also states should be transformed from body frame to inertial frame. These transformations are done usually under the scope of Euler transformation matrices and Euler angles using ZYZ rotation axes (Christoph Aoun & Shammas, 2019). This is shown in Figure (2.1) which shows the difference between the iniertial and body frames.

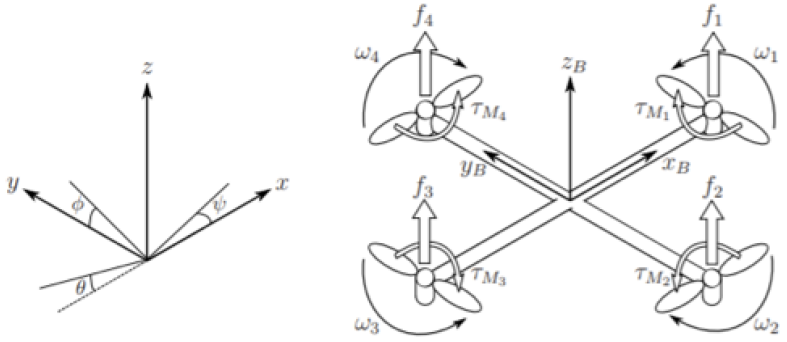


Figure 2.1 Quadrotor Dynamics and Frames (Christoph Aoun & Shammas, 2019).

Specifically, a transformation matrix is given by:

$$R_I^B(\phi,\theta,\psi) = \begin{bmatrix} c_\phi c_\psi - c_\theta s_\phi s_\psi & -c_\psi s_\phi - c_\phi c_\theta s_\psi & s_\theta s_\psi \\ c_\theta c_\psi s_\phi + c_\phi s_\psi & c_\phi c_\theta c_\psi - s_\phi s_\psi & -c_\psi s_\theta \\ s_\phi s_\theta & c_\phi s_\theta & c_\theta \end{bmatrix} \tag{2.1}$$

where c represents a cos function and s represents a sin function as well as the Euler angles are represented by ϕ, θ, ψ which are roll, pitch and yaw, respectively.

The forces that govern the motion of the quadrotor are generated by the rotors of the quadcopter. Each rotor has two components of force which are created when the rotor blades swirl. These forces are lateral and longitudinal known as the drag and thrust forces of the rotor blades.

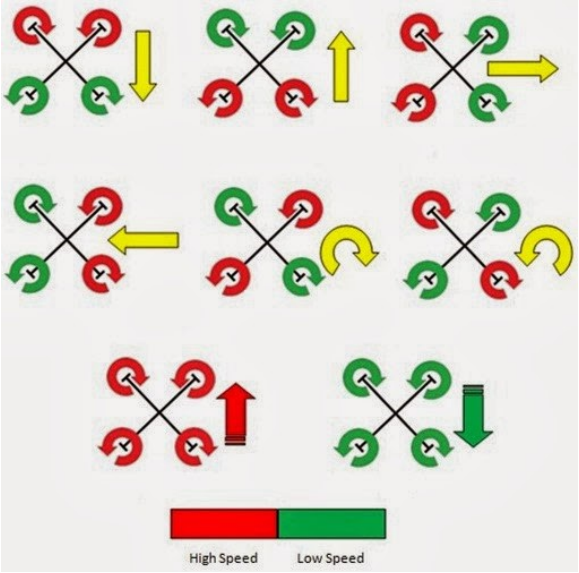


Figure 2.2 Quadrotor Motion Variations (Christoph Aoun & Shammas, 2020).

As shown in Figure (2.2), the motion of the quadrotor varies with the differnt combinations by the relative variations of the rotor speeds. To formulate this into actual dynamics, the following equations are used.

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} k(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\ k(\omega_2^2 - \omega_4^2) \\ k(\omega_3^2 - \omega_1^2) \\ b_t(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{pmatrix}. \tag{2.2}$$

where ω represent the rotor speeds, k is the thrust coefficient, b_t is the drag coefficient. u_1, is the total thrust performed by all rotors, while u_2 and u_3 are the difference in thrusts which will be used to calucalte the roll and pitch moments respectively, while u_4 is the yaw moment (Aoun, 2019).

Using Equation (2.1) along with Newton's second law of motion, the following quadrotor equations of motion in the inertial frame are established.

$$\begin{aligned}
m\ddot{x} &= k(\sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta)u_1,\\
m\ddot{y} &= k(\cos\phi\sin\theta\sin\psi - \cos\psi\sin\phi)u_1,\\
m\ddot{z} &= k(\cos\phi\cos\theta)u_1 - mg,\\
I_x\ddot{\phi} &= (I_y - I_z)\dot{\theta}\dot{\psi} + lu_2,\\
I_y\ddot{\theta} &= (I_z - I_x)\dot{\phi}\dot{\psi} + lu_3,\\
I_z\ddot{\psi} &= (I_x - I_y)\dot{\phi}\dot{\theta} + u_4.
\end{aligned} \tag{2.3}$$

where x, y, z are the positions in the inertial frame and I_x, I_y, and I_z are the moments of inertia of the respective axes. m is the mass of the quadrotor, $g = 9.81m/s^2$ is the acceleration due to gravity, and l is length of the arm. Considering Equations (2.3) and (2.2), the following equation of motion result (Aoun, 2019):

$$\begin{aligned}
m\ddot{x} &= k(\sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta)(\omega_1^2+\omega_2^2+\omega_3^2+\omega_4^2),\\
m\ddot{y} &= k(\cos\phi\sin\theta\sin\psi - \cos\psi\sin\phi)(\omega_1^2+\omega_2^2+\omega_3^2+\omega_4^2),\\
m\ddot{z} &= k(\cos\phi\cos\theta)(\omega_1^2+\omega_2^2+\omega_3^2+\omega_4^2) - mg,\\
I_x\ddot{\phi} &= (I_y - I_z)\dot{\theta}\dot{\psi} + lk(\omega_4^2 - \omega_2^2),\\
I_y\ddot{\theta} &= (I_z - I_x)\dot{\phi}\dot{\psi} + lk(\omega_3^2 - \omega_1^2),\\
I_z\ddot{\psi} &= (I_x - I_y)\dot{\phi}\dot{\theta} + b(\omega_2^2 - \omega_1^2 + \omega_4^2 - \omega_3^2).
\end{aligned} \tag{2.4}$$

2.2. UAV Operational Challenges

There are three generalized categories of problems that might arise in UAVs. These might be categorized into system problems, external problems, and program issues. In particular, the categories are known as Failures, Errors, and Threats. These might have

various effects on the system whether it changes the system dynamics, or the sensor data, communication, or even might just be considered as an additional input to the system.

2.2.1. Failures

Failures in the system are generally associated with major changes occurring in the physical attributes of the system. For a quadrotor UAV in particular, there are several failures that might occur.

2.2.1.1. Propeller and Motor Failure

One example would be a propeller breakage or motor failure for one or more sides of the quadrotor. This would cause a total exclusion of one or more of the ω terms from the equations of motion in Equation (2.4) depending on which rotor or motor is affected or destroyed.

Another example of this failure is reduced efficiency of the motor or bent propellers which might also reduce efficiency or change the thrust coefficient k. This would result in having an efficiency coefficient ϵ_ω multiplied to each ω^2 in Equations (2.4).

One of the most common limitations of the motor is saturation. This could be very eccentrically formulated into the system where there is a range of speed that the rotor can sustain such as $\omega_{min} < \omega < \omega_{max}$. This could be formulated as an external input which increases proportionally with the motor command to keep it limited within the maximum limit of the system (Mueller & D'Andrea, 2014).

2.2.1.2. Arm Failure

A change in the arms first and foremost would result in the change of the constant l in the Equations (2.4). It might also result in an array of variations. Having an arm break would not only result in the elimination of the ω component of the respective side, but also would affect the whole dynamics of the system including the moments of inertia I_x, I_y, and I_z along with the mass m. As a consequence, the center of gravity will change.

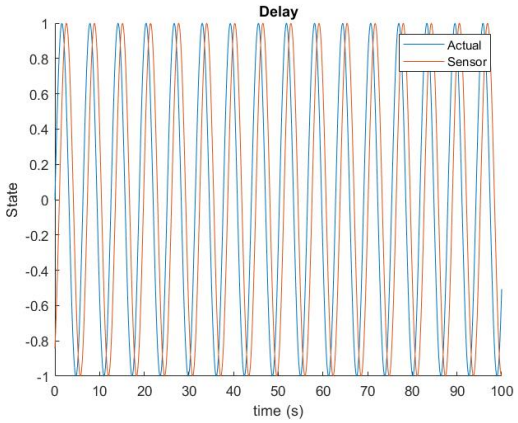
If the arm is twisted several components are changed. The thrust contributed by each rotor will result in two main forces on the body frame which include a lateral and a

longitudinal thrust force. This would result in the augmentation of the equations of motion. Moreover, several dynamic constants would be augmented to fit the contribution and the center of gravity and mass shifting in the quadrotor. As per the severity of the bend, it might also cause certain changes to the dynamic constants of Equations (2.4) as it might move the center of gravity causing changes in the moments of inertia I_x, I_y, and I_z (M. Rizon, 2020).

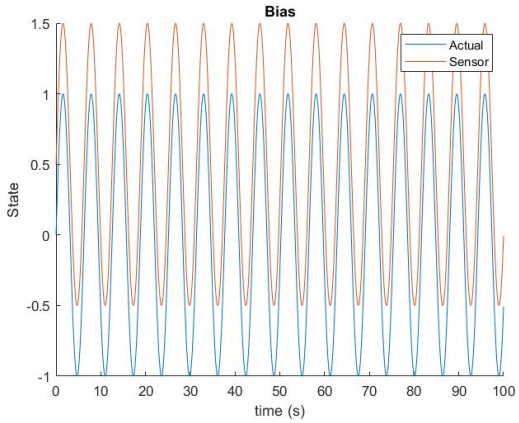
2.2.2. Sensor Errors

Sensor errors mostly pertain to more sensory inputs and virtual system failures. These might affect the general observability of the system dynamics. In the context of a state-space format, the observation matrix C will be modified. Sensor errors are generally established in different levels that include ceasing of functionality, biases, drift, noise, and delays.

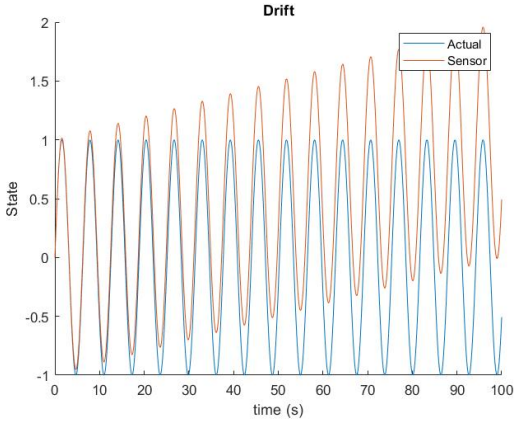
Ceasing of sensor functionality means that the sensor has totally failed and would result in zero observability of the state that was provided by that sensor. Bias is described by an offset in sensory input. Noise is usually described as small additions to the actual condition of the sensed state that might average out in the long run to a zero mean error. These could be found in a range of amplitudes and frequencies. Delays, on the other hand, might come from the rate of sensing that might occur and the variations in signal timing between sensory inputs (Vignesh Kumar Chandhrasekaran, 2010). Figure (2.3) shows the various sensor errors that could occur.



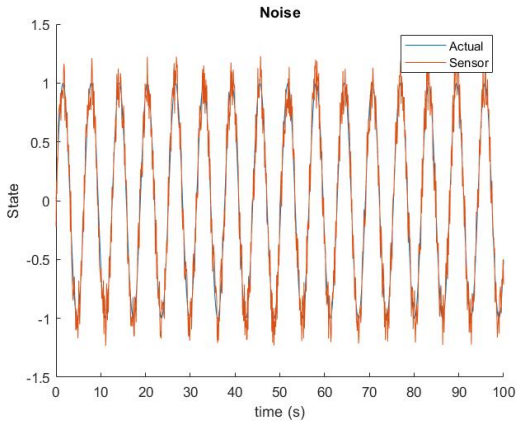
(a) Delay Example(1 second delay)



(b) Bias Example (+0.5 offset)



(c) Drift Example (+0.01/s slope drift)



(d) Noise Example (max amplitude ± 0.25)

Figure 2.3 Various Examples of Errors.

The main sensors used in the quadrotor are the GPS and the Inertial Measurement Unit (IMU) which includes both the gyroscope and the accelerometer sensors. These sensors measure linear and angular accelerations in body frames (Olson & Atkins, 2013). Figure (2.4) describes the various sensors included in the IMU.

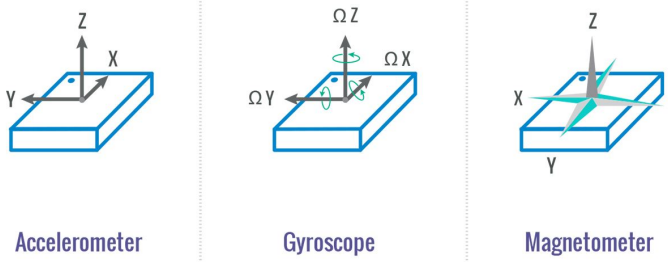


Figure 2.4 IMU input.

Another type of commonly used sensor is the height sensor which mostly uses infrared or ultrasound to determine the height of the object from the ground. Moreover, in order to determine the rest of the coordinates and the heading of the system other sensors are used to determine the lateral states. This includes vision systems which determine the lateral speed of the system along with its relative location to its original position based on variations in pixels. Another form of determining location is a Global Position System (GPS) which provides position and velocity observability. Finally, another form of location system is using a magnetometer to find out the position and the heading of the system (Cuenca, 2021).

2.2.3. External Threats

One of the most difficult disturbances to determine or estimate are external threats. These could range from simple forces, to change in loads, to wind gusts and many more.

2.2.3.1. Wind Gusts

Wind gusts are a common threats that UAV's experience very often. One way to model a wind gust is considering it simply as a form of force that is proportionate to the angle of impact and relative to the body frame. The quadrotor can be simplified into a cylindrical representation whilst having the the wind being represented as a distributed force (Solovyev Viktor V. & A., 2015). This wind gust force effect is portrayed in Figure (2.5).

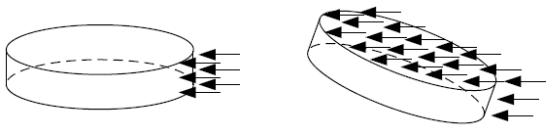


Figure 2.5 Wind Force Representation (Solovyev Viktor V. & A., 2015).

This formulation could be portrayed using the following equations:

$$F_{wx} = S_{ex} A_w (V_{cz})^2 \cos\psi_w \tag{2.5}$$

$$S_{ex} = \beta\pi r^2 \sin\theta + \alpha\pi r h \cos\theta \tag{2.6}$$

$$F_{wy} = S_{ey} A_w (V_{cz})^2 \sin\psi_w \tag{2.7}$$

$$S_{ey} = \beta\pi r^2 \sin\phi + \alpha\pi r h \cos\phi \tag{2.8}$$

where ψ_w is the relative the yaw of the quad, while α and β are fill factors of the cylinder area which depends on the quadrotor design. Moreover, h and r represent the cylinder height and radius respectively. F_{wx} and F_{wy} are the forces to the body frame along the lateral axes. V_{cz} is the relative wind velocity compared to the quadrotor speed. $A_w = 0.61$ is the rate of conversion from wind velocity (m/s^2) to pressure (N/m^2) and S_{ex} and S_{ey} are the effective area of impact of the wind pressure.

An alternative way for modeling wind speed to the system is analyzing how the quadrotor produces thrust through an action-reaction process by pushing wind through its rotors and causing a thrust force. However, when wind is applied to Equation (2.4), it may cause a variation in the air direction and speed which might disrupt the expected reactionary thrust outcome from the rotor speed (Solovyev Viktor V. & A., 2015).

In an outdoor domain, the lateral airflow acting on the propeller can be depicted using the Figure (2.6) (Ding & Wang, n.d.):

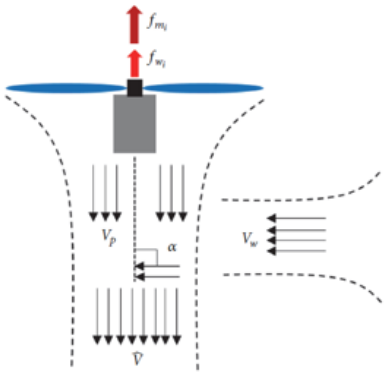


Figure 2.6 Lateral Wind Effect.

The resultant force from each quadrotor is modeled as follows:

$$f_T = 2\rho A_p \hat{V} V_p \tag{2.9}$$

$$\hat{V} = \sqrt{(V_w \cos\alpha_w + V_p)^2 + (V_w \sin\alpha_w)^2} \tag{2.10}$$

$$f_\omega = f_T - k_t\omega^2 \tag{2.11}$$

where V_w and α_w are the wind speed and impact angle with the qaudrotor vertical axis, V_p

is the induced wind speed from the propellers, A_p is the propeller area, and ρ is the air density. Moreover, the aerodynamic drag is defined as $m_{drag} = \rho A_w V_w^2/2$

$$\begin{pmatrix} m_{\omega_\phi} \\ m_{\omega_\theta} \\ m_{\omega_\psi} \end{pmatrix} = \begin{pmatrix} (f_{\omega_4} - f_{\omega_2})l \\ (f_{\omega_3} - f_{\omega_1})l \\ \sum_{i=1}^{4} m_{drag_i} \end{pmatrix}. \tag{2.12}$$

This results in a more complete set of equations of motion (Ding & Wang, n.d.):

$$\begin{aligned}
m\ddot{x} &= k(\sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta)(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) + \sum_{i=1}^{4} f_{\omega_i}, \\
m\ddot{y} &= k(\cos\phi\sin\theta\sin\psi - \cos\psi\sin\phi)(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) + \sum_{i=1}^{4} f_{\omega_i}, \\
m\ddot{z} &= k(\cos\phi\cos\theta)(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) - mg + \sum_{i=1}^{4} f_{\omega_i}, \\
I_x\ddot{\phi} &= (I_y - I_z)\dot{\theta}\dot{\psi} + lk(\omega_4^2 - \omega_2^2) + m_{\omega_\phi}, \\
I_y\ddot{\theta} &= (I_z - I_x)\dot{\phi}\dot{\psi} + lk(\omega_3^2 - \omega_1^2) + m_{\omega_\theta}, \\
I_z\ddot{\psi} &= (I_x - I_y)\dot{\phi}\dot{\theta} + b(\omega_2^2 - \omega_1^2 + \omega_4^2 - \omega_3^2) + m_{\omega_\psi}.
\end{aligned} \tag{2.13}$$

2.2.3.2. Payload

Payloads are an addition to the m mass of the system. This, however, can change depending on the quality of the payload. Some payloads include fluids and varying payloads and even swinging payloads, which affect the moment of inertia. A variation of load could be formulated as a peripheral force added or subtracted to the moments and forces that affect the dynamics of the system (S. Sadr & Zarafshan, 2014). One of the swinging payload examples is portrayed in Figure (2.7).

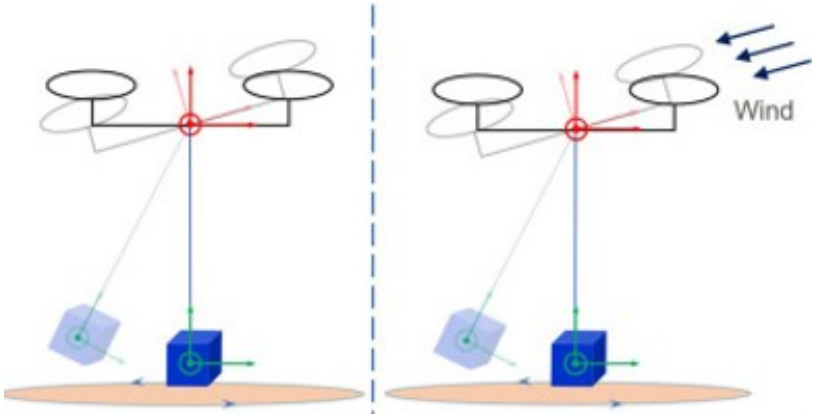


Figure 2.7 Swinging Payload.

2.2.3.3. Cyber Attacks

One threat that has been rising during system operations is cyber-attacks, particularly zero-dynamic cyber attacks. This would entail changes to the dynamic equations of the system. This includes the motor inputs as well as the observability provided from certain sensors (Hamidreza Jafarnejadsani & Voulgaris, 2018). Another form of cyber-attack is hijacking where a malicious individual takes control of the drone. Moreover, a different aspect could be obscuring vision detection systems and replacing them with unrealistic data which inevitably ends in a crash.

2.3. Control Systems

Throughout time, there have been proposed several types of controllers that range from linear to model-based to learning algorithms. In this chapter, a review of different types of controllers along with their pros and cons are provided.

2.3.1. PID Controller

Proportional-Integrator-Differential (PID) controllers are divided into three gains each of which is used to fix certain attributes of the tracking system response. Proportional deals primarily with the tracking, Differential deals with the transition phase to prevent strong spikes and peaks, and finally the integrator is used to take out the offset which might not be compensated by a proportional gain. It is considered one of the most stable controllers and

includes a certain degree of robustness. It is known to be slightly difficult to tune, but also there are differences and variations which could help such as an online PID tuner (Qingsong Jiao, 2018).

2.3.2. LQR/LQG

Optimal Control is usually implemented to a Linear Quadratic Regulator (LQR) or Linear Quadratic Gaussian (LQG), which involves an estimator of the states along with an LQR. The main concept within Optimal Control is minimizing a generalized cost function such as:

$$J(e, u_o) = \int_0^\infty e^T Q e + u_o^T R u_o dt \tag{2.14}$$

where $Q \geq 0$ and $R > 0$ are the weight matrices of the state error (e) and input (u_o) respectively. This cost function is expected to be minimized along with a linearized state space model, which is $\dot{x} = Ax + Bu_o$. In order to optimize the solution (Araar & Aouf, 2014), the Hamiltonian is used:

$$H = e^T Q e + u_o^T R u_o + \Lambda^T (Ae + Bu_o) \tag{2.15}$$

continuous time Algebraic Ricatti Equation (ARE) (Brian D.O. Anderson, 1989).
where Λ is a Lagrange multiplier. Using the Euler-Lagrange method results in the

$$A^T S_\infty + S_\infty A - S_\infty B R^{-1} B^T S_\infty + Q = 0 \tag{2.16}$$

feedback input to the system, which is as follows:
where S_∞ is the solution of the ARE. Solving this equation would result in determining the

$$u_o = -R^{-1} B^T S_\infty e \tag{2.17}$$

This algorithm is optimal with a clear-cut analytical solution, but it retains problems of adaptability and robustness.

2.3.3. H_∞

This type of controller relies on frequency domain optimization. It is mainly concerned with robustness and sub-optimality when creating a feedback controller. The problem is generally formulated in a way that the system is considered with two inputs u_o as the

controlled input and w as the exogenous input. This results in two outputs z representing the error signal and y which are the measurable variables. This formulation is placed in the Lagrangian domain and results in the following formulation (Harm BartMarinus, 2010):

$$\begin{pmatrix} \hat{z} \\ \hat{y} \end{pmatrix} = \begin{pmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{pmatrix} \begin{pmatrix} \hat{w} \\ \hat{u} \end{pmatrix} \tag{2.18}$$

However, placing $\hat{u} = K(s)\hat{y}$ as well as $\hat{z} = F_l(s)\hat{w}$ are determined by the H_∞ norm which is expressed as follows:

$$H_\infty = \begin{pmatrix} A & \gamma^{-2}BB^T \\ -CC^T & -A^T \end{pmatrix} \tag{2.19}$$

where $\gamma > ||G||_\infty$. This results into two Hamiltonian equations and two separate Ricatti equations that provide a solution and its solution to both $K(s)$ and $F_l(s)$ (Araar & Aouf, 2014). This system lacks adaptability and does not satisfy total optimality.

2.3.4. Sliding Mode Controller (SMC)

SMC is usually considered an adaptive case where it is used in nonlinear control cases. It is usually an attempt at adjusting several gains in terms of sliding them slightly in order to stabilize the system or minimize the error according to a certain nonlinear requirements. This, however, lacks robustness and optimality as this is simply relying on the dynamical equations without a generalized optimality at hand. The main concept of the system is based off Lyapunov Stability Theory which is often used as a complementary adaptive mechanism to manipulate more basic forms of controllers such as PID (Abdel-Razzak Merheb & Bateman, 2014).

2.3.5. Model Reference Adaptive Control (MRAC)

In general, adaptive control methods deal mostly with uncertainties. That is why they do not require a priori knowledge of the system or the plant at hand. It relies on the concept of parameter identification. These identifications can be usually done through recursive least square or gradient descent.

There are various types of adaptive controllers including MRAC where the system parameters are estimated by comparing its outputs with a model reference system based on ideal terms. This helps identify uncertainties which might not have been taken into consideration when considering the ideal case or condition of the system at hand. A correcting input is used in order to help the system follow and track the required trajectory based on the reference model system at hand.

There are two forms of MRAC which are Direct and Indirect MRAC. Direct MRAC use the estimated parameters directly for the adaptive controller. In indirect MRAC, the system parameters are estimated and used to calculate required controller parameters such as the various matrices of the linearized system. MRAC usually includes a controller that might be as simple as a proportional controller that has adaptable gains (Shekhar & Sharma, 2018). The various forms of MRAC are shown in Figure (2.8).

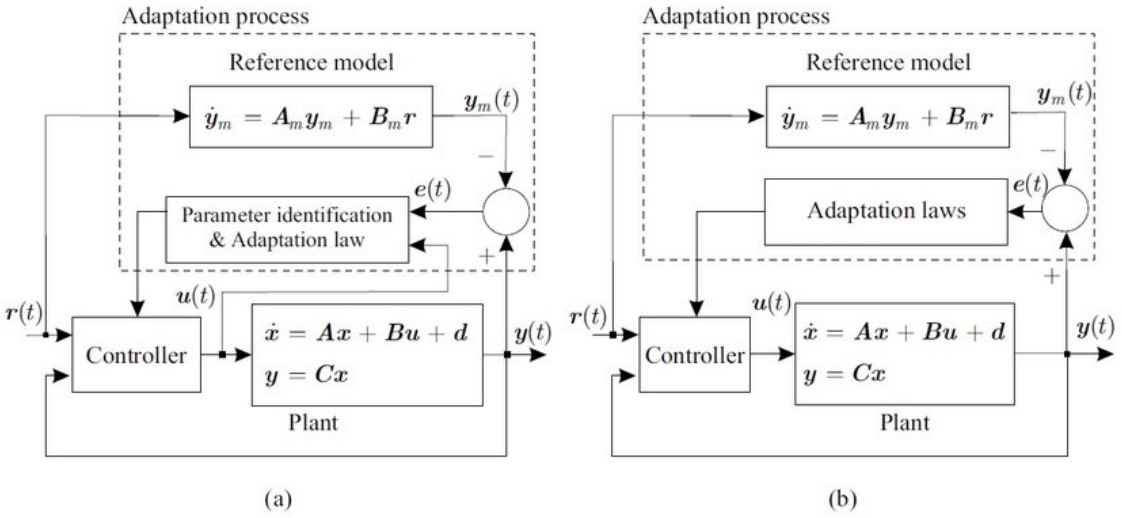


Figure 2.8 Indirect Vs Direct MRAC (Shekhar & Sharma, 2018).

2.3.6. Nonlinear Dynamic Inversion (NLDI)

When dealing with a nonlinear system it is usually preferred to transform it into a linearized system. This helps to turn the dynamical equations into an invertible form.

Consider the following nonlinear system:

$$\dot{x} = f(x) + g(x)u_o, \tag{2.20}$$

$$\begin{pmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_n \end{pmatrix} = \begin{pmatrix} x_2 \\ \vdots \\ b(x) \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ a(x) \end{pmatrix} u_o \tag{2.21}$$

As the system is linearized, similar to a Feedback Linearization system, there is a virtual controller that can be established $v = b(x) + a(x)u_o \Leftrightarrow u_o = a^{-1}(x)(v - b(x))$ where v can be used to control the system. This however, is very susceptible to errors and uncertainties as well as external threats which might cause it to go unstable. Thus, this is neither optimal nor robust nor adaptive (Qing Lin, 2016).

2.3.7. Fuzzy Logic-Based Controller

The main idea of this controller revolves around the concepts of fuzzy rules in pilot interactions. It is considered quite robust and adaptable, but lacks in initial stability and accuracy in state prediction and compensation. It has capabilities of fault tolerance, but might suffer when the scope of the threat is out of the logic base (Andrew Zulu, 2014).

2.3.8. Artificial Neural Network(ANN)

This form of controller is an umbrella term for variable forms of controllers. Neural Networks are a combination of Neural equations cascaded into weighted sums of other combinations. These weights are mostly updated through learning or optimization algorithms, however complex the algorithms might be. They fall into generalized categories which will be explained in later sections.

ANN has been hailed as the new innovative domain of feedback controllers since its learning algorithms could incorporate several aspects which can include robustness, adaptability, optimality, fault tolerance, accuracy, and stability. However, it is important to note that proving ANN stability in an analytical perspective is challenging and it might be quite difficult to predict how the system would react based on various situations and

conditions. As such, ANN is still a wide field of research and the controller is as efficient and as strong as its learning algorithm (Lebao Li, 2015).

2.3.9. Comparison Table

In order to compare the various stated algorithm along with many others, Table (2.1) outlines the various attributes of the array of control algorithms proposed for quadrotor controller.

Table 2.1

Comparison of Quadrotor control Algorithms.

Characteristics												
Name	R	A	O	I	T	F	P	S	D	U	M	N
PID	1	0	0	0	1	1	1	2	0	0	2	2
LQR	0	2	1	0	1	1	0	1	1	0	1	1
LQG	0	2	2	0	1	1	0	0	2	0	1	0
H_∞	2	1	2	0	2	0	1	0	2	2	0	0
SMC	2	2	1	0	2	2	2	1	2	1	0	0
FL-NLDI	1	1	0	0	2	2	2	1	0	1	0	1
BC	0	2	0	0	2	2	1	0	2	1	0	0
MPC	2	1	2	0	2	1	1	0	1	1	0	1
NST	2	2	0	0	2	2	1	0	1	1	0	1
FLB	2	1	1	2	1	1	1	1	1	0	1	0
ANN	2	1	2	2	1	1	1	0	1	1	0	0
RL	2	1	2	2	1	1	1	0	1	1	0	1
IL	1	2	1	2	1	1	2	1	1	1	0	1
MB	1	1	1	2	1	1	1	1	2	1	0	1
BEL	2	2	2	2	1	1	1	0	2	2	0	1
I-PID	1	0	0	2	1	1	1	1	0	0	0	1
L_1	0	2	2	0	1	2	2	0	1	0	0	0
GA	1	2	2	2	1	1	1	0	1	2	0	0
H_1	2	1	2	0	2	0	1	0	1	1	0	0

Table (2.1) describes all the different controllers and their respective advantages and disadvantages. The grading is done as follows (Lebao Li, 2015):
$0 - Low, 1 - Average, 2 - High$. The Abbreviations of the table are shown as follows; R-robust; A-adaptive; O-optimal; I-intelligent; T-tracking ability; F-fast

convergence/response; P-precision; S-simplicity; D-disturbance rejection; U-unmodeled parameter handling; M-manual tuning; N-(signal) noise (Andrew Zulu, 2014).

The names are as follows (Lebao Li, 2015): PID -Poportional Integral Derivative; LQR-Linear Quadratic Regulator; LQG- Linear Quadratic Gaussian; SMC- Sliding Mode Controller; FL-Feedback Linearizer; NLDI- Nonlinear Dynamics Inversion; MPC-Model Predictive Controller; NST- Nested Saturation Technique; BC- Backstepping Controller; FLB- Fuzzy Logic Based; ANN-Artificial Neural Network; RL- Reinforcement Learning; IL- Iterative Learning; MB-Memory Based; BEL-Brain Emotional Learning; I-PID-Intelligent PID; GA-Genetic Algorithm (Andrew Zulu, 2014).

2.4. Adverse Estimation (Li, 2016)

One of the most difficult aspects in any control system is predicting, detecting, or quantifying the adverse inputs or disturbances that arise or affect the general dynamics of the system. These threats can be identified in many ways which will be discussed briefly in this section.

In a Frequency Domain Disturbance Observer (IDO), formulation the disturbance is is considered a lumped disturbance, where the generalized diagram of this formulation is shown in Figure (2.9).

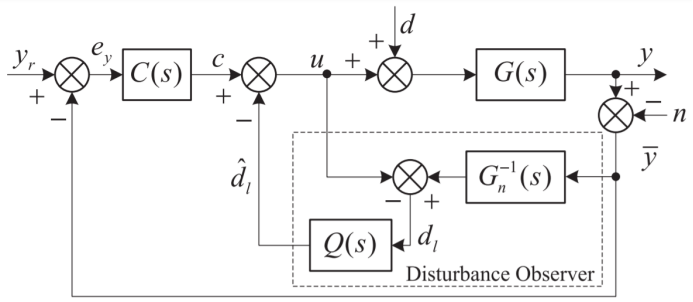


Figure 2.9 Disturbance Observer Based Controller.

To be able to estimate the disturbance in the frequency domain, the lumped disturbance is considered to be made up of three elements based on Figure (2.9).

$$d_l(s) = [G(s)^{-1} - G_n(s)^{-1}]y(s) + d(s) - G_n(s)n(s) \tag{2.22}$$

where $G(s)$ is the physical system frequency domain model while $G_n(s)$ is the nominal model. $d(s)$ is the actual disturbance and $n(s)$ is the measurement noise. Thus, d_l includes all the system noises and disturbances. It is generally funneled through a filter $Q(s)$, which ends up estimates the lumped force using the following:

$$\hat{d}_l(s) = G_{ud}u(s) + G_{yd}(s)\bar{y}(s) \tag{2.23}$$

where the G_{ud} and G_{yd} are the components of the filter $Q(s)$ pertaining to the input and sensor data components respectively. Moreover, it is important to note that $Q(s)$ is usually a low pass filter since it helps channel out the noise which is of high frequency and retains a certain estimation of the disturbance which is of low to medium frequencies.

Furthermore, an Extended State Observer (ESO) estimator proposal focuses on transforming a nonlinear system into a linearized one similar to that of a Feedback linearizer or NLDI. In this estimation, $b(x)$ also includes the disturbance dynamics, thus an extra state is chosen.

$$\begin{aligned} x_{n+1} &= b(x, d) \\ \dot{x}_{n+1} &= h(t) \end{aligned} \tag{2.24}$$

It is also important to note the configuration of the output formulation which is established as follows:

$$y^{(n)} = f(y(t), ..., y^{(n-1),t}, u(t)) \tag{2.25}$$

where y^l is the l^{th} derivative of the output. The ESO concept is designed and formulate to estimate all the states along with the lumped disturbance term which is found in $b(x, d)$ in Equation (2.24).

$$\begin{aligned} \dot{\hat{x}}_i &= \hat{x}_{i+1} + \beta_i(y - \hat{y}), i = 1, ..., n \\ \dot{\hat{x}}_{n+1} &= \beta_{n+1}(y - \hat{y}) \end{aligned} \tag{2.26}$$

where β is a gain used as a correcting factor. This helps in determining the external disturbances, but up to a relative degree depending on the proper estimation of the dynamics and how they are reflected in cascaded linearization.

Since the 1960's Unknown Input Observer (UIO) has been proposed and developed by NASA for various types of projects. This is considered one of the first milestones in disturbance estimation using filtering methods. The linearized system can be shown as follows:

$$\begin{aligned} \dot{x} &= Ax + B_u u_o + B_d d \\ y &= Cx. \end{aligned} \tag{2.27}$$

where B_u and B_d are input matrices for control and disturbance respectively. However, it considers a peripheral exogenous system with its exogenous state to be determined as follows:

$$\begin{aligned} \dot{\zeta} &= W\zeta \\ d &= V\zeta. \end{aligned} \tag{2.28}$$

where W is the exogenous feedback dynamic matrix V is the mapping matrix of exogenous state to disturbance input. In this system, both states and disturbances are estimated using a Kalman filter concept. This results in an estimation system as follows.

$$\begin{aligned} \dot{\hat{x}} &= A\hat{x} + B_u u_o + L_x(y - \hat{y}) + B_d\hat{d} \\ \hat{y} &= C\hat{x}. \\ \dot{\hat{\zeta}} &= W\hat{\zeta} + L_d(y - \hat{y}) \\ \hat{d} &= V\hat{\zeta}. \end{aligned} \tag{2.29}$$

where L_x and L_d are the observer gains for the state and the disturbance estimator respectively such that the estimated system is still stable. This allows the system to determine a wide range of disturbance estimation, but this suffers from a limited predictability to uncertainties and noise as it might be considered under the notion of lumped disturbance.

In addition, the Uncertainty and Disturbance Estimator (UDE) formulation considers mainly a lump sum of uncertainties and disturbances and noise assumptions. In general, the linearized system can be shown as follows:

$$\dot{x} = Ax + B_u u_o + \Delta Ax + \Delta B_u u_o + d \tag{2.30}$$

$$d_l = \Delta Ax + \Delta B_u u_o + d \tag{2.31}$$

$$d_l = \dot{x} - Ax - B_u u_o \tag{2.32}$$

Since $\dot{x}$ is not observable, the proposed algorithm surpasses this problem by approximating its estimate as:

$$\hat{d}_l = d_l \star q \tag{2.33}$$

where $\star$ is a convolution operator with q, which is the impulse response of a filter $Q(s)$. This proposed filter helps detect uncertainties more prominently, but requires extensive considerations in filter design which could be difficult and might lead to unrealistic results.

On the other hand, in an Equivalent Disturbance (EID) system, $B_u = B_d$; this consideration makes slight changes and eases certain restrictions of wide range considerations. In this, the filter $Q(s)$ is formulated as follows:

$$Q(s) = \frac{1}{T_q s + 1} \tag{2.34}$$

where T_q is the time constant used for filtration. However, using this filter results in the disturbance estimation formulation which is shown as follows:

$$\dot{\hat{d}} = \frac{1}{T_q}(B_u^T B_u)^{-1} B_u^T L(y - \hat{y}) \tag{2.35}$$

where L is a error gain between estimated sensor output and actual sensor output. This has several advantages and disadvantages depending on what frequency of disturbances the user is trying to identify.

There are two main categories of nonlinear systems which are similar to the linear systems. However, when dealing with a Nonlinear system a Nonlinear Disturbance

Observer (NDOB) is used which is generally under the following formulation:

$$\dot{x} = Ax + g_1(x)u_o + g_2(x)d$$
$$y = h(x). \tag{2.36}$$

where $g_1(x)$ is the nonlinear dynamics of the system or controller input, $g_2(x)$ is the nonlinear external disturbance dynamics, while u_o and d are the controller and disturbance inputs respectively, and $h(x)$ is the nonlinear sensor input.

One of the nonlinear methods for disturbance estimation is called Unknown Constant Disturbance (UCD). This nonlinear formulation is used to estimation unknown slow time varying disturbances. It is generally implemented as follows:

$$\dot{z} = -l(x)g_2(x)z - l(x)[g_2(x)p(x) + f(x)g_1(x)u_o]$$
$$\hat{d} = z = p(x). \tag{2.37}$$

where z is an internal pseudo-state while $l(x)$ and $p(x)$ are nonlinear function gains which are designed such that:

$$l(x) = \frac{\delta p(x)}{\delta x} \tag{2.38}$$

Moreover, they are both chosen such that $\dot{e}_d = -l(x)g_2(x)e_d$ is asymptotically stable regardless of x while $e_d = d - \hat{d}$ is the disturbance error.

Moreover, there exists a nonlinear algorithm with similar formulation as the UIO. This is called Geberal Exogenous Disturbance (GED). This uses a cascade of three interrelated equations of nonlinear formulation, which are shown as follows:

$$\dot{z} = [W - l(x)g_2(x)V]z + Wp(x) - l(x)[g_2(x)Vp(x) + f(x) + g_1(x)u_o]$$
$$\hat{\zeta} = z + p(x)$$
$$\hat{d} = V\hat{\zeta} \tag{2.39}$$

where W, V are similar to that of UIO, while $l(x)$ is similar to that of the UCD In order to calculate the nonlinear gain $l(x)$ the gains must be chosen such that
$\dot{e}_\zeta = [W - l(x)g_2(x)V]e_\zeta$ is globally exponentially stable regardless of x. This is a

preferred algorithm since it includes observable aspects of the system as well as an exogenous and internal state which can provide further filtering of noise and identification of uncertainties.

There are various other observers, which include Generalized Proportional Integral Observer (GPIO), which builds upon the UIO with an integral component for settling error or offsets. This can help with the final fine tuning of the system which usually has a constant error.

Moreover, there are many other nonlinear observers which include Extended High-Gain State Observer (EHGSO), which uses a complex cascade of various linearized and nonlinear assumptions of the system to evaluate both state and disturbance. However, this system lacks in robustness and ability to reject noise and uncertainties especially with rapid disturbance changes even as simple as a large step input.

2.5. Artificial Intelligence

One of the most popular topics of the current age is artificial intelligence, that has risen to fruition with the high speed computing and increase capabilities of retaining and processing memory data. The primary start to entering this field is understanding the difference between different types of learning algorithms along with their pros and cons. In general, artificial intelligence is denoted to John McCarthy who is considered the godfather of AI. It is defined as the branch of computer science that deals with the simulation of intelligent behavior in computers which give capabilities to imitate human behaviour. It enables computer systems to perform tasks that normally require human intelligence such as complex visual perception, speech recognition, decision-making, and translation between languages (A.I., 2021).

2.5.1. Machine Learning (ML)

Machine Learning is a subset of Artificial Intelligence which took the concept training to a whole new level. Machine learning is based on the concept that a system is trained in various manners, but develops decision making capabilities where it can tackle issues it did

not encounter during training up to a certain extent. This basically revolutionized the concept of Machine Learning as a new form of controls which required prior training, but doesn't have to go through every scenario. ML has several categories where each entails different formulations, but they can be sectioned into four main processes: Supervised, Semi-Supervised, Unsupervised, Reinforcement Learning (Middleton, 2021). Figure (2.10) clearly portrays the different Machine learning algorithms.

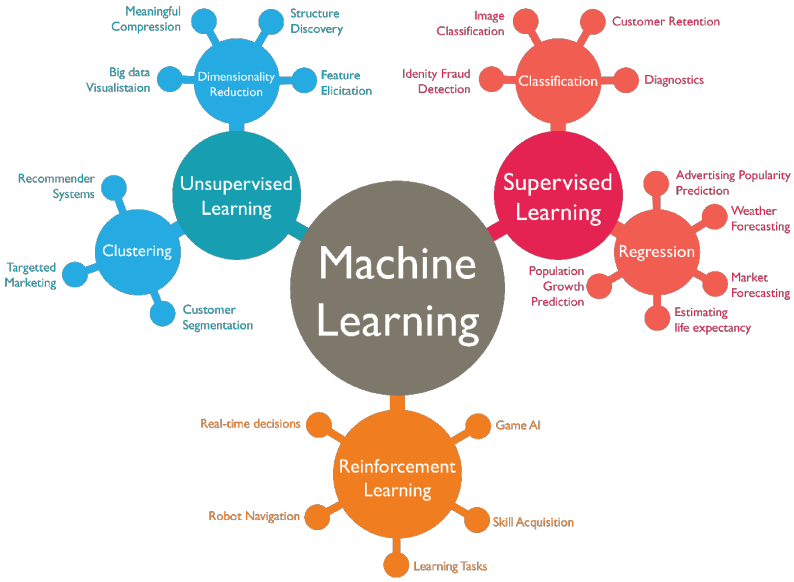


Figure 2.10 Machine Learning Categories (A.I., 2021).

Supervised learning is learning based on training material which are already labeled with a clear input-output. The system learns and modifies every time it adds to its attributes as it learns from the pre-compiled and played input-output combinations and fits them according to standard (Middleton, 2021).

Due to the vast numbers required to train a Supervised Learning algorithm, it is quite vexing to collect all the necessary labeling and data . Thus, Semi-supervised Learning includes both labeled and unlabeled data sets where a small quantity of input-output pairs are presented which enables the system to correct itself every time it trains with unlabeled material. Sometimes, Semi-supervised is placed under the Supervised Category due to their

similarities. However, there are times when data labeling is not present and training is required. This leads to the Unsupervised Learning (Middleton, 2021).

Unsupervised Learning is created when labeling is not present at all and it is up to the ML algorithm to determine patterns and similarities between data sets and create formats of clustering. It samples all data and adds them into clusters based on similar patterns on behaviours whether it is visual aspects or control input-output combinations. This method includes lots of uncertainties and might lead to unpredictable results. However, when data sets are not present, there is a fourth process that can take place which is known as Reinforcement Learning (Middleton, 2021).

2.5.2. Reinforcement Learning

Reinforcement Learning can be classified as model based or model free. In its definition, it is a form of ML where the system acquires random data and compiles input-output combinations on its own. In the meantime, it uses several episodes and several steps to modify its own attributes whilst acquiring the input-output labelling. This is done mainly using rewards and punishments, which are given at every step based off a generalized cost function and the output acquired from the input decision taken by the system. The concept is summarized in Figure (2.11) (Draguna Vrabie, 2012).

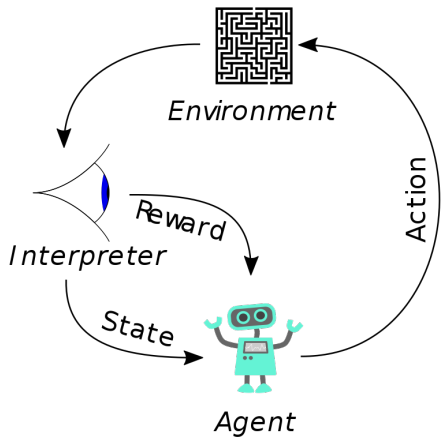


Figure 2.11 Reinforcement Learning Process (A.I., 2021).

Deep Learning, used in this thesis is the learning element of RL and is a subset of Machine Learning that uses deep neural networks. It also uses complex algorithms in order to optimize and learn. It requires less intervention from human, and is mostly determined by the algorithms and might not even require previously acquired data (A.I., 2021). To further explain where Deep Learning lies within the categories, Figure (2.12) portrays the subsections of AI.

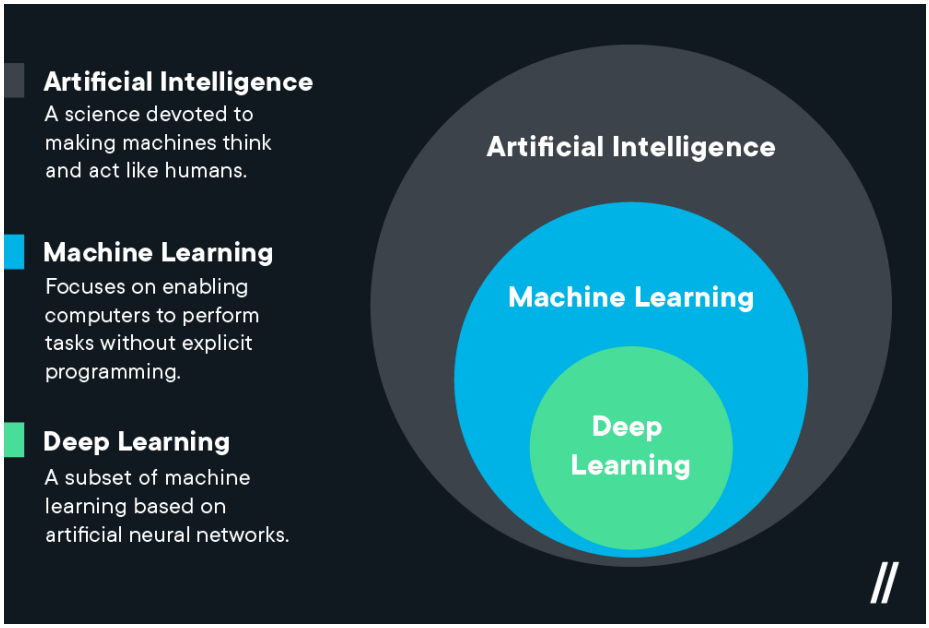


Figure 2.12 Learning Subsets (A.I., 2021).

The main neural networks that are used in DL fall under three main aspects which are Convolution Neural networks which are usually used for vision systems and are mainly binary in their output, Regressive Neural Networks that include weights and activation functions, and Recursive Neural Networks that are a form of ANN that retain a memory in its contents (Middleton, 2021).

Deep Learning algorithms are designed to mimic human brain activity and might use an array of various ”neurons” which are known as activation functions. These activation functions can be found in various formulations (Kumawat, 2019). The different types of NN are outlined in Table (2.2).

Table 2.2

Activation Functions.

Identity	Binary Step	Sigmoid	Tanh
x	$\begin{matrix} 0 & x < 0 \\ 1 & x \geq 0 \end{matrix}$	$\sigma(x) = \frac{1}{1+e^{-x}}$	$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
ReLU	GELU	Softplus	ELU
$\begin{matrix} 0 & x \leq 0 \\ x & x > 0 \end{matrix}$	$\frac{1}{2}x\left(1 + erf\left(\frac{x}{\sqrt{2}}\right)\right)$	$\ln\left(1 + e^x\right)$	$\begin{matrix} \alpha(e^x - 1) & x \leq 0 \\ x & x > 0 \end{matrix}$
SELU	Leaky ReLU	PReLU	SiLU
$\begin{matrix} \lambda\alpha(e^x - 1) & x \leq 0 \\ \lambda x & x > 0 \end{matrix}$	$\begin{matrix} 0.01x & x \leq 0 \\ x & x > 0 \end{matrix}$	$\begin{matrix} \alpha x & x \leq 0 \\ x & x > 0 \end{matrix}$	$\frac{x}{1+e^{-x}}$
Mish	Gaussian	Softmax	Maxout
$x \tanh\left(\ln\left(1 + e^x\right)\right)$	e^{-x^2}	$\frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}}$	$\max x_i$

Imitation Learning is a form of machine learning that is similar to Reinforcement Learning and Semi-supervised learning. The system tries to mimic a behaviour of another observed system by performing iterations. This provides the system both observability to its own input-output labeling as well as that of the other system. However, it is important to note that Imitation learning can only be as good as the system it is imitating. Generally optimization is done by sampling as well as changing attributes might sometimes be only done by using the data provided from the mimicked individual which might deprive the system from optimality (Lőrincz, 2019).

2.5.3. Generative Adversarial Neural Networks (GAN)

Having a system which is required to deal with an array of different disturbances and problems, it is quite difficult to create a library which can include all possible scenarios. As such, an adversarial system was proposed in 2014. This formulation is made in a way where

there are two main neural networks are working together known as the critic and the actor while a third one is working to disturb the system and create a new form of problems as the simulations progress. The actor tries to do the correct action, the adverse tries to disrupt, and the critic observes and assesses the main input-adverse-output combination (Goodfellow, 2014). This concept is clearly portrayed in Figure (2.13).

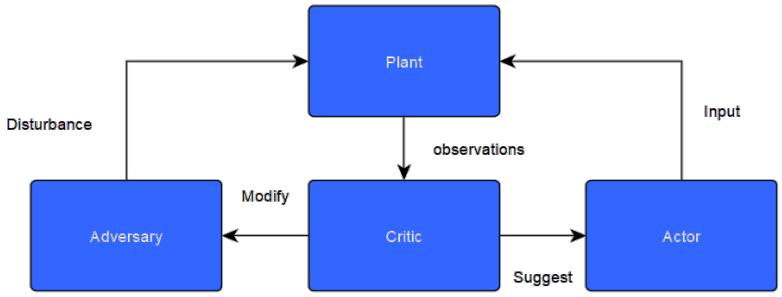


Figure 2.13 Adverserial System Diagram.

This adverse concept helps robustify the system and some could even be implemented as an online trainer or even during a training session for a pre-training protocol. This method can be used in all categories of Deep and Machine learning. Moreover, this system provides adaptability to the system according to the learning protocol under robust circumstances.

3. Methodology

In this chapter, an extensive discussion of the proposed machine learning architecture is provided. The system includes a design based on a Deep Learning Online Learning Algorithm with optimal, robust, and adaptive capabilities. The system could be used as a main controller or an augmentation to another baseline controller. The discussion for stability and initial conditions is primarily tackled as well as its capabilities of rejecting disturbances. It is based on various concepts from robust control, optimal control, and game theory. The architecture includes main NN modules for actor-critic-adversary.

As it will be noticed later in this chapter, this formulation requires observability of disturbance inputs. As such, a cascaded system of Kalman filters is proposed using the main concept from UIO. However, the main contribution to this is splitting the system into an inner and outer filter formulation that deals with position and attitude control separately while guaranteeing optimality both in estimation and control.

3.1. Optimal Control with Disturbance

Primarily, the general nonlinear and linear state space dynamics can be defined as follows:

$$\dot{x} = f(x) + g(x)u_o + k(x)d_a \tag{3.1}$$

$$\dot{x} = Ax + Bu_o + Kd_a \tag{3.2}$$

where x is the state, $f(x)$ or A is the state input dynamics, $g(x)$ or B is the input dynamics, $k(x)$ or K is the disturbance dynamics, u_o is the actor input,and d_a is the disturbance or adverse input to the system which is either a "lumped disturbance" or individual disturbance.

3.1.1. Hamiltonian

In order to incorporate the disturbance input into the optimal control cost function an H_∞ formulation is used resulting in the following cost function (Luis Carrillo, 2019):

$$J(e, u_o, d_a) = \int_0^\infty e^T Qe + u_o^T Ru_o - \gamma^2 ||d_a||^2 dt \tag{3.3}$$

where e is the tracking error; $Q \geq 0$ and $R > 0$ are the weighting matrices of the state error (e) and input (u_o) respectively. The term γ is considered the H_∞ gain, and $\gamma > 0$ is the weight of the adversarial input to the system (d_a).

$$H = e^T Q e + u_o^T R u_o - \gamma^2 ||d_a||^2 + \nabla \Lambda^T (f(e) + g(e)u_o + k(e)d_a) \tag{3.4}$$

where $\Lambda > 0$ is the Lagrange Multiplier, ∇ symbolizes a gradient operation. The formulation aims to minimize the Hamiltonian and find the optimal cases for the inputs (Luis Carrillo, 2019).

3.1.2. Hamilton Jacobi Isaacs (HJI) and Zero-Sum Game Theory

In order to optimize the Hamiltonian equation usually a Hamilton Jacobi Bellman Equation is established, but with the introduction of the adverse input a different formulation arises to the Hamilton-Jacobi-Isaacs equations.

This system is presumed to be a two-player game where the advancement of one is considered the loss of the other. This concept is mainly translated as a zero-sum game theory is a limited amount of rewards and where only one side can advance while the other recedes. This assumption results in what is known as the Nash condition and equilibrium (Kyriakos Vamvoudakis, n.d.).

$$\min_{u_o} \max_{d_a} J(e, u_o, d_a) = \max_{u_o} \min_{d_a} J(e, u_o, d_a) \tag{3.5}$$

$$J(e, u_o^*, d_a) \leq J(e, u_o^*, d_a^*) \leq J(e, u_o, d_a^*) \tag{3.6}$$

where (u_o^*, d_a^*) is considered the optimal saddle point optimal solution for two players (actor vs. adverse). Using Equation (3.4), the Λ can be calculated as a solution to the Hamiltonian and the path to minimization of the cost function.

$$\Lambda^* = \min_{u_o} \max_{d_a} J(e, u_o, d_a) \tag{3.7}$$

However, in order to determine the saddle point within the zero-sum game assumption, the following equations are established.

$$\frac{\partial H}{\partial u_o} = 0 \implies u_o^* = -\frac{1}{2} R^{-1} g^T(e) \nabla \Lambda^* \tag{3.8}$$

$$\frac{\partial H}{\partial d_a} = 0 \implies d_a^* = -\frac{1}{2\gamma^2} k^T(e) \nabla \Lambda^* \tag{3.9}$$

Replacing these equations back into the Hamiltonian results in a maximized Hamiltonian (Luis Carrillo, 2019).

$$H^* = e^T Q e + \nabla \Lambda^{*T} f(e) - \frac{1}{4} \nabla \Lambda^{*T} g(e) R^{-1} g^T(e) \nabla \Lambda^* + \frac{1}{4\gamma^2} \nabla \Lambda^{*T} k(e) k^T(e) \nabla \Lambda^* \tag{3.10}$$

with $\nabla \Lambda = \frac{\partial \Lambda}{\partial e}$ and $\Lambda^*(0) = 0$ and in convergence $H^* = 0$.

3.1.3. Critic Neural Network Approximations of the Value Function

The function of the critic is to estimate a cost or value function. This estimation is improved upon every time step in order to estimate the value correctly. At the same time, the critic influences the Actor and Adverse NN's to change their weights.

In this problem, the primary value function is assumed to be the cost function of the optimal control. This results in the following NN assumption:

$$\Lambda^*(e) = W_c^{*T} \Phi_c(e) + \epsilon_c(e) \tag{3.11}$$

where $W_c \in \mathrm{R}^N$ is the critic weights, Φ_c is the NN with N number of activation functions and ϵ_c is the critic approximation error . In order to build the Hamiltonian, however, the following gradient is calculated:

$$\nabla \Lambda^*(e) = \nabla \Phi_c^{T*}{}_c + \nabla \epsilon_c(e) \tag{3.12}$$

The main concept behind NN approximations is based off of the Weistrass approximation which states that as $N \longrightarrow \infty$ the error and its gradient $\epsilon \longrightarrow 0$ and $\nabla \epsilon \longrightarrow 0$. Despite the free form that exists with the NN approximation it should be noted that there is an underlying assumption of boundedness that proceeds with this formulation as the NN, its weights, and their respective errors and its gradients are all considered to be bounded between terms. As such, Equation (3.12) is replaced in Equation (3.10) which results in the

following equation:

$$\hat{H} = \hat{W}_c^T \sigma(e) + e^T Q e + u_o^T R u_o - \gamma^2 ||d_a||^2 \xrightarrow{t \longrightarrow \infty} H^* \quad \forall e, u_o, d_a \tag{3.13}$$

where $\sigma = \nabla \Phi_c(f(e) + g(e)u_o + k(e)d_a)$. In this case $\hat{W}_c \xrightarrow{t \longrightarrow \infty} W_c^*$. Moreover, the methodology proposed for updating the critic weights is based on minimization of the approximation error (Luis Carrillo, 2019).

$$e_H = \hat{H} - H^* \quad \forall e, u_o, d_a \tag{3.14}$$

In this case, $H^* = 0$ is similar to the consideration done for $\Lambda^*(0) = 0$. This results in $\hat{H} = e_H$. However, to minimize this error the square residual error is calculated.

$$E_H = \frac{1}{2} \frac{e_H^T e_H}{(\sigma^T \sigma + 1)^2} \tag{3.15}$$

The residual error gradient descent method is used in order to build the adaptive weight updating equation. This results in the following:

$$\begin{aligned} \dot{\hat{W}}_c &= -\alpha_c \frac{\partial E_c}{\partial \hat{W}_c} \\ &= -\alpha_c \frac{\sigma}{(\sigma^T \sigma + 1)^2} (\sigma^T W_c + e^T Q e + u_o^T R u_o - \gamma^2 ||d_a||^2) \end{aligned} \tag{3.16}$$

where $\alpha_c > 0$ which is the rate of convergence coefficient. However, this formulation has a dependency for Persistence of Excitation(PE). In order to surpass that obstacle without including additional lemmas and assumptions, a buffer is used (Kyriakos Vamvoudakis, n.d.).

3.1.4. Critic Buffer

The main purpose of the buffer is built upon two basic needs. One is the necessity of persistence of excitation which is a requirement for every deep learning algorithms. Two, the buffer allows the system to compare its current status as not only serving the current state, but also all the previous states and inputs acquired which occupy the buffer size. This means that the estimated critic or modified weights are compared not only to the current

status of the cost function, but also all previous cost functions observed in prior steps. Thus, a buffer estimation error is defined as follows:

$$e_{buff} = \hat{H}_{buff,i} - H^* \quad \forall e, u_o, d_a \tag{3.17}$$

where the subscript i represents each step saved in the buffer memory. Using the same assumptions as the previous section along with the buffer, the combination of residual errors is shown as follows:

$$E_{total} = \frac{1}{2}\frac{e_H^T e_H}{(\sigma^T\sigma + 1)^2} + \frac{1}{2}\sum_{i=1}^{k}\frac{e_{buff,i}^T e_{buff,i}}{(\sigma_i^T\sigma_i + 1)^2} \tag{3.18}$$

The main reason for using $(\sigma^T\sigma + 1)^2$ is in order to normalize and bound the function to restrict divergence and ensure further convergence. Moreover, $k \in \mathbb{Z}^+$ is the number of saved events the buffer memory contains. Performing the gradient descent method on the modified residual mean error results in the following critic updating formulation:

$$\begin{aligned}\dot{\hat{W}}_c = &-\alpha_c\frac{\sigma}{(\sigma^T\sigma + 1)^2}(\sigma^T W_c + e^T Q e + u_o^T R u_o - \gamma^2||d_a||^2) \\ &- \alpha_{buff}\sum_{i=1}^{k}\frac{\sigma_i}{(\sigma_i^T\sigma_i + 1)^2}(\sigma_i^T W_c + e_i^T Q e_i + u_{o_i}^T R u_{o_i} - \gamma^2||d_{a_i}||^2)\end{aligned} \tag{3.19}$$

where $\alpha_{buff} > 0$ and $\alpha_c > 0$ are the updating rates respective to current or buffer memory status contribution. Moreover, it is important to determine the buffer size since this is computationally heavy and requires a big slot of memory for the processor, as well as the fact that this might keep some data which are unfavorable for a longer time that might disturb the clean convergence expected.(Luis Carrillo, 2019)

3.1.5. Actor and Adverse Neural Networks

Going back to Equations (3.8) and (3.9), if we replace the Lagrange constant with the NN as suggested in Equation (3.12) the resultant would be the following for the controller and adverse inputs (Kyriakos G. Vamvoudakis, 2010):

$$\frac{\partial H}{\partial u_o} = 0 \implies u_o^* = -\frac{1}{2}R^{-1}g^T(e)\nabla\Phi_c^T W_c^* \tag{3.20}$$

$$\frac{\partial H}{\partial d_a} = 0 \implies d_a^* = -\frac{1}{2\gamma^2}k^T(e)\nabla\Phi_c^T W_c^* \tag{3.21}$$

Replacing those in Equation (3.10) results in the following:

$$e^T Q e + W_c^T \nabla \Phi_c f(e) - \frac{1}{4} W_c^T D_o W_c + \frac{1}{4} W_c^T D_a W_c = 0 \tag{3.22}$$

where $D_o = \nabla \Phi_c g(e) R^{-1} g^T(e) \nabla \Phi_c^T$ and $D_a = \frac{1}{\gamma^2} \nabla \Phi_c k(e) k^T(e) \nabla \Phi_c^T$. However,using the critic weights directly would result in constant fluctuations and near instability especially in highly unstable systems. That is why a different weight is chosen for the actor and adverse NN. This enables the weights to update in a filtered manner. Thus, the critic weights should be different than that of the critic as they are influenced by the critic weights. As such the following is established (Luis Carrillo, 2019).

$$u_o = -\frac{1}{2} R^{-1} g^T(e) \nabla \Phi_c^T W_o \tag{3.23}$$

$$d_a = \frac{1}{2\gamma^2} k^T(e) \nabla \Phi_c^T W_a \tag{3.24}$$

where W_o and W_a represent the actor and the adversary NN respectively. In order to maintain the consistency of the HJI equation the following tuning of the actor and adverse are defined as follows (Igelnik & Yoh-Han Pao, 1995):

$$\dot{\hat{W}}_o = -\alpha_o \left\{ \frac{1}{2} D_o \left(W_o - W_c \right) - \frac{1}{4} D_o W_o \left(\frac{\sigma}{(\sigma^T \sigma + 1)^2} \right)^T W_c \right\} \tag{3.25}$$

$$\dot{\hat{W}}_a = -\alpha_a \left\{ \frac{1}{2} D_o \left(W_a - W_c \right) - \frac{1}{4} D_a W_a \left(\frac{\sigma}{(\sigma^T \sigma + 1)^2} \right)^T W_c \right\} \tag{3.26}$$

The presence of $\frac{\sigma}{(\sigma^T \sigma + 1)^2}$ guarantees persistence of excitation for both the actor and adverse in addition to a boundary limit to the tuning level. α_o and α_a are the actor and the adversary NN weight update rate or learning rate, respectively (Kyriakos G. Vamvoudakis, 2010).

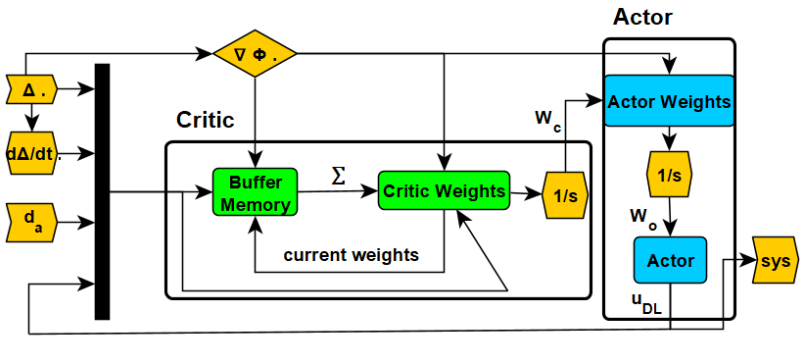


Figure 3.1 DL Schema without Adverse NN.

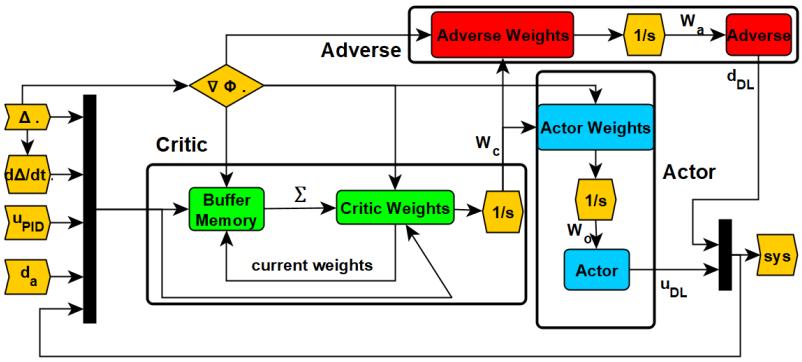


Figure 3.2 DL Schema with Adverse NN.

Figure (3.1) shows the formulation of the system where the critic and buffer update an actor NN weight while retaining observability of an external disturbance. On the other hand, Figure (3.2) shows the scheme of how the critic and the buffer update not only the actor, but also an adverse NN, which are both then inputed into the plant/system creating an internal adverse input.

3.2. Adverse Estimation

When the system is not generating its own adverse input via the adverse NN, various forms of threats, uncertainties and adverse inputs are present to disrupt the system or even destabilize it. Moreover, as the DL algorithm and specifically Equation (3.19) show, there is a need for a full state, dynamic, and disturbance observability (Li, 2016).

As such, primarily a UIO formulation is chosen which is shown as follows:

$$\dot{\hat{x}} = A\hat{x} + B_u u_o + L_x(y - \hat{y}) + B_d\hat{d}$$
$$\hat{y} = C\hat{x}. \tag{3.27}$$

$$\dot{\hat{\zeta}} = W\hat{\zeta} + L_d(y - \hat{y})$$
$$\hat{d} = V\hat{\zeta}. \tag{3.28}$$

However, the main issue behind performing this solution is determining out the value of both L_x and L_d. In this case, Linear Quadratic Estimators are used in order to determine the

L_x. For this consider the linearized state space with noise as follows (Li, 2016):

$$\dot{x} = Ax + B_u u_o + B_d d_a + G v_n$$
$$y = Cx + Du_o + w_n. \tag{3.29}$$

where v_n is the process noise and w_n is the measurement noise, usually white Gaussian noise with an estimated variance of Q_e and R_e, respectively with a mean of 0. Considering the estimation error to be $e_e = x - \hat{x}$, the resulting error dynamics are as follows (Brian D.O. Anderson, 1989):

$$\dot{e}_e = \dot{x} - \dot{\hat{x}}$$
$$= (A - L_x C)e_e + G v_n - L_x w_n \tag{3.30}$$

In order to minimize the error there are two aspects that need to be attenuated which are the mean and variance of the estimation error. Thus, consider the covariance matrix $P = E_e[e_e e_e^T]$. This results in the following covariance dynamics:

$$\dot{P} = (A - L_x C)P + P(A - L_x C)^T + G Q_e G^T - L_x R_e L_x^T \tag{3.31}$$

with this covariance equation the following assumption could be made for an infinite-horizon case:

$$\dot{P} \longrightarrow 0 \quad t \longrightarrow \infty \tag{3.32}$$
$$P \longrightarrow P_\infty \quad t \longrightarrow \infty \tag{3.33}$$

As such, an L_x should be chosen that minimizes the following cost function (Brian D.O. Anderson, 1989):

$$J_e = trace(P_\infty) \tag{3.34}$$

However, the dynamics of the covariance are portrayed as follows:

$$0 = (A - L_x C)P + P(A - L_x C)^T + G Q_e G^T - L_x R_e L_x^T = N \tag{3.35}$$

This results in a Hamiltonian as follows:

$$H_e = trace(P_\infty) + trace(\Lambda_e^T N) \tag{3.36}$$

where Λ_e is the Lagrange multiplier. Solving the Hamiltonian would result in the following Algebraic Ricatti Equation (ARE):

$$AP_\infty + P_\infty A^T - P_\infty C^T R_e^{-1} P_\infty + GQ_e G^T = 0 \tag{3.37}$$

Moreover, maximizing the Hamiltonian would result in the following equation (Brian D.O. Anderson, 1989):

$$L_x = P_\infty C^T R_e^{-1} \tag{3.38}$$

Furthermore, L_d is usually influenced by several considerations which includes a small premonition of the dynamics that might be affected by the external disturbance. This includes distributing weights according to the effect of the presumed disturbance on each state. However, it should be set in such a manner that the dynamics of ζ are stable. This is also affected by the values of W and V.

4. Numerical Simulation and Result Analysis

In this chapter, numerical simulations are described and different results are presented to analyze the proferomance of the DL proposed algorithm. It builds up culminating with the application of the quadrotor split algorithm along with the adverse estimator

4.1. High Order Dynamics

Controlling a quadrotor has been an extensively researched subject that has had several solutions. In general, the main states that need to be controlled are $[x, \dot{x}, y, \dot{y}, z, \dot{z}, \phi, \dot{\phi}, \theta, \dot{\theta}, \psi, \dot{\psi}]$.

In order to address this problem, the system is designed using two separate dynamical systems: one is concerned with stability and altitude (inner loop) and another one which is concerned with the horizontal position and attitude (outer loop).As shwon in Figure (4.1), The formulation of the inner loop arises by linearizing the system at hovering condition and with very small angles. Hovering input for quadrotors results in $k(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) = mg$. In consequence, we obtain the following outer loop dynamical equations:

$$\begin{aligned} \ddot{x} &= g\theta \\ \ddot{y} &= -g\phi \end{aligned} \tag{4.1}$$

whereas the inner loop dynamics could stay as follows.

$$\begin{aligned} m\ddot{z} &= k(\cos\phi\cos\theta)(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) - mg, \\ I_x\ddot{\phi} &= (I_y - I_z)\dot{\theta}\dot{\psi} + lk(\omega_4^2 - \omega_2^2), \\ I_y\ddot{\theta} &= (I_z - I_x)\dot{\phi}\dot{\psi} + lk(\omega_3^2 - \omega_1^2), \\ I_z\ddot{\psi} &= (I_x - I_y)\dot{\phi}\dot{\theta} + b(\omega_2^2 - \omega_1^2 + \omega_4^2 - \omega_3^2). \end{aligned} \tag{4.2}$$

In this configuration, each loop contains its own Deep Learning algorithm as well as estimator.

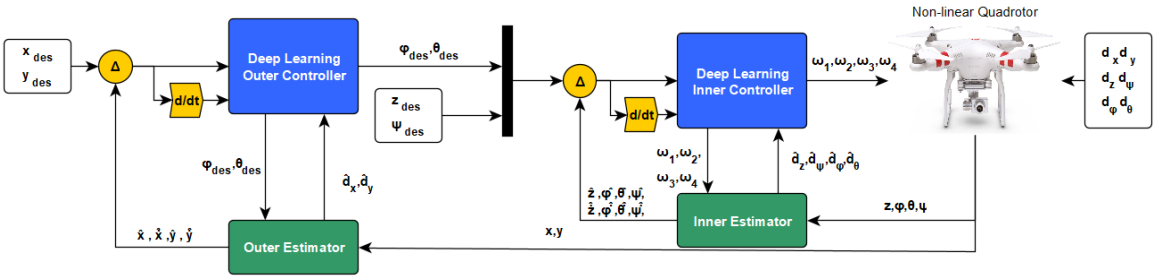


Figure 4.1 Quadrotor Loops and schema.

4.2. Initial Conditions

One of the most important aspects of DL algorithms is the initial conditions. This even gets more important with unstable systems such as the quadcopter dynamics. In such cases, the system would collapse or go through irreversible instability before the weights of the DL could converge to the desired values. Thus, it is preferable to have at least a safe first guess of optimal weights in order to guarantee stability and further optimization. Going back to the optimal control, the optimal input to the system is defined as:

$$u_o = -R^{-1}g^T(e)S_\infty e \tag{4.3}$$

While the one for the DL, the optimal input to the system is.

$$u_o = -\frac{1}{2}R^{-1}g^T(e)\nabla\Phi_c^T W_o \tag{4.4}$$

It is important to make a first assumption of $W_o = W_a = W_c$ since it is considered at this point that the critic weights are already optimal and thus can be used in all equations. This results in the following:

$$\frac{1}{2}\nabla\Phi_c^T W_c \equiv S_\infty e \tag{4.5}$$

Moreover, if the Φ_c is quadratic then the "$\equiv$" could turn into an "$=$". This results in a meticulous format of coefficient matching which results in the finalized initial weight guesses.

Note that if such initial condition process doesn't have solution, it is important to have the unstable system stabilized either by a form of another controller and then use the DL as

a complementary system or augmentation to the main controller. For example, as the configuration shown in Figure (4.2).

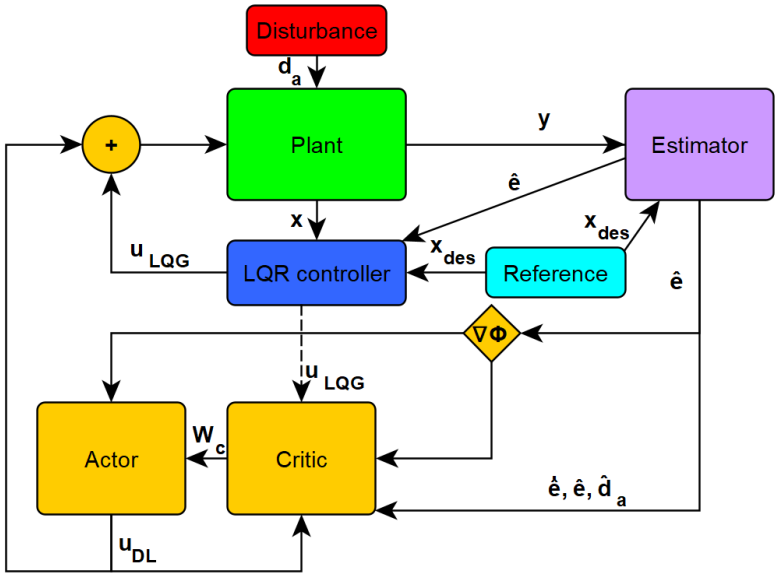


Figure 4.2 DL augmentation of an LQG with Disturbance Estimation.

4.3. DL Augmentation for Linear System

One of the first experimental phases was testing on an unstable linear system. This experiment did not include an Adverse Neural Network since the disturbance is considered an external one. The state space was as follows. Table (4.1) lists the various constants for the linearized simulation.

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}, B_u = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, B_d = B_u, C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Table 4.1

DL constants.

Constants	γ	Q	R	α_o	α_c	α_{buff}	Q_E	R_E	G	W	V	L_d
Value	4	$10\times\mathbb{I}^{3\times3}$	1	0.1	1	0.001	$10\times\mathbb{I}^{3\times3}$	$10\times\mathbb{I}^{2\times2}$	$\mathrm{I}^{3\times3}$	-1	0.9	$[0, 10]$

The initial conditions for $W_c = W_o$ are a $[6 \times 1]$ ones-matrix. The quadratic form of NN was chosen for both actor and critic and their respective gradients are as follows:

$$\Phi_c = \begin{bmatrix} x_1^2 & x_1x_2 & x_2^2 & x_1x_3 & x_2x_3 & x_3^2 \end{bmatrix}$$

$$\nabla\Phi_c = \begin{bmatrix} 2x_1 & 0 & 0 \\ x_2 & x_1 & 0 \\ 0 & 2x_2 & 0 \\ x_3 & 0 & x_1 \\ 0 & x_3 & x_2 \\ 0 & 0 & 2x_3 \end{bmatrix}$$

As this is not a tracking test, but rather a stability check, the initial conditionsfor the states are chosen as are $[2, 2, 2]$. At ten seconds mark a step input of magnitude 2 is placed as an adverse input along with a chirp signal of amplitude 0.5 as shown in Figure (4.3). The initial condition for ζ is 0. It is important to note that the open loop poles are $[1, 1, 1]$ while the LQR closed loop poles are $[-0.767, -2.128, -2.8077]$. This shows that the system is now stabilized. Figures (4.4) to (4.9) show the main results obtained for all three states as well as the inputs between LQR and Deep Learning Augmentation of LQR along with the critic and actor weights.

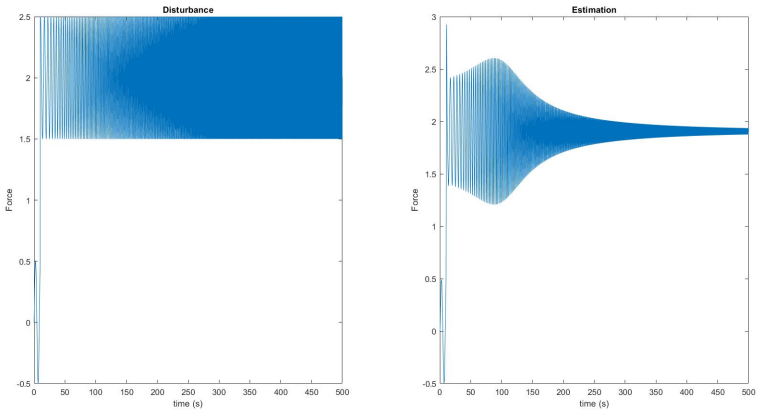


Figure 4.3 Disturbance vs Disturbance Estimator .

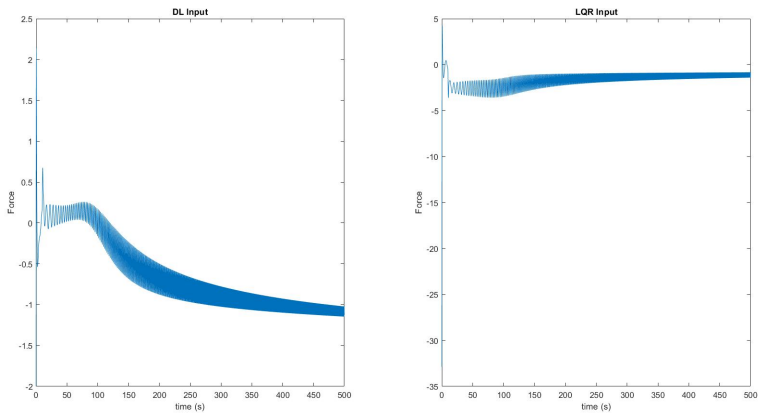


Figure 4.4 DL vs LQR input.

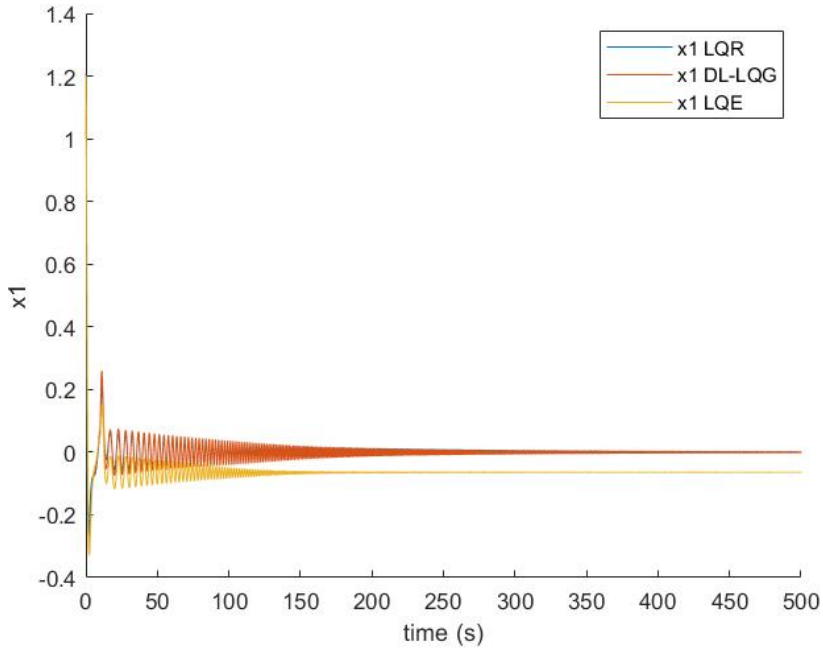


Figure 4.5 x1 State Estimation, LQR Controller, and DL-LQG Controller .

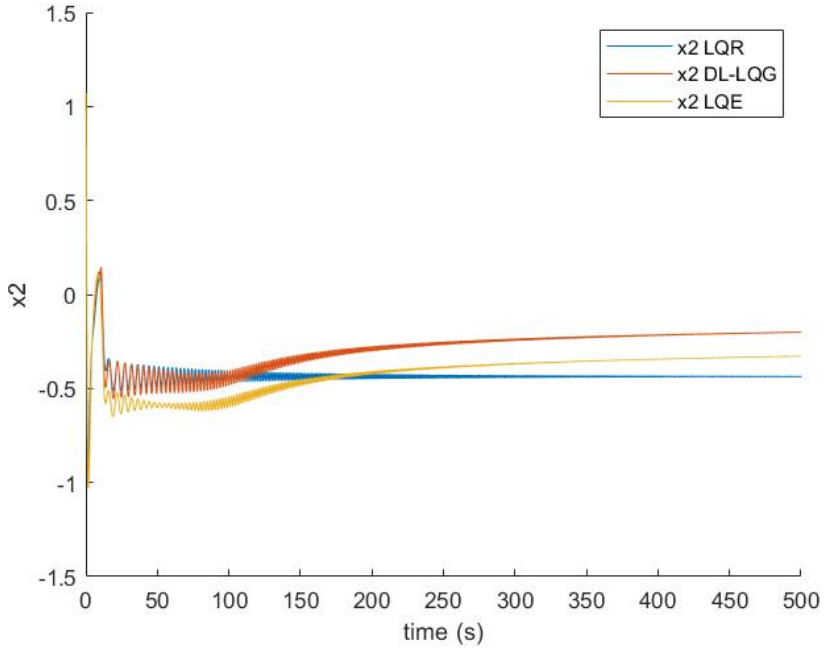


Figure 4.6 x2 State Estimation, LQR Controller, and DL-LQG Controller .

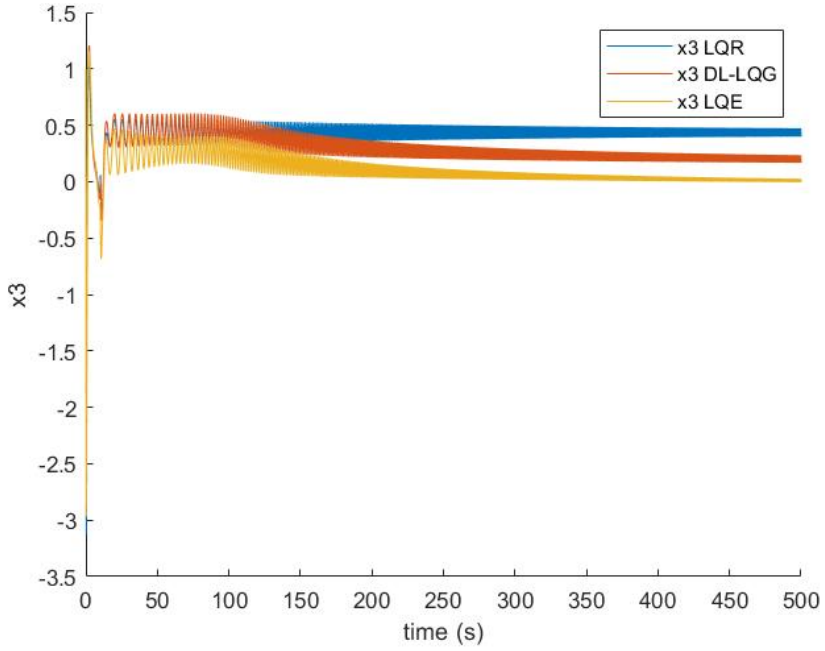


Figure 4.7 x3 State Estimation, LQR Controller, and DL-LQG Controller .

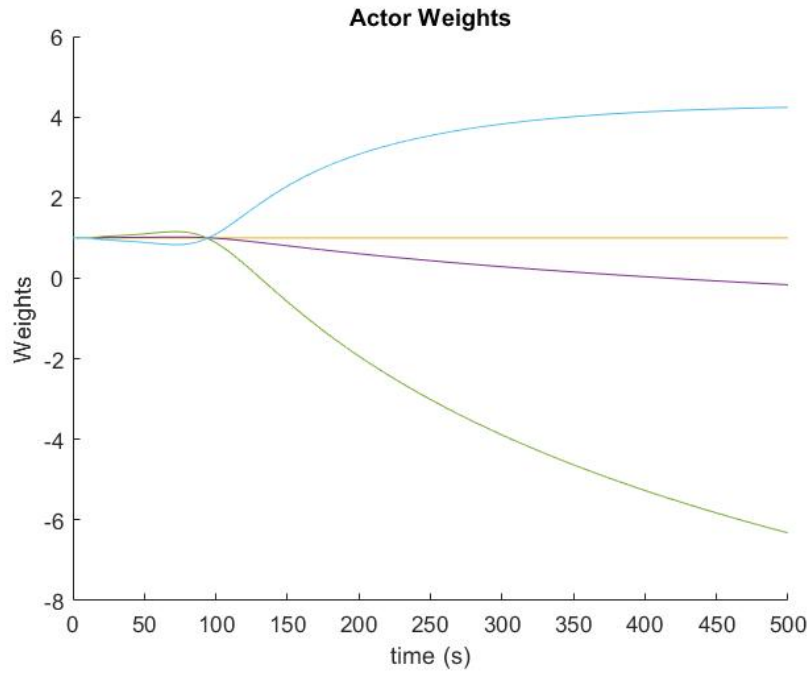


Figure 4.8 Actor Weights.

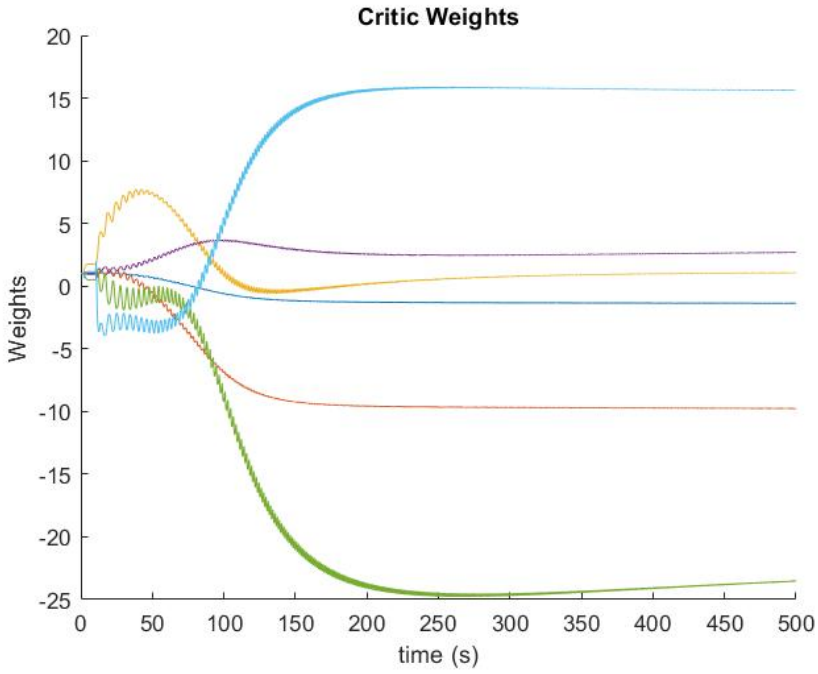


Figure 4.9 Critic Weights.

From these figure, there is not much of a difference when it came to LQR and DL-LQG controllers during nominal conditions when observing solely the states. However, the main

difference appears in the inputs and states. The DL algorithm added an additional form of constant push back which caused the states to return back to their stable 0 position, but with the LQR alone there was a significant offset that did not decrease.

On the other hand, the disturbance estimator shows to be accurate in its estimation, but failed at retaining the chirp magnitude, the higher the chirp frequency got the more attenuated it became. Moreover, the DL weights converged which a sign that the algorithm is stable and does not diverge.

4.4. Linear Quadrotor Simulation

One of the first trials for Quadrotor stability analysis was an LQR augmented with DL applied to a linearized version of a quadrotor state space dynamics:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, B_u = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $B_d = B_u$, and inputs are similar to Equation (2.3) with m, I_x, I_y, and I_z are 1. The vector of the states is $[x, \dot{x}, y, \dot{y}, z, \dot{z}, \phi, \dot{\phi}, \theta, \dot{\theta}, \psi, \dot{\psi}]$. Moreover, the adaptive law for the actor and adverse weights are not taken as prior, but use the formulation derived from

(Kyriakos Vamvoudakis, n.d.). Which are:

$$\dot{\hat{W}}_o = -\alpha_o \left\{ \left(F_o W_o - F_c \frac{\sigma}{\sigma^T \sigma + 1} W_c \right) - \frac{1}{4} D_o W_o \left(\frac{\sigma}{(\sigma^T \sigma + 1)^2} \right)^T W_c \right\} \quad (4.6)$$

$$\dot{\hat{W}}_a = -\alpha_a \left\{ \left(F_a W_a - F_c \frac{\sigma}{\sigma^T \sigma + 1} W_c \right) - \frac{1}{4} D_a W_a \left(\frac{\sigma}{(\sigma^T \sigma + 1)^2} \right)^T W_c \right\} \quad (4.7)$$

where F_a, F_o and F_c are the confidence rates of the adverse, actor and critic respectively.

The two main scenarios where derived from Figure (4.10) and (4.11).

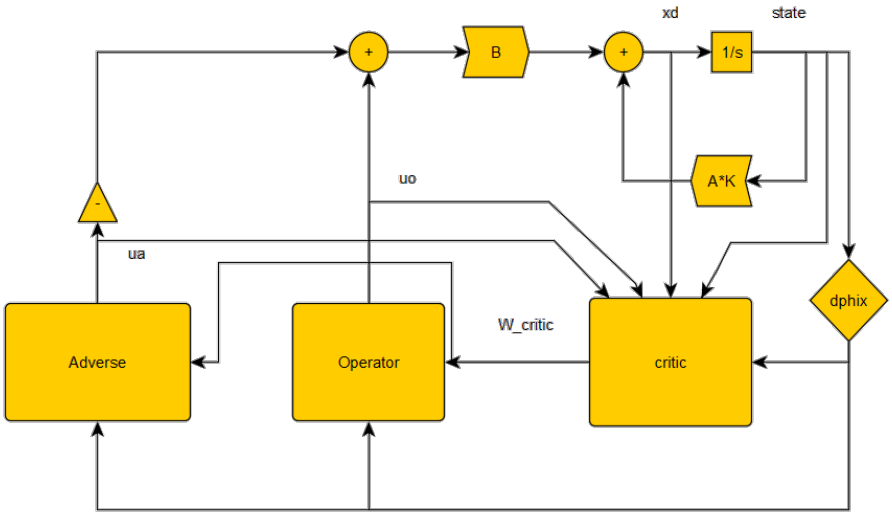


Figure 4.10 Scenario 1.

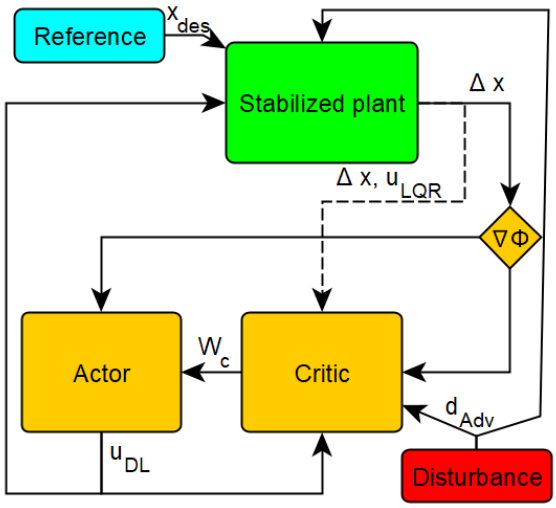


Figure 4.11 Scenario 2.

The schema and the constants used in these trial scenarios are shown in Table (4.2) while the constants used in the simulations are shown in Table (4.3).

Table 4.2

Scenarios.

	Scenario 1 (Internal Adverse)		Scenario 2 (External Disturbance)	
Systems	LQR	LQR+DL	LQR	LQR+DL
Closed Loop	LQR Input	LQR + Operator input + Adverse Input	LQR Input	LQR input + Operator input + Extraneous input
Observability	States	States + Dynamics	States	States + Dynamics + Extraneous Input
Adverse Input	N/A	Adversary NN	Step input Magnitude 20 at 10 seconds	

Table 4.3

Control constants.

Constants	γ	Q	R	α_o	α_c	α_{buff}	α_a	F_o	F_a	F_c	X_o
Value	1	$10\times\mathbb{I}^{12\times12}$	$I^{4\times4}$	0.01	10	1	1	10	1	1	$1^{12\times1}$

It is important to note that the initial guess for all NN's is randomly generated and the critic, actor, and adverse Neural Networks all used in this simulation are a twelve state quadratic NN. Figure (4.12) shows the difference in system inputs coming from the Actor and Adverse and LQR respectivley for Scenario 1, while Figures (4.13) to (4.15) show the progression of the Weights of the different NN's through time, and Figure (4.16) compare all the states between an LQR and a DL augmented LQR in Scenario 1. Moreover, Figure (4.17) shows the inputs to the system of actor, disturbance, and LQR to the system, while Figures (4.18) and (4.19) show the progression of actor and critic weights respectively for Scenario 2, and Figure (4.20) show the comparison of the states between an LQR system vs. an DL augmented LQR.

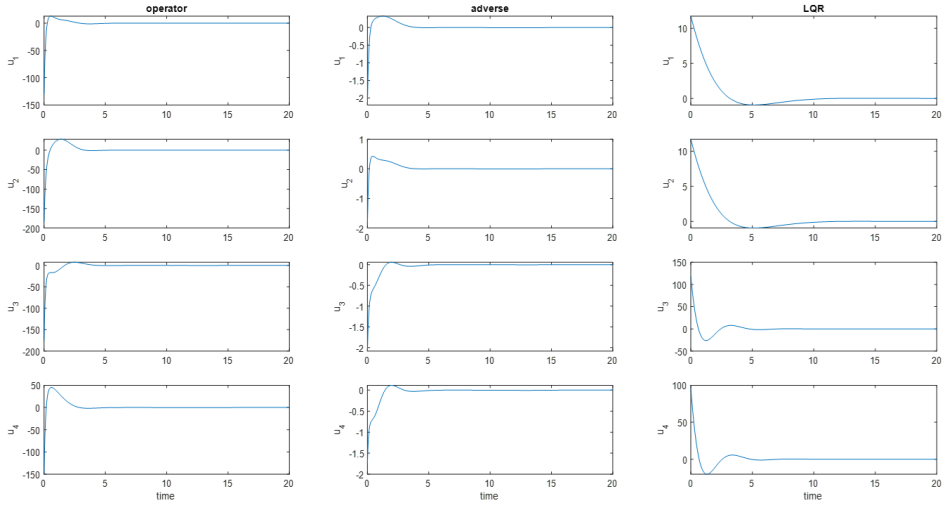


Figure 4.12 Inputs to the system (Scenario 1).

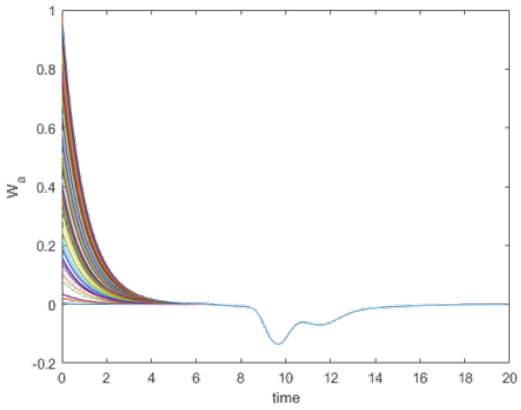


Figure 4.13 Adverse Weights (Scenario 1).

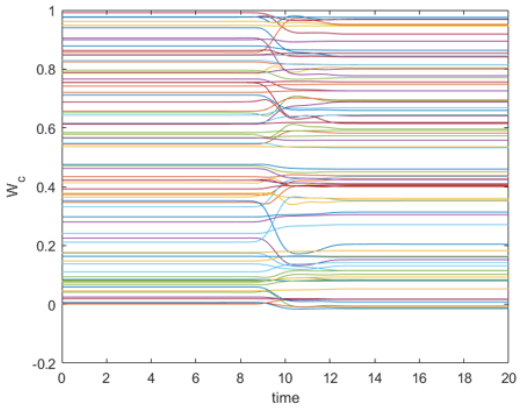


Figure 4.14 Critic Weights (Scenario 1).

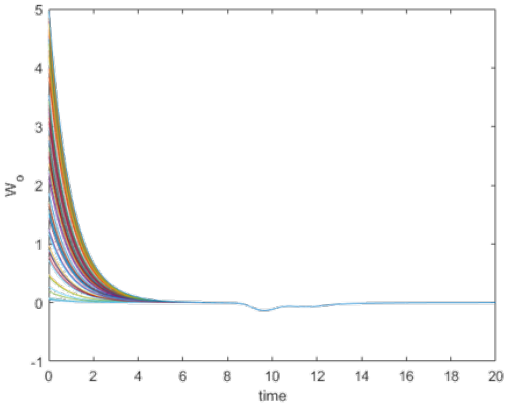


Figure 4.15 Actor Weights (Scenario 1).

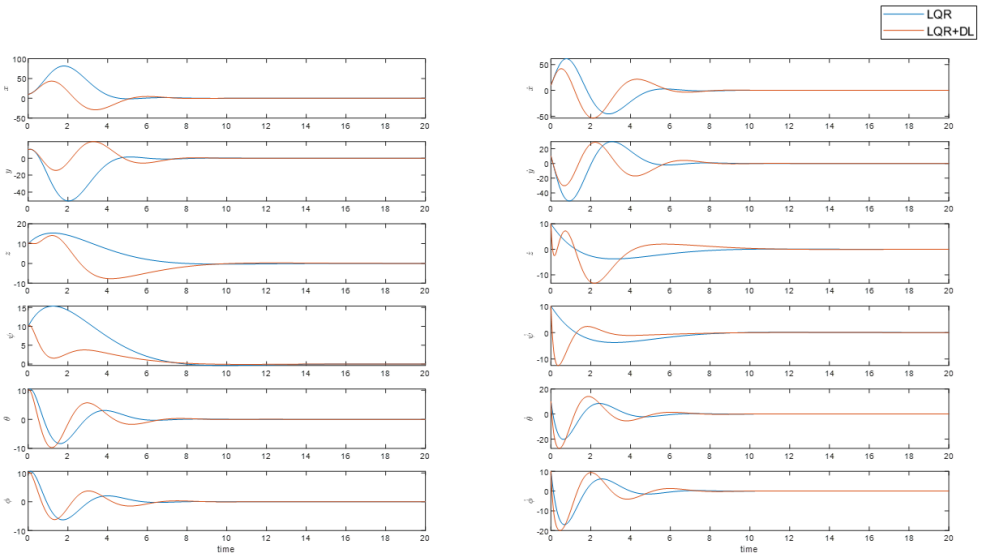


Figure 4.16 States LQR vs DL+LQR (Scenario 1).

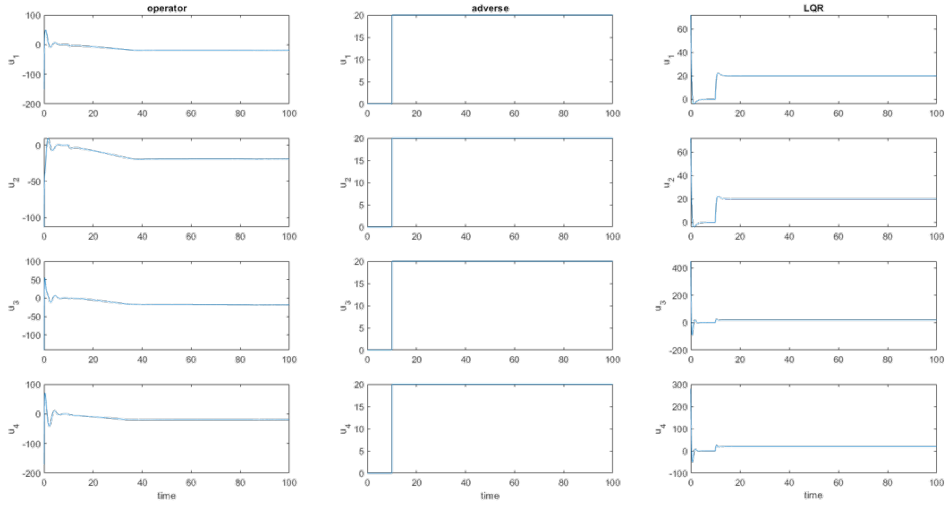


Figure 4.17 System Inputs (Scenario 2).

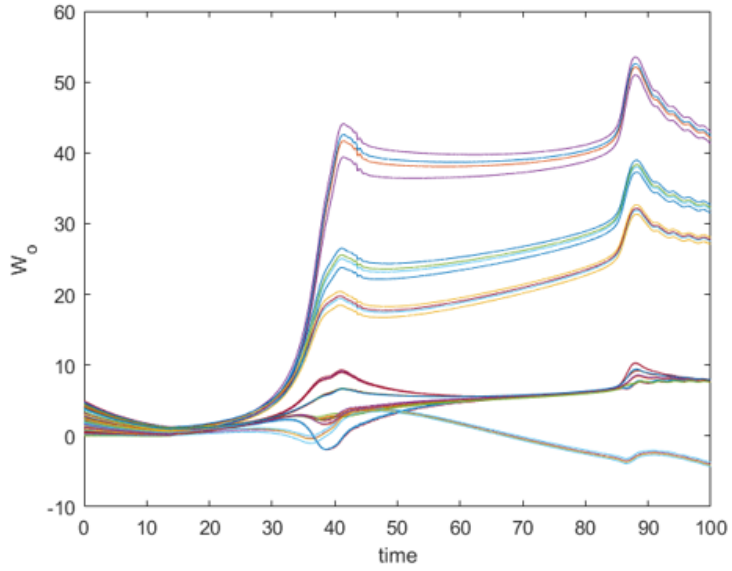


Figure 4.18 Actor Weights (Scenario 2).

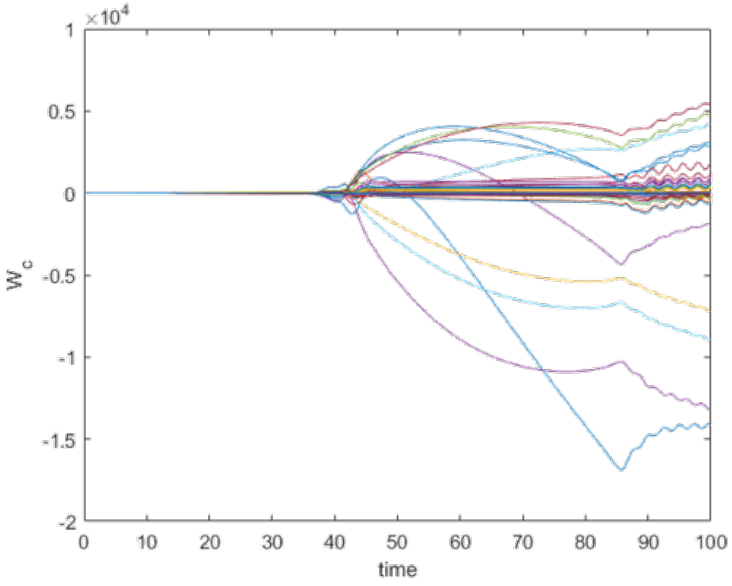


Figure 4.19 Critic Weights (Scenario 2).

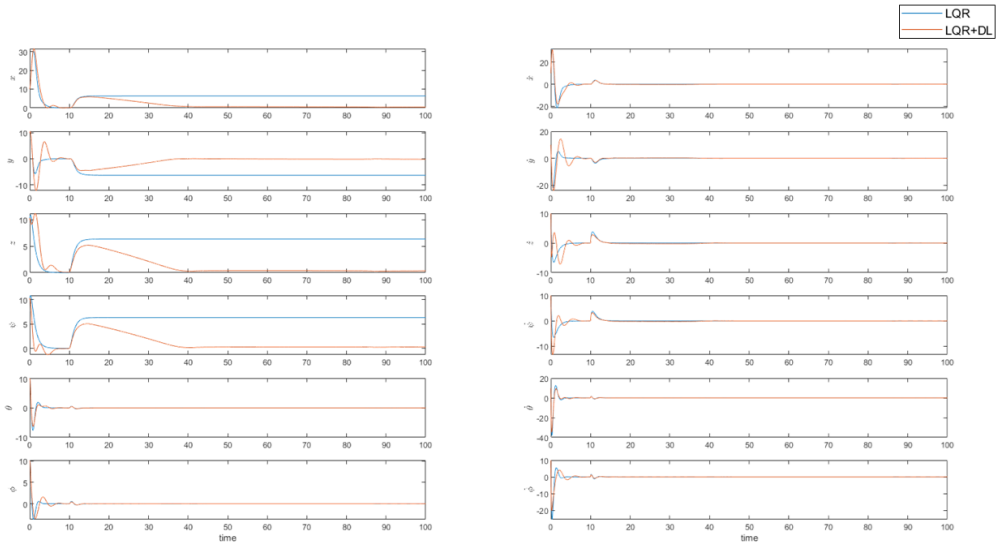


Figure 4.20 States LQR vs DL+LQR (Scenario 2).

As seen in Figure (4.16) the DL augmented LQR controller shows good signs of stability with less fluctuations even with the adverse NN input disturbaing the system. Moreover, Figure (4.20) clearly portrays the difference between the DL augmentation and a sole LQR controller as the system tends to bring back the states to the equilibrium point of 0 with the DL augmentation, while maintaining an offset in the states when LQR is used alone in Scenario 2.

4.5. DJI Quadrotor Two-loop DL Controller

In this section the format shown in Figure (4.1) is used along with an actor-critic controller, with the dynamic equations that are portrayed as follows:

$$
\begin{aligned}
m\ddot{x} &= k(\sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta)(\omega_1^2+\omega_2^2+\omega_3^2+\omega_4^2) + d_x,\\
m\ddot{y} &= k(\cos\phi\sin\theta\sin\psi - \cos\psi\sin\phi)(\omega_1^2+\omega_2^2+\omega_3^2+\omega_4^2) + d_y,\\
m\ddot{z} &= k(\cos\phi\cos\theta)(\omega_1^2+\omega_2^2+\omega_3^2+\omega_4^2) - mg + d_z,\\
I_x\ddot{\phi} &= (I_y - I_z)\dot{\theta}\dot{\psi} + lk(\omega_4^2-\omega_2^2) + d_\phi,\\
I_y\ddot{\theta} &= (I_z - I_x)\dot{\phi}\dot{\psi} + lk(\omega_3^2-\omega_1^2) + d_\theta,\\
I_z\ddot{\psi} &= (I_x - I_y)\dot{\phi}\dot{\theta} + b(\omega_2^2-\omega_1^2+\omega_4^2-\omega_3^2) + d_\psi.
\end{aligned}
\tag{4.8}
$$

the DL is applied to the dynamics on a DJI Phantom 2. Table (4.4) lists the physical parameters of the DJI Phantom 2 (Christoph Aoun & Shammas, 2019).

Table 4.4

DJI Phantom 2 Parameters

$m_{\text{quad}} = 1.3$ kg	$I_x = 0.081$ kgm^2
$I_y = 0.081$ kgm^2	$I_z = 0.142$ kgm^2
$k = 3.8305 \times 10^{-6}$	$b_t = 2.2518 \times 10^{-8}$
$\omega_{\max} = 1047.197$ rad/s	$l = 0.175$ m

4.5.1. Linearization

In order to Linearize the system. Primarily, the conditions should be in hovering state. This is done by have a rotor speed of 913.12 rad/s. The outer loop follows Equation (4.1) and the inner loop uses the Equations (4.2). This results in the following linear state spaces:

For Outer:

$$A_o = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, B_{uo} = \begin{bmatrix} 0 & 0 \\ 9.81 & 0 \\ 0 & 0 \\ 0 & -9.81 \end{bmatrix}, B_{do} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

For Inner:

$$A_i = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B_{ui} = 10^{-5}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0.2947 & 0.2947 & 0.2947 & 0.2947 \\ 0 & 0 & 0 & 0 \\ 0 & 0.8276 & 0 & -0.8276 \\ 0 & 0 & 0 & 0 \\ -0.8276 & 0 & 0.8276 & 0 \\ 0 & 0 & 0 & 0 \\ 0.0159 & -0.0159 & 0.0159 & -0.0159 \end{bmatrix}, B_{di} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where the subscript of i indicates inner and a subscript of o indicates outer. The inputs to the outer loop are $[\phi_{des}, \theta_{des}]$ while the inputs to the inner loop are $[\omega_1^2, \omega_2^2, \omega_3^2, \omega_4^2]$. The states for the outer loop are $[x, \dot{x}, y, \dot{y}]$ and for the inner loop are $[z, \dot{z}, \phi, \dot{\phi}, \theta, \dot{\theta}, \psi, \dot{\psi}]$. The initial condition for all the states is zero.

4.5.2. Deep Learning Design

To determine the initial conditions of weights using LQR, the system must be linearized. A quadratic NN activation function was defined to create a primary weight values. Table (4.5) summarizes the values for the main parameters used within the DRL architecture.

Table 4.5

DL constants.

Inner loop Parameters										
Q=diag([$10^{15}, 10^{15}, 10^{15}, 10^{10}, 10^{12}, 10^{12}, 10^{16}, 10^{16}$])										
γ	R	α_o	α_c	α_{buff}	Q_E	R_E	G	V	T_p	T_d
10^6	10 $\times\mathbb{I}^{4\times4}$	0.05	0.05	0.05	10$\times\mathbb{I}^{8\times8}$	$I^{8\times8}$	$I^{8\times8}$	1	-0.1 $\times\mathbb{I}^{2\times2}$	-0.1 $\times\mathbb{I}^{2\times2}$
$L_d = 10 \times [1, 10, 0, 0, 0, 0, 0, 0; 0, 0, 1, 1, 0, 0, 0, 0, ; 0, 0, 0, 0, 1, 1, 0, 0; 0, 0, 0, 0, 0, 0, 1, 1]$										
Outer loop Parameters										
Q=diag([0.05,0.05,0.05,0.05])										
γ	R	α_o	α_c	α_{buff}	Q_E	R_E	G	V	T_p	T_d
1	$I^{2\times2}$	0.05	0.05	0.05	10$\times\mathbb{I}^{4\times4}$	$I^{4\times4}$	$I^{4\times4}$	1	-1 $\times\mathbb{I}^{2\times2}$	-1 $\times\mathbb{I}^{2\times2}$
$L_d = [10, 1, 0, 0; 0, 0, 10, 1]$										

Initial numerical simulations were run for a case where a step input disturbance was introduced into the y while the system is hovering at attitude zero. This could represent, for example, an external force input due to wind.

4.5.3. Initial Conditions

The function (Φ_c) has a quadratic form for both inner and outer loops. Using this NN, we solve the following equivalence: $S_\infty x = \frac{1}{2}\nabla\Phi_c^T W_c$.

For the outer loop, the gradient of the quadratic NN would result in the following $\nabla\Phi_c$:

$$\nabla\Phi_{co} = \begin{bmatrix} 2x_1 & 0 & 0 & 0 \\ x_2 & x_1 & 0 & 0 \\ 0 & 2x_2 & 0 & 0 \\ x_3 & 0 & x_1 & 0 \\ 0 & x_3 & x_2 & 0 \\ 0 & 0 & 2x_3 & 0 \\ x_4 & 0 & 0 & x_1 \\ 0 & x_4 & 0 & x_2 \\ 0 & 0 & x_4 & x_3 \\ 0 & 0 & 0 & 2x_4 \end{bmatrix}$$

where $[x_1, x_2, x_3, x_4] = [x, \dot{x}, y, \dot{y}]$. Taking into consideration the constants of the outer loop used in Table (4.5) the following equivalence is made:

$$\frac{1}{2}\nabla\Phi_c^T W_c = S_\infty x$$

$$\begin{bmatrix} w_1x_1 + \frac{w_2x_2}{2} + \frac{w_4x_3}{2} + \frac{w_7x_4}{2} \\ \frac{w_2x_1}{2} + w_3x_2 + \frac{w_5x_3}{2} + \frac{w_8x_4}{2} \\ \frac{w_4x_1}{2} + \frac{w_5x_2}{2} + w_7x_3 + \frac{w_9x_4}{2} \\ \frac{w_7x_1}{2} + \frac{w_8x_2}{2} + \frac{w_9x_3}{2} + w_{10}x_4 \end{bmatrix} = \begin{bmatrix} 0.0691x_1 + 0.0228x_2 - 3.62 \times 10^{-18}x_3 + 7.6029 \times 10^{-18}x_4 \\ 0.0228x_1 + 0.0315x_2 - 1.5313 \times 10^{-18}x_3 + 3.8237 \times 10^{-18}x_4 \\ -3.62 \times 10^{-18}x_1 - 1.5313 \times 10^{-18}x_2 + 0.0709x_3 + 0.0228x_4 \\ 7.6029 \times 10^{-18}x_1 + +3.8237 \times 10^{-18}x_2 + 0.0228x_3 + 0.0323x_4 \end{bmatrix}$$

This results in the following initial outer critic and actor weights:

$$W_{co} = \begin{bmatrix} 0.06911 \\ 0.0456 \\ 0.0315 \\ -7.24 \times 10^{-18} \\ -3.0626 \times 10^{-18} \\ 0.0709 \\ 1.5206 \times 10^{-17} \\ 7.6475 \times 10^{-18} \\ 0.0456 \\ 0.0323 \end{bmatrix}$$

These weights are used used as an optimal initial condition for the outer loop DL controller.

Similarly to the prior formulation, the following is the NN gradient for the inner loop.

$$\nabla\Phi_{ci} = \begin{bmatrix}
2x_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
x_2 & x_1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2x_2 & 0 & 0 & 0 & 0 & 0 & 0 \\
x_3 & 0 & x_1 & 0 & 0 & 0 & 0 & 0 \\
0 & x_3 & x_2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 2x_3 & 0 & 0 & 0 & 0 & 0 \\
x_4 & 0 & 0 & x_1 & 0 & 0 & 0 & 0 \\
0 & x_4 & 0 & x_2 & 0 & 0 & 0 & 0 \\
0 & 0 & x_4 & x_3 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 2x_4 & 0 & 0 & 0 & 0 \\
x_5 & 0 & 0 & 0 & x_1 & 0 & 0 & 0 \\
0 & x_5 & 0 & 0 & x_2 & 0 & 0 & 0 \\
0 & 0 & x_5 & 0 & x_3 & 0 & 0 & 0 \\
0 & 0 & 0 & x_5 & x_4 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2x_5 & 0 & 0 & 0 \\
x_6 & 0 & 0 & 0 & 0 & x_1 & 0 & 0 \\
0 & x_6 & 0 & 0 & 0 & x_2 & 0 & 0 \\
0 & 0 & x_6 & 0 & 0 & x_3 & 0 & 0 \\
0 & 0 & 0 & x_6 & 0 & x_4 & 0 & 0 \\
0 & 0 & 0 & 0 & x_6 & x_5 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 2x_6 & 0 & 0 \\
x_7 & 0 & 0 & 0 & 0 & 0 & x_1 & 0 \\
0 & x_7 & 0 & 0 & 0 & 0 & x_2 & 0 \\
0 & 0 & x_7 & 0 & 0 & 0 & x_3 & 0 \\
0 & 0 & 0 & x_7 & 0 & 0 & x_4 & 0 \\
0 & 0 & 0 & 0 & x_7 & 0 & x_5 & 0 \\
0 & 0 & 0 & 0 & 0 & x_7 & x_6 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 2x_7 & 0 \\
x_8 & 0 & 0 & 0 & 0 & 0 & 0 & x_1 \\
0 & x_8 & 0 & 0 & 0 & 0 & 0 & x_2 \\
0 & 0 & x_8 & 0 & 0 & 0 & 0 & x_3 \\
0 & 0 & 0 & x_8 & 0 & 0 & 0 & x_4 \\
0 & 0 & 0 & 0 & x_8 & 0 & 0 & x_5 \\
0 & 0 & 0 & 0 & 0 & x_8 & 0 & x_6 \\
0 & 0 & 0 & 0 & 0 & 0 & x_8 & x_7 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2x_8
\end{bmatrix}$$

where $[x1, x2, x3, x4, x5, x6, x7, x8] = [z, \dot{z}, \phi, \dot{\phi}, \theta, \dot{\theta}, \psi, \dot{\psi}]$. Using similar processes as the section above the following equivalencies are established.

$$\begin{bmatrix} w1 \times x1 + (w2 \times x2)/2 + (w4 \times x3)/2 + (w7 \times x4)/2 + (w11 \times x5)/2 + (w16 \times x6)/2 + (w22 \times x7)/2 + (w29 \times x8)/2 \\ (w2 \times x1)/2 + w3 \times x2 + (w5 \times x3)/2 + (w8 \times x4)/2 + (w12 \times x5)/2 + (w17 \times x6)/2 + (w23 \times x7)/2 + (w30 \times x8)/2 \\ (w4 \times x1)/2 + (w5 \times x2)/2 + w6 \times x3 + (w9 \times x4)/2 + (w13 \times x5)/2 + (w18 \times x6)/2 + (w24 \times x7)/2 + (w31 \times x8)/2 \\ (w7 \times x1)/2 + (w8 \times x2)/2 + (w9 \times x3)/2 + w10 \times x4 + (w14 \times x5)/2 + (w19 \times x6)/2 + (w25 \times x7)/2 + (w32 \times x8)/2 \\ (w11 \times x1)/2 + (w12 \times x2)/2 + (w13 \times x3)/2 + (w14 \times x4)/2 + w15 \times x5 + (w20 \times x6)/2 + (w26 \times x7)/2 + (w33 \times x8)/2 \\ (w16 \times x1)/2 + (w17 \times x2)/2 + (w18 \times x3)/2 + (w19 \times x4)/2 + (w20 \times x5)/2 + w21 \times x6 + (w27 \times x7)/2 + (w34 \times x8)/2 \\ (w22 \times x1)/2 + (w23 \times x2)/2 + (w24 \times x3)/2 + (w25 \times x4)/2 + (w26 \times x5)/2 + (w27 \times x6)/2 + w28 \times x7 + (w35 \times x8)/2 \\ (w29 \times x1)/2 + (w30 \times x2)/2 + (w31 \times x3)/2 + (w32 \times x4)/2 + (w33 \times x5)/2 + (w34 \times x6)/2 + (w35 \times x7)/2 + w36 \times x8 \end{bmatrix}$$

$$= \begin{bmatrix} 1.016e+15 & 1.6969e+13 & 1.881 & 0.1616 & -0.0441 & 0.0124 & 0.7756 & -0.5861 \\ 1.6969e+13 & 17255e+13 & -0.0763 & -0.0022 & -0.0014.9714e-4 & -0.0899 & -0.0249 & \\ 1.8811 & -0.0763 & 1.3076e+14 & 8.5444e+12 & -0.0525 & -0.055 & -0.4826 & -0.107 \\ 0.1616 & -0.0022 & 8.5444e+12 & 1.1173e+12 & -0.0034 & -0.0029 & -0.1975 & -0.0129 \\ -0.0441 & -0.001 & -0.0525 & -0.0034 & 1.2411e+11 & 3.3535e+11 & -0.2104 & -0.0275 \\ 0.7756 & -0.0899 & -0.4826 & -0.1975 & -0.091 & -0.2104 & 1.0952e+16 & 9.9706e+14 \\ -0.5861 & -0.0249 & -0.107 & -0.0129 & -0.0147 & -0.0275 & 9.9706e+14 & 1.0920e+15 \end{bmatrix} \times \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \\ x5 \\ x6 \\ x7 \\ x8 \end{bmatrix}$$

The following Critic and Actor initial weights for the inner loop are found:

$$\mathrm{W}_{ci} = \begin{bmatrix} 1.0168e+15 \\ 3.3938e+13 \\ 1.7255e+13 \\ 3.7622 \\ -0.1526 \\ 0.3232 \\ -0.0043 \\ 1.7089e+13 \\ 1.1173e+12 \\ -0.0881 \\ -0.002 \\ -0.105 \\ -0.0069 \\ 1.2411e+12 \\ 0.0248 \\ 9.9427e-4 \\ -0.11 \\ -0.059 \\ 5.4039e+11 \\ 3.3535e+11 \\ 1.5512 \\ -0.1798 \\ -0.9653 \\ -0.3951 \\ -0.1821 \\ -0.4208 \\ 1.0952e+16 \\ -1.1721 \\ -0.0499 \\ -0.214 \\ -0.0257 \\ -0.0293 \\ -0.0549 \\ 1.9941e+15 \\ 1.092e+15 \end{bmatrix}$$

4.5.4. Results

An adverserial input d_y of step input of value 2 N at 100 seconds as shown in Figure (4.21). Figure (**??**) shows all the weights of the actors and critics of both the inner and outer loops. Figures (4.25) to (4.35) show the comparison between the states when the controllers are either LQG or DL controllers. Furthermore, Figure (4.36) shows the difference between controller inputs of the LQG and DL controllers.

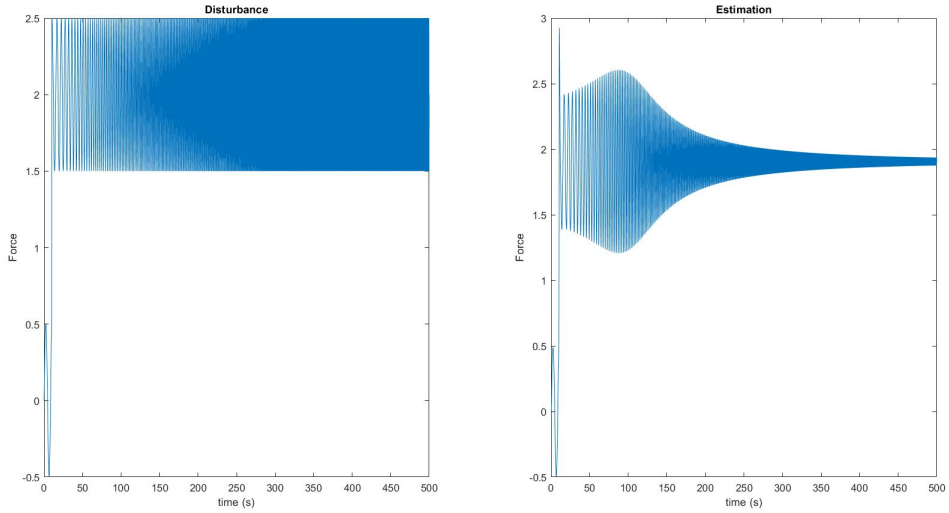


Figure 4.21 Adverse Estimation vs Actual Disturbance.

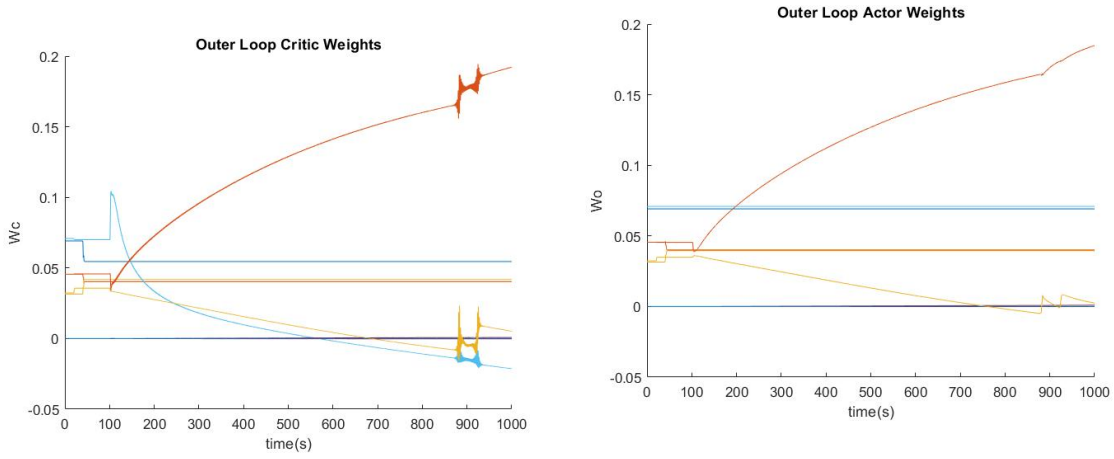


Figure 4.22 DL Outer Weights.

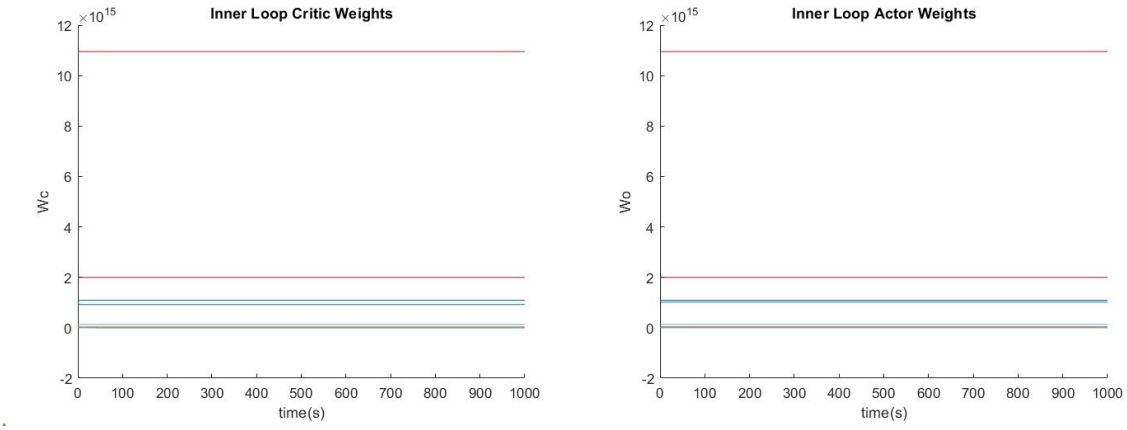


Figure 4.23 DL Inner Weights.

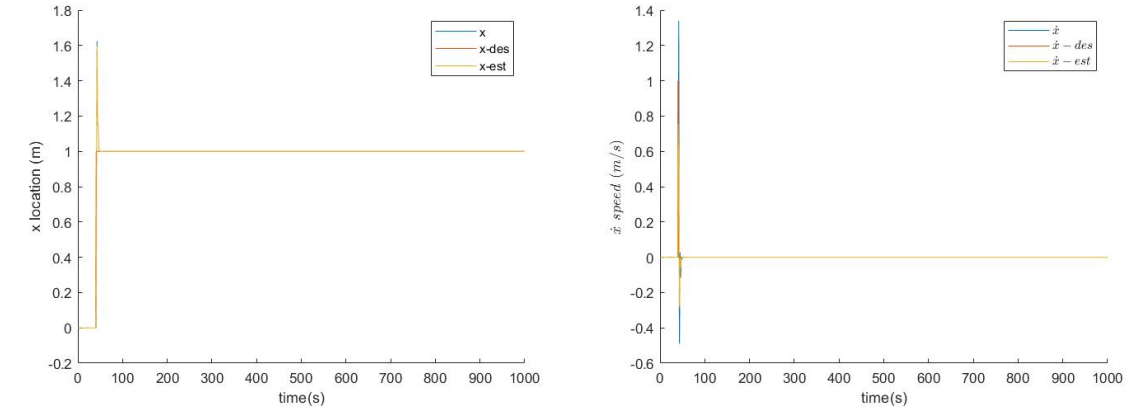
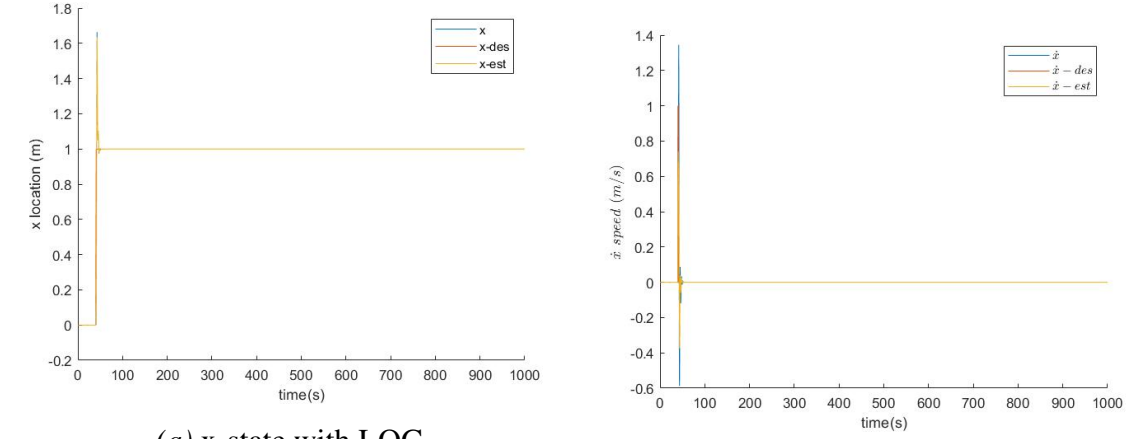


Figure 4.24 x and $\dot{x}$ states with DL Controller.



(a) x-state with LQG

Figure 4.25 x and $\dot{x}$ states with LQG Controller.

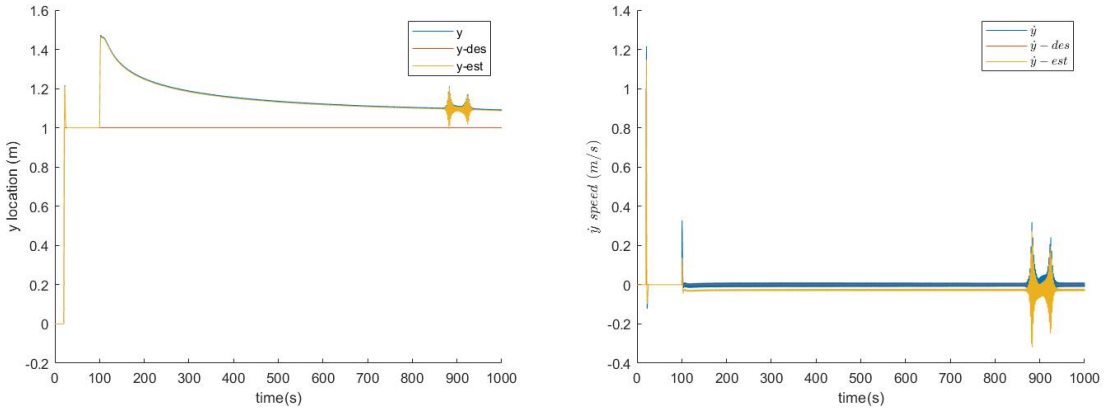


Figure 4.26 y and $\dot{y}$ states with DL Controller.

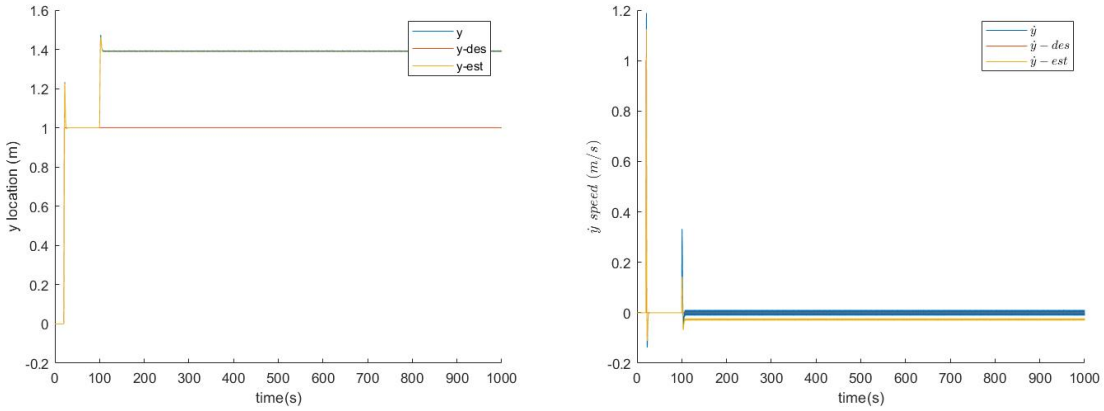


Figure 4.27 y and $\dot{y}$ states with LQG Controller.

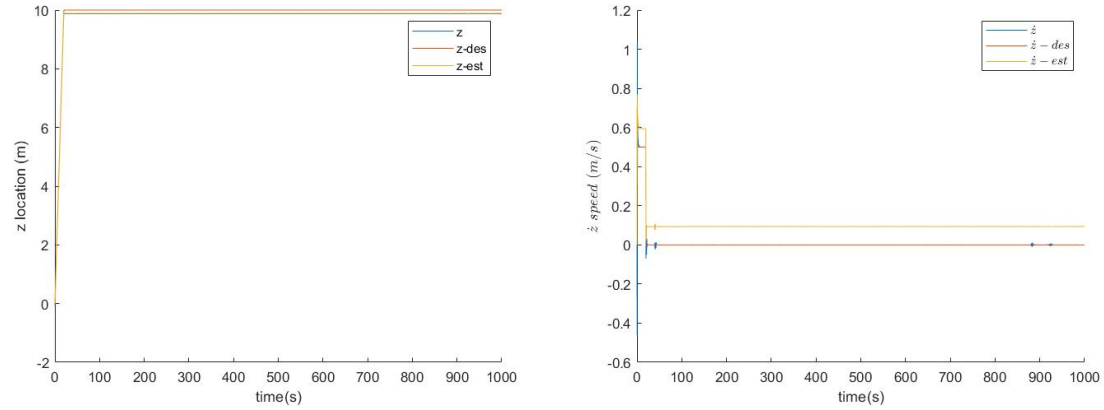


Figure 4.28 z and $\dot{z}$ states with DL Controller.

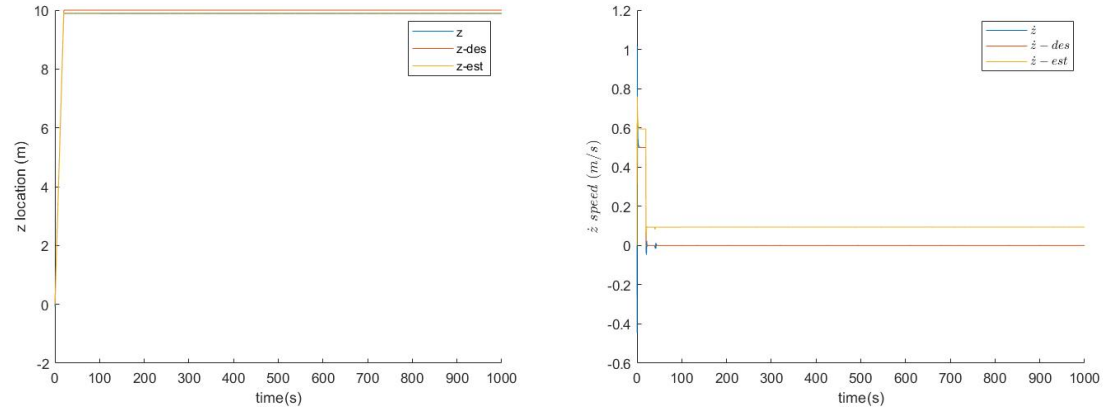


Figure 4.29 z and $\dot{z}$ states with LQG Controller.

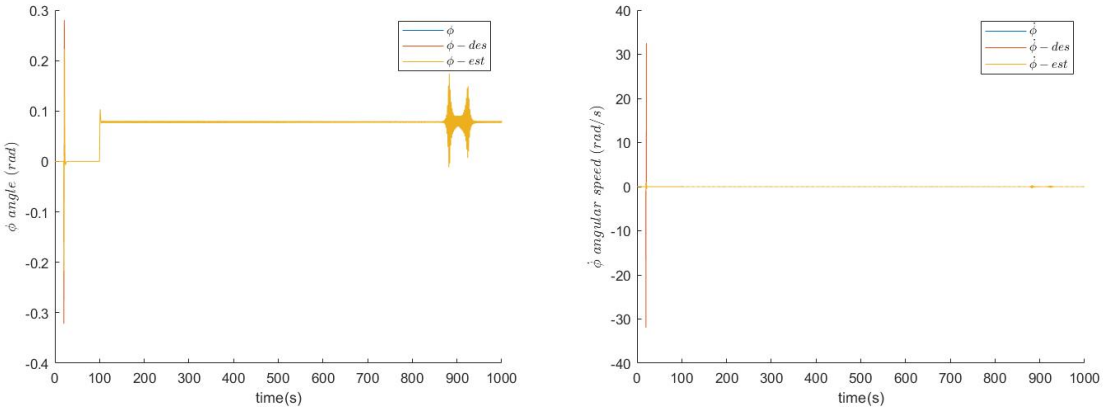


Figure 4.30 ϕ and $\dot{\phi}$ states with DL Controller.

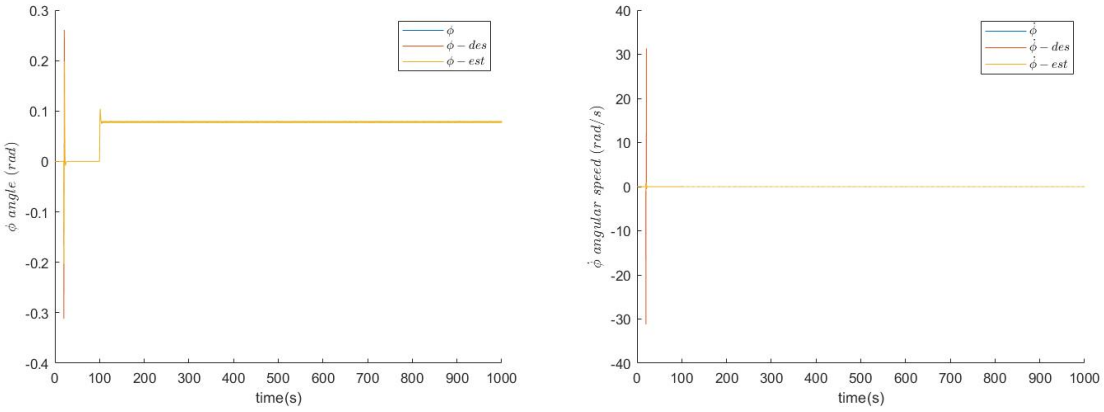


Figure 4.31 ϕ and $\dot{\phi}$ states with LQG Controller.

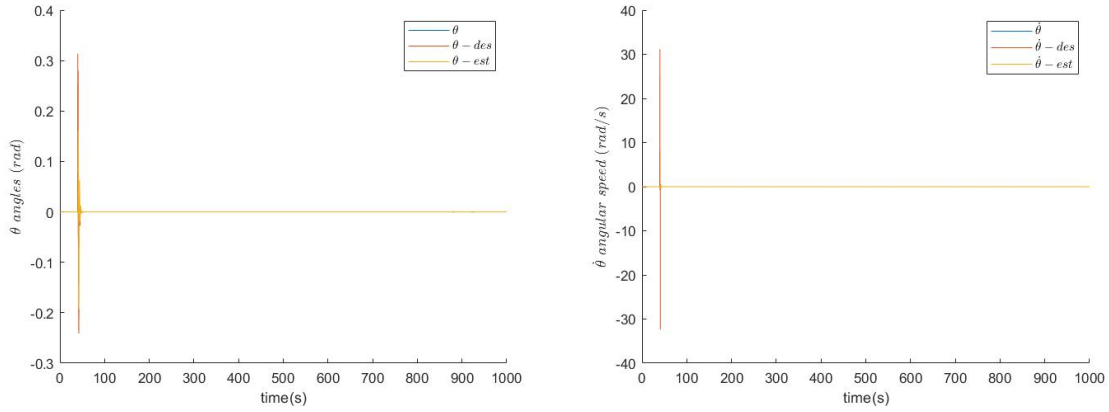


Figure 4.32 θ and $\dot{\theta}$ states with DL Controller.

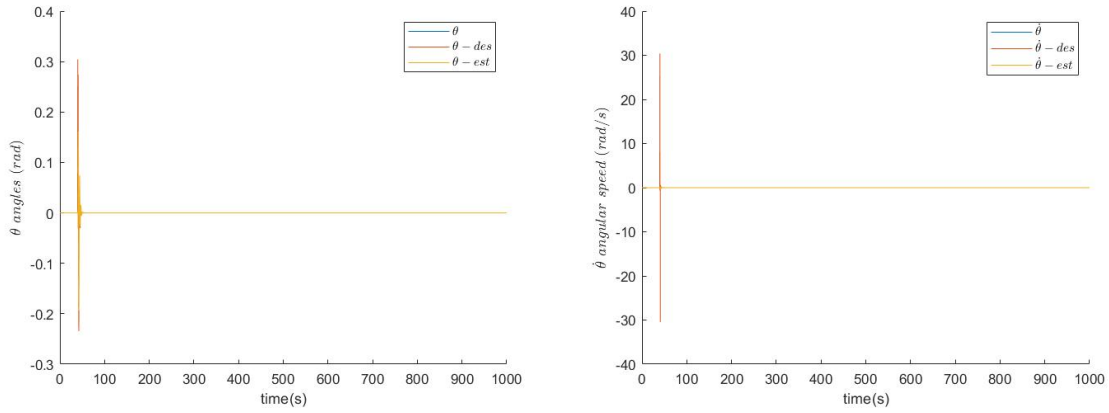


Figure 4.33 θ and $\dot{\theta}$ states with LQG Controller.

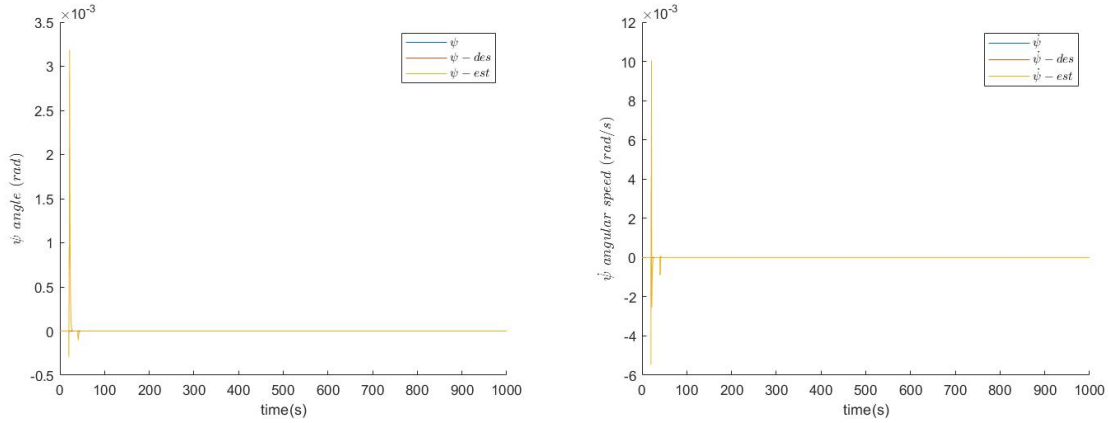


Figure 4.34 ψ and $\dot{\psi}$ states with DL Controller.

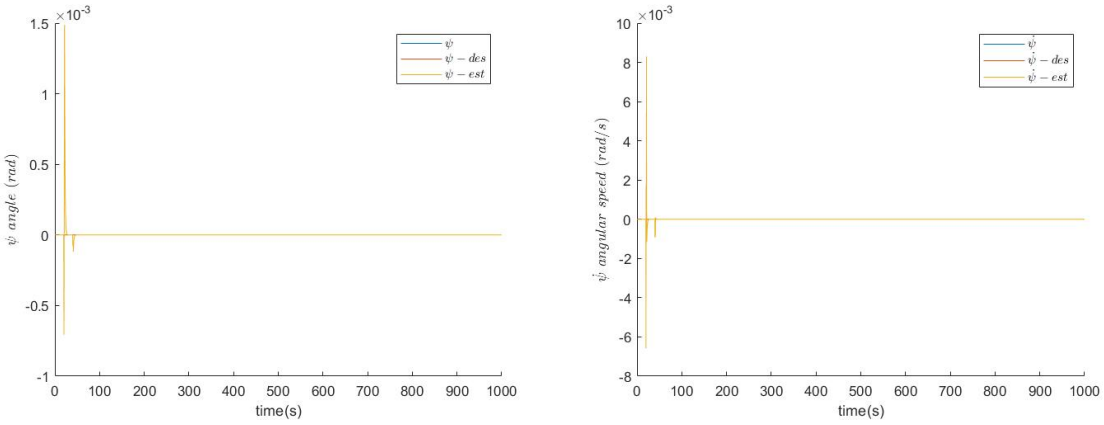
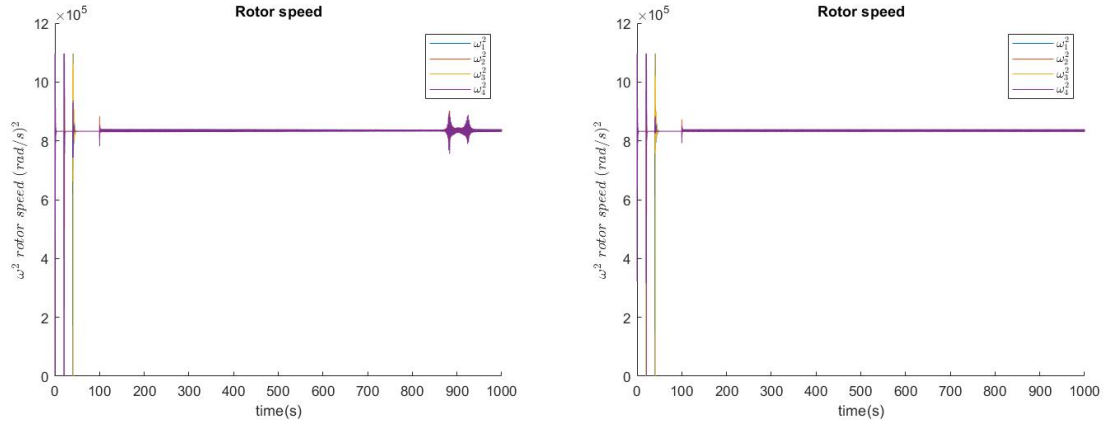


Figure 4.35 ψ and $\dot{\psi}$ states with LQG Controller.



(a) Rotor input with DL *(b)* Rotor input with LQG

Figure 4.36 Comparison of Rotor Inputs with DL vs. LQG.

As shown in Figure (4.27), the LQG controller maintains an offset from the desired position, while the DL controller pushes back trying to retrieve the system back to its desired position. The rest of the states are shown to be stable and tracking.

4.5.5. Sinusoidal Disturbance

Similarly a sine wave of magnitude 1 N and frequency 0.1 Hz is applied to the y-direction of the inertial frame as shown in Figure (4.37). Figure (4.38) shows all the weights of the actors and critics of both the inner and outer loops. Figures (4.40) to (4.51) show the comparison between the states when the controllers are either LQG or DL

controllers. Furthermore, Figure (4.52) shows the difference between controller inputs of the LQG and DL controllers.

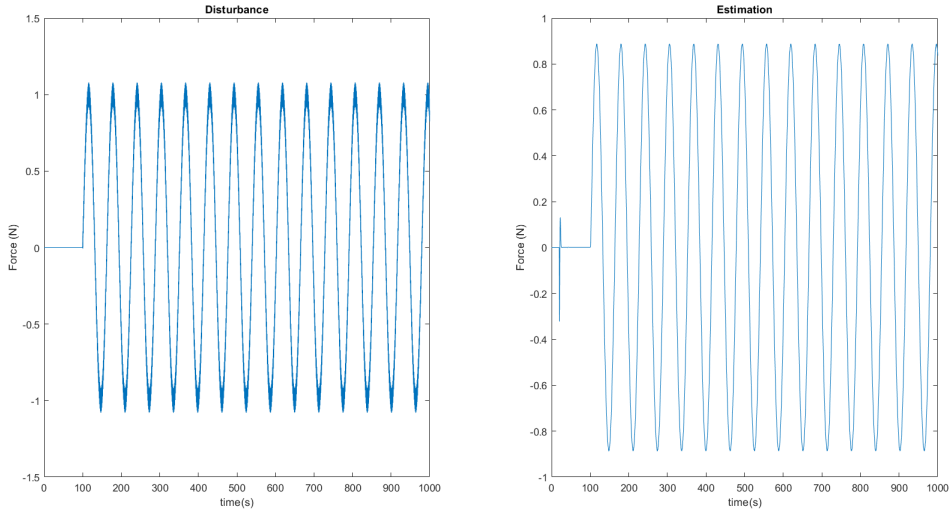


Figure 4.37 Adverse Estimation vs Actual Disturbance(Sinusoidal).

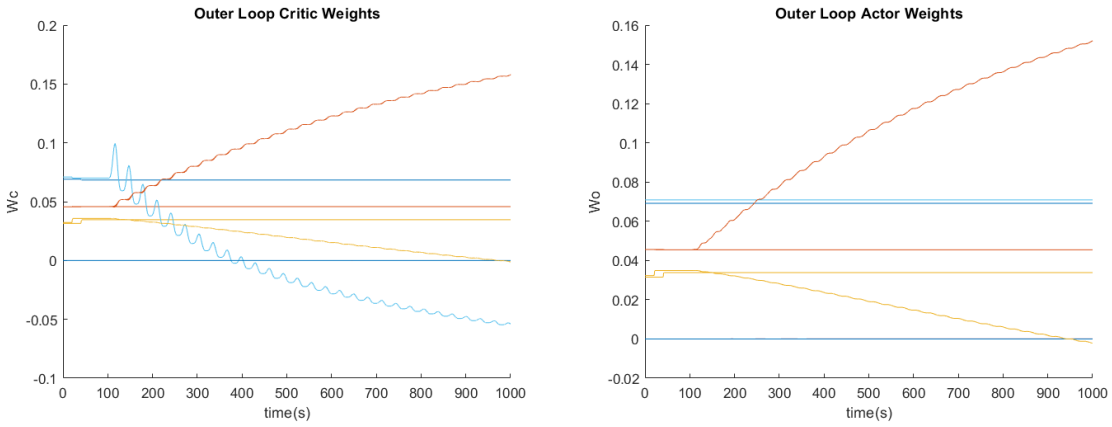


Figure 4.38 DL Outer Weights (Sinusoidal).

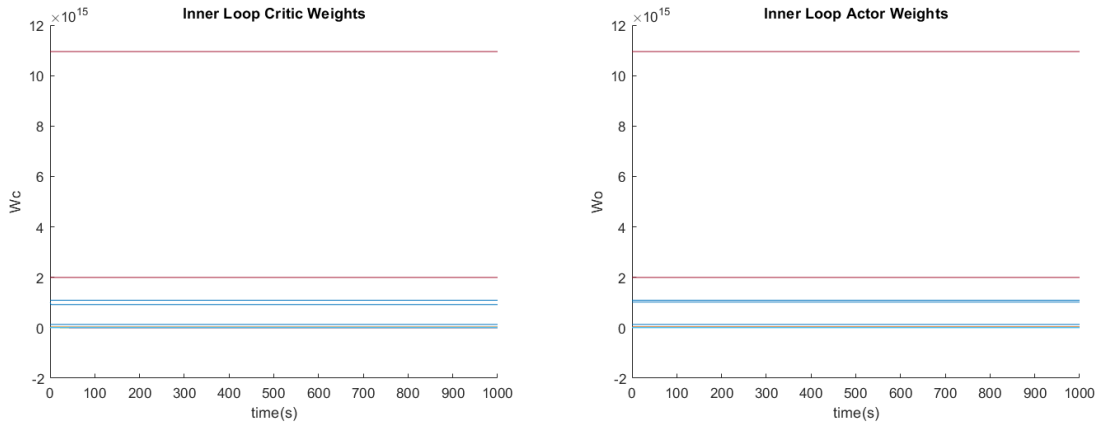


Figure 4.39 DL Inner Weights (Sinusoidal).

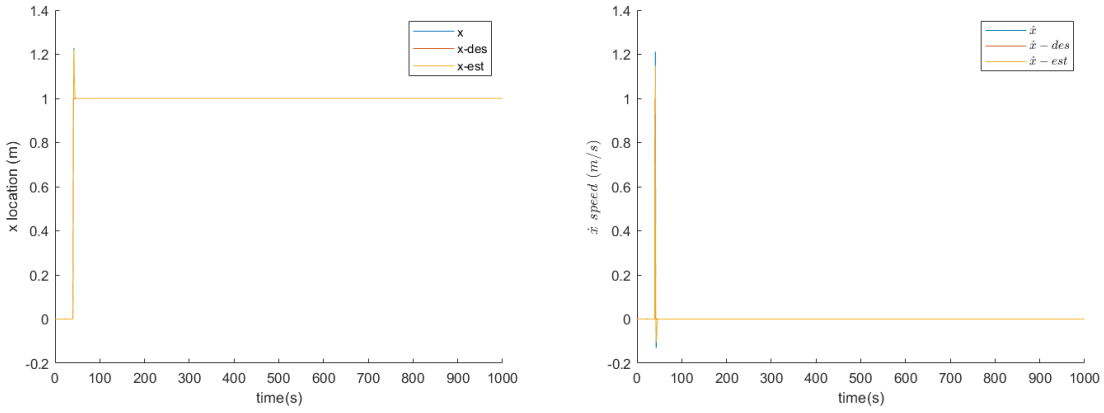


Figure 4.40 x and $\dot{x}$ states with DL Controller (Sinusoidal).

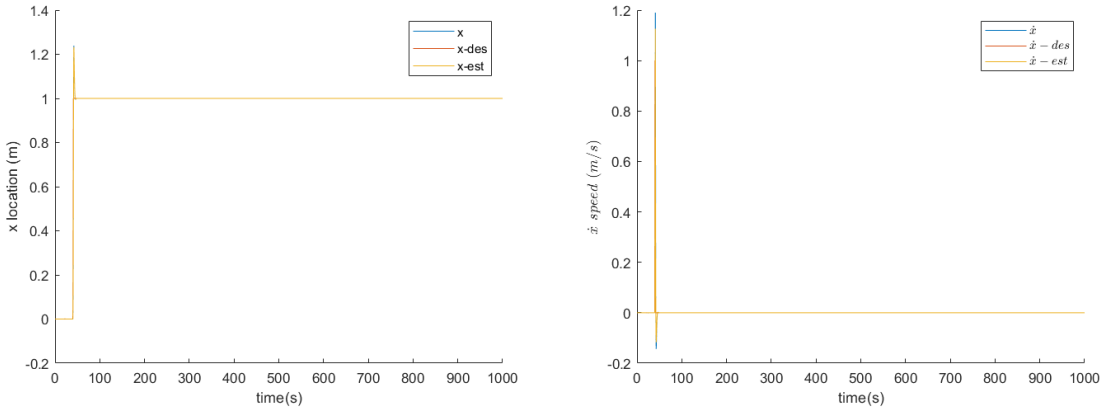


Figure 4.41 x and $\dot{x}$ states with LQG Controller (Sinusoidal).

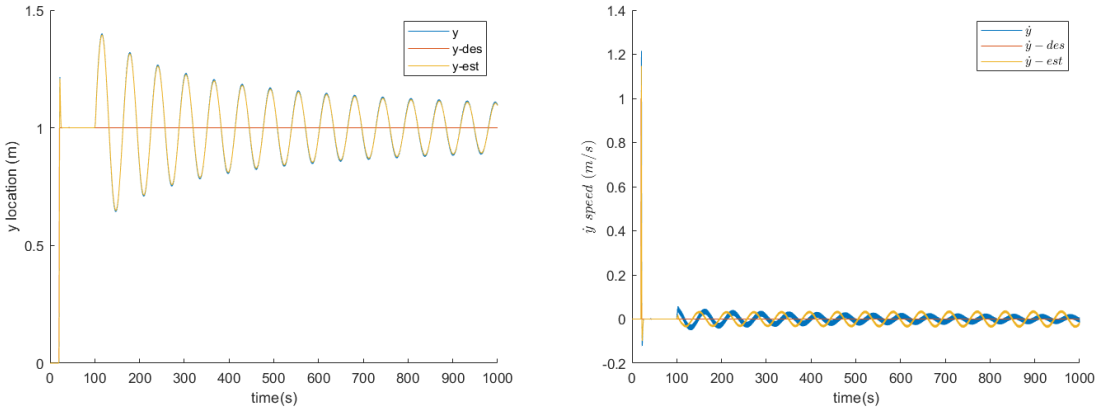


Figure 4.42 y and $\dot{y}$ states with DL Controller (Sinusoidal).

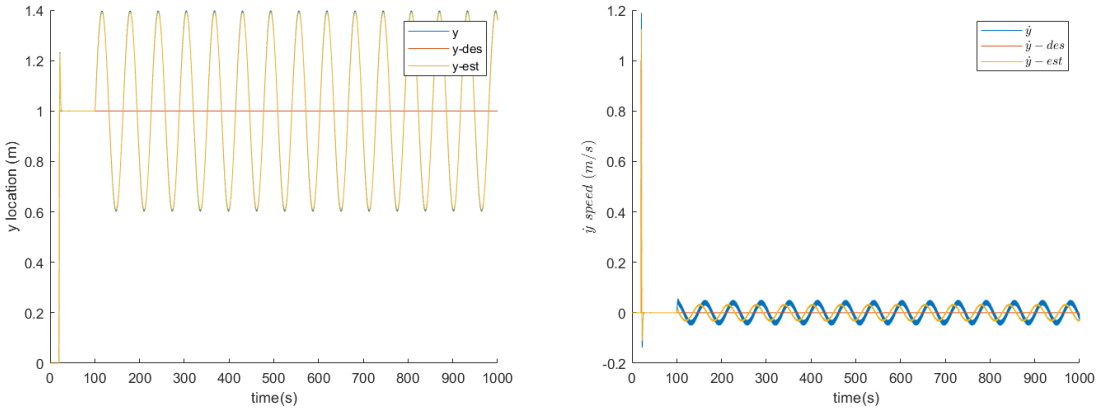


Figure 4.43 y and $\dot{y}$ states with LQG Controller (Sinusoidal).

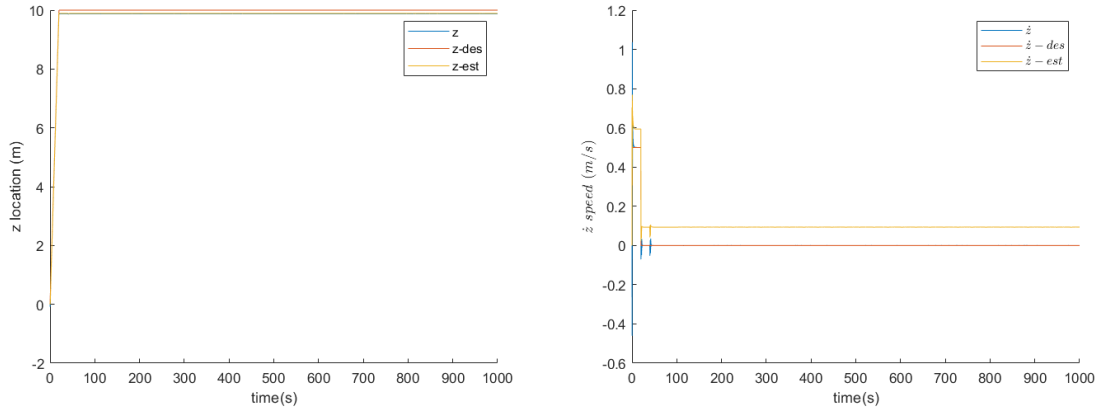


Figure 4.44 z and $\dot{z}$ states with DL Controller (Sinusoidal).

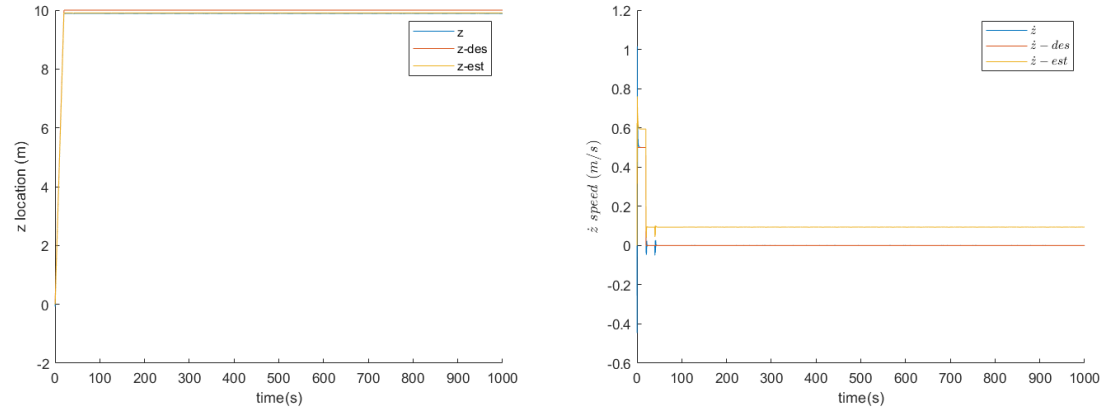


Figure 4.45 z and $\dot{z}$ states with LQG Controller (Sinusoidal).

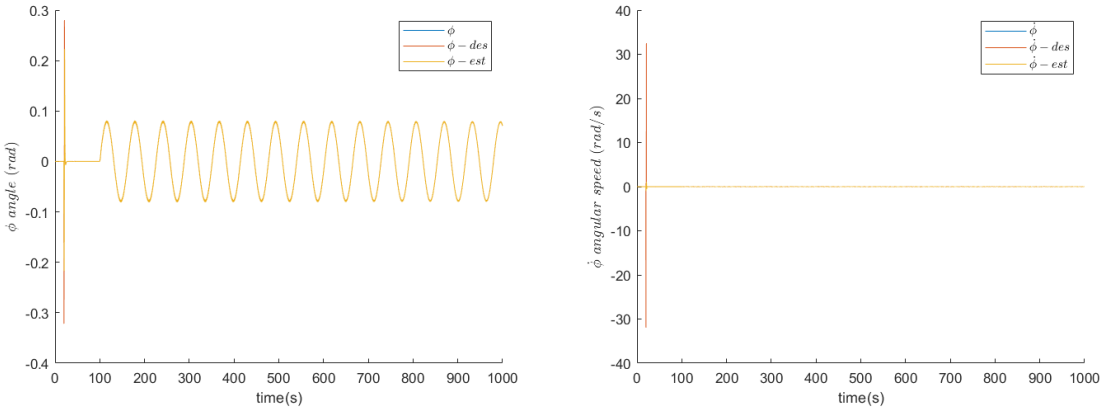


Figure 4.46 ϕ and $\dot{\phi}$ states with DL Controller (Sinusoidal).

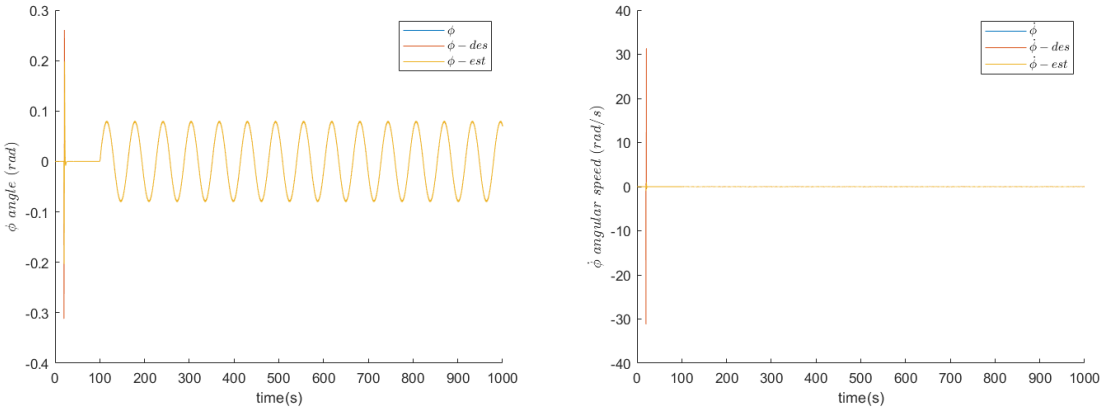


Figure 4.47 ϕ and $\dot{\phi}$ states with LQG Controller (Sinusoidal).

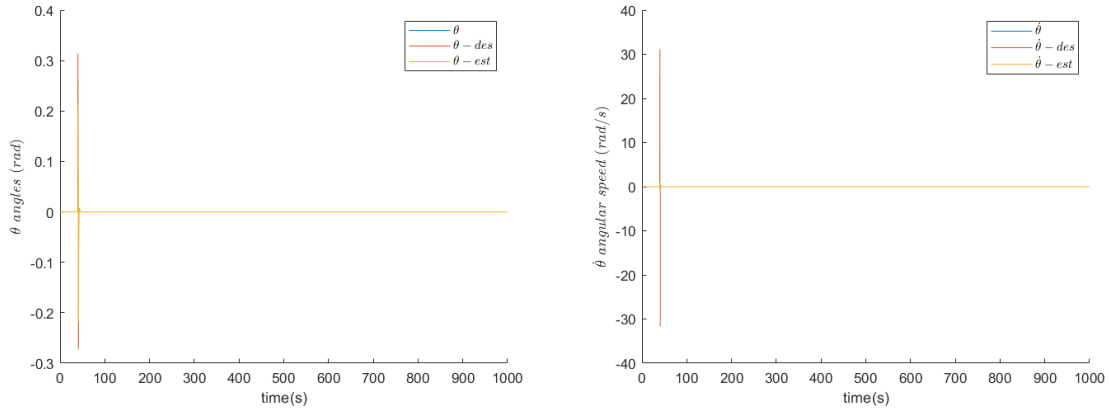


Figure 4.48 θ and $\dot{\theta}$ states with DL Controller (Sinusoidal).

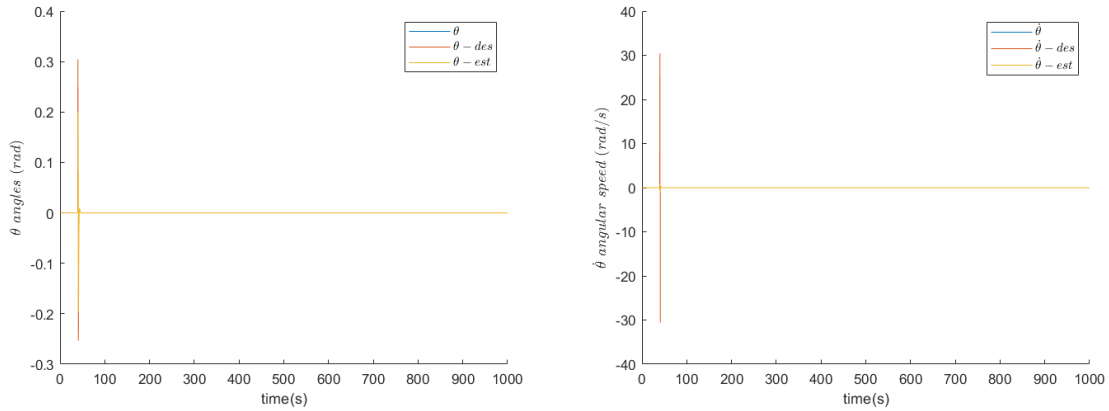


Figure 4.49 θ and $\dot{\theta}$ states with LQG Controller (Sinusoidal).

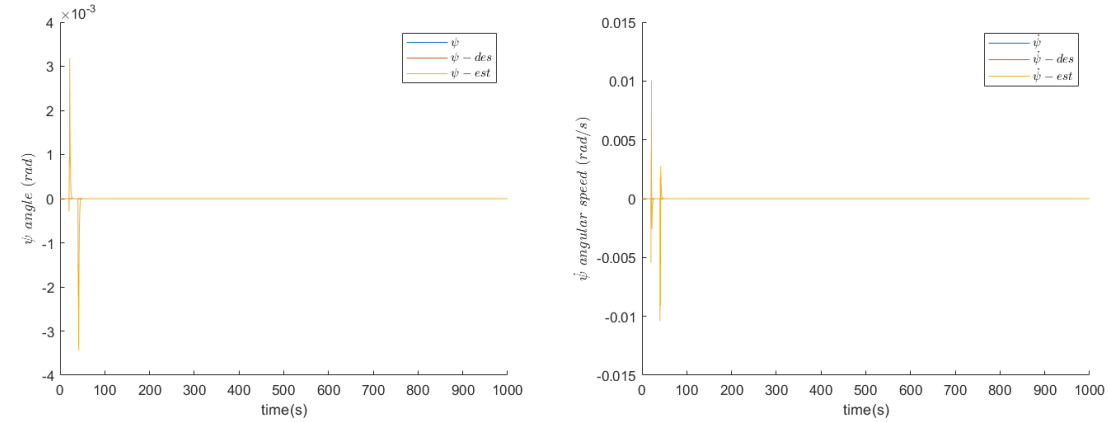


Figure 4.50 ψ and $\dot{\psi}$ states with DL Controller (Sinusoidal).

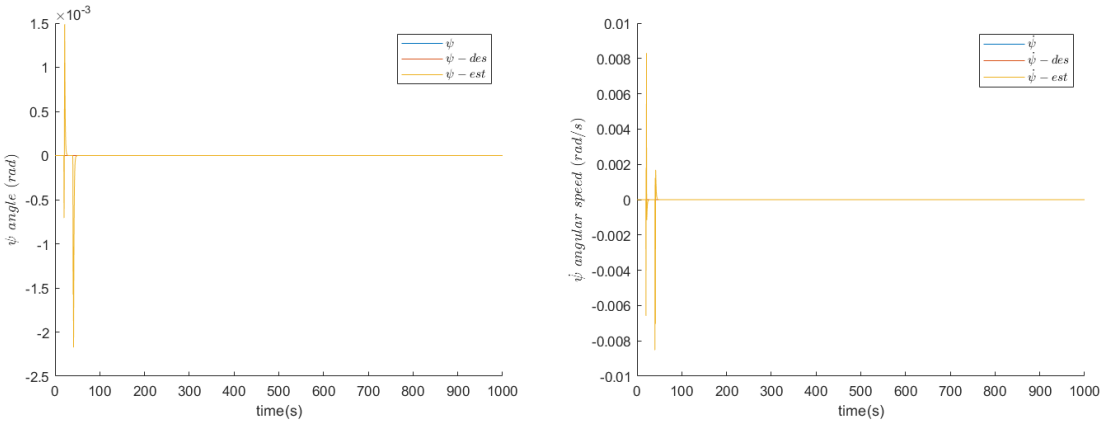
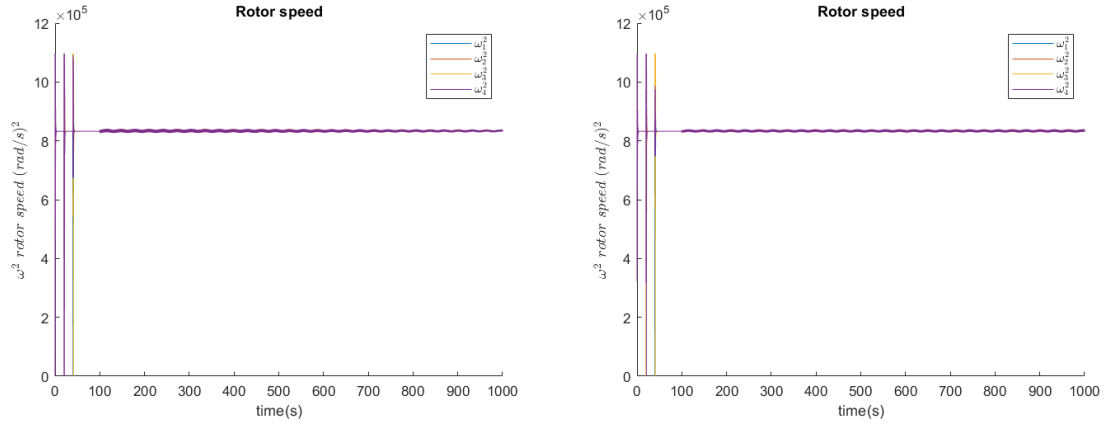


Figure 4.51 ψ and $\dot{\psi}$ states with LQG Controller (Sinusoidal).



(a) Rotor input with DL *(b)* Rotor input with LQG

Figure 4.52 Comparison of Rotor Inputs with DL vs. LQG (Sinusoidal) .

As shown in Figure (4.42), the LQG controller maintains a constant fluctuation from the desired position, while the DL controller pushes back trying to retrieve the system back to its desired position while attenuating and lowering the magnitude of the fluctuation. The rest of the states are shown to be stable and tracking.

4.5.6. Helix trajectory Analysis

A helical shape is proposed with a disturbance to d_x as shown in Figure (4.53). Figure (4.54) shows all the weights of the actors and critics of both the inner and outer loops. Figures (4.56) to (4.67) show the comparison between the states when the controllers are

either LQG or DL controllers. Furthermore, Figure (4.68) shows the difference between controller inputs of the LQG and DL controllers.

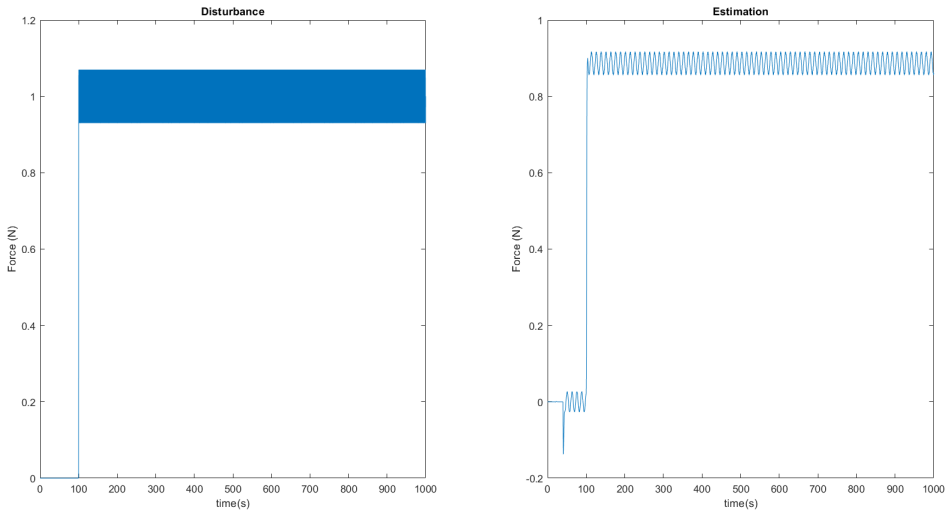


Figure 4.53 Adverse Estimation vs Actual Disturbance(HELIX).

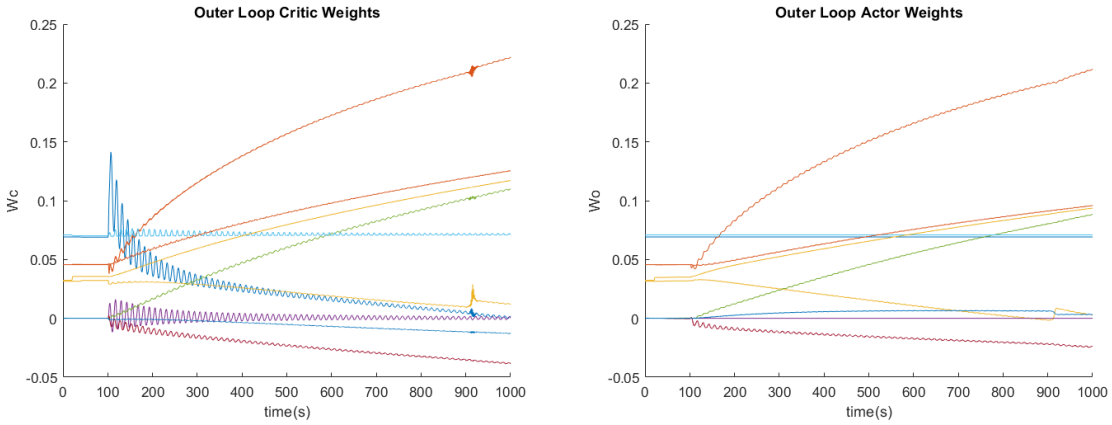


Figure 4.54 DL Outer Weights (HELIX).

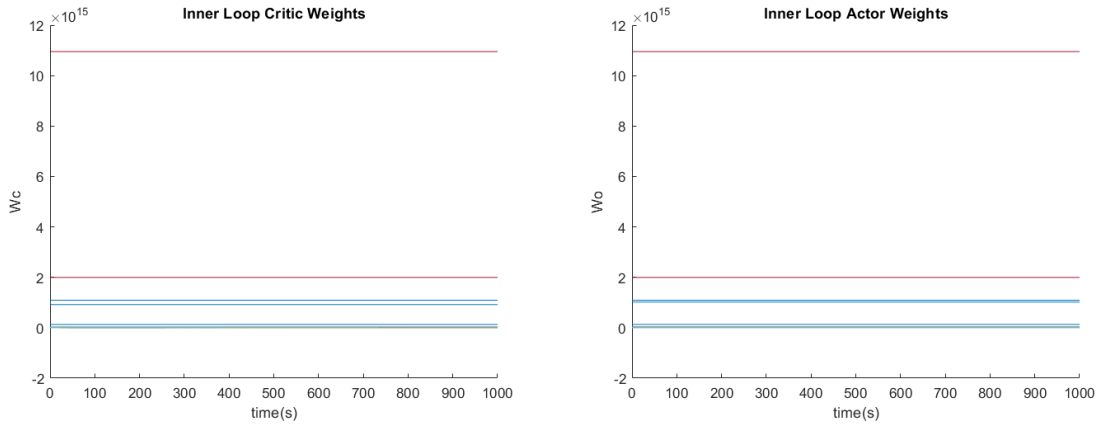


Figure 4.55 DL Inner Weights (HELIX).

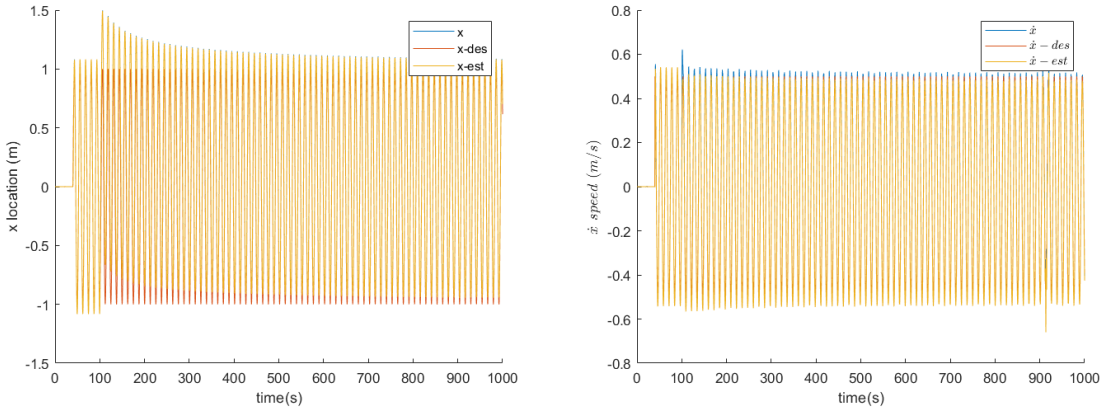


Figure 4.56 x and $\dot{x}$ states with DL Controller (HELIX).

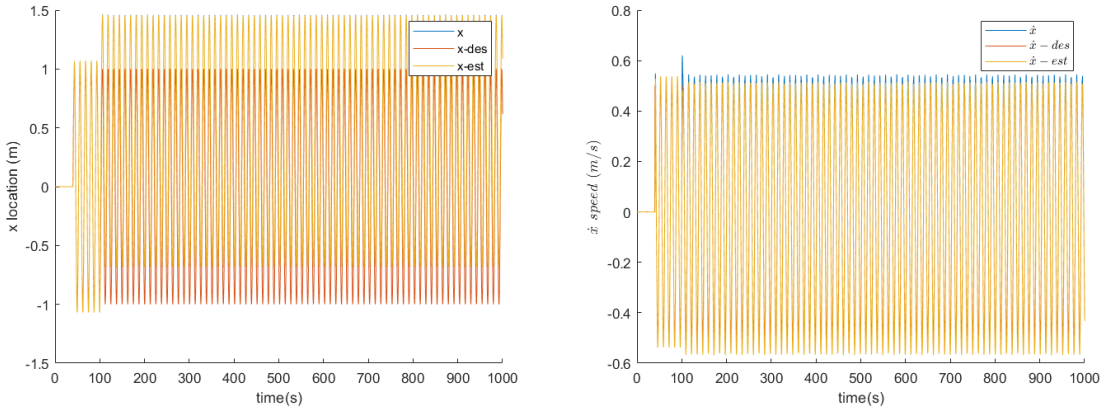


Figure 4.57 x and $\dot{x}$ states with LQG Controller (HELIX).

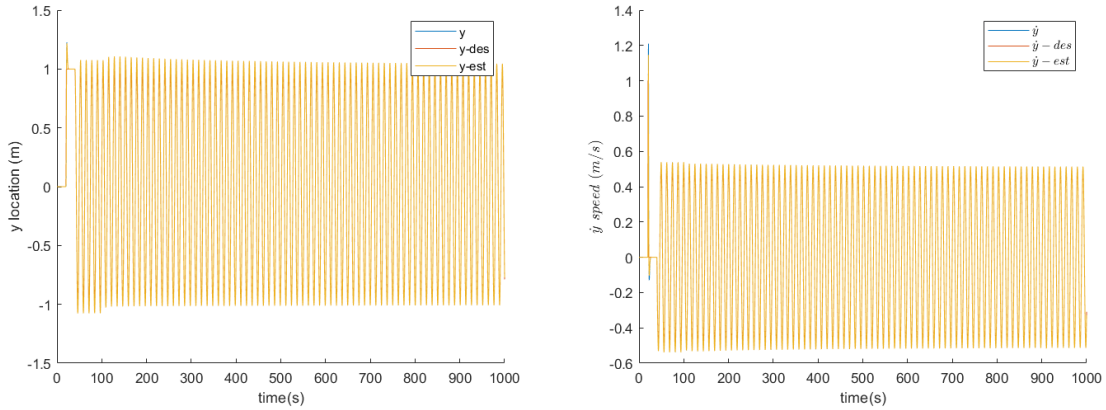


Figure 4.58 y and $\dot{y}$ states with DL Controller (HELIX).

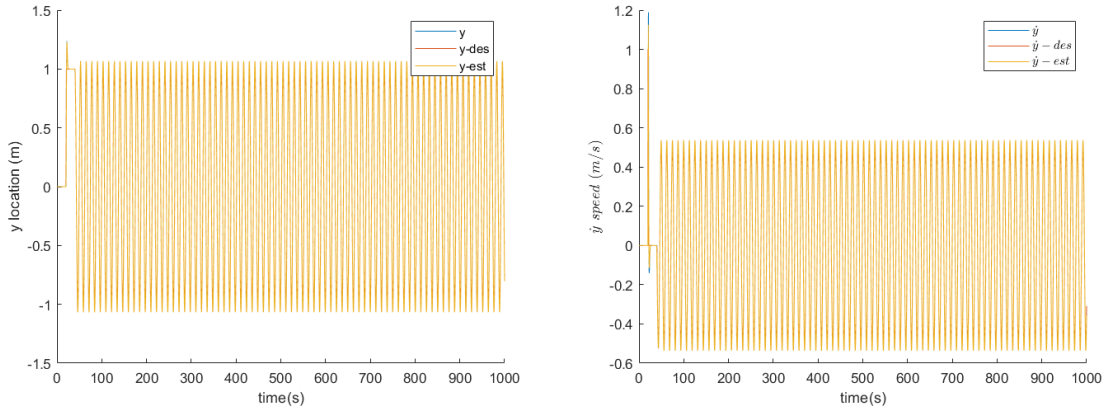


Figure 4.59 y and $\dot{y}$ states with LQG Controller (HELIX).

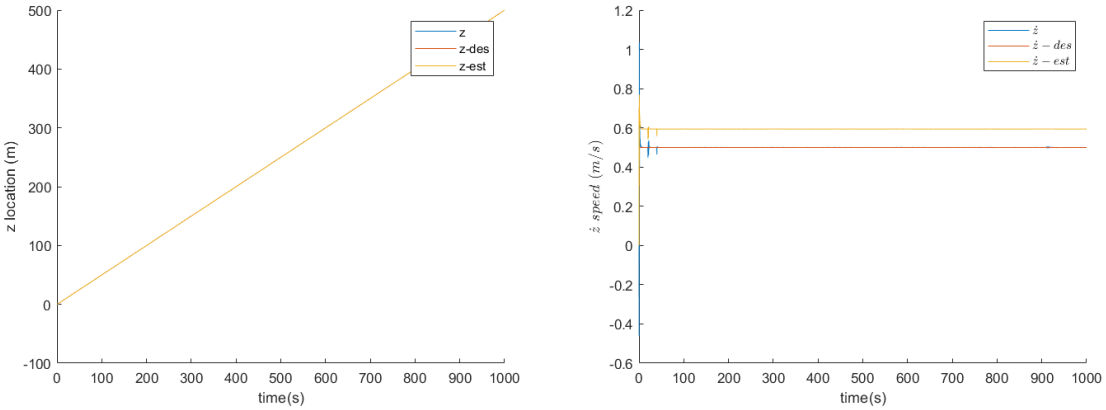


Figure 4.60 z and $\dot{z}$ states with DL Controller (HELIX).

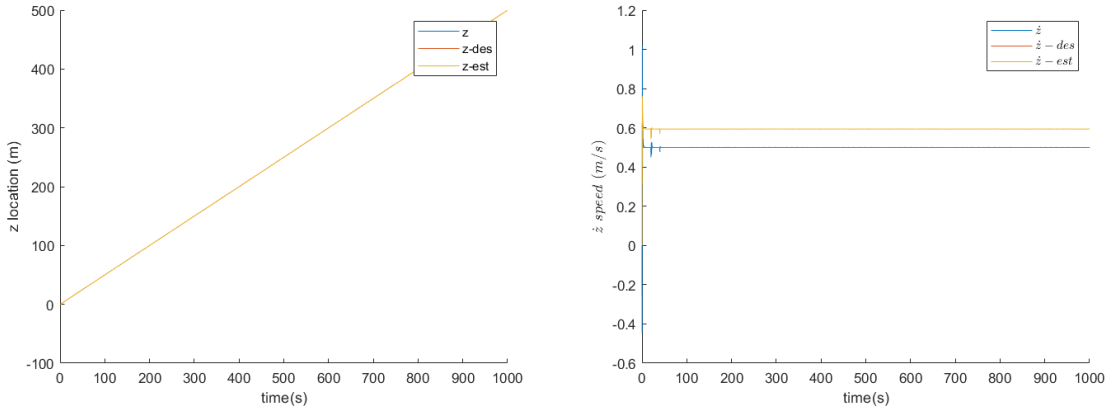


Figure 4.61 z and $\dot{z}$ states with LQG Controller (HELIX).

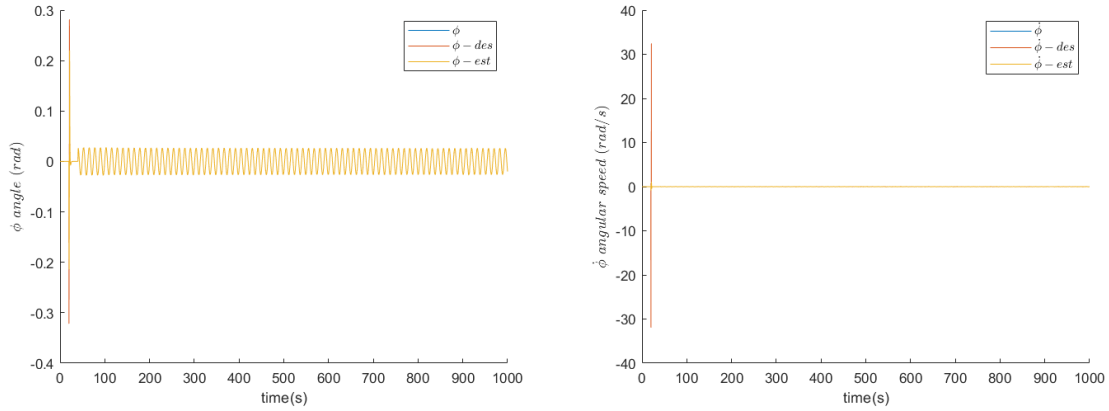


Figure 4.62 ϕ and $\dot{\phi}$ states with DL Controller (HELIX).

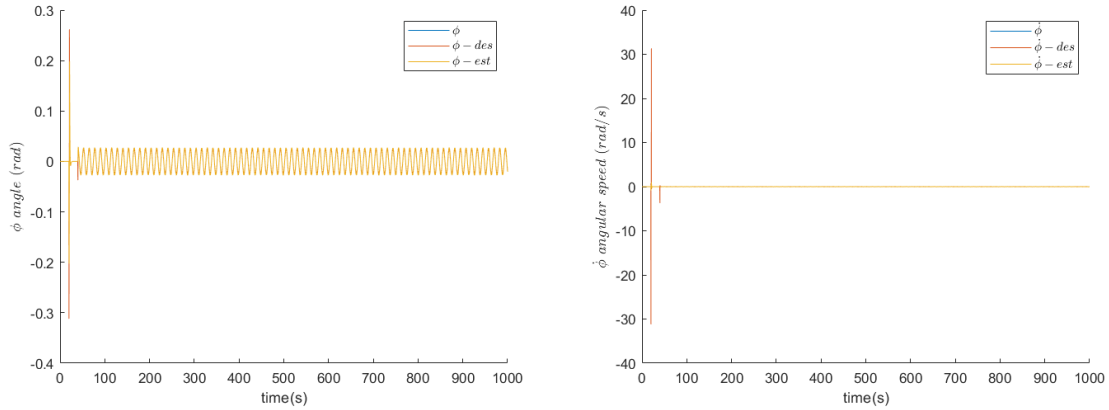


Figure 4.63 ϕ and $\dot{\phi}$ states with LQG Controller (HELIX).

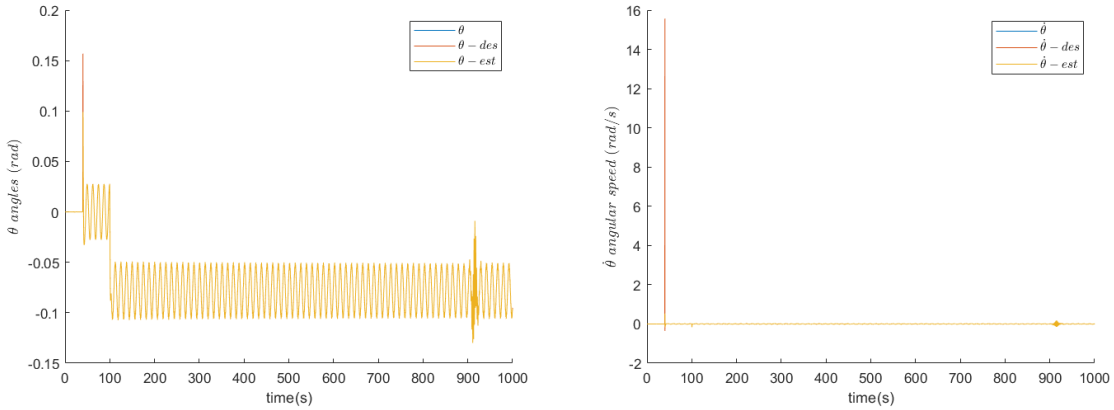


Figure 4.64 θ and $\dot{\theta}$ states with DL Controller (HELIX).

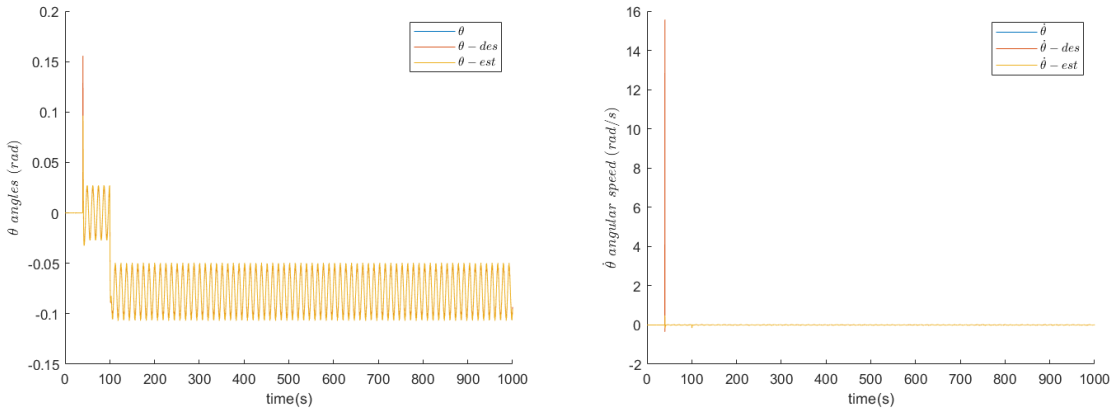


Figure 4.65 θ and $\dot{\theta}$ states with LQG Controller (HELIX).

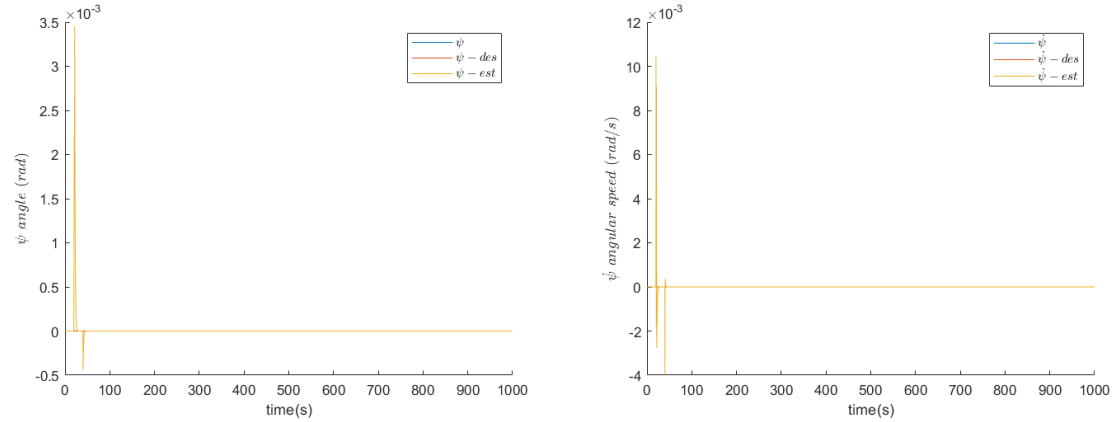


Figure 4.66 ψ and $\dot{\psi}$ states with DL Controller (HELIX).

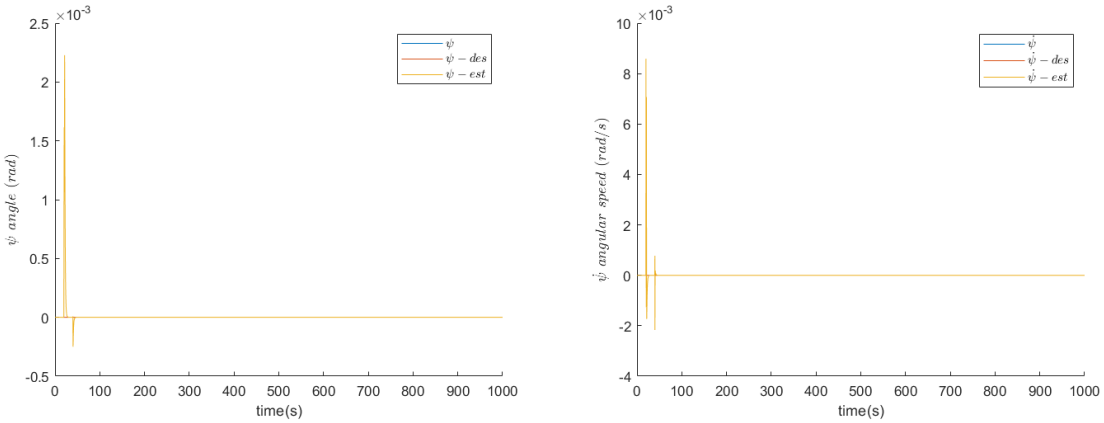
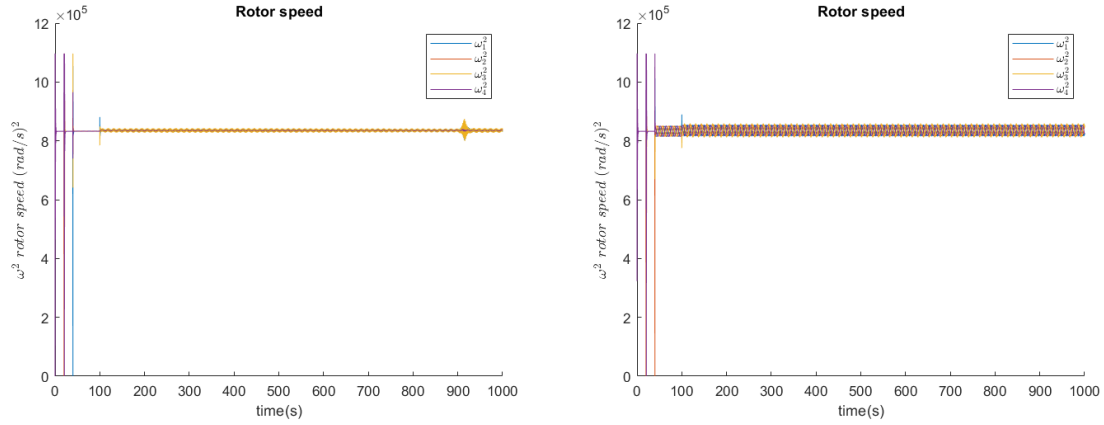


Figure 4.67 ψ and $\dot{\psi}$ states with LQG Controller (HELIX).



(a) Rotor input with DL *(b)* Rotor input with LQG

Figure 4.68 Comparison of Rotor Inputs with DL vs. LQG (HELIX) .

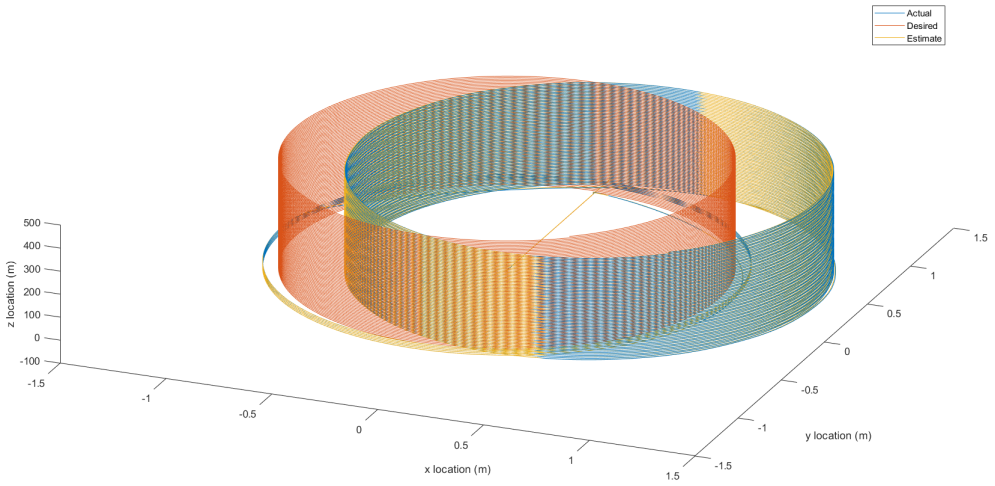


Figure 4.69 Actual, Desired, an Estimated LQG path of Helical Shape.

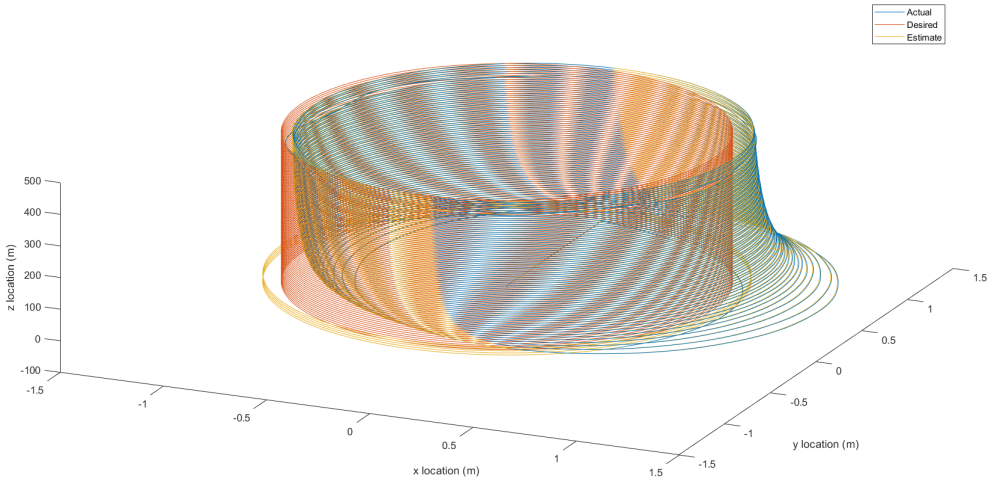


Figure 4.70 Actual, Desired, an Estimated Deep Learning path of Helical Shape.

As shown in Figure (4.69), the LQG does not reject the disturbance completely, but keeps the system at a constant offset of the helical desired path. However, as shown in Figure (4.70) the DL controller pushes back and tries to get the system back on its original desired track.

4.6. Crazyflie Quadrotor

Similarly to the previous simulations a test for the Crazyflie 2.0 is used. This mainly is different from the DJI in terms of its stability and it's high coupling of dynamic factors. This offers a great platform to test the stability of the system proposed since Crazyflie is highly unstable and can be easily drifted with any amount of force. Crazyflie dynamic constants are defined in Table (4.6) and the DL algorithm constants are shown in Table (4.7). The hovering rotor speed for the Crazyflie is $606.9384 rad/s$. Moreover, in order to initialize the DL (Aoun, 2019).

Table 4.6

Crazyflie 2.0 Parameters .

$m_{\text{quad}} = 30$ g	$I_x = 1.395 \times 10^{-5}$ kgm^2
$I_y = 1.395 \times 10^{-5}$ kgm^2	$I_z = 2.173 \times 10^{-5}$ kgm^2
$k = 1.9973 \times 10^{-7}$	$b_t = 2.4411 \times 10^{-9}$
$\omega_{\max} = 2513.27$ rad/s	$l = 40 \times 10^{-3}$ m

Table 4.7

DL constants (Crazyflie 2.0).

Inner loop Parameters										
Q=diag([$10^{10}, 10^{10}, 10^{10}, 10^{10}, 10^{10}, 10^{10}, 10^{10}, 10^{10}$])										
γ	R	α_o	α_c	α_{buff}	Q_E	R_E	G	V	T_p	T_d
10	10 $\times \mathbb{I}^{4\times4}$	5×10^{-5}	5×10^{-5}	5×10^{-5}	$10\times\mathbb{I}^{8\times8}$	$I^{8\times8}$	$I^{8\times8}$	1	-0.1 $\times \mathbb{I}^{2\times2}$	-0.1 $\times \mathbb{I}^{2\times2}$
$L_d = 10 \times [1, 10, 0, 0, 0, 0, 0, 0; 0, 0, 1, 1, 0, 0, 0, 0, ; 0, 0, 0, 0, 1, 1, 0, 0; 0, 0, 0, 0, 0, 0, 1, 1]$										
Outer loop Parameters										
Q=diag([0.1,0.1,0.1,0.1])										
γ	R	α_o	α_c	α_{buff}	Q_E	R_E	G	V	T_p	T_d
2	$I^{2\times2}$	5×10^{-3}	5×10^{-3}	5×10^{-3}	$10\times\mathbb{I}^{4\times4}$	$I^{4\times4}$	$I^{4\times4}$	0.9	-0.1 $\times \mathbb{I}^{2\times2}$	-10 $\times \mathbb{I}^{2\times2}$
$L_d = [10, 1, 0, 0; 0, 0, 10, 1]$										

The initialization of the Critic and Actor weights using the Table (4.7) inputs resulted in the following optimal initial weights.

$$\mathbf{W}_{co} = \begin{bmatrix} 0.1282 \\ 0.0654 \\ 0.0413 \\ -8.1529 \times 10^{-18} \\ -3.0413 \times 10^{-18} \\ 0.1282 \\ 9.0828 \times 10^{-18} \\ -1.1436 \times 10^{-17} \\ 0.0645 \\ 0.0413 \end{bmatrix}, \quad \mathbf{W}_{ci} = \begin{bmatrix} 2.3979e+10 \\ 4.7499e+10 \\ 5.6948e+10 \\ -7.2817e-05 \\ -5.845e-05 \\ 1.0383e+10 \\ 3.3467e-06 \\ 5.236e-06 \\ 7.8089e+8 \\ 4.0540e+8 \\ 1.6063e-05 \\ 1.7887e-05 \\ 3.3329e-05 \\ 1.6559e-06 \\ 1.0383e+10 \\ -1.1233e-06 \\ -2.4833e-07 \\ -2.4833e-07 \\ 9.6055e-07 \\ 7.8089e+8 \\ 4.0540e+8 \\ -1.7875e-07 \\ 5.4532e-06 \\ 1.1677e-05 \\ 4.5870e-07 \\ 4.8760e-06 \\ 1.9662e-07 \\ 1.1320e+10 \\ 6.3776e-06 \\ 7.8953e-06 \\ 3.4298e-06 \\ 1.5340e-07 \\ 1.1757e-07 \\ 2.8149e+9 \\ 1.5933e+9 \end{bmatrix}$$

A helical path is proposed and a disturbance is of magnitude $0.05N$ in the d_x of the x-direction as shown in Figure (4.71). Figure (4.72) shows all the weights of the actors and critics of both the inner and outer loops. Figures (4.74) to (4.85) show the comparison between the states when the controllers are either LQG or DL controllers. Furthermore, Figure (4.86) shows the difference between controller inputs of the LQG and DL controllers.

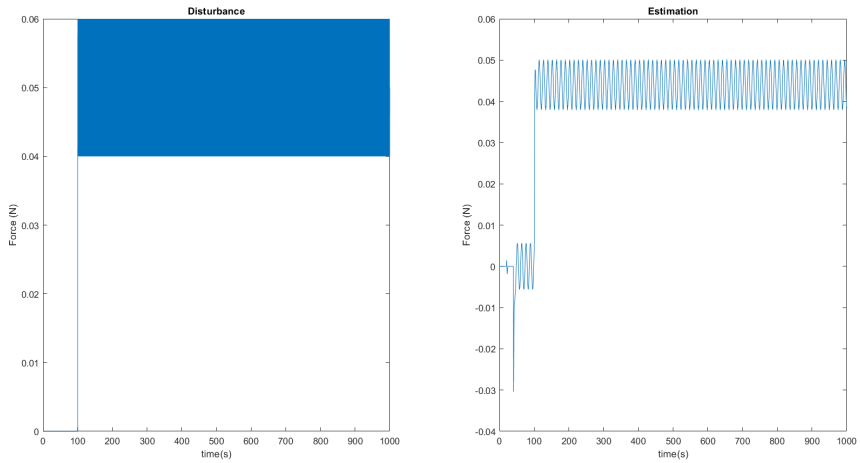


Figure 4.71 Adverse Estimation vs Actual Disturbance (Crazyflie).

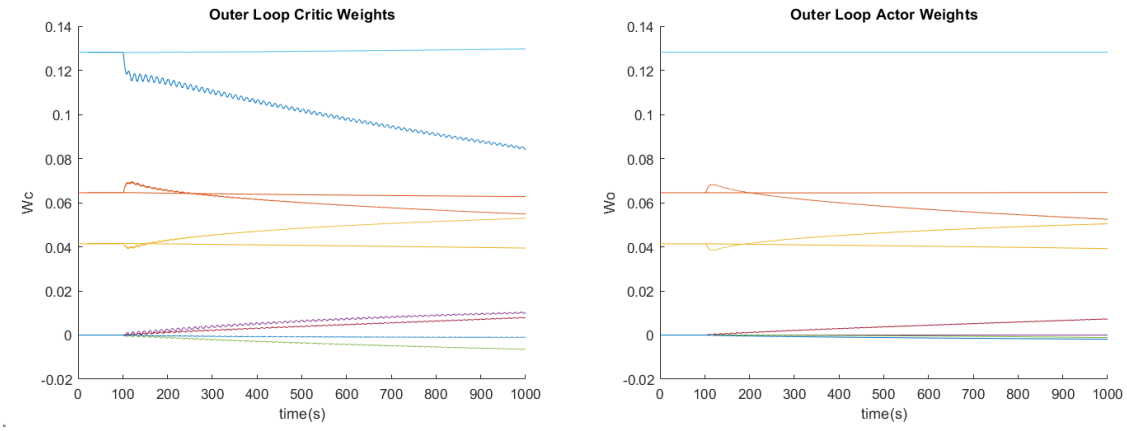


Figure 4.72 DL Outer Weights (Crazyflie).

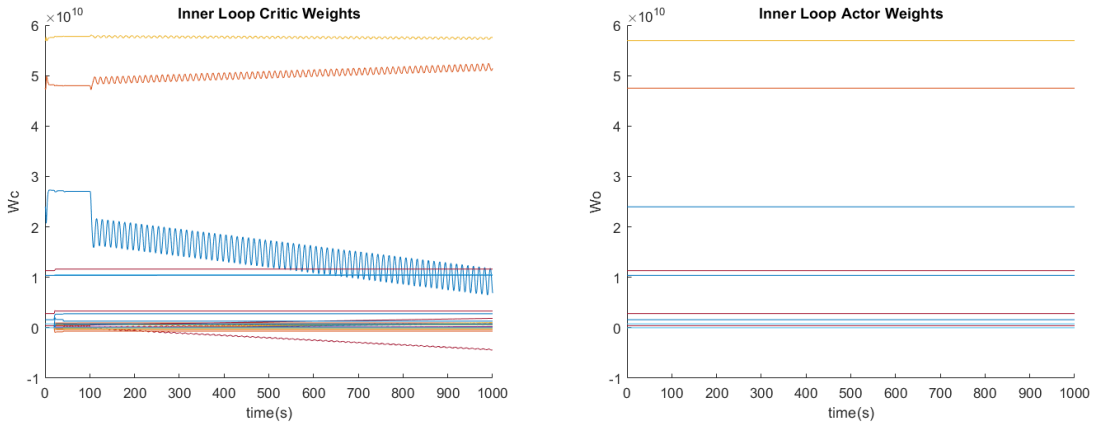


Figure 4.73 DL Inner Weights (Crazyflie).

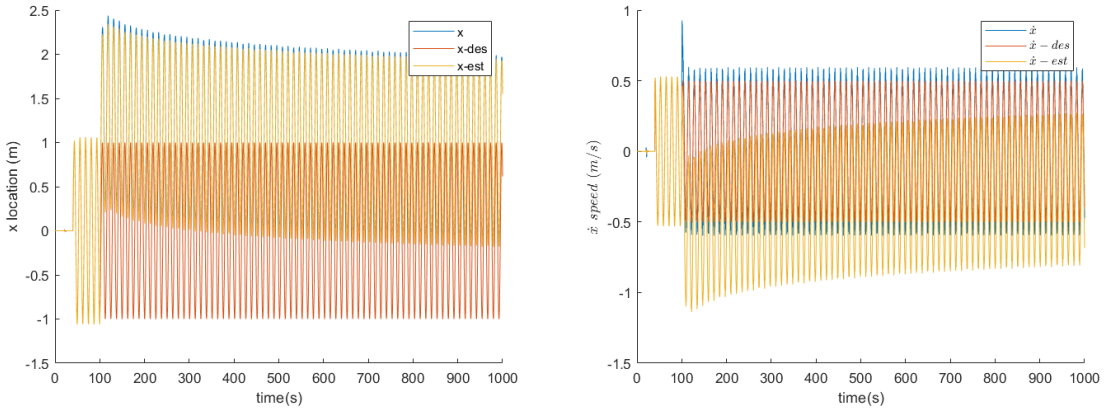


Figure 4.74 x and $\dot{x}$ states with DL Controller (Crazyflie).

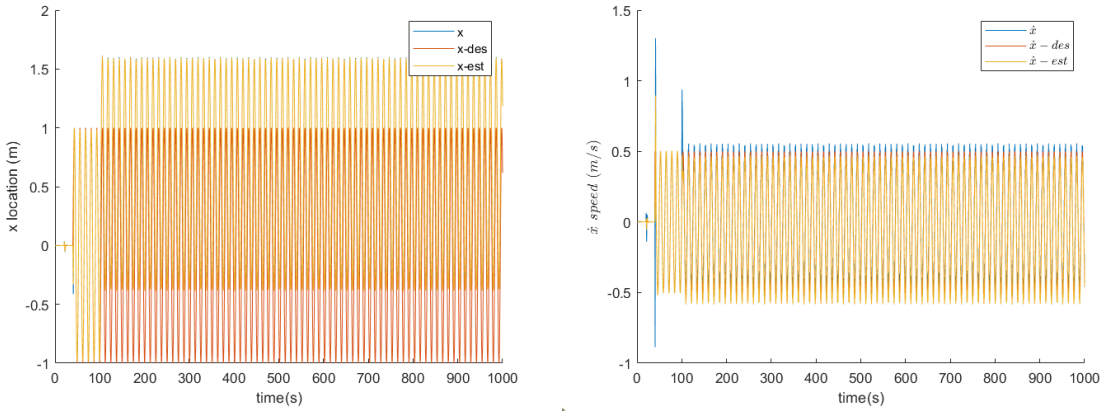


Figure 4.75 x and $\dot{x}$ states with LQG Controller (Crazyflie).

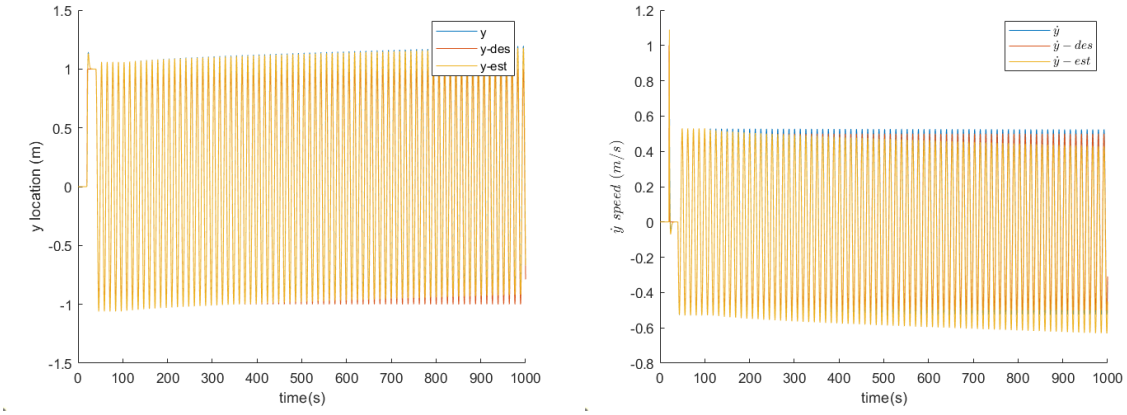


Figure 4.76 y and $\dot{y}$ states with DL Controller (Crazyflie).

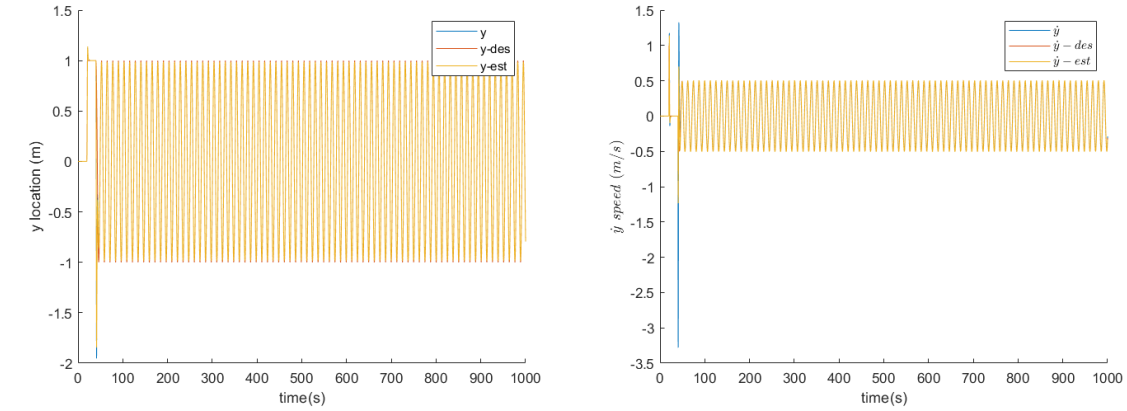


Figure 4.77 y and $\dot{y}$ states with LQG Controller (Crazyflie).

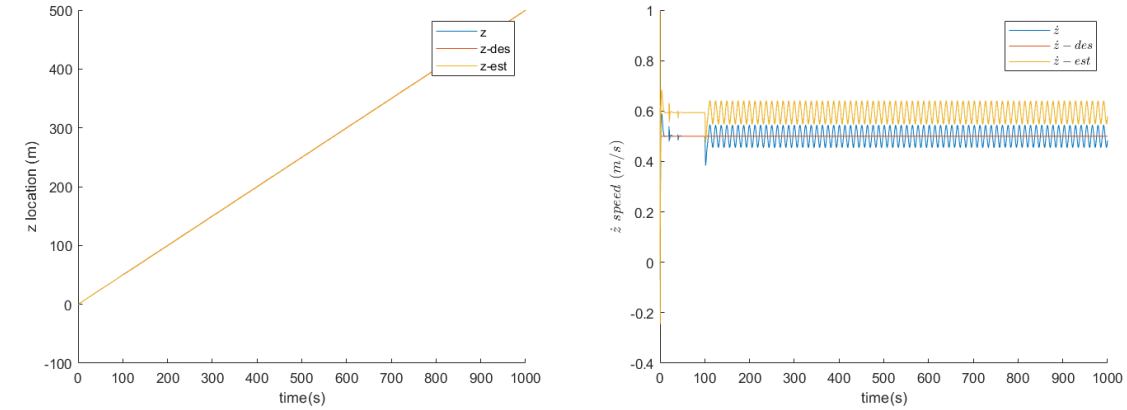


Figure 4.78 z and $\dot{z}$ states with DL Controller (Crazyflie).

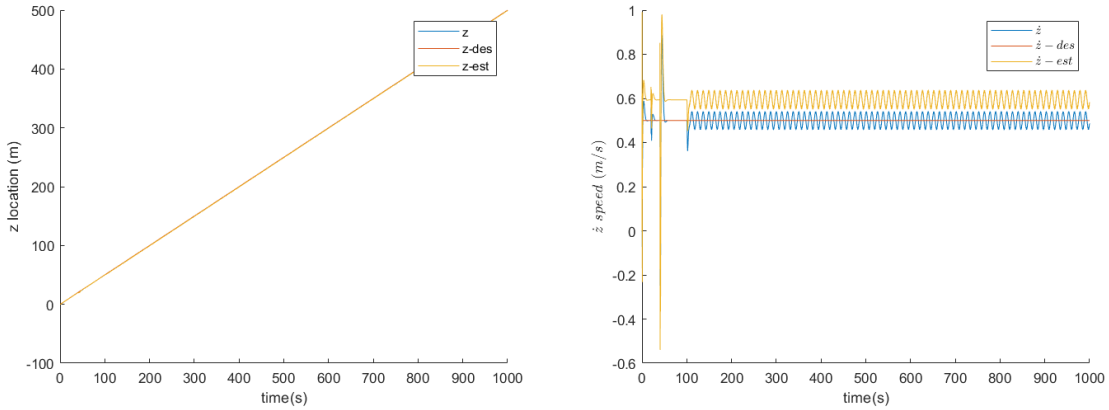


Figure 4.79 z and $\dot{z}$ states with LQG Controller (Crazyflie).

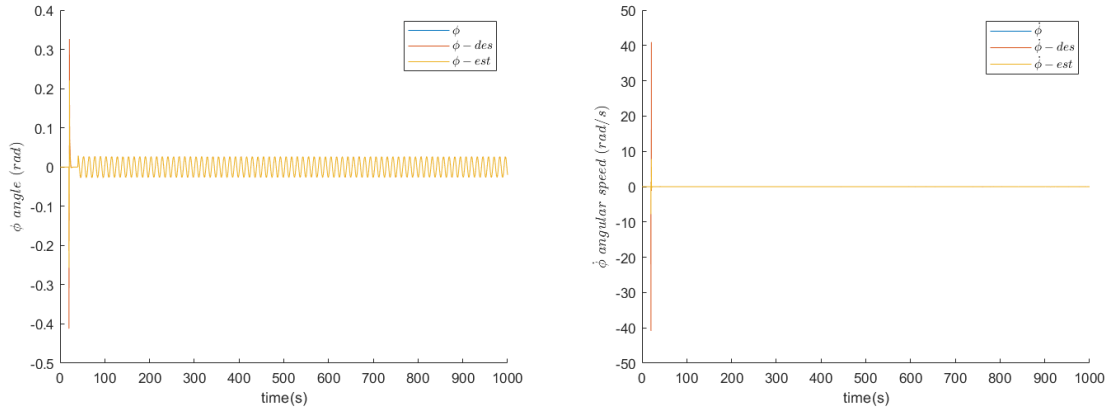


Figure 4.80 ϕ and $\dot{\phi}$ states with DL Controller (Crazyflie).

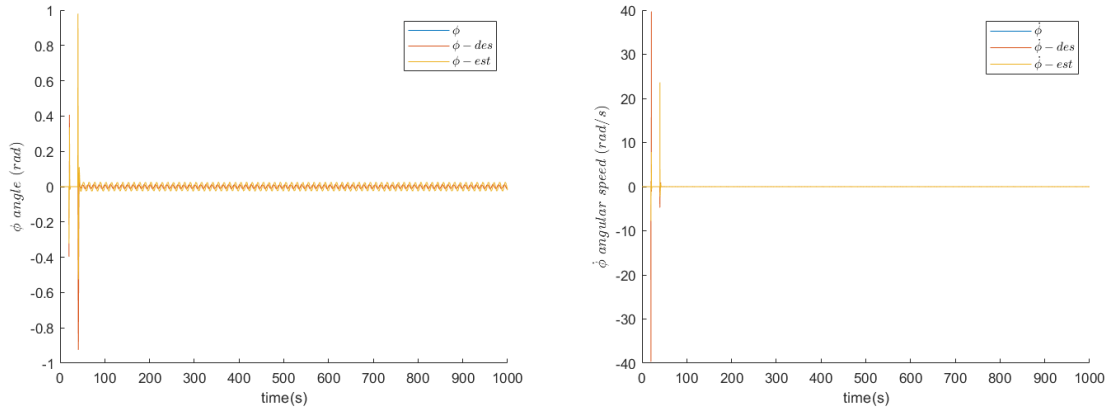


Figure 4.81 ϕ and $\dot{\phi}$ states with LQG Controller (Crazyflie).

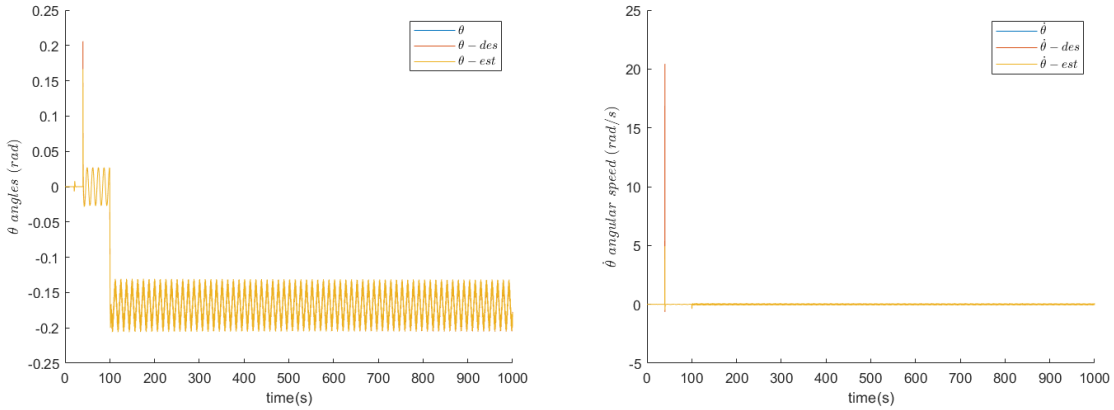


Figure 4.82 θ and $\dot{\theta}$ states with DL Controller (Crazyflie).

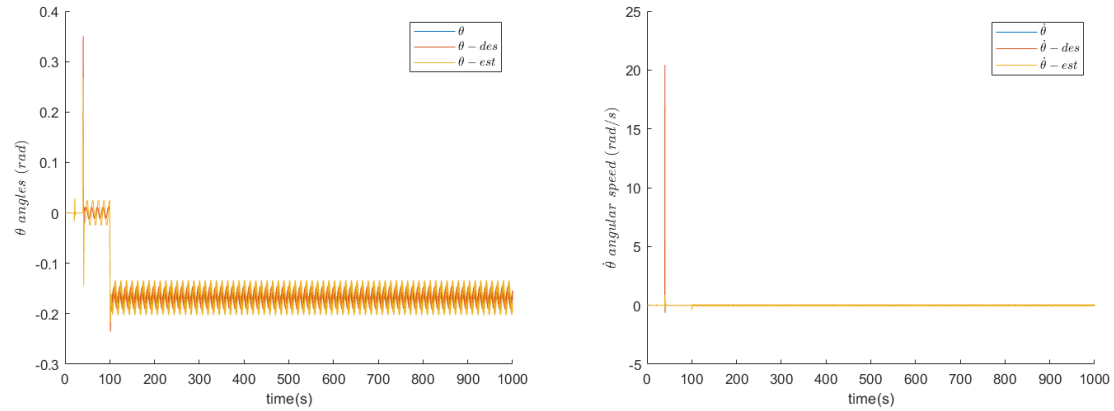


Figure 4.83 θ and $\dot{\theta}$ states with LQG Controller (Crazyflie).

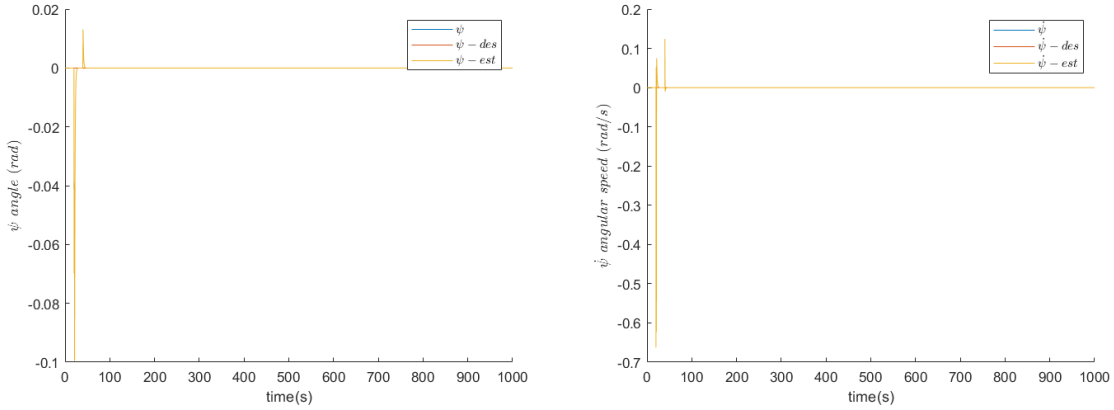


Figure 4.84 ψ and $\dot{\psi}$ states with DL Controller (Crazyflie).

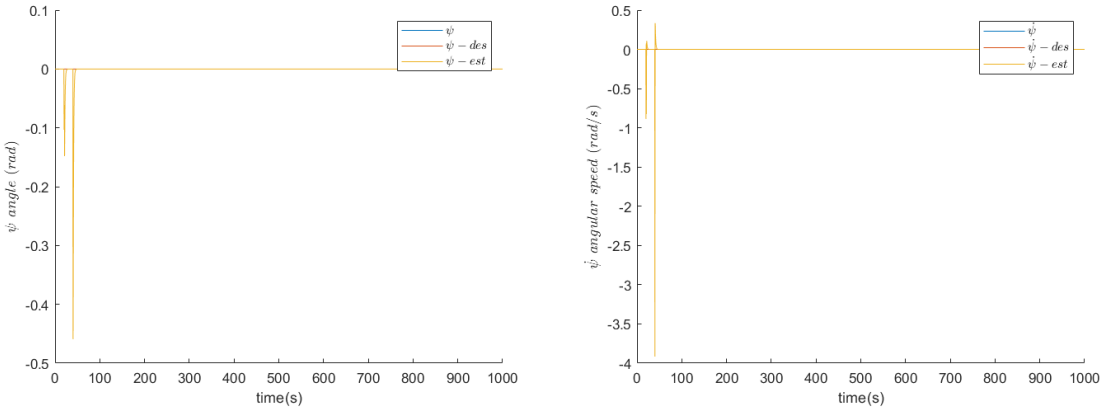
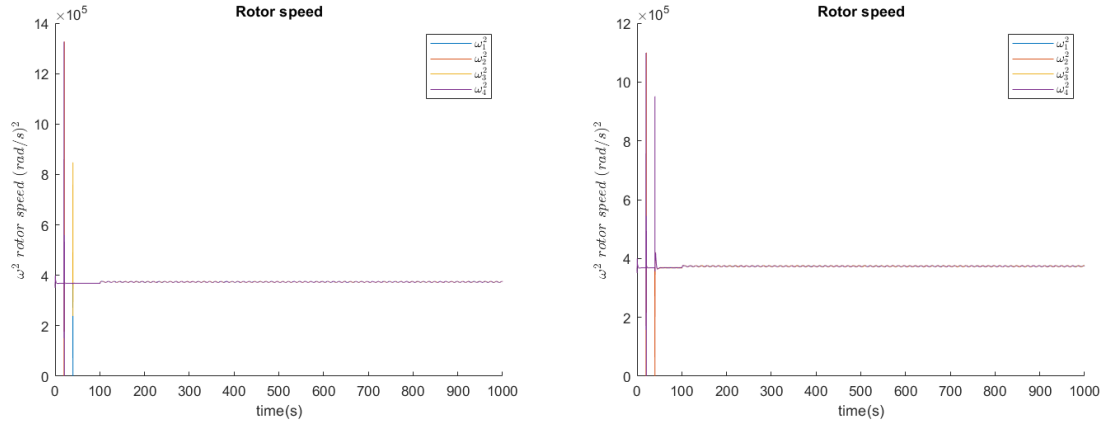


Figure 4.85 ψ and $\dot{\psi}$ states with LQG Controller (Crazyflie).



(a) Rotor input with DL *(b)* Rotor input with LQG

Figure 4.86 Comparison of Rotor Inputs with DL vs. LQG (Crazyflie) .

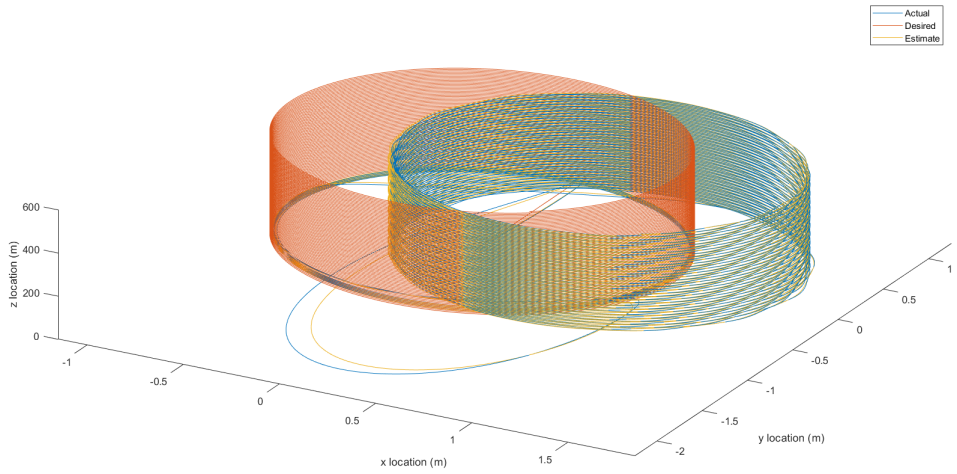


Figure 4.87 Actual, Desired, an Estimated LQG path of Helical Shape for Crazyflie.

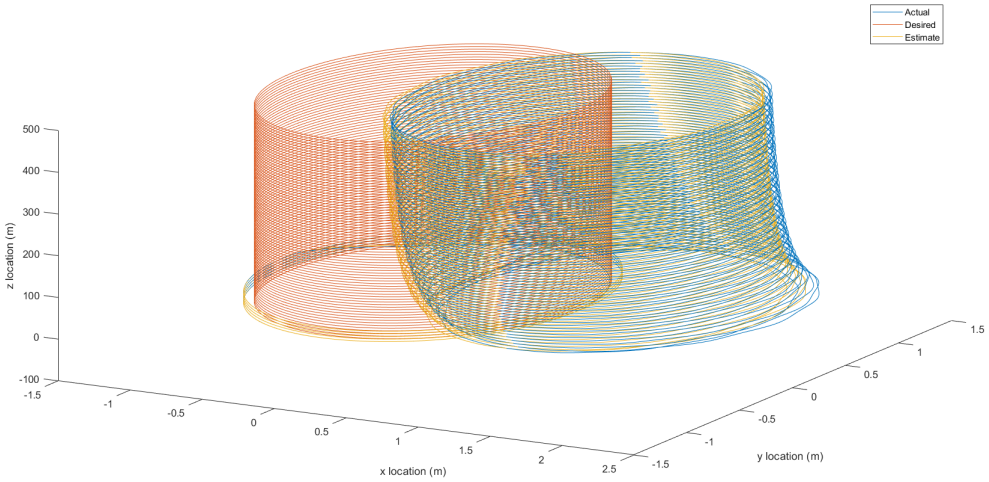


Figure 4.88 Actual, Desired, an Estimated DL path of Helical Shape for Crazyflie.

As shown in Figure (4.87), the LQG does not reject the disturbance completely, but keeps the system at a constant offset of the helical desired path. However, as shown in Figure (4.88) the DL controller pushes back and tries to get the system back on its original desired track.

It is also important to note how the LQG controller in Figure (4.87) digressed significantly from the path even in the y-direction when struck by the wind force which

resulted in having circular paths way off the desired path, while the DL in Figure (4.88) did not undergo this chaotic behaviour.

5. Conclusions and Future Work

With the results provided in Chapter (4), it is safe to say that the DL algorithm for controls has proven to be empirically stable and capable of disturbance rejection. This established the basis for later endeavours which involve DL. It presented an algorithm which is optimal, adaptable, and robust.

In this research a Fault Tolerancing Deep learning algorithm was studied which is based off of a Actor-Critic-Adverse System. This system is updated and optimized using concepts from optimal control, robust control, and game theory. It's adaptability law is done using gradient descent methods. This proposed algorithm was tested on various dynamical systems ranging from unstable to stable systems. As such, it has proven that the system is effective if dealt with in a stable or stabilized environment. Otherwise, an initial stable conditioning should be established. Moreover, it has proved to reject disturbances both as a controller or as an augmentation algorithm to a stabilizing controller.

Due to the necessity for total observability to the system dynamics including adverse inputs, states, and their respective dynamics, a Kalman Filter based observer was successful in estimating said requirements. A UIO is used which ensures that the system maintains fidelity to the system dynamics and states status especially after linearization.

5.1. Future Work

To conclude this research, optional concepts are proposed to further review aiming for performance improvement of the algorithms proposed:

- Expanding the DL algorithm to a Kalman Filter such as LQE's
- Explore the possibility of limited obesrvability such as output feedback controller that could be formulated into a DL algorithm.
- Stability analysis using Lyapounov stability.
- Real field experiments with various uncertainties and disturbances.

REFERENCES

Abdel-Razzak Merheb, H. N., & Bateman, F. (2014, June). Active fault tolerant control of quadrotor uav using sliding mode control. In *2014 international conference on unmanned aircraft systems.* Orlando, FL, USA.

Aeronautics, N., & Administration, S. (2021). *6 things to know about nasa's ingenuity mars helicopter.* Retrieved from https://www.nasa.gov/feature/jpl/6-things-to-know-about-nasas-ingenuity-mars-helicopter

A.I., W. (2021). *Artificial intelligence (ai) vs. machine learning vs. deep learning.* Retrieved from https://wiki.pathmind.com/ai-vs-machine-learning-vs-deep-learning

Andrew Zulu, S. J. (2014, September). A review of control algorithms for autonomous quadrotors. *Open Journal of Applied Sciences*, *4*, 547–556.

Aoun, C. (2019). *Energy optimal path planning for quadrotors in forests* (M.Sc. Thesis). American University of Beirut, Beirut, Lebanon.

Araar, O., & Aouf, N. (2014, October). Full linear control of a quadrotor uav, lq vs h. In *2014 ukacc international conference on control.* Loughborough, UK.

Brian D.O. Anderson, J. B. M. (1989). *Optimal control lienar quadratic methods*. Prentice-Hall International.

Christoph Aoun, N. D., & Shammas, E. (2019, December). An energy optimal path-planning scheme for quadcopters in forests. In *58th conference on decision and control (cdc).* Nice, France.

Christoph Aoun, N. D., & Shammas, E. (2020, July). Energy-optimal tours for quadrotors to scan moth-infested trees in densely-packed forests. In *2020 american control conference (acc).* Denver, Colorado.

Cuenca, A. (2021). *Geomagnetic aided dead-reckoning navigation* (MSc. Thesis). Embry Riddle Aeronautical University Daytona Beach.

Ding, L., & Wang, Z. (n.d., August). A robust control for an aerial robot quadrotor under wind gusts. *Journal of Robotics*, *2018*, 1–8.

Draguna Vrabie, F. L. L., Kyriakos G. Vamvoudakis. (2012). *Optimal adaptive control and differential games by reinforcement learning principles* (Vol. 200). The Institute of Engineering Technologies.

Foote, K. D. (2016). *A brief history of artificial intelligence.* Retrieved from http://www.dataversity.net/brief-history-artificial-intelligence/#

Foote, K. D. (2017). *A brief history of deep learning.* Retrieved from https://www.dataversity.net/brief-history-deep-learning/#

Goodfellow, J. M. M. X. B. W.-F. D. O. S. C. A. B. Y., Ian; Pouget-Abadie. (2014, June). Generative adversarial nets. In *International conference on neural information processing systems (nips 2014).* Montreal, QC.

Hamidreza Jafarnejadsani, N. H., Hanmin Lee, & Voulgaris, P. (2018, December). A multirate adaptive control for mimo systems with application to cyber-physical security. In *2018 ieee conference on decision and control (cdc).* Miami, FL, USA.

Harm BartMarinus, M. R., A. KaashoekAndré C. (2010). *A state space approach to canonical factorization with applications* (Vol. 200). Springer Basel.

Igelnik, B., & Yoh-Han Pao, F. (1995, November). Stochastic choice of basis functions in adaptive function approximation and the functional-link net. *IEEE Transaction on Neural Networks*, *6*, 1320–1329.

Kumawat, D. (2019). *Types of activation functions in neural network.* Retrieved from https://www.analyticssteps.com/blogs/7-types-activation-functions-neural-network

Kyriakos G. Vamvoudakis, F. L. (2010, July). Online actor critic algorithm to solve the continuous-time infinite horizon optimal control problem. In *International joint conference on neural networks.* Atlanta, GA, USA.

Kyriakos Vamvoudakis, F. L. (n.d., May). Online actorcritic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica*, *46*.

Lebao Li, J. J., Lingling Sun. (2015, October). Survey of advances in control algorithms of quadrotor unmanned aerial vehicle. In *16th international conference on communication technology.* Hangzhou, China.

Li, W.-H. C. J. Y. L. G. S. (2016, February). Disturbance-observer-based control and related methods—an overview. *IEEE Transactions on Industrial Electronics*, *63*, 1083–1095.

Luis Carrillo, K. V. (2019, July). Deep-learning tracking for autonomous flying systems under adversarial inputs. *IEEE Transactions on Aerospace and Electronic Systems*, *56*, 1444–1459.

Lőrincz, Z. (2019). *A brief overview of imitation learning.* Retrieved from https://smartlabai.medium.com/a-brief-overview-of-imitation-learning-8a8a75c44a9c

Middleton, M. (2021). *Deep learning vs. machine learning — what's the difference?* Retrieved from https://flatironschool.com/blog/deep-learning-vs-machine-learning

M. Rizon, Z. M. R. H. D.-S. A. B. W. K. Z. I., Mahmud Iwan Solihin. (2020, June). Effects of variable arm length on uav control systems. *Journal of Robotics, Networking and Artificial Life*, *7*, 91-97.

Mueller, M. W., & D'Andrea, R. (2014, September). Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers. In *International conference on robotics automation.* Hong Kong, China.

Nagyfi, R. (2018). *The differences between artificial and biological neural networks.* Retrieved from https://towardsdatascience.com/ the-differences-between-artificial-and-biological-neural-networks-a8b46db828b7

Olson, I. J., & Atkins, E. M. (2013, August). Qualitative failure analysis for a small quadrotor unmanned aircraft system. In *Guidance, navigation, and control (gnc) conference.*

Qing Lin, Y. W. J. Y. L. C., ZhiHao Cai. (2016, September). Adaptive flight control design for quadrotor uav based on dynamic inversion and neural networks. In *Third international conference on instrumentation, measurement, computer, communication and control.* Shenyang, China: IEEE.

Qingsong Jiao, Y. Z. W. L., Jia Liu. (2018, May). Analysis and design the controller for quadrotors based on pid control method. In *2018 33rd youth academic annual conference of chinese association of automation (yac).* Nanjing, China.

Shekhar, A., & Sharma, A. (2018, August). Review of model reference adaptive control. In *2018 international conference on information, communication, engineering and technology (icicet).* Pune, India.

Solovyev Viktor V., Z. Y. A. S. I. O., Finaev Valery I., & A., B. D. (2015, March). Simulation of wind effect on a quadrotor flight. *ARPN Journal of Engineering and Applied Sciences*, *10*(4), 1535–1538.

S. Sadr, S. A. A. M., & Zarafshan, P. (2014, November). Dynamics modeling and control of a quadrotor with swing load. *Journal of Robotics*, *2014*, 1–12.

Vignesh Kumar Chandhrasekaran, E. C. (2010, May). Fault tolerance system for uav using hardware in the loop simulation. In *4th international conference on new trends in information science and service science.* Gyeongju, Korea: IEEE.