DESIGN OF AN ONLINE ADAPTIVE CONTROLLER FOR ACTIVE DISTURBANCE REJECTION IN A FIXED WING UAV USING REINFORCEMENT LEARNING AND DIFFERENTIAL GAMES

STEPHEN MUCHAI KIMATHI

MASTER OF SCIENCE

(Electrical and Electronic Engineering)

JOMO KENYATTA UNIVERSITY OF AGRICULTURE AND TECHNOLOGY

2018

Design of an Online Adaptive Controller For Active Disturbance Rejection In a Fixed Wing UAV Using Reinforcement Learning And Differential Games

Stephen Muchai Kimathi

A thesis submitted in partial fulfilment for the degree of Master of Science in Electrical and Electronic Engineering in the Jomo Kenyatta University of Agriculture and Technology

2018

DECLARATION

This thesis is my original work and has not been presented for a degree in any other university.

Signature : .... Date : ....

Stephen Muchai Kimathi

This thesis has been submitted with our approval as University Supervisors:

Signature : .... Date : ....

Prof. Samuel Kang'ethe, PhD

JKUAT, Kenya

Signature : .... Date : ....

Dr. Peter Kihato, PhD

JKUAT, Kenya

DEDICATION

I dedicate this work to my late Dad for his encouragement and continued support towards my education.

ACKNOWLEDGEMENT

I would like to pass my sincere gratitude to my supervisors Prof. Samuel Kang'ethe and Dr. Peter Kihato for their input, direction and guidance towards the achievement of this work. Finally, I thank my mum, siblings, classmates, colleagues and friends for their encouragement, motivation and support they gave me during the period of this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

LIST OF ABBREVIATIONS

ADRC	Active Disturbance Rejection Control
CoG	Center of Gravity
DOF	Degrees of Freedom
ESO	Extended State Observer
GARE	Generalized Algebraic Riccati Equation
GNC	Guidance, Navigation and Control
HJI	Hamilton-Jacobi-Isaacs
NARX	Nonlinear Auto-Regressive Network
NTSB	National Transportation and Safety Board
ODE	Ordinary Differential Equations
PI	Proportional-Integral
PID	Proportional-Integral-Derivative
RL	Reinforcement Learning
SARSA	State Action Reward State Action
TCP	Transmission Control Protocol
TD	Temporal Differences
UAV	Unmanned Aerial Vehicle
UDP	User Datagram Protocol

ABSTRACT

The challenge of coping with highly nonlinear and rapidly time-varying dynamics is a prevailing factor when designing controllers for next-generation Unmanned Aerial Vehicles (UAVs). Stochastic disturbances such as wind gusts and atmospheric turbulence form the biggest challenge which the control structure must be capable of minimizing, while handling the errors in "dynamic modelling".

This work presents an online adaptive controller for active disturbance rejection in a fixed wing UAV using reinforcement learning. The approach is based on modelling the UAV system dynamics as a two player zero-sum differential game against nature, which will represent external disturbances affecting the UAV such as wind. Then an online learning algorithm using reinforcement learning is developed to solve the continuous-time two-player differential game. Game theoretic methods together with SARSA, a reinforcement learning technique were used to calculate a cost function for each state action pair. A one step gradient search of the cost function was done which was implemented as the reinforcement signal in the differential game. Back propagation technique was used to update the feed forward neural network weights in real time to compensate for the tracking errors in the heading from a commanded reference for each time step.

The simulation tests carried out under various disturbances in MATLAB and X-Plane, showed satisfactory performance of the proposed method to eliminate the disturbances and maintain the UAV in the desired target path. The responses got were compared with responses from of a well tuned PID controller both in MATLAB and X-Plane platforms. The adaptive method responses were better.

CHAPTER ONE

INTRODUCTION

1.1 Background

An unmanned aerial vehicle is a space traversing vehicle that flies without a human crew on board and, can be remotely controlled or can fly autonomously. The five general categories of UAVs depending on their configuration are: fixed wing, vertical takeoff and landing (VTOL), rotary-wing, and helicopters [1]. Among the many classifications made based on size, there is none which is universally accepted as a standard. The European Association of Unmanned Vehicle Systems (EUROUVS) drew up a classification of UAV systems. The classification was not created for certification purposes but for compiling a universal catalog of UAVs categories as well as their associated acronyms [2]. Here is the four categories based on size and wingspan [3];

1. Very small UAVs
2. Small UAVs
3. Medium UAVs
4. Large UAVs

UAVs have a wide range of application including surveillance, search and rescue, target tracking and weather observations. Recent technology developments allow unmanned aerial vehicles to displace manned aircraft in many commercial and military roles. As these roles are expanded from simple reconnaissance missions to more complex missions, there is an increasing need for control systems that are robust to model uncertainty due to incomplete modelling, system dynamics, parameter uncertainty and malfunction.

The control of the heading i.e. roll and yaw angles under various disturbances has been a challenging research due to the high nonlinearity and high coupling of an UAV model [4]. These present rapidly time-varying dynamics which make it difficult to analyse, and design control laws using traditional methods. Aerodynamic modelling is concerned with the development of mathematical models to describe the aerodynamic forces and moments acting on the airframe. As the flow conditions around the airframe are generally complex, any attempt to describe the aerodynamics phenomona mathematically must result in compromise [5]. This compromise is referred as “errors in dynamic modelling”, and depends on the confidence in the aerodynamic modelling process and the fidelity of the aircraft dynamics derived from the aerodynamic model. Adaptive flight control designs provide a way to deal with the uncertainties in the system and environment, without sacrificing performance hence ideally suited for this application [6].

Game theory is the mathematical study of the interaction of self-interested independent agents; each agent having his own description of which states of the world he likes and he acts in an attempt to bring these states of the world. A dynamic game is said to be differential if the evolution of the decision process takes place in continuous time and generally involves a differential equation. A game is zero-sum if the sum of the payoffs of the two players is zero for any choice of strategies. The goal of game theory techniques is to find a saddle point equilibrium in which each player has an outcome that cannot be improved unilaterally by changing his strategy.

Reinforcement Learning (RL), is based on the common sense idea that if a action is followed by a satisfactory state, or by an improvement in the state, then the tendency to produce that action is strenghtened i.e. reinforced [7]. Reinforcement learning framework provide algorithms with a reward function which indicates to the learning agent when it is doing well and when it is doing poorly. This learning can be extended along two dimensions [7];

i) the number of decision makers (single or multi)

ii) nature of interaction (collaborative or adversarial)

Reinforcement learning include a number of techniques; Monte Carlo methods, Dynamic programming and Temporal Differences (TD) learning methods. TD method has Q-learning which is an Off-policy learning method and SARSA algorithm an On-policy learning method.

1.2 Problem Statement

The various non-linear disturbance control strategies in use have certain drawbacks; Active Disturbance Rejection Control (ADRC), introduces an extra state into which untrackable terms like disturbances and modelling errors are lumped. An extended state observer is used to track the non-linear gains in the extra state. This increases the order of the system and hence complexity of the control algorithm. H_∞ control method requires the complete nonlinear model of a system. Its application in unmanned aerial vehicles assumes the existence of deterministic disturbances while trivializing the existence of unmodelled system dynamics and parameter variation during flight. Therefore, there is need to use adaptive heuristics to minimize stochastic disturbances and errors in "dynamic modelling" during flight without sacrificing performance while also improving the control of a UAV in real time. This is because path following in an unmanned aerial vehicles is the most important function which accounts for flight safety and mission survivability.

1.3 Justification

UAVs are subject to disturbances, perturbations and system dynamics for example atmospheric turbulence and wind gusts which are too complex to be characterized by explicit mathematical models. An adaptive scheme which gathers data from online operation and uses adaptive heuristics to determine the parameters of the controller is highly desirable. Reinforcement learning based on neural networks offer distinct advantages for improving control performance; their nonlinearity enables neural networks to implement a wider range of control functions and their adaptability permits them to improve control performance via on-line, trial and error learning. Moreover, temporal differences reinforcement learning techniques do not require an accurate model of the environment.

In view of these advantages, it is imperative to develop an adaptive technique which result in optimal feedback controllers for dynamical systems that can be described by differential equations.

1.4 Objectives

1.4.1 Main Objective

The main objective of this research is to design an online adaptive controller using differential games and reinforcement learning for active disturbance rejection in a fixed wing unmanned aerial vehicle autopilot.

1.4.2 Specific Objectives

i) Model a fixed wing UAV system dynamics as a two player differential game.

ii) Design a controller using reinforcement learning (SARSA algorithm) by exploring saddle points in the differential game.

iii) Simulate the controller on a UAV model in X-Plane.

1.5 Scope

This research deals with active disturbance rejection in a fixed wing UAV using SARSA algorithm, a reinforcement learning technique and optimal control strategies. The focus is designing an online adaptive controller in a miniature (small) fixed wing UAV to cater for disturbances in the lateral directional motion using information gathered during flight by adaptively minimizing the effect that these disturbances have on the heading of a UAV.

1.6 Research Contributions and Publications

The main contribution of this thesis are;

i) A UAV is modelled as a two player differential game. Where the second player is taken as wind disturbance. Game theoretic methods are used convert the modelled system to a differential game against nature.

ii) An online adaptive controller using *SARSA*, a reinforcement learning technique is introduced where a generalized solution of the differential game is achieved using an iterative process.

iii) The designed controller is tested using X-Plane and MATLAB/SIMULINK simulation tools for the performance of the designed controller under different wind conditions.

The related publications and conferences emanating from this work include;

Publications

i) Kimathi, S., Kangethe, S., and Kihato, P. , "Heading Control of a Fixed Wing UAV under Windy and Turbulent conditions Using Reinforcement Learning", IOSR, Journal of Electrical and Electronic Engineering, Vol.2, 2018.

ii) Kimathi, S., Kangethe, S., and Kihato, P. , "Application of Reinforcement Learning in Heading control of a Fixed Wing UAV using X-Plane Platform", International Journal of Scientific and Technology Research, Vol. 6, Issue 02, 2017.

Conference Proceedings

i) Kimathi, S., Kangethe, S., and Kihato, P. , “UAV heading controller using Reinforcement Learning”, Proceedings of Pan African Conference on Science, Computing and Telecommunications (PACT), Strathmore University, March 2017.

ii) Kimathi, S., Kangethe, S., and Kihato, P. , “A review of control algorithms for fixed wing UAVs”, Proceedings of the 2^{nd} DeKUT International Conference on Science, Technology, Innovation and Entrepreneurship, Nov. 2016.

1.7 Organization of the Thesis

This thesis is organized as follows. An introductory background, objectives and the scope of the research are given in Chapter 1. Chapter 2 introduces the reinforcement learning and differential game principles. It also provides the basis of learning in games and finally a short review of methods used in disturbance rejection in UAVs is presented. Chapter 3 describes the methodology of the research, that is the modelling of the system and the design of the adaptive controller. In Chapter 4, an analysis of the modelled system is carried out. Simulation tests of heading control in a UAV using both the mathematical model and a UAV model in X-Plane and the discussions are presented. The conclusion and future recommendations for the research are outlined in Chapter 5.

CHAPTER TWO

LITERATURE REVIEW

2.1 Reinforcement Learning

Reinforcement learning (RL), is learning what to do that is, how to map situations to actions, so as to maximize a numerical reward signal. The learning agent is not told the correct actions; instead it explores the possible actions and remembers the reward it receives [7]. It is inspired by natural learning mechanisms where animals adjust their actions based on the reward or punishment stimuli they receive from interacting with the environment[8]. In machine learning, RL is a method for solving optimization problems that involve an agent that interacts with an environment and modifies its actions based on stimuli it receives in response to its actions. The learner and decision maker is called the *agent* while everything outside the agent that it interacts with is called its *environment*. The agent and the environment interact with each other through a sequence of discrete time steps, $t = 0, 1, 2, 3, \ldots$. At each timestep t, the agent receives some representation of the environment's *state*, $s_t \in S$, where S is the set of possible states, and on that basis executes an *action* $a_t \in A(s_t)$ where $A(s_t)$ is the set of actions available in the state s_t. One time step later, as a consequence of its action, the agent receives a numerical reward r_{t+1} and finds itself in a new state s_{t+1} [9]. At each time step, the agent implements a mapping from states to probabilities of selecting each possible action. This mapping is called the agent's *policy* and is

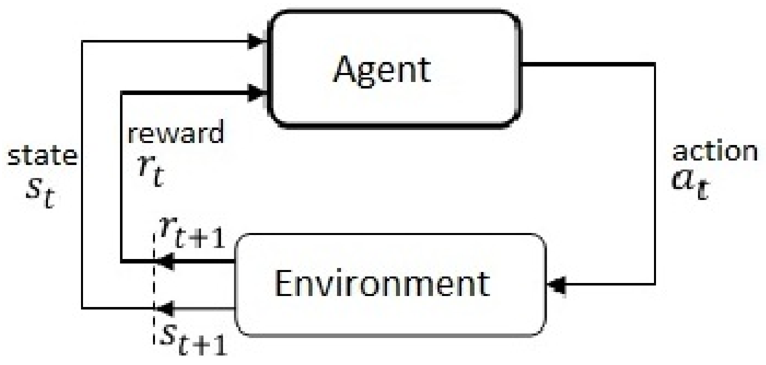


Figure 2.1: Agent-environment interaction in reinforcement learning

denoted as π_t and it is which maximizes the cumulative reward of an agent over time [9]. The cumulative reward function is given by equation (2.1),

$$R = \sum_{t=o}^{\infty} \gamma^t r_t \tag{2.1}$$

where $0 < \gamma < 1$ is a discount factor, which discounts the value of future rewards. To obtain a preferable reward, a reinforcement learning agent must prefer actions that it has tried in the past and found to be effective in producing a good reward. The agent has to *exploit* what it already knows and also *explore* the environment in order to make better action selections in future. In reinforcement learning there is dynamic programming(DP), Monte Carlo methods and temporal difference methods; which comprise of Q-learning and sarsa algorithm where the latter is an online learning method [9].

Like Monte Carlo methods, TD methods can learn directly from raw experience without a model of the environment dynamics and like DP, TD methods updates estimates based in part on other learned estimates without having to wait for a final outcome. This makes TD methods more robust and effective as they bootstrap information they gather from interacting with the environment. The simplest TD method is given as

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \tag{2.2}$$

where the temporal difference update is

$$r_{t+1} + \gamma V(s_{t+1}) \tag{2.3}$$

where

- α is the learning rate

- r is the reward for being in that state

- γ is a discounting factor

- $V(s_t)$ is the previous state value function

- $V(s_{t+1})$ is the present state value function

SARSA being an on line TD method, estimates $Q^{\pi}(s, a)$ for the current behavior policy π and for all states s and actions a. This is done using the same TD method described in equation (2.2) but the transitions from state-action pair to state-action pair is considered rather than from one state to another state and hence the value of the state-action pairs.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{2.4}$$

This update is done after every transition from a non terminal state s_t . The word SARSA comes from the transitions $s_t - a_t - r - s_{t+1} - a_{t+1}$ as Figure 2.2.

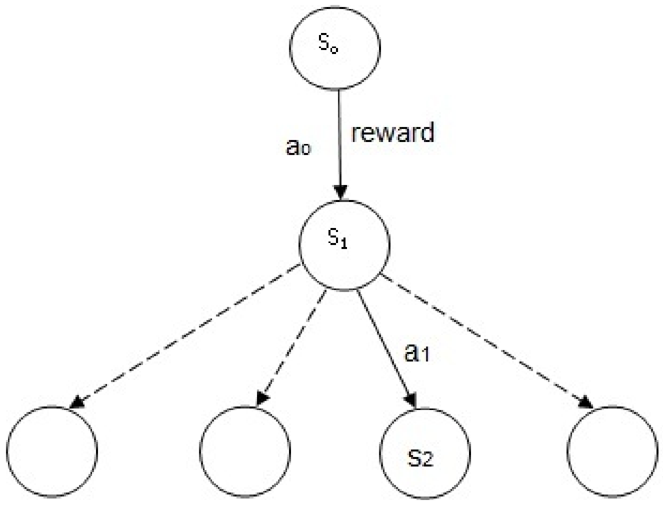


Figure 2.2: SARSA algorithm illustration

Therefore in SARSA we continually estimate Q^{π} for the behavior policy π and at the same time change π towards greediness with respect to Q^{π}. This is referred to as generalized policy iteration as is shown in Figure 2.3. RL is different from

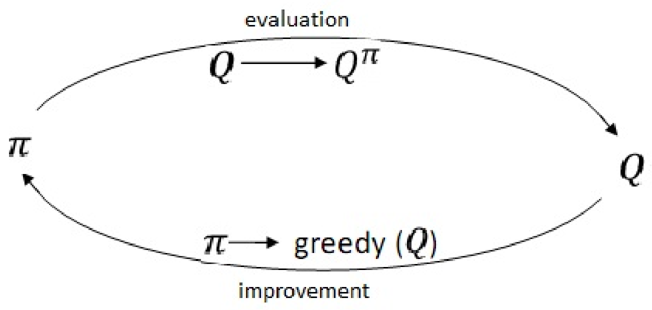


Figure 2.3: Generalized Policy iteration

supervised learning; which is learning from examples provided by a knowledgeable external supervisor, as it is not adequate for learning from interaction without a priori information or pattern [10]. Reinforcement learning as a class of learning methods can and has been used to design adaptive controllers that learn on line, in real time, the solutions of user-prescribed control problems [7][10].

2.2 Differential Games

A game is defined as any situation in which there are at least two players, each player having a number of possible strategies and courses of actions which he/she may follow. The strategies chosen by each player determine the outcome of the game. Also associated with each possible outcome of the game is a collection of numerical *payoffs* [11]. Each player would like the game to end in an outcome which gives him as large a payoff as possible. A game is zero-sum if the sum of the payoffs of the two players is zero for any choice of strategy. Games can either be static or dynamic. A static game is one in which a single decision is made by each player, and each player has no knowledge of the decision made by the other players before making their own decision. Decisions are made simultaneously. In dynamic games, the actions available to each agent depends on its current state which evolves according to a certain dynamical system i.e. sets of states and actions are usually in a continuum. A dynamic game is said to be a differential game if the evolution of the decision process takes place in continuous time and generally involves a differential equation. Differential games provide a natural extension to the standard control theory model, where two or more individuals are present and each one of them seeks to maximize his own payoff. The state of a system is described by a variable $x \in \chi \subseteq \mathbf{R}^n$. This state evolves in time according to an ODE

$$\dot{x}(t) = f(t, x(t), u_1(t), u_2(t)) \qquad t \in [0, T] \tag{2.5}$$

Here $t \mapsto u_i(t) \in \mathbf{U_i} \subseteq \mathbf{R}^m, \quad i = 1, 2,$ are the control functions implemented by the two players. The goal of the $i - th$ player is to

$$maximize: \qquad J_i = \int_0^T L_i(t, x(t), u_1(t), u_2(t))dt \tag{2.6}$$

where J_i is the performance index and L_i represent $x^T Q x + u^T R u - \gamma^2 \left\|d\right\|^2$, a cost function. Since the gain of player 1 represents a loss of player 2, such scenerios are referred to as *zero-sum games*. Saddle point equilibrium strategy arises in zero-sum differential games where a Player 1 wants to maximize and a Player 2 wants to minimize, i.e. $(U^{1\star}, U^{2\star})$ is a saddle point equilibrium if;

$$J(U^1, U^{2\star}) \leq J(U^{1\star}, U^{2\star}) \leq J(U^{1\star}, U^2) \tag{2.7}$$

The analysis of differential games relies heavily on the concepts and techniques of optimal control theory whereby equilibrium strategies in feedback are best studied by looking at a system of Hamiltonian-Jacobi-Bellman PDEs for the value functions of the various players [12].

2.3 Learning in Differential Games

Learning in games can be formalized as a multi-agent reinforcement learning problem. Agents select actions simultaneously at the current state and receive rewards at the next state. The agents may not know the transition function or the reward function from the environment, instead the agents are required to select actions and observe the received immediate reward and the next state in order to gain information of the transition function or reward function. Learning in differential games has attracted a lot of attention [10] [13, 14, 15] . Here, reinforcement learning algorithms have been applied to linear quadratic differential games where the game is a Markov decision process with continuous time, states and actions, linear dynamics and a quadratic cost function. The games are of predator-prey/pursuit-evasion systems [10][14]. In [16], an online concurrent reinforcement learning algorithm based on neural networks to solve the H_∞ control problem of partially unknown continuous time systems is presented. The control input acts as one player and attempts to make the optimal control while the other player is a disturbance, which tries to make the worst-case possible disturbance. The approximate solution is achieved using integral RL and policy iteration for finding the optimal value function of the corresponding Hamilton Jacobi Isaacs (HJI) equations. In [10][16], closed loop stability using Lyapunov technique and convergence of the algorithm was achieved.

In this work, a similar approach based on linear quadratic differential games and general policy iteration will be used, but a slightly different RL technique called SARSA will be used due to its key advantages over other RL techniques [9];

1. Its an on-policy algorithm since it updates value functions strictly on the basis of the experience gained from executing some policy i.e. it does not represent the policy as a separate entity.

2. Due to the dependence of the policy on the value function, there is enhanced convergence of the learning process hence less computational power is required.

2.4 Review of methods used in disturbance rejection in UAVs

2.4.1 Active disturbance rejection control (ADRC)

ADRC was originally proposed by J. Han and comprises of three parts: a tracking differentiator, extended state observer (ESO), and a non-linear state-error feedback [17]. It is designed such that the ESO control algorithm actively estimates and compensates for, in real time the effects of the unknown dynamics and disturbances, forcing an otherwise unknown plant to behave like a nominal one [18]. That is, instead of depending on the model of the plant, the controller draws the information from the ESO.

The basic principle of ADRC is directly estimating the system dynamics and the total disturbances which are extended as a new system state in real time using an ESO and then compensating for them [4]. This method suffers from a few drawbacks including;

i) it lumps untrackable terms into “total disturbances” hence augmenting the observer to include an extra state [18, 19]

ii) while the non-linear gains in the state observer may be effective, they also produce complexity in the control algorithm implementation and tuning [18].

2.4.2 Non-linear H_∞ state feedback

Non-linear H_∞ control has been applied to aerospace problems such as to stabilise the rotational movements of a quadrotor helicopter but the application has been restricted to lateral or longitudinal motion alone [20]. The main difficulty in applying non-linear H_∞ control theory to flight dynamics is solving the associated Hamilton-Jacobi-Isaacs equations. A solution for this problem was proposed by [21] for a general 6-DOF motion which can be applied to various types of vehicles such as airplanes, missiles or helicopters. The drawback of this work is that the design of the controller is based on deterministic external forces acting on the vehicle and the aerodynamics characteristics of the aircraft are neglected.

2.4.3 Proportional Integral Derivative Method

The birth and deployment of Proportional-Integral-Derivative (PID), control dates back to the period of the 1920s-1940s, in response to the pressing demands of industrial automation before, during and after World War II. Its dominance is evident even today in various sectors of the industry. The classical PID linear controller has the advantage that parameter gains are easy to adjust, is simple to design and tune and has rather good robustness. In [22], PID controllers were used in autopilot design, one for the lateral motion autopilot and the other for longitudinal autopilot. The former was converting the roll rate error to aileron commands with an output saturation mechanism while the latter was converting heading error to a roll attitude command which was fed to the former as the roll rate error. This same sequentially tandem PID control was used in [23]. In both [22, 23] the PID gains were designed using root locus technique in order to achieve the desired heading successfully. The performance showed good response in terms of both transient and steady state performance. In [24], the performance of PID controller in control of a fixed wing UAV is compared with other control methods and its performance was satisfactory but with some limitations; steady state errors and tuning of the gains is done off line about an equilibrium point hence not adaptive.

2.4.4 Intelligent Methods

Intelligent control refers to the use of artificial intelligent techniques in controlling a system. Most notable intelligent techniques include fuzzy logic, genetic algorithms and reinforcement learning.

Fuzzy logic represents data in fuzzy sets and uses fuzzy membership functions to classify data. Prabhudas and Nagababu in [25] used a fuzzy logic controller in controlling the altitude of a fixed wing UAV. The fuzzy logic controller was used to adapt the already designed PID controller so as to improve robustness and adaptability of the controller. In [26], a full fuzzy controller was implemented for takeoff, cruise and landing. Simulation results revealed the usability of this method although it had average tracking performance and sluggish response. The tracking capability of fuzzy logic controllers in UAV autopilot design is also highlighted in [24]. Reinforcement learning is learning through punishment and reward. It is an intelligent method as a 'punishment' is made to a controller if it deviates from the set path. This technique although old, its application in flight control has been scarce due to problem definition that suits formulation of the algorithm. However, in [27] reinforcement learning was used to harness energy from a horizontal wind shear to achieve dynamic soaring. Simulation results showed the RL controller was able to achieve better performance than its baseline teacher. Moreover, it also followed its own optimum path other than that which it was supposed to follow. In [28], RL was used to improve the baseline policy through a constructive relationship between a planner and a learner so as to mitigate the learning risk. Simulation results showed impressive improvement in performance and reduction in risk.

2.4.5 Other nonlinear methods

Adaptive Backstepping Backsteping has been applied to control the movements of altitude, yaw and roll in UAVs with comparison to other linear controllers such as proportional-integral (PD) and proportional-integral-derivative (PID), and nonlinear controllers such as fuzzy, sliding mode and nested saturation methods as in [24]. In [29], adaptive backstepping is used to obtain directional control of a fixed wing UAV in presence of unknown crosswind. The control strategy was focused on reducing the position deviation of the airplane with respect to a desired path in the lateral dynamics. Its drawback is that in addition to being applied to only lateral dynamics, it is an offline method.

Adaptive Super Twisting Technique It uses the adaptive super twisting algorithm whose goal is to design a controller without overestimating the gain so as to drive the sliding state variable and its derivative to zero in finite time under boundary disturbances (additives/multiplicative) with unknown bounds [30]. The drawback is that it includes a differentiator to estimate inertial states i.e. $n - th$

derivatives of $f(t)$ which introduces complexity of control algorithm. Also, it is inferior in terms of system response and robustness as compared to ADRC.

Feedback Linearization Dynamic inversion and dynamic surface control belong to a class of algorithms known as feedback linearization where a nonlinear system model is transformed into an equivalent system using a change of variables. In [31], a multi-loop structure is used that comprised an outer guidance loop which acted as a virtual target and an inner control loop. The control loop based on dynamic surface control is derived so as to follow the command generated by the outer guidance loop. The method exploits the relationship between the two loops thus avoids 'explosion of complexity'. Simulations showed good tracking performance. In [32], two approaches are used; one employed a two-stage dynamic inversion and the other employed feedback dynamic inversion on the command augmented system. The two methods permit adaptation of the parametric uncertainity and unmodelled dynamics. Numerical simulations showed average performance while used on their own. The tracking performance was greatly improved through the use of pseudo-control hedging and neural-network adaptation designs.

2.5 Summary

In this chapter, reinforcement learning has been discussed with specific emphasis on SARSA algorithm, a temporal differences method. Differential games were introduced and how learning in differential games occurs. Methods used in disturbance rejection in UAVs are discussed where under each method, its disadvantage is stated.

Since the UAV will be modelled as a two player differential game, the use of non-linear H_∞ technique is not feasible due to the complexity of finding the solution of the associated Hamilton-Jacobi-Isaacs equations. Moreover, it is an off line method and relies on deterministic external forces whereas wind disturbances are stochastic. Therefore, the need to use adaptive heuristics contained in reinforcement learning for the solution of a differential game without increasing the complexity of the system.

CHAPTER THREE

METHODOLOGY

This chapter covers UAV mathematical modelling and the design of an adaptive controller.

3.1 Introduction: UAV Control Basics

An autopilot consists of three subsystems; Guidance, Navigation and Control (GNC) systems. The *guidance system* is defined as a group of components that measure the position of the guided vehicle with respect to the target and changes its flight path in accordance with a guidance law to achieve the flight mission goal [1]. Therefore guidance systems takes care of the system inputs; way-points, desired speed and path. The *navigation system* determines the position and altitude of the aircraft at a given time using generated coordinates. The control system ensures that the aircraft follows the desired path and altitude from the guidance system to the target position by manipulating the aircraft's control surfaces. The subsystems are related as illustrated in Figure 3.1. An unmanned aerial vehicle is a 6 degrees of freedom (DOF), rigid model with the propeller's motor throttle (δ_T) and three control surfaces; elevator (δ_e), aileron (δ_a) and rudder (δ_r) which act as control surfaces. Figure 3.2 shows a UAV axes of motion where the plane can rotate about the three principle axes (x, y, z) from its center of gravity. The position control of the UAV is converted to angular control in the three principle axes; roll ($\varPhi$), pitch (θ) and yaw (ψ)as shown in Figure 3.3. The

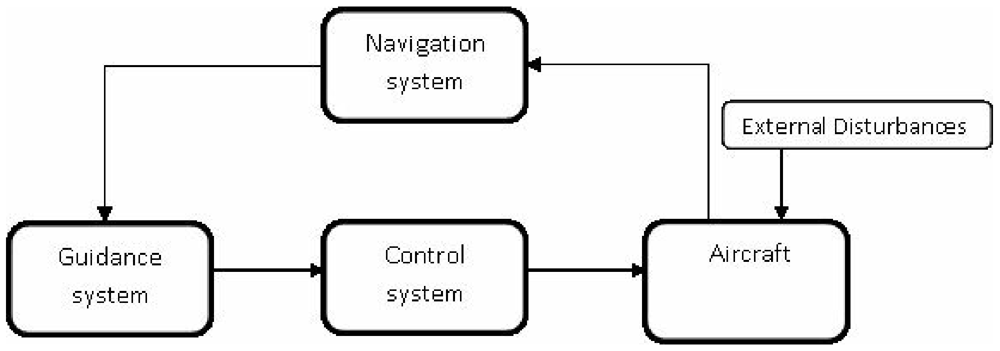


Figure 3.1: UAV guidance, navigation and control model

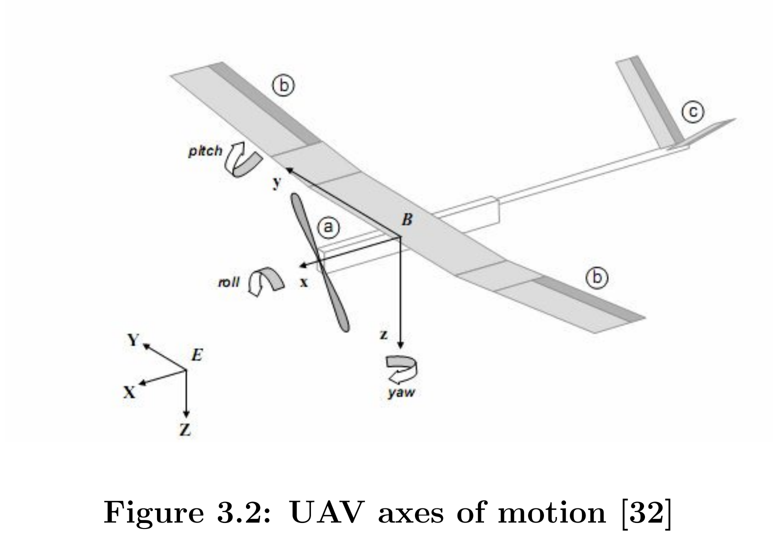


Figure 3.2: UAV axes of motion [32]

effect of the control surfaces to a fixed wing UAV are;

- Ailerons to control the rolling
- Elevator to control the pitching
- Rudder to control the yawing
- Throttle to control the engine power

as shown in Figure 3.3

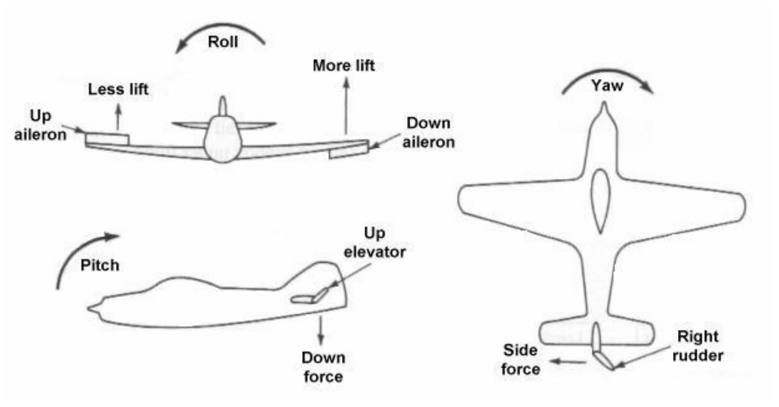


Figure 3.3: Control Surfaces

3.2 UAV Mathematical modeling

As identified in [4], in addition to the control surfaces above, the derivation of a complete nonlinear model of a UAV includes velocities (u, v, w), accelerations (a_x, a_y, a_z) and angular rates (p, q, r) measured along the three principle axes of motion *x, y, z* respectively. In [31], the velocities are converted into α, the angle of attack and β, the sideslip angle for easier manipulation and representation of the model. The derivation of equations of motion for a fixed wing UAV is given in [33, 34]. The complete nonlinear equations of motion are linearized about a level flight trim condition. The linear model thus obtained is decoupled into longitudinal and lateral directional sub-systems to approximate the UAV dynamics in the longitudinal and lateral directions of motion respectively [35]. Thus a lateral mode state space model decoupled from within the linear model is then used with aileron and rudder as inputs to control the heading of a UAV [23][20]. The decoupled lateral mode state space model from [33] and [36] is given as

$$\dot{x} = A_{lat}x_{lat} + B_{lat}u_{lat} \tag{3.1}$$

where x_{lat} is the decoupled lateral state space model with $[\rho \quad \beta \quad r \quad \Phi]^T$ as the state variables. ρ is the roll rate, β is the sideslip angle, r is the yawing rate and Φ is the roll angle of a UAV , u_{lat} is the control input and comprises of δ_a, aileron deflection and δ_r, rudder deflection as shown in Appendix I. A_{lat} is the state matrix and B_{lat} the input matrix. The state space model as derived in Appendix I [34] is

$$\begin{bmatrix} \dot{\beta} \\ \dot{\rho} \\ \dot{r} \\ \dot{\Phi} \end{bmatrix} = \begin{bmatrix} Y_p & Y_\beta & Y_r - 1 & mgcos\theta_e \\ L_p & L_\beta & L_r & 0 \\ N_p & N_\beta & N_r & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \rho \\ \beta \\ r \\ \Phi \end{bmatrix} + \begin{bmatrix} L_{\delta a} & L_{\delta r} \\ Y_{\delta a} & Y_{\delta r} \\ N_{\delta a} & N_{\delta r} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} \tag{3.2}$$

where L, N are moment equations, and Y is a force equation, all acting along the motion variables p, r and v, $(\beta = \frac{v}{U_e})$.

The aerodynamic and stability coefficients shown in Figure 3.4 are taken from [20] and [36]. Other coefficients include $C_{m_{\delta e}} = -1.13$, $C_{m_q} = -5.08 \times 10^1$, $C_{n_\beta} = 3.44 \times 10^-2$, $C_{n_{\delta r}} = -3.45 \times 10^-2$.

The UAV properties are;

- Mass of the UAV, $m = 1.9kg$,

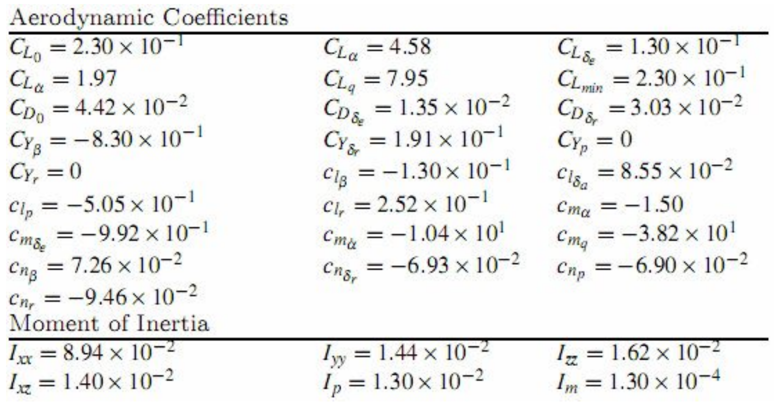
- The wingspan, $b = 1.2m$,

- The wing area, $S = 0.32m^2$,

- The density of the UAV, $\rho = 1.225kg/m^3$,

- Earth's gravitational pull, $g = 9.8m/s^2$.

These values are substituted into equation (3.2) using the relations in Appendix 1 labelled as **Lateral motion derivatives functions**. The resulting linear state space model is given in equation (3.3) and equation (3.4).



Aerodynamic Coefficients		
$C_{L_0} = 2.30 \times 10^{-1}$	$C_{L\alpha} = 4.58$	$C_{L_{\delta_e}} = 1.30 \times 10^{-1}$
$C_{L_\alpha} = 1.97$	$C_{L_q} = 7.95$	$C_{L_{min}} = 2.30 \times 10^{-1}$
$C_{D_0} = 4.42 \times 10^{-2}$	$C_{D_{\delta_e}} = 1.35 \times 10^{-2}$	$C_{D_{\delta_r}} = 3.03 \times 10^{-2}$
$C_{Y_\beta} = -8.30 \times 10^{-1}$	$C_{Y_{\delta_r}} = 1.91 \times 10^{-1}$	$C_{Y_p} = 0$
$C_{Y_r} = 0$	$c_{l_\beta} = -1.30 \times 10^{-1}$	$c_{l_{\delta_a}} = 8.55 \times 10^{-2}$
$c_{l_p} = -5.05 \times 10^{-1}$	$c_{l_r} = 2.52 \times 10^{-1}$	$c_{m\alpha} = -1.50$
$c_{m_{\delta_e}} = -9.92 \times 10^{-1}$	$c_{m_{\dot\alpha}} = -1.04 \times 10^{1}$	$c_{m_q} = -3.82 \times 10^{1}$
$c_{n_\beta} = 7.26 \times 10^{-2}$	$c_{n_{\delta_r}} = -6.93 \times 10^{-2}$	$c_{n_p} = -6.90 \times 10^{-2}$
$c_{n_r} = -9.46 \times 10^{-2}$		
Moment of Inertia		
$I_{xx} = 8.94 \times 10^{-2}$	$I_{yy} = 1.44 \times 10^{-2}$	$I_{zz} = 1.62 \times 10^{-2}$
$I_{xz} = 1.40 \times 10^{-2}$	$I_p = 1.30 \times 10^{-2}$	$I_m = 1.30 \times 10^{-4}$

Figure 3.4: Stability Coefficients [20]

$$A = \begin{bmatrix} 0 & -1.4000 & 0 & 9.4953 \\ -12.8000 & -30.9000 & 14.4000 & 0 \\ -0.4480 & 1.4781 & -6.0800 & 0 \\ 1.0000 & 0 & 0 & 0 \end{bmatrix} \tag{3.3}$$

and

$$B = \begin{bmatrix} 0 & 0.7412 \\ 61.4000 & 12.4000 \\ -3.6700 & -15.0000 \\ 0 & 0 \end{bmatrix} \tag{3.4}$$

The C matrix is chosen as in equation(3.5).

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

and the D matrix is taken to be a zero matrix of the required dimensions.

3.2.1 Differential game modeling

The Optimal Control Problem

Consider the nonlinear time-invariant affine input dynamical system given by

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t); x(0) = x_0 \tag{3.6}$$

with the state as $x(t) \in \mathbb{R}^n$, $f(x(t)) \in \mathbb{R}^{n \times n}$, $g(x(t)) \in \mathbb{R}^{n \times m}$, $u(x(t)) \in \mathbb{R}^m$ as the A matrix , B matrix and control input respectively [8]. We assume that $f(0) = 0, f(x) + g(x)u$ is Lipschitz continuous on a set $\Omega \subseteq \mathbb{R}^m$ that contains the origin and that the system is stabilizable on Ω, i.e. there exists a continuous control function $u(t) \in U$ such that the system is asymptotically stable on a set Ω. Define an integral cost function

$$V(x_o) = \int_0^\infty r(x(\tau)), u(\tau) d\tau \tag{3.7}$$

where $r(x,u) = Q(x) + u^T R u$ with $Q(x)$ being $(x^T Q x)$ as positive definite and $R \in \mathbf{R}^{m \times m}$, a symmetric positive definite matrix.

Definition 1 *A control policy $\mu(x)$ is defined as admissible with respect to equation* (3.7) *on Ω, if $\mu(x)$ is continuous on $\Omega, \mu(0) = 0, u(x) = \mu(x)$ stabilizes equation* (3.6) *on Ω and $V(x_0)$ is finite $\forall x_0 \in \Omega$ [36].*

The optimal control problem can now be formulated. Given the continuous-time system equation (3.6), the set $\mu \in \Psi(\Omega)$ of admissible control policies and the infinite horizon cost function equation (3.7), find an admissible control policy such that the cost index equation (3.7) associated with the system equation (3.6) is minimized. Defining the Hamiltonian of the problem as

$$H(x, u, V_x) = r(x(t)), u(t) + V_x^T(f(x(t)) + g(x(t)\mu(t)) \tag{3.8}$$

where V_x^T is the cost function, and the optimal cost function $V^*(x)$ as

$$V^*(x_0) = \min_{\mu \in \Psi(\Omega)} (\textstyle\int_0^\infty r(x(\tau)), u(\tau) d\tau)$$

with $x_0 = x$ is known as the value function and satisfies the HJB equation

$$0 = \min_{\mu \in \Psi(\Omega)} [H(x, \mu, V_x^*)] \tag{3.9}$$

Assuming that the minimum on the right hand side of equation (3.9) exists and is unique, then the optimal control function for the given problem is

$$u^*(x) = -\tfrac{1}{2} R^{-1} g^T(x) V_x^*(x) \tag{3.10}$$

For the linear case, considering the quadratic cost functional, the equivalent of the HJB equation is the Riccati equation. The optimal control solution for the

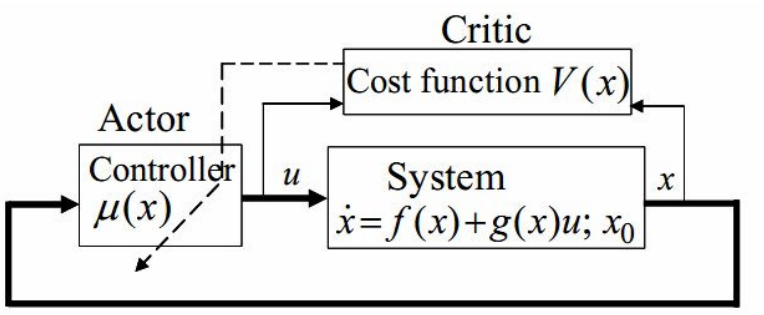


Figure 3.5: Actor Critic Structure

problem is achieved by solving the HJB equation (3.9) for the value function and then substitute this into equation (3.10).

Due to the nonlinear nature of the HJB equation, finding its solution is generally difficult. But this is achieved using policy iteration [9], which is an iterative method of reinforcement learning for solving optimal control problems and consists of policy evaluation and policy improvement based on equation (3.10). In the linear-time invariant case, policy iteration method reduces to the Kleinman's algorithm [37] for the solution of the Riccati equation and hence a Lyapunov equation by extension. The policy iteration algorithm like other reinforcement learning algorithms can be implemented on an actor-critic whose structure [9] is shown in Figure 3.5; *actor* because it is used to select actions, and the estimated value function is known as the *critic* because it critizes the actions made by the actor. Learning is always on-policy that is, the critic must learn about and critique whatever policy is currently being followed by the actor; it takes the form of a TD error. This scalar signal (dotted line in Figure 3.5)is the sole output of the critic and drives all learning in both actor and critic. The critic network is tuned to solve the value function and the actor is tuned to solve the control policy in equation (3.10). The policy iteration algorithm then consists of tuning the two networks iteratively. At each time step, the critic neural network is tuned to evaluate the performance of the current control policy.

Two Player Differential game

The formulation of a single input optimal control problem to a two player optimal control problem commonly referred to as a differential game is put forth. The solution of a two player zero-sum infinite horizon game is presented where saddle points strategies for both player are learned in real-time. From equation (3.6) a second player is included and the dynamical system is reformulated as

$$\dot{x}(t) = f(x(t)) + g(x(t))u(x(t)) + k(x)d(x); \tag{3.11}$$

with state $x(t) \in \mathbb{R}^n, f(x(t)) \in \mathbb{R}^n, g(x(t)) \in \mathbb{R}^{n\times m}$ and control input $u(x(t)) \in \mathbf{R}^m, k(x(t)) \in \mathbb{R}^{n\times q}$ and a disturbance $d(x(t)) \in \mathbb{R}^q$. We assume that $f(x)$, is locally Lipschitz, $f(0) = 0$ so that $x = 0$ is an equilibrium point of the system. The corresponding performance index for equation (3.11) is given by [8] as

$$J(x(0), u, d) = \int_0^\infty (Q(x) + u^T Ru - \gamma^2 \parallel d \parallel^2)dt \equiv \int_0^\infty r(x, u, d)dt \tag{3.12}$$

For feedback policies $u(x)$ and disturbance policies $d(x)$, a cost function is defined as

$$V(x(t), u, d) = \int_0^\infty (Q(x) + u^T Ru - \gamma^2 \parallel d \parallel^2)dt \tag{3.13}$$

For a two player zero-sum differential game [8], it is defined as in equation (3.14)

$$V^* x(0) = \min_u \max_d J(x(0), u, d) = \min_u \max_d \int_0^\infty (Q(x) + u^T Ru - \gamma^2 \parallel d \parallel^2)dt \tag{3.14}$$

which is subject to the constraints in equation (3.11). Thus, u is the minimizing player and d is the maximizing player. This optimal control problem has a unique solution, if a game theoretic saddle point exists; equation (2.7) highlights the condition for existence of a saddle point.

A necessary condition for this, is the Isaac's condition

$$\min_u \max_d H(x, \nabla V, u, d) = \max_u \min_d H(x, \nabla V, u, d) \tag{3.15}$$

for some saddle point (u^*, d^*).

The Hamiltonian is given as

$$H(x, \nabla V, u, d) = r(x, u, d) + (\nabla V)^T (f(x) + g(x)u(t) + k(x)d)[8] \tag{3.16}$$

Given the solution of the value function $V^*(x) \geq 0$, the associated control and disturbance policies can be denoted as

$$u^* = -\tfrac{1}{2} R^{-1} g^T(x) \nabla V^* \tag{3.17}$$

and

$$d^* = \tfrac{1}{2\gamma^2} k^T(x) \nabla V^* \tag{3.18}$$

Proof of above equation (3.17) and equation (3.18) is given in [38]. The system equation (3.11) is linearized about the origin to obtain the Generalized Algebraic Riccati Equation (GARE). Of the non-negative definite solutions to the GARE, the one with the corresponding stable invariant manifold of the Hamiltonian matrix is selected. The minimum non negative solution of the corresponding HJI equation is the one having the stabilizing GARE as its Hessian matrix. The HJI equation (3.16) is usually intractable to solve directly but iterative methods like

[39, 40] have been applied in solving it. In [39], a policy iteration algorithm was applied where an inner loop with iterations on the disturbance was used. In this work the policy iteration algorithm is not implemented, instead only one loop is implemented; the feedback control loop. The effect of the inner disturbance loop is achieved from the system's interaction with the environment. In the linear case, the solution of the HJI equation is given by the solution of the GARE [41] as

$$A^T P + PA + Q - PBR^{-1}B^T + \tfrac{1}{\gamma^2} PKK^T P = 0 \tag{3.19}$$

where A is given in equation (3.3), B in equation (3.4), K from the solution of equation (3.19), Q and R are as described in equation (3.20) and equation (3.21) respectively, where the basis of their choice is informed from the perspective of maintaining a desired heading angle while placing constraints on the maximum permissible roll rate, pitch rate, aileron and rudder deflections respectively [33].

$$Q = \begin{bmatrix} 1/p_{max}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/\Phi_{max}^2 \end{bmatrix} \tag{3.20}$$

$$R = -\gamma^2 \times \begin{bmatrix} 1/\delta r_{max}^2 & 0 \\ 0 & 1/\delta r_{max}^2 \end{bmatrix} \tag{3.21}$$

where $\delta_{max} = \pm\pi/3°$, $\Phi_{max} = 10° = 0.174$ rad, $p_{max} = 300°/s = 5.23$ rad/s and $\gamma = 0.5$.

Two Player Differential game formulation of the System

Wind disturbance is a stochastic process defined by a velocity spectra. A vector representing a gust which varies spatially over an aircraft can be represented by including a rotational gust velocity around the pitch, roll and yaw axes [42]. The purpose of this, is to approximate its effect on flight dynamics on the three coordinate axes which translate to being the control surfaces.

From equation (3.1) and including the principle of two player zero-sum differential game as given in equation (3.11), the fixed wing UAV flight dynamics is represented as a two player game as in [42] to be

$$\dot{x} = A_{lat}x_{lat} + B1_{lat}u_{lat} + B2_{lat}u_g \tag{3.22}$$

where $B1_{lat}$ is the B matrix as described in equation (3.4), $B2_{lat}$ is the disturbance matrix and u_g is the wind gust angular velocity input and comprises of rotational gusts around the β, p, and r. The $B2_{lat}$ matrix derivation proceed with values as of the state space model in equation (3.3) as shown in Appendix.

$$\begin{bmatrix} \dot{\beta} \\ \dot{\rho} \\ \dot{r} \\ \dot{\Phi} \end{bmatrix} = \begin{bmatrix} Y_p & Y_\beta & Y_r - 1 & mgcos\theta_e \\ L_p & L_\beta & L_r & 0 \\ N_p & N_\beta & N_r & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \rho \\ \beta \\ r \\ \Phi \end{bmatrix} + \begin{bmatrix} L_{\delta a} & L_{\delta r} \\ Y_{\delta a} & Y_{\delta r} \\ N_{\delta a} & N_{\delta r} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} - \begin{bmatrix} Y_\beta & Y_p & Y_r \\ L_\beta & L_p & L_r \\ N_\beta & N_p & N_r \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \beta_g \\ p_g \\ r_g \\ \Phi_g \end{bmatrix} \tag{3.23}$$

The state space model in equation (3.23) can be thought of as a game against nature - the second player (wind disturbance); an unreasoning entity whose strategic choices affects 'your' payoff but which has no awareness of, or interest in the outcome of the game.

Definition 2 *When one player in a two person zero-sum game is not a reasoning entity capable of forethought and/or adaptive play which game theory assumes of players, the minimax solution concept may not apply. However, it is applicable depending on the goals of the rational player [11].*

This is exploited with the sole objective of coming up with a single controller countering the effects of wind disturbances. Therefore equation (3.17) will be used and equation (3.18) will be ignored (the effect of the disturbance is 'felt' from the system's interaction with the environment) in the design of the controller, as the main interest is to counter it's effect as is put forth by Savage in his principle of maximum regret; the rational player should minimize the maximum regret that he might feel after the nature's choices become known [11].

In practice, wind disturbance alleviation is accomplished by choosing one or more variables to be controlled in some manner and designing an algorithm around this goal [42]. The $B2_{lat}$ matrix in equation (3.23) is reduced to equation (3.24) by taking a rotational gust about the side slip angle, i.e. along Y with respect to β, L with respect to β, and N with respect to β moments such that it reduces to equation (3.24)

$$\begin{bmatrix} \dot{\beta} \\ \dot{\rho} \\ \dot{r} \\ \dot{\Phi} \end{bmatrix} = \begin{bmatrix} Y_p & Y_\beta & Y_r - 1 & mgcos\theta_e \\ L_p & L_\beta & L_r & 0 \\ N_p & N_\beta & N_r & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \rho \\ \beta \\ r \\ \Phi \end{bmatrix} + \begin{bmatrix} L_{\delta a} & L_{\delta r} \\ Y_{\delta a} & Y_{\delta r} \\ N_{\delta a} & N_{\delta r} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} - \begin{bmatrix} Y_\beta \\ L_\beta \\ N_\beta \\ 0 \end{bmatrix} \begin{bmatrix} \beta_g \\ p_g \\ r_g \\ \Phi_g \end{bmatrix}$$

$$(3.24)$$

This rotational gust becomes the competing input(second player) to the system. Thus, the system matrices become as in equation (3.25);

$$\begin{bmatrix} \dot{\beta} \\ \dot{\rho} \\ \dot{r} \\ \dot{\Phi} \end{bmatrix} = \begin{bmatrix} 0 & -1.4000 & 0 & 9.4953 \\ -12.8000 & -30.9000 & 14.4000 & 0 \\ -0.4480 & 1.4781 & -6.0800 & 0 \\ 1.0000 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \rho \\ \beta \\ r \\ \Phi \end{bmatrix} + \begin{bmatrix} 0 & 0.7412 \\ 61.4000 & 12.4000 \\ -3.6700 & -15.0000 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix}$$

$$+ \begin{bmatrix} 1.4000 \\ 30.9000 \\ -1.4781 \\ 0 \end{bmatrix} \begin{bmatrix} \beta_g \\ p_g \\ r_g \\ \Phi_g \end{bmatrix} \qquad (3.25)$$

3.3 Adaptive Controller

This chapter covers the basics of adaptive control, and then discusses the design of an online adaptive controller using reinforcement learning and differential games for active disturbance rejection for a smooth heading control in a UAV.

3.3.1 Adaptive Control

Adaptive control refers to a set of techniques that provide a systematic approach for automatic adjustment of controllers in *real time*, in order to achieve or maintain a desired level of control system performance when the parameters of the plant dynamic system are unknown and/or change in time [43].

An adaptive control system measures a certain performance index of the control system using the inputs, states, outputs and the known disturbances. From the comparison of the measured performance index and a set of given ones, the adaptation mechanism modifies the parameters of the adjustable controller in order to maintain the performance index of the control sytem close to the set of given ones. It is important to note that, while the design of a conventional feedback control system is oriented firstly toward the elimination of the effect of disturbances upon the controlled variables, the design of adaptive control systems is oriented firstly towards the elimination of the effect of parameter disturbances upon the performance of the control system. An adaptive control system can be viewed as a hierarchical feedback system where;

- Level 1: conventional feedback control
- Level 2: adaptation loop

The operation of the adaptation loop and its design relies upon the fundamental hypothesis: *For any possible values of plant model parameters, there is a controller with a fixed structure and complexity such that the specified performances can be achieved with appropriate values of the controller parameters* [43]. Hence, the work of the adaptation loop is to search for the “good” values of the controller parameters.

This stresses the importance of a control design for the underlying control problem, as well as the necessity of a priori information about the structure of the plant model. Therefore, an adaptive controller is not a “black box” which can solve a control problem in real time without an initial knowledge about the plant to be controlled.

Adaptive control consists of direct adaptive and indirect adaptive control.

Direct Adaptive Control

In most cases, the desired performance of a feedback control system can be specified in terms of the characteristics of a dynamic system which is a *realization* of the desired behaviour [43]. For instance, for both tracking and regulation objectives, the controller is designed such that for a given plant model, the closed loop system has characteristics of the desired dynamic system. Thus, the design problem can be formulated as in Figure 3.6 where the *reference model* is a realization of the system with desired performances. The use of direct adaptive control is

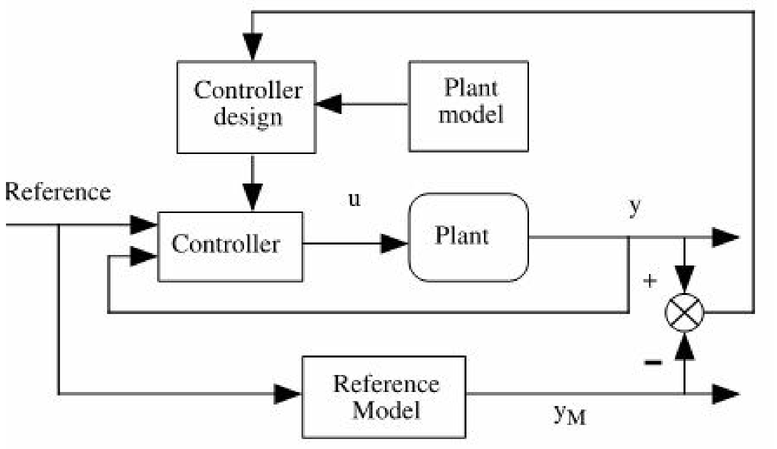


Figure 3.6: Direct Adaptive control principle [43]

limited by the hypotheses related to the underlying linear design such that the conditions for the existence of a feasible controller allowing for the closed loop to match the reference model are restrictive. This problem becomes even more difficult for multi-input-multi-output systems.

Indirect Adaptive Control

The idea behind indirect adaptive control is that a suitable controller can be designed on-line if a model of the plant is estimated on-line from the available input-output measurements. In order to design and tune a good controller, one has to specify the desired control loop performances [43]. This control method

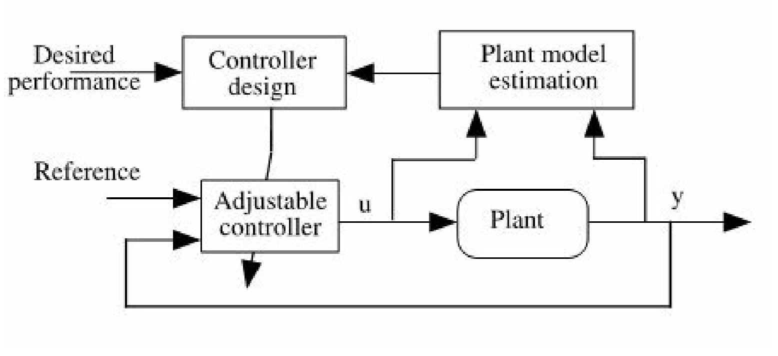


Figure 3.7: Indirect Adaptive control principle [43]

uses current plant model parameter estimates as if they are equal to the true ones in order to compute the controller parameters. The method is termed *indirect* because; there is on-line estimation of plant parameters and there is on-line computation of the controller parameters based on the current estimated plant model.

This idea of an indirect adaptive control scheme is presented in this work, where the controller is designed on line using a model of the plant which is estimated on line from the available input-output signals.

3.4 Design of an Adaptive Controller

A controller is developed using reinforcement learning and optimal control techniques. It is sub-divided into three parts;

- Feedforward neural network
- SARSA algorithm coupled with optimal control
- Back propagation algorithm.

The above components are discussed next to explain how they fit in, to form an adaptive controller.

Adaptive Control Block Diagram

The diagram in Figure 3.1 was transformed into an operational block diagram for adaptive control as shown in Figure 3.8. The guidance system is the basic reference tracking signal method.

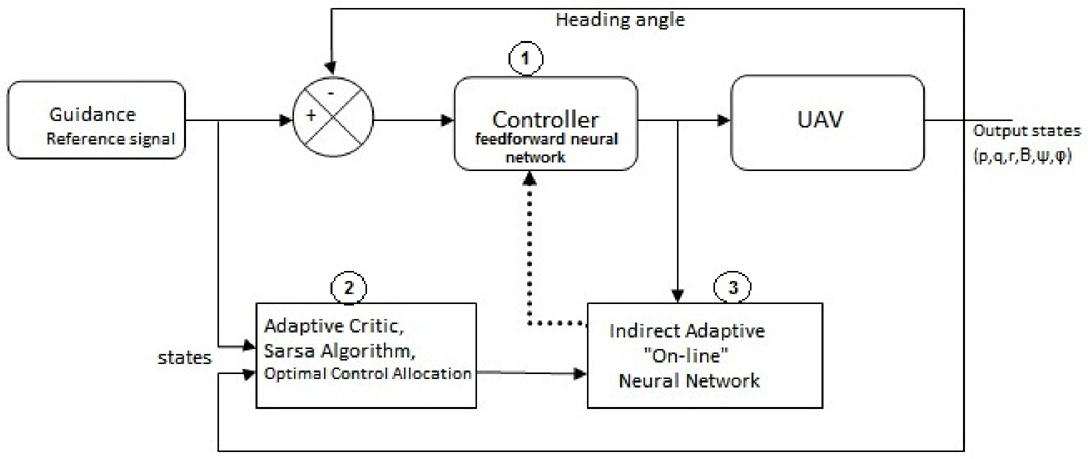


Figure 3.8: Adaptive control block diagram

3.4.1 Feedforward Neural network

It consists of a two-layer neural network with one hidden layer with three neurons as shown in Figure 3.9, that use *tanh* activation functions since *tanh* has stronger gradients, and one output layer with two neurons that use *tansig* activation functions since they are perform better for half spaces. It has two inputs; the heading error and the roll error.

At the first instance, the neural network weights are initialized arbitrarily.

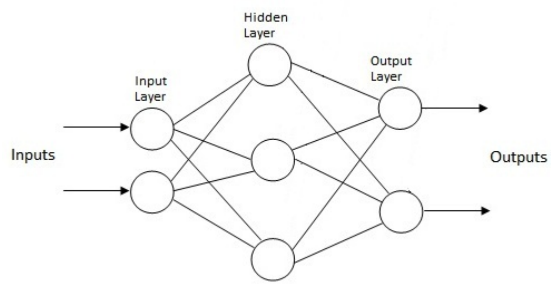


Figure 3.9: Feedforward neural network

The outputs of this feedforward net are the two the control signals to the system

- aileron deflection for the roll control, and
- rudder deflection for the yaw control.

This is shown in the block diagram of Figure 3.8 as block **1**.

3.4.2 SARSA Algorithm

The system is subjected to the control deflections (inputs) from the feedforward neural network and the outputs from the system are tapped.

SARSA algorithm coupled with optimal control is then implemented. The reference signals for the yawing angle (yawing rate) and roll angle (roll rate) are also directed to this block.

Value Function using Riccatti Co-efficient

From the state space model in equation (3.25), the associated Ricatti coefficient, **P** is calculated 'OFFLINE' using equation (3.19). It is formulated as the Algebraic Ricatti Equation(ARE); the system's operation time is infinite.

For easier calculation of **P** in MATLAB, the B2 matrix in equation (3.25) is conjoined with the disturbance matrix to form one input matrix. The Ricatti coefficient is used to calculate the State-Action value function (Cost function)

$$Q = X^T P X \tag{3.26}$$

for each state and action (control deflection) using a Lyapunov function as was discussed in Section 3.2.1. This equation is reformulated according to optimal control literature [8] to include the reference signals as

$$J = (X - Z)^T P (X - Z) \tag{3.27}$$

where X are states and Z are the reference signals. The states will capture the effect of wind disturbances has on the system in real time.

Reward function

The reward function for the corresponding action (control deflections) is calculated as the deviation of the actual state from the desired position - reference path [44]

$$r = -C_1(\psi - \psi_{ref})^2 - C_2(\theta - \theta_{ref})^2 - C_3(\phi - \phi_{ref})^2 \tag{3.28}$$

where C_s is a constant giving the ability to focus on the control of one of the angles rather than the other. In this work, the heading (ψ) error is used hence equation (3.28) reduces to

$$r = -C(\psi_{actual} - \psi_{ref})^2 \tag{3.29}$$

Value function

The total value function is then calculated, which is a sum of previous the state-action value function, the current reward and state-action value function hence, State-Action – Reward – State-Action (SARSA) Algorithm as

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{3.30}$$

with values as explained in Section 2.2. This is updated as the value function for the next cycle of learning.

According to [45], it is allowed to have a one step gradient search of the value function for ease of real time implementation and less computational burden but it is not the most efficient. This is calculated as the discounted current value function minus the previous value function plus the current reward which reduces to

$$\delta_t = r_{t+1} + \gamma Q(s_{t+1}) - Q(s_t) \tag{3.31}$$

This is referred to as a temporal difference error in SARSA. After each action selection, the critic evaluates the new state to determine whether things have gone better or worse. This temporal difference error will be the reinforcement signal.

Optimal Control Signal

According to optimal control principles, an optimal control effort is given as

$$u = -KX \tag{3.32}$$

but according to differential games this is represented as in equation (3.17)

$$u^{\star} = -1/2R^{-1}B^{T}\nabla V^{\star} \tag{3.33}$$

where $\nabla V^{\star}$ is taken as the change in the value function – which in this work, the temporal difference is used as the reinforcement as explained above. This optimal control implemented this way, is considered to be compensated of the wind disturbances, error(s) and deviation from the target states hence the best control effort.

As Section 3.2.1 puts it, "wind disturbance alleviation is accomplished by choosing one or more variables to be controlled in some manner and designing an algorithm around this goal". This is what has been formulated herein as active disturbance rejection without explicitly measuring the disturbance affecting the system but rather getting the real effect of how the disturbance affect the system with reference to the target states/reference and designing a controller around that. This part is shown in Figure 3.8 as **2**.

3.4.3 Back Propagation

The back propagation algorithm updates the feedforward neural network weights using back propagation algorithm. This is achieved by getting the deltas for a normal back propagation algorithm

$$\delta_1 = \Phi(\hat{u}^\star - \hat{u}) \tag{3.34}$$

$$\delta_2 = V(1 - \Phi^2)(\hat{u}^\star - \hat{u}) \tag{3.35}$$

where

- δ_1 is the update for the input weights of the neural network
- δ_2 is the update for the middle layer weights
- Φ is the feedforward neural network middle layer weight matrix.
- $\hat{u}^\star$ is the best action (control deflection) that should have been implemented using the State Action Reward State Action algorithm as discussed above; and
- $\hat{u}$ the actual control deflection implemented through the feedforward neural network.

A learning rate of $\alpha = 0.6$ was arbitrary chosen in the update of the neural network feedforward weights.

$$V = V + \alpha\delta_1 \tag{3.36}$$

$$W = W + \alpha\delta_2 \tag{3.37}$$

where

- V is the output weights and,
- W the input weights.

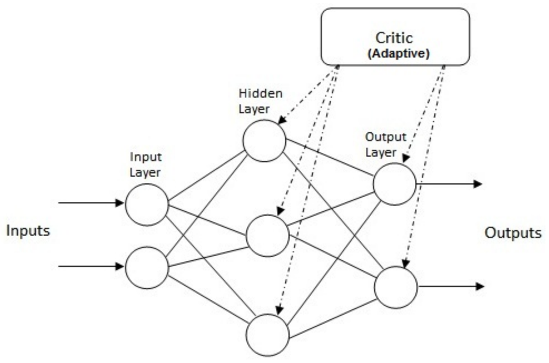
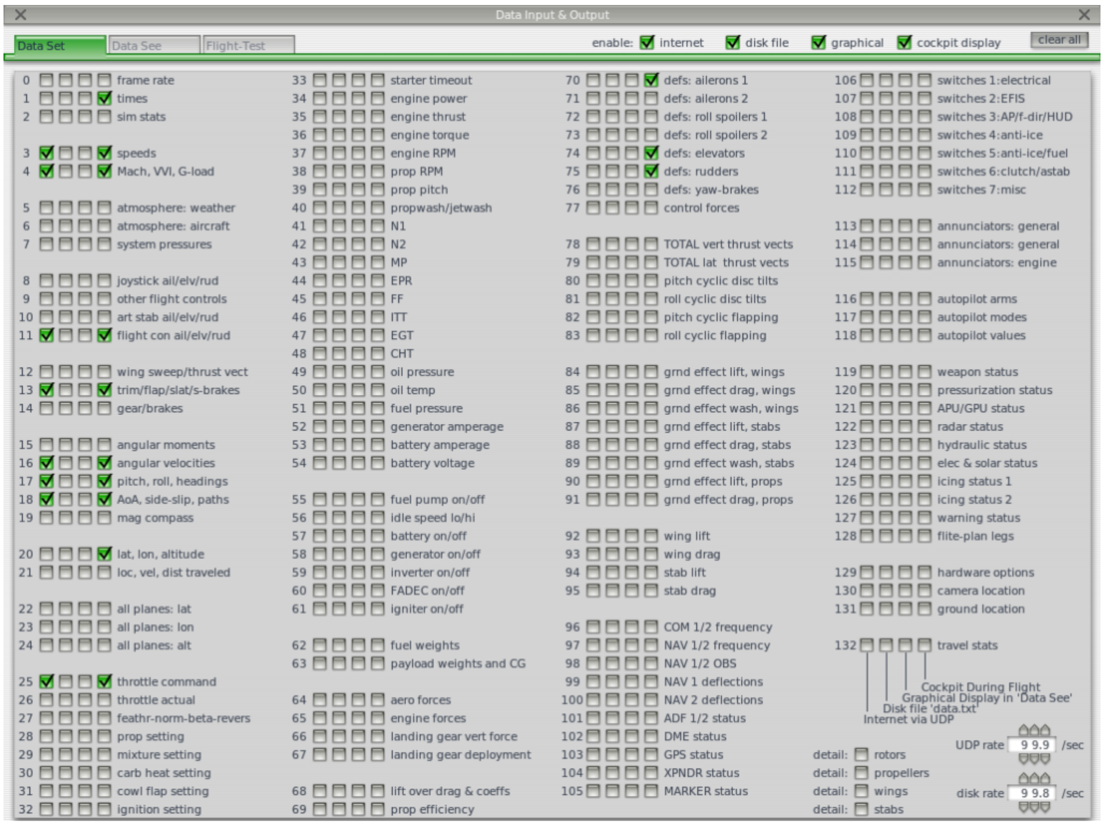


Figure 3.10: Evaluative Feedback

The process of back propagation is as illustrated in Figure 3.10. The error in the control deflection is being corrected in the next control action through an update of feedforward neural network weights thus slowly taking our actions to the best available control effort in each consecutive cycle thus adaptation as was illustrated in Figure 2.3. This is consistent with the actor - critic structure of reinforcement learning discussed in Sec. 3.2.1 and illustrated in Figure 3.5. This process in shown as **block 3** in the experimental block diagram of Figure 3.8.

3.5 X-Plane Integration with MATLAB

X-Plane is a powerful flight simulator for personal computers. It is not a game but an engineering tool that can be used to predict the flying qualities of fixed and rotary wing aircraft with considerable accuracy [46]. This makes it a useful tool to predict and test the performance of an aircraft and its characteristics. It has the capacity to send and receive data to and from other devices using the User Datagram Protocol(UDP). UDP uses a simple transmission method without explicit handshaking protocol, ordering or data integrity and thus, provides fast communication due to less overhead of network level processing thus suit-



Data Input & Output

Data Set | Data See | Flight-Test

enable: ✓ internet ✓ disk file ✓ graphical ✓ cockpit display   clear all

#	Internet via UDP	Disk file 'data.txt'	Graphical Display in 'Data See'	Cockpit During Flight	
0	☐	☐	☐	☐	frame rate
1	☐	☐	☐	✓	times
2	☐	☐	☐	☐	sim stats
3	✓	☐	☐	✓	speeds
4	✓	☐	☐	✓	Mach, VVI, G-load
5	☐	☐	☐	☐	atmosphere: weather
6	☐	☐	☐	☐	atmosphere: aircraft
7	☐	☐	☐	☐	system pressures
8	☐	☐	☐	☐	joystick ail/elv/rud
9	☐	☐	☐	☐	other flight controls
10	☐	☐	☐	☐	art stab ail/elv/rud
11	✓	☐	☐	✓	flight con ail/elv/rud
12	☐	☐	☐	☐	wing sweep/thrust vect
13	✓	☐	☐	✓	trim/flap/slat/s-brakes
14	☐	☐	☐	☐	gear/brakes
15	☐	☐	☐	☐	angular moments
16	✓	☐	☐	✓	angular velocities
17	✓	☐	☐	✓	pitch, roll, headings
18	✓	☐	☐	✓	AoA, side-slip, paths
19	☐	☐	☐	☐	mag compass
20	☐	☐	☐	✓	lat, lon, altitude
21	☐	☐	☐	☐	loc, vel, dist traveled
22	☐	☐	☐	☐	all planes: lat
23	☐	☐	☐	☐	all planes: lon
24	☐	☐	☐	☐	all planes: alt
25	✓	☐	☐	✓	throttle command
26	☐	☐	☐	☐	throttle actual
27	☐	☐	☐	☐	feathr-norm-beta-revers
28	☐	☐	☐	☐	prop setting
29	☐	☐	☐	☐	mixture setting
30	☐	☐	☐	☐	carb heat setting
31	☐	☐	☐	☐	cowl flap setting
32	☐	☐	☐	☐	ignition setting
33	☐	☐	☐	☐	starter timeout
34	☐	☐	☐	☐	engine power
35	☐	☐	☐	☐	engine thrust
36	☐	☐	☐	☐	engine torque
37	☐	☐	☐	☐	engine RPM
38	☐	☐	☐	☐	prop RPM
39	☐	☐	☐	☐	prop pitch
40	☐	☐	☐	☐	propwash/jetwash
41	☐	☐	☐	☐	N1
42	☐	☐	☐	☐	N2
43	☐	☐	☐	☐	MP
44	☐	☐	☐	☐	EPR
45	☐	☐	☐	☐	FF
46	☐	☐	☐	☐	ITT
47	☐	☐	☐	☐	EGT
48	☐	☐	☐	☐	CHT
49	☐	☐	☐	☐	oil pressure
50	☐	☐	☐	☐	oil temp
51	☐	☐	☐	☐	fuel pressure
52	☐	☐	☐	☐	generator amperage
53	☐	☐	☐	☐	battery amperage
54	☐	☐	☐	☐	battery voltage
55	☐	☐	☐	☐	fuel pump on/off
56	☐	☐	☐	☐	idle speed lo/hi
57	☐	☐	☐	☐	battery on/off
58	☐	☐	☐	☐	generator on/off
59	☐	☐	☐	☐	inverter on/off
60	☐	☐	☐	☐	FADEC on/off
61	☐	☐	☐	☐	igniter on/off
62	☐	☐	☐	☐	fuel weights
63	☐	☐	☐	☐	payload weights and CG
64	☐	☐	☐	☐	aero forces
65	☐	☐	☐	☐	engine forces
66	☐	☐	☐	☐	landing gear vert force
67	☐	☐	☐	☐	landing gear deployment
68	☐	☐	☐	☐	lift over drag & coeffs
69	☐	☐	☐	☐	prop efficiency
70	☐	☐	☐	✓	defs: ailerons 1
71	☐	☐	☐	☐	defs: ailerons 2
72	☐	☐	☐	☐	defs: roll spoilers 1
73	☐	☐	☐	☐	defs: roll spoilers 2
74	☐	☐	☐	✓	defs: elevators
75	☐	☐	☐	✓	defs: rudders
76	☐	☐	☐	☐	defs: yaw-brakes
77	☐	☐	☐	☐	control forces
78	☐	☐	☐	☐	TOTAL vert thrust vects
79	☐	☐	☐	☐	TOTAL lat thrust vects
80	☐	☐	☐	☐	pitch cyclic disc tilts
81	☐	☐	☐	☐	roll cyclic disc tilts
82	☐	☐	☐	☐	pitch cyclic flapping
83	☐	☐	☐	☐	roll cyclic flapping
84	☐	☐	☐	☐	grnd effect lift, wings
85	☐	☐	☐	☐	grnd effect drag, wings
86	☐	☐	☐	☐	grnd effect wash, wings
87	☐	☐	☐	☐	grnd effect lift, stabs
88	☐	☐	☐	☐	grnd effect drag, stabs
89	☐	☐	☐	☐	grnd effect wash, stabs
90	☐	☐	☐	☐	grnd effect lift, props
91	☐	☐	☐	☐	grnd effect drag, props
92	☐	☐	☐	☐	wing lift
93	☐	☐	☐	☐	wing drag
94	☐	☐	☐	☐	stab lift
95	☐	☐	☐	☐	stab drag
96	☐	☐	☐	☐	COM 1/2 frequency
97	☐	☐	☐	☐	NAV 1/2 frequency
98	☐	☐	☐	☐	NAV 1/2 OBS
99	☐	☐	☐	☐	NAV 1 deflections
100	☐	☐	☐	☐	NAV 2 deflections
101	☐	☐	☐	☐	ADF 1/2 status
102	☐	☐	☐	☐	DME status
103	☐	☐	☐	☐	GPS status
104	☐	☐	☐	☐	XPNDR status
105	☐	☐	☐	☐	MARKER status
106	☐	☐	☐	☐	switches 1:electrical
107	☐	☐	☐	☐	switches 2:EFIS
108	☐	☐	☐	☐	switches 3:AP/f-dir/HUD
109	☐	☐	☐	☐	switches 4:anti-ice
110	☐	☐	☐	☐	switches 5:anti-ice/fuel
111	☐	☐	☐	☐	switches 6:clutch/astab
112	☐	☐	☐	☐	switches 7:misc
113	☐	☐	☐	☐	annunciators: general
114	☐	☐	☐	☐	annunciators: general
115	☐	☐	☐	☐	annunciators: engine
116	☐	☐	☐	☐	autopilot arms
117	☐	☐	☐	☐	autopilot modes
118	☐	☐	☐	☐	autopilot values
119	☐	☐	☐	☐	weapon status
120	☐	☐	☐	☐	pressurization status
121	☐	☐	☐	☐	APU/GPU status
122	☐	☐	☐	☐	radar status
123	☐	☐	☐	☐	hydraulic status
124	☐	☐	☐	☐	elec & solar status
125	☐	☐	☐	☐	icing status 1
126	☐	☐	☐	☐	icing status 2
127	☐	☐	☐	☐	warning status
128	☐	☐	☐	☐	flite-plan legs
129	☐	☐	☐	☐	hardware options
130	☐	☐	☐	☐	camera location
131	☐	☐	☐	☐	ground location
132	☐	☐	☐	☐	travel stats

detail: ☐ rotors
detail: ☐ propellers
detail: ☐ wings
detail: ☐ stabs

UDP rate 9 9.9 /sec
disk rate 9 9.8 /sec

Figure 3.11: X-Plane Test platform

able for time sensitive real time applications as X-Plane [47]. X-Plane is able to send and receive data at 99.9 data packets per second via UDP configurable sockets. Each data packet is configured to carry specific aircraft parameters that has been selected in the check boxes provided in the X-Plane's Input and Output data interface as shown in Figure 3.11. The selected parameters on X-Plane, can be received by MATLAB/SIMULINK using UDP through loopback addresses. This is made possible by using X-Plane Communication Library which enables integration between MATLAB/SIMULINK and X-Plane as in Figure 3.12. The received packets are repackaged for use in MATLAB/SIMULINK environment using the X-Plane Fixed Wing UDP Receiver block and the data to be sent to X-Plane is repacked in a format that can be received and processed by X-Plane

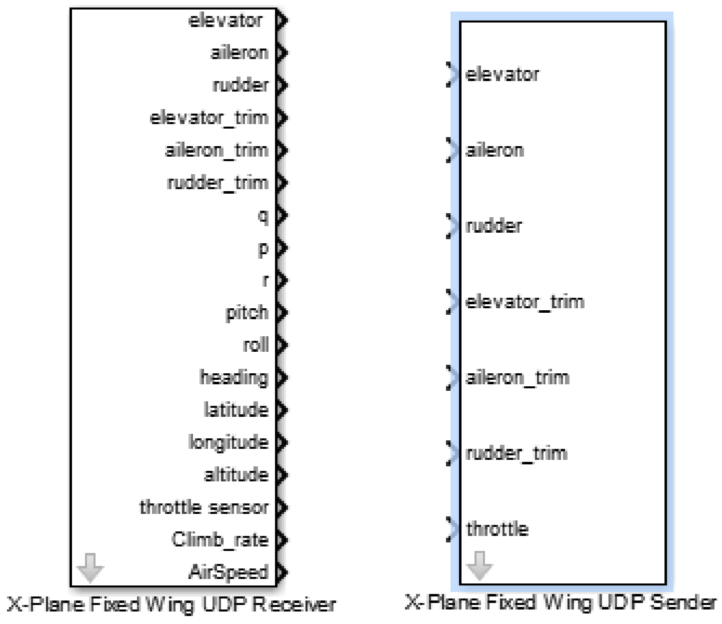
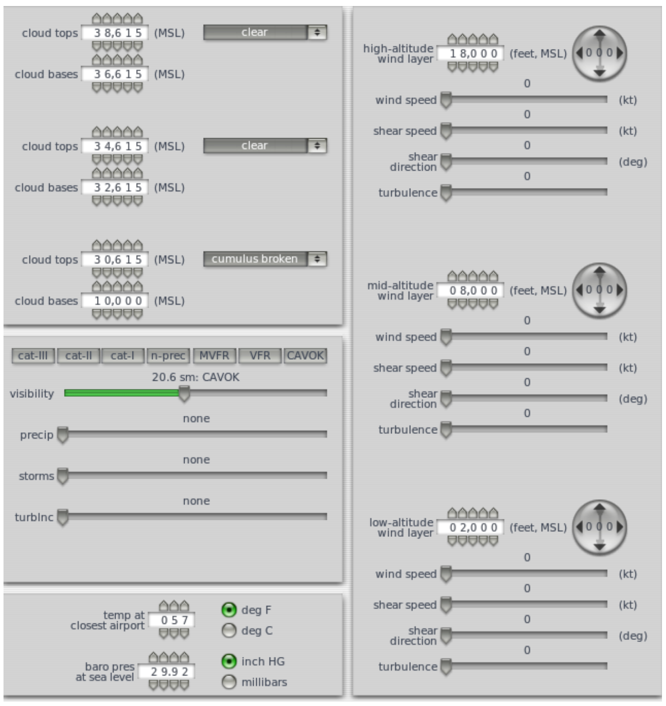


Figure 3.12: X-Plane SIMULINK communication library for fixed wing UAVs

using the X-Plane Fixed Wing UDP Sender block.

X-Plane also has the functionality of altering the weather conditions i.e. wind speed, shear speed and direction, and turbulence of an altitude layer as shown in Figure 3.13. This allows for close to real life flying conditions hence a robust simulation environment. X-Plane will be used to visualize the effectiveness and robustness of the designed controller through an actual flight envelope using different weather conditions which will be adjusted in real-time. Classification of weather factors, wind speeds and their effect, and the impact of weather factors on recorded aviation accidents [48] is provided in Appendix III.



cloud tops 38,615 (MSL) clear
cloud bases 36,615 (MSL)

cloud tops 34,615 (MSL) clear
cloud bases 32,615 (MSL)

cloud tops 30,615 (MSL) cumulus broken
cloud bases 10,000 (MSL)

cat-III | cat-II | cat-I | n-prec | MVFR | VFR | CAVOK

visibility: 20.6 sm: CAVOK
precip: none
storms: none
turbInc: none

temp at closest airport 057 [✓ deg F] deg C
baro pres at sea level 29.92 [✓ inch HG] millibars

high-altitude wind layer 18,000 (feet, MSL) 000
wind speed 0 (kt)
shear speed 0 (kt)
shear direction 0 (deg)
turbulence 0

mid-altitude wind layer 08,000 (feet, MSL) 000
wind speed 0 (kt)
shear speed 0 (kt)
shear direction 0 (deg)
turbulence 0

low-altitude wind layer 02,000 (feet, MSL) 000
wind speed 0 (kt)
shear speed 0 (kt)
shear direction 0 (deg)
turbulence 0

Figure 3.13: X-Plane atmospheric layers

3.5.1 X-Plane UAV plane model

The UAV model plane that was used for simulations is an inbuilt model that comes with the X-Plane software and was designed by the Great Planes company and has the specifications as shown in the Table 3.1; The values that were used for

Table 3.1: Great Planes, GP-PT-60 UAV specification

Mass, m	3180 g
Wingspan, b	1.8 m
Wing area, S	0.54 m^2
Fuselage length (main body of the UAV)	1.35 m

mathematical modelling of the UAV are different from the X-Plane UAV model. The characteristics of the modelled Ultra Stick 25E UAV are as shown in Table

3.2;

Table 3.2: Characteristics of UltraStick

Mass, m	1900 g
Wingspan,b	1.2 m
Wing area, S	0.32 m^2
Fuselage length	1.09 m
Payload	2250 g

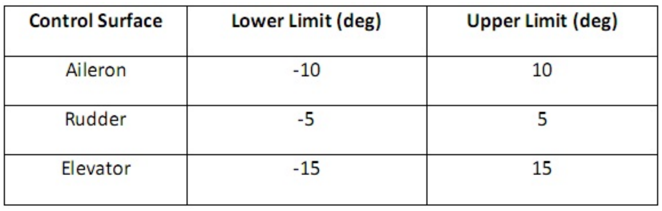
As [49] puts it, much of the lateral dynamics of a UAV are determined by the aerodynamic stability derivatives, which are functions of the vehicle's configuration and not of inertia; weight, size and speed. But the relative mass which is a dimensionless number ($= m/\rho Sb$) affects the vehicle response. Thus the vehicle response will be affected as opposed to the vehicle dynamics.

3.5.2 Actuator Models

Actuator dynamics of the aileron and rudder are both modelled by first order dynamics to account for actuator time lag in the simulation model. The actuators rotation angle saturation limits are limited to the maximum mechanical deflection angles of the control surfaces. These limits are given in Table 3.3 for the GP-PT 60 UAV model in X-Plane. They are described by the transfer function:

$$\delta_{a,r} = \tfrac{20}{s+20} \tag{3.38}$$

Table 3.3: Control surface saturation limits



Control Surface	Lower Limit (deg)	Upper Limit (deg)
Aileron	-10	10
Rudder	-5	5
Elevator	-15	15

where $\delta \leq \delta_{max}$. The outputs from the actuators are the per unit control surface deflections which are given by;

$$\delta_{a,r} = \tfrac{20}{s+20} u_{a,r} \tag{3.39}$$

where $u_{a,r}$ is the aileron and/or rudder control signal.

3.6 Experimental Setup

The experimental setups are as in Figure 3.14 and Figure 3.15. Figure 3.14 shows the setup that was used for the mathematical simulation in MATLAB/SIMULINK, and Figure 3.15 shows the setup used for real time simulation in MATLAB/SIMULINK and linked with X-Plane platform.

The adaptive critic block of Figure 3.14 and Figure 3.15 is expanded to view inside for the MATLAB functions implementing the reinforcement learning algorithm as shown in Figure 3.16. The experiments using this setups are;

i) Heading rate simulations using the mathematical model in Figure 3.14

ii) Disturbance rejection test using (i) above.

iii) Real time heading angle simulations in X-Plane using Figure 3.15.

iv) Disturbance rejection tests in X-Plane using (iii) above.

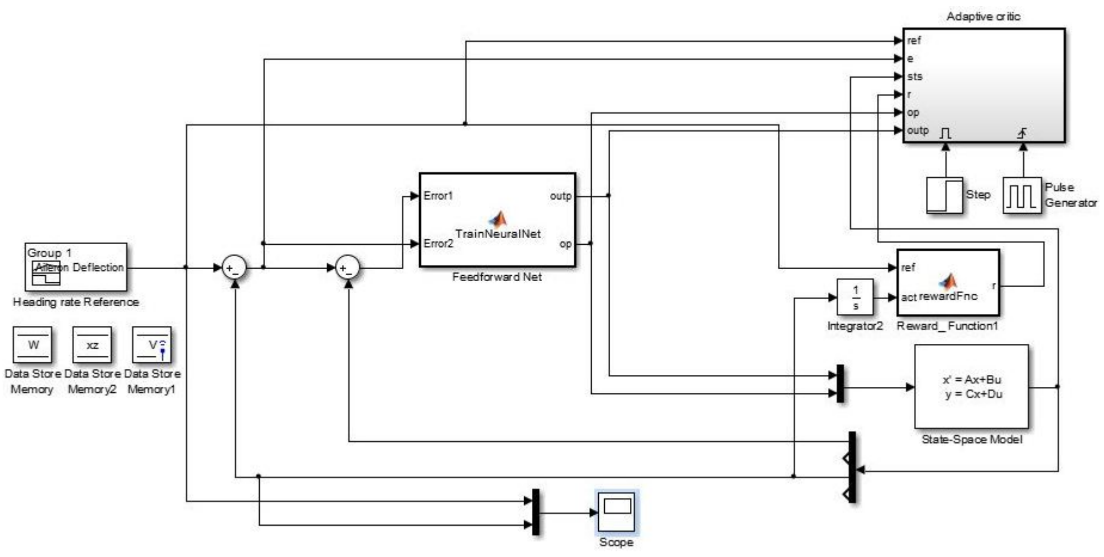


Figure 3.14: Experimental Setup for Mathematical Model Simulations

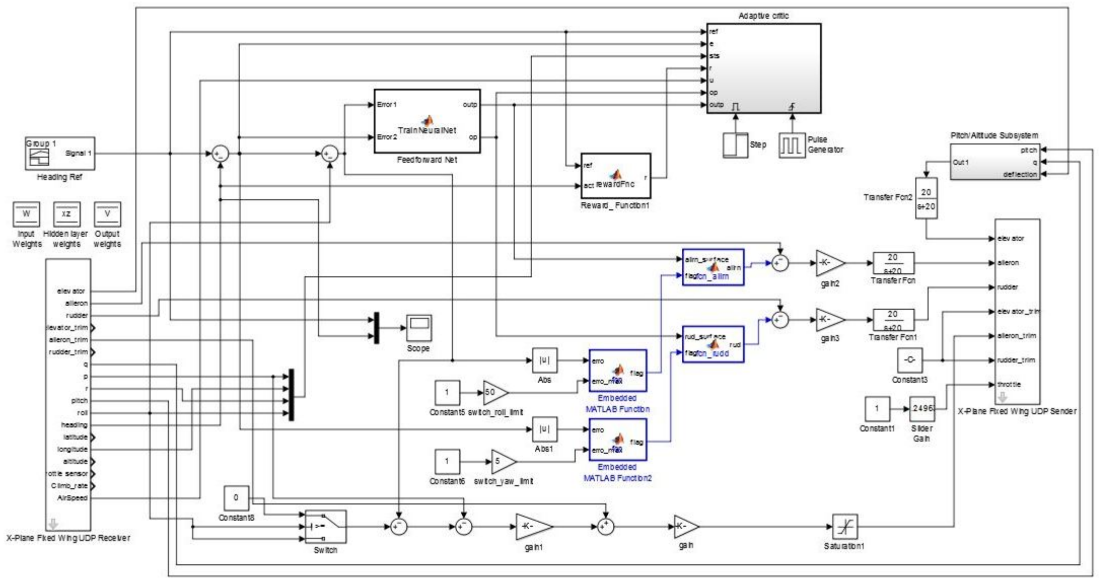


Figure 3.15: Experimental Setup for real-time Simulations in X-Plane

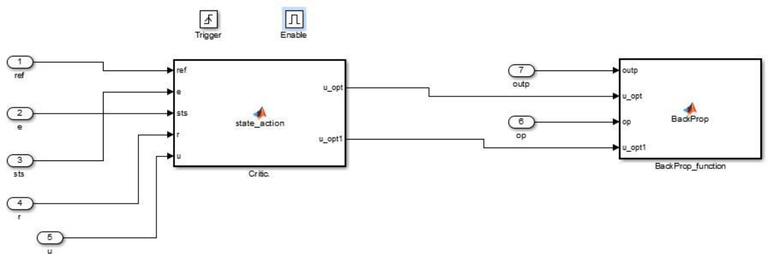


Figure 3.16: The functions performing Adaptation in the Setups

3.7 Summary

In this chapter, the basics of flight control were introduced and thereafter a UAV mathematical model was been derived from the given properties. The model was formulated as a two player differential game, with wind disturbances acting as the second competing player. Game theoretic methods were used to reduce the control problem into a game against nature. An iterative method for a generalized solution of the differential game using SARSA, a reinforcement learning technique was presented where an adaptive approach using an actor-critic structure utilizing neural networks and back propagation algorithm is used. The actor implements control structure while the critic approximates the value function and hence an approximation of the optimal control law which improves the control effort being implemented by the actor. Thereafter, a brief introduction of X-Plane, a visualization software for aircrafts is discussed, and its integration with MATLAB/SIMULINK to aid visualization of the designed controller.

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 Analysis of the mathematical model

The state space matrices for the system are lifted from equation (3.25) as;

$$A = \begin{bmatrix} 0 & -1.4000 & 0 & 9.4953 \\ -12.8000 & -30.9000 & 14.4000 & 0 \\ -0.4480 & 1.4781 & -6.0800 & 0 \\ 1.0000 & 0 & 0 & 0 \end{bmatrix}$$

$$B1 = \begin{bmatrix} 0 & 0.7412 \\ 61.4000 & 12.4000 \\ -3.6700 & -15.0000 \\ 0 & 0 \end{bmatrix}$$

$$B2 = \begin{bmatrix} 1.4000 \\ 30.9000 \\ -1.4781 \\ 0 \end{bmatrix}$$

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

And D matrix is taken to zero matrix of size 4 by 2.

The system open-loop poles are;

- -32.2627
- 3.4009
- -5.4576
- -2.6606

This shows that the system is open-loop unstable, because of the pole at 3.4009. Figure 4.1 shows the singular value decomposition of the model. It can be seen that the model has two dominant modes. Both modes have a constant gain at low

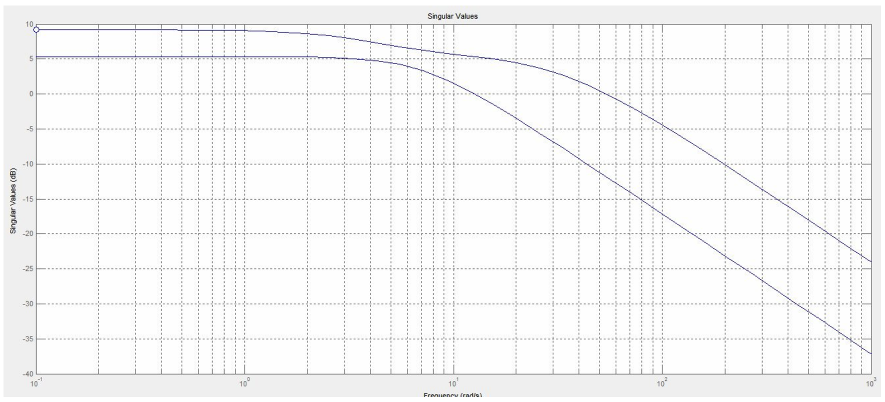


Figure 4.1: Singular value decomposition of the model

frequencies and relatively low gain at high frequencies. The bottom mode has a slightly lower gain as compared to the first. The Hankel singular values[1] were also plotted to see the interaction of the modes in the model and if the model could be reduced to a lower order. As the plot shows, the first three states contribute

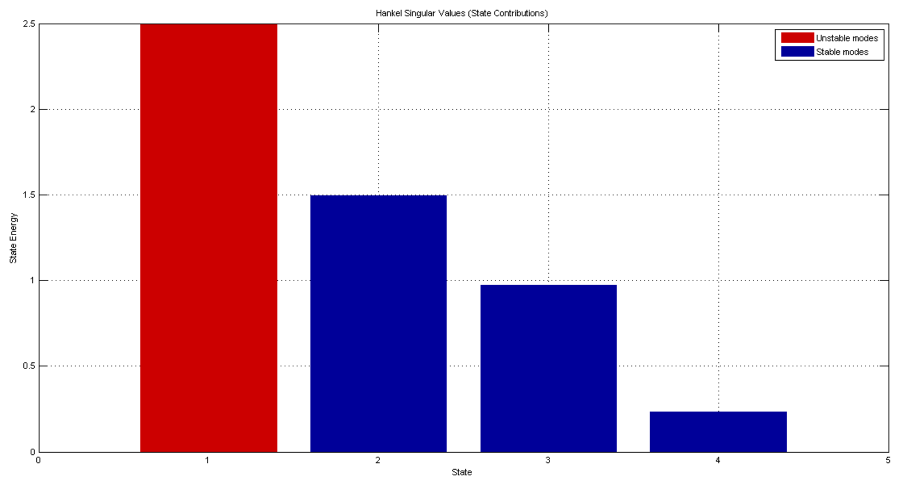


Figure 4.2: Hankel Singular Value Analysis

immensely in the system. The fourth state contributes the least. The first two states have the highest state energies and thus contribute more i.e. rolling rate and side-slip angle as compared to the third state i.e. yawing rate. It should be noted that the first state which also has the highest energy is an unstable state. Due to the interaction of the different states, a non-dominant state/mode can affect the response of a 'dominant' mode. It was not feasible to reduce the model to a lower order. Pole-zero mapping of the open-loop system was done and is as shown in Figure 4.3.

Three things can be noted from the pole-zero mapping;

[1]Hankel singular values define the "energy" of each state in the system. Keeping larger energy states of a system preserves most of its characteristics in terms of stability, frequency, and time responses.

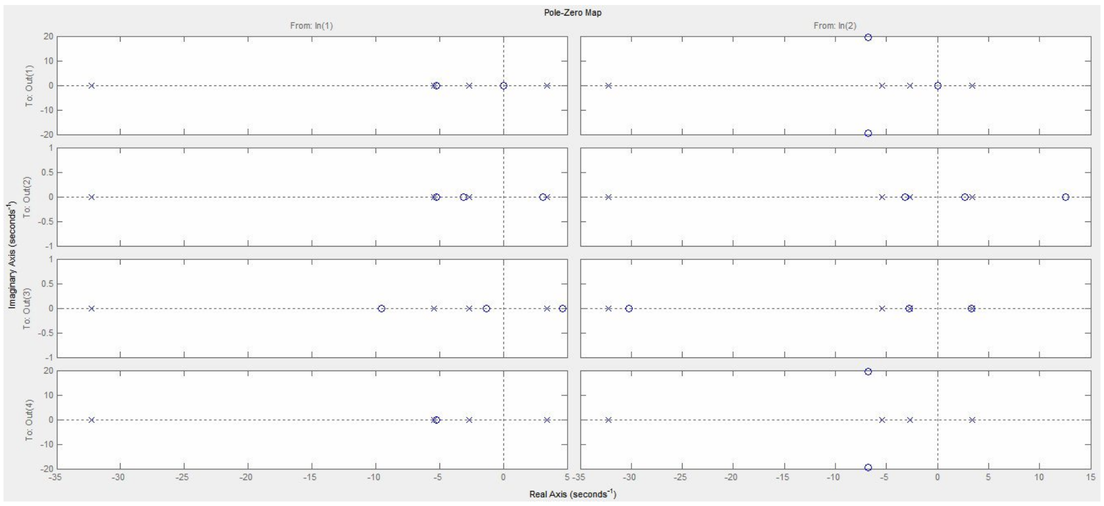


Figure 4.3: Input-to-Output Pole-zero Map of the open loop system

- All the four states have one pole in the right hand side of the jw axis.
- All the states have at least a zero to the right of the s-plane. This implies that with the two inputs, the system will exhibit non-minimum phase behaviour.
- State 1, has a zero at the origin from both inputs. The system will have zero gain at zero frequency.

4.2 Configuration of the Reference Controller

In order to verify the performance of the designed controller, a reference controller is introduced so that a comparison is done between the performance of the two controllers.

4.2.1 Proportional Integral Derivative (PID) Controller

The state space model given in equation (3.1) and equation (3.2) was used to design PID controllers and hence get the response of the heading rate in MATLAB/SIMULINK using the block diagram shown in Figure 4.4. This similar architecture of PIDs in cascade was applied in [23].

Due to the complex interaction of the states, normal PID tuning rules were

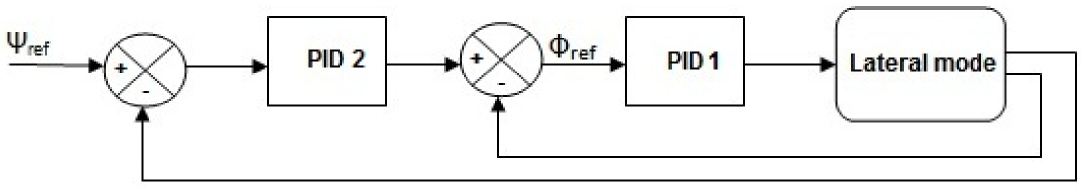


Figure 4.4: PID controllers in cascade

not feasible. Thus, gains were tuned using MATLAB/SIMULINK PID interactive tuner[2] where gains of the two PIDs were tuned sequentially. A parallel PID structure of the form

$$G_c = P + I\tfrac{1}{s} + D\tfrac{N}{1+N\frac{1}{s}} \tag{4.1}$$

was used, where P represents proportional gain, K integral gain, D derivative gain and N the filter coefficient.

The PID tuning requirements for the reference controller were;

- Closed loop stability since it was established that the system is open-loop unstable.

[2]Automatically computes an initial PID design with a balance between performance and robustness. A GUI helps one interactively refine the performance of the PID controller to meet the design requirements.

- Minimum overshoot as a result from both inputs operating simulteanously
- Realistic PID gains
- Robust reference tracking ability

The gains realized using the PID interactive tuner in Matlab/Simulink were as shown in Table 4.1 for the corresponding PID 1 and PID 2 as shown in Figure 4.4. The overall transfer function using this gains were;

Table 4.1: PID gains

	PID 1, δ_a	PID 2, δ_r
K_p	-3.707	-3.749
K_i	-14.548	-10.825
K_d	0.043	-0.080
N	13.514	42.346

$$PID1, \delta_a = \frac{-3.704s^2 - 14.82s - 1.077}{s^2 + 0.074s}$$

due to aileron input, and

$$PID2, \delta_r = \frac{-3.751s^2 - 10.91s - 0.2556}{s^2 + 0.02362s}$$

due to rudder input.

A MATLAB function, *loopsens* which stands for loop sensitivity was used to investigate the stability and sensitivity of the tuned system in closed loop. The architecture for the *loopsens* is as shown in Figure 4.5. Notice that the function does not include a reference signal.

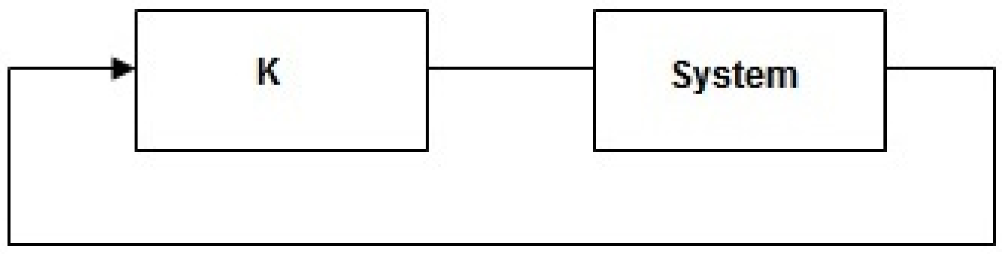


Figure 4.5: One DoF architecture for a closed loop system

The function implemented was as;

$[Loops.Stable] = loopsens(sys, K)$

where the *sys* was the open loop state space model and K the controller which was a matrix of the two inputs The output from the *loopsens* function was; $Loops.Stable \quad ans = 1$ which indicates that the system is closed loop stable using the tuned PID gains given above.

Proportional Integral (PI) Controller

A similar procedure and design specifications was used for PI controllers as was for PID controllers. The structure for the PI controller is

$$G_c = P + I\frac{1}{s}$$

The gains realized through interactive tuning were as

	PI 1, δ_a	PI 2, δ_r
K_p	-3.707	-0.976
K_i	-14.548	-3.565

The function *loopsens* was used to investigate the closed loop stability status of the system. The output showed that the system is closed loop stable using the

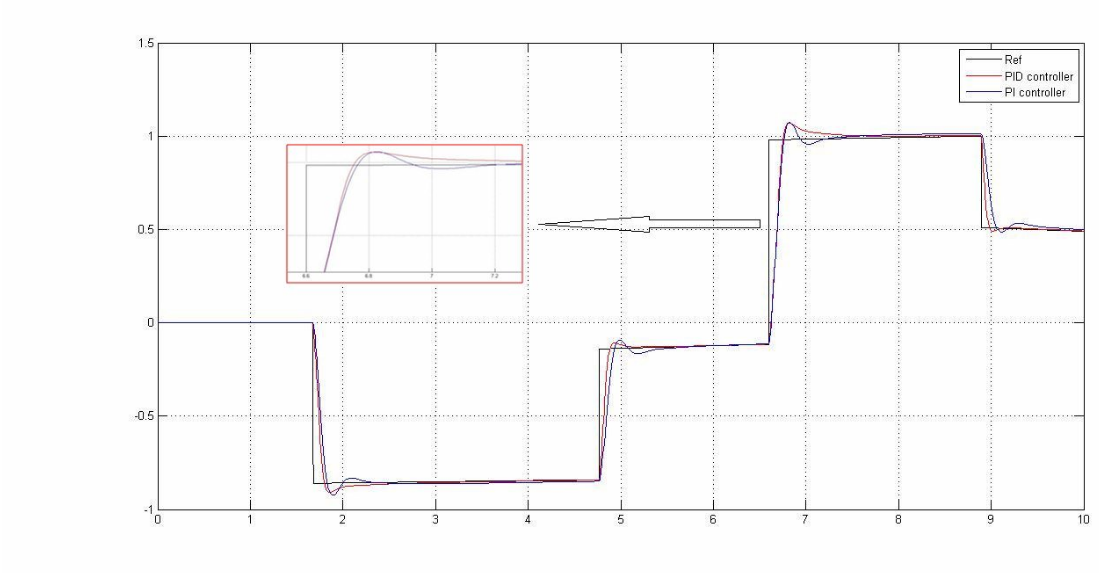


Figure 4.6: Heading rate response of the PID and PI controllers

PI gains above.

4.2.2 Comparison of PI and PID Control

Since the two control methods satisfied most of the design specifications, we investigated more to see which of the two had superior performance. The responses of the two controllers were plotted alongside each other as shown in Figure 4.6. A closer look at Figure 4.6 as is zoomed shows that the response of the PID control

Table 4.2: Comparison of PID and PI controllers

Step Info	PID Controllers	PI Controllers
$Risetime$	0.0597	0.1089
$SettlingTime$	5.2604	Did not settle within the set bounds of 0.5%
$Overshoot(Percentage)$	4.4260	7.5474

is slightly better than that of PI control; it has a lesser overshoot and tracks the reference better. Comparison was made based on a step input to get their characteristics as shown in Table 4.2. It can be deduced from the comparison, that PID control performs better than the corresponding PI control for this application. Therefore, the PID controllers were chosen and assimilated as the **reference** controller.

4.2.3 Validating the reference Controller

In [23], PID controllers in cascade as in Figure 4.4 were used to control the heading of a fixed wing UAV. The design technique employed was the root locus method and the heading response to a step input was obtained as in Figure 4.7. There

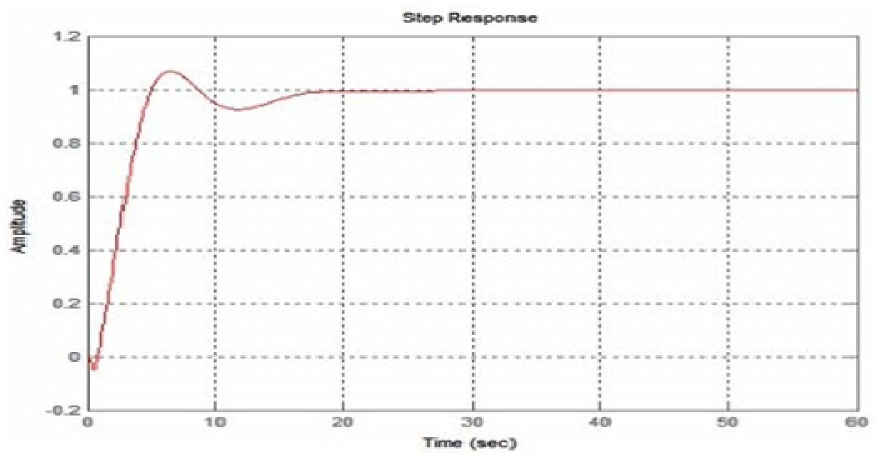


Figure 4.7: Heading angle response [23]

is a relationship in the graphs with what was achieved using the PID interactive tuning method in Matlab as shown in Figure 4.8. The main difference is the time scale whereby the reference PID controller system settled after 0.253 seconds for a 2% settling time bound whereas that given by [23] settled after 38 seconds. This big difference could be attributable to the fact that results shown in Figure

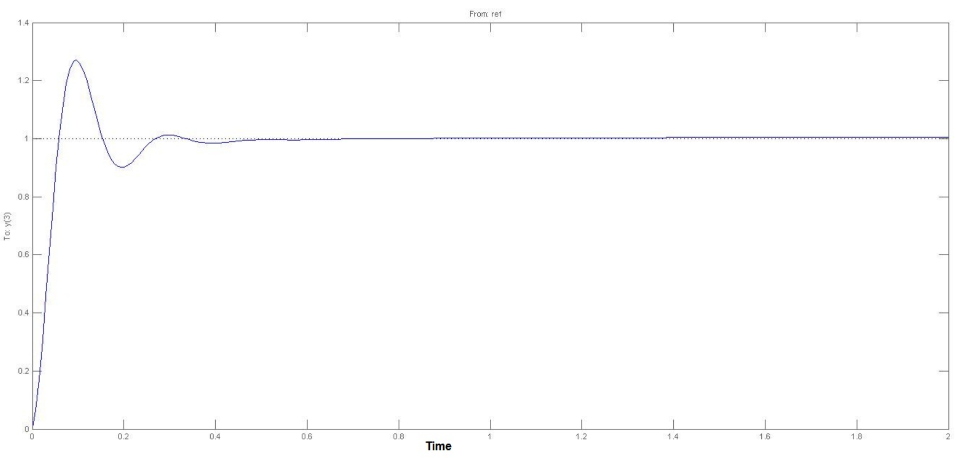


Figure 4.8: Heading rate response of PID control

4.7 show the heading angle response whereas Figure 4.8 shows the heading rate response. Even so, there is a great relationship between the two and thus the PID controller was good enough to be used as a reference controller.

4.3 Performance of the Controllers

4.3.1 Performance of adaptive controller using the model

The response, that is the heading rate is as shown in Figure 4.9 for the UAV mathematical model. It can be seen that the controller utilizing reinforcement learning has better tracking response as compared to a well tuned PID reference controller; has no overshoots and tracks the reference as closely as possible. Similar results were achieved in [32] where PCH and neural network adaptation was used.

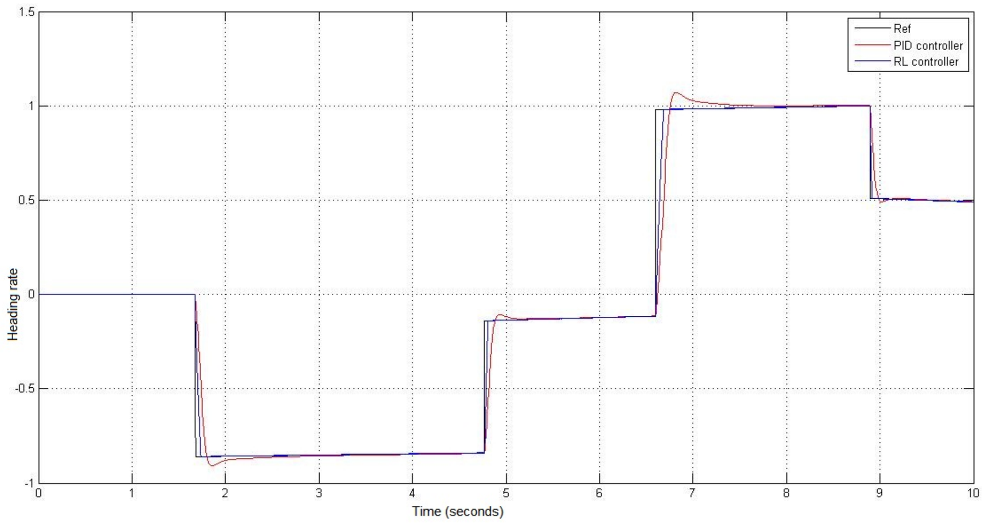


Figure 4.9: Heading rate response of the PID and RL controllers

4.3.2 Performance of the adaptive controller using UAV model in X-Plane

The designed controllers above were integrated to become real time controllers for a UAV described in Section 3.5.1 in the X-Plane platform. The UAV in X-Plane should first be taken airborne for the purpose of this simulations tests since this work does not cover taking off and landing, but concentrated on a controller during airborne flight.

At first, a single step heading reference was designed for an actual flight regime of 50° initial heading to 100° final heading and the results were as shown in Figure 4.10.

For the PID response to the step reference, the rise time was 1.4007 seconds and the system does not settle within the bounds of 0.5% of final value that was specified. The percentage overshoot was calculated to be 6.3119%. The rise time for the reinforcement learning controller is 1.2663 seconds as compared to PID's 1.4007 seconds and the system settles after 21.371 seconds. The percentage overshoot is 3.1083% for the RL controller. As can be seen the overshoot is lesser than the PID's. The results in Figure 4.11 and Figure 4.12 show UAV simulations where the UAV is commanded from an initial heading of 40°, then commanded to go to 80° and then 50° and so on. The heading in degrees is plotted on the vertical axis against time in seconds on the horizontal axis. Figure 4.11 shows the performance of the PID controller for real time heading control of a UAV. PID controller performs well in tracking the reference initially from any random position, but its tracking response is poor as it overshoots in every step reference heading change and does not track the reference accurately as is illustrated in Figure 4.11. Figure 4.12 shows the response of the Reinforcement

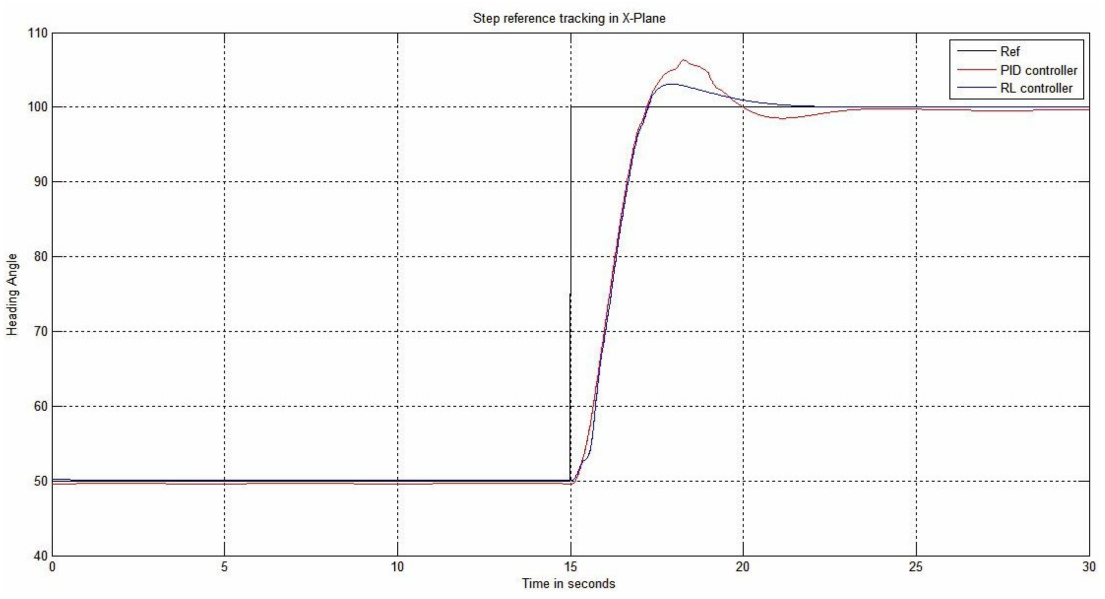


Figure 4.10: Comparison of the two responses to a step heading reference

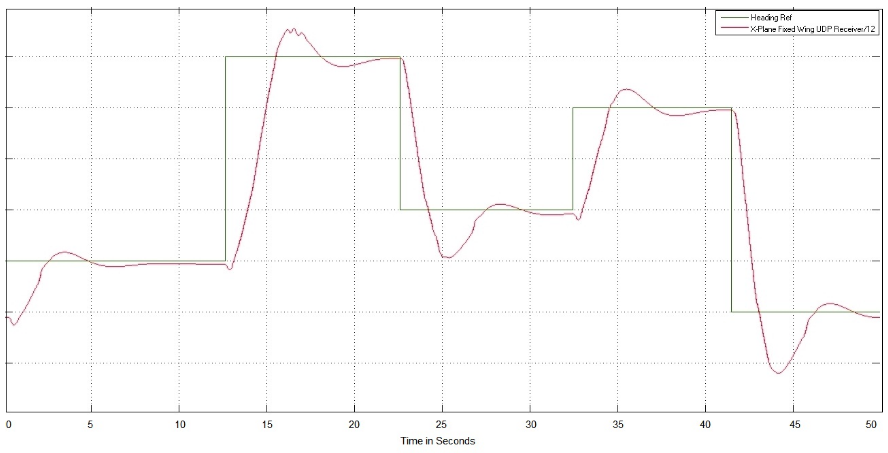


Figure 4.11: PID controller response to real time heading control in X-Plane

learning controller for the same heading control of a UAV. For the first 10 seconds, the response is poor as compared to the PID controller; this is due to the fact that the artificial neural network weights are being continually adjusted where initially big adjustments are expected then they settle around the optimum weights. After 10 seconds the response stabilizes and follows the reference better than the PID controller. It can also be seen that there is an overshoot on the first step heading

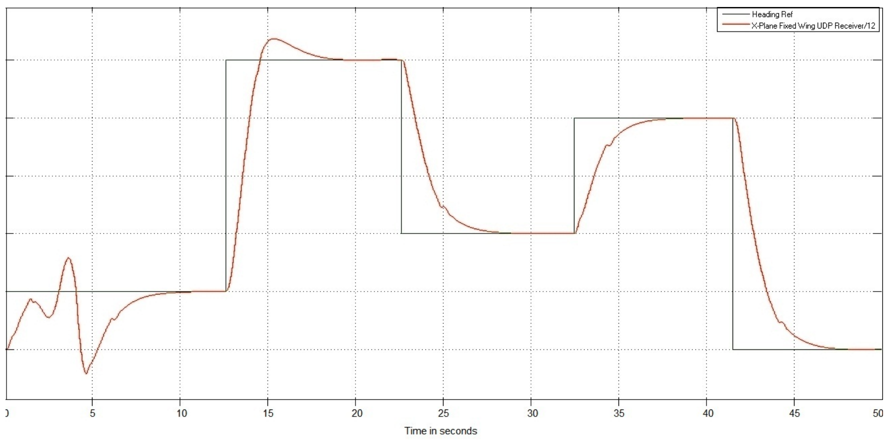


Figure 4.12: RL controller response to real time heading control in X-Plane

change, but due to adaptation that overshoot is eliminated in the consecutive step heading angle changes as is evident from the Figure 4.12.

4.3.3 Performance upon training the Initial weights of Controller

The results presented in Figure 4.12 were achieved using randomly initialized weights in the RL controller. The impact of this is seen in the first 5 seconds where there are large oscillations about the reference point. This is due to the fact that the neural network weights are adjusting rapid to optimal weights in the designed RL controller.

Training of Initial Neural network Weights

The data from Figure 4.12 was used to train neural network weights using non-linear autoegressive network with exogenous inputs, NARX in MATLAB. The equation for a NARX model is

$$y(t) = f(y(t-1), y(t-2), \ldots, y(t-n_y), u(t-1), u(t-2), \ldots, u(t-n_u)) \tag{4.2}$$

This is a linear ARX model where the next value of the dependent output signal $y(t)$ is regressed on previous values of the input signal and previous values of an independent (exogenous) input signal. The optimum neural network weights were achieved as;

$$Input \quad weights = -\begin{bmatrix} 0.6623 & 0.1961 \\ 0.0701 & 0.7474 \\ 0.9348 & 0.5801 \end{bmatrix} \tag{4.3}$$

and

$$Hidden \quad layer \quad weights = -\begin{bmatrix} 1.1392 \\ 1.1389 \\ -4.7608 \times 10^{-4} \end{bmatrix} \tag{4.4}$$

These weights were set as initial weights in the controller and simulation was ran again. The response was as shown in Figure 4.13. It can be seen that there is

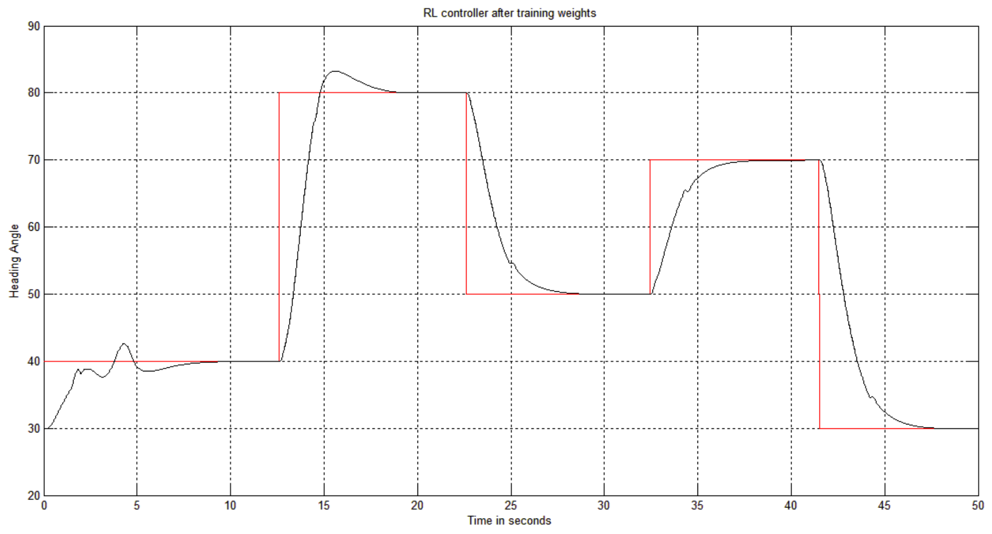


Figure 4.13: RL controller response after training neural network

some improvement on the initial tracking; there is less oscillations and overshoots during the initial stages as compared to Figure 4.12.

4.4 Performance of adaptive controller in the presence of disturbances

Different disturbances are introduced into the mathematical model and, wind disturbances on a UAV model in X-Plane. The simulation responses of the adaptive controller under these disturbances is compared with those of the PID reference controller.

4.4.1 Performance using the UAV mathematical model

The mathematical model in equation (3.2) was used and different disturbances were added to the two control inputs $\delta_a \quad and \quad \delta_r$. The responses were compared to those of a PID controller.

Impulse Disturbance

An impulse test signal of amplitude 1 rad and 0.4 seconds duration [20] as shown in Figure 4.14, was injected in the control surfaces. The response of the yawing

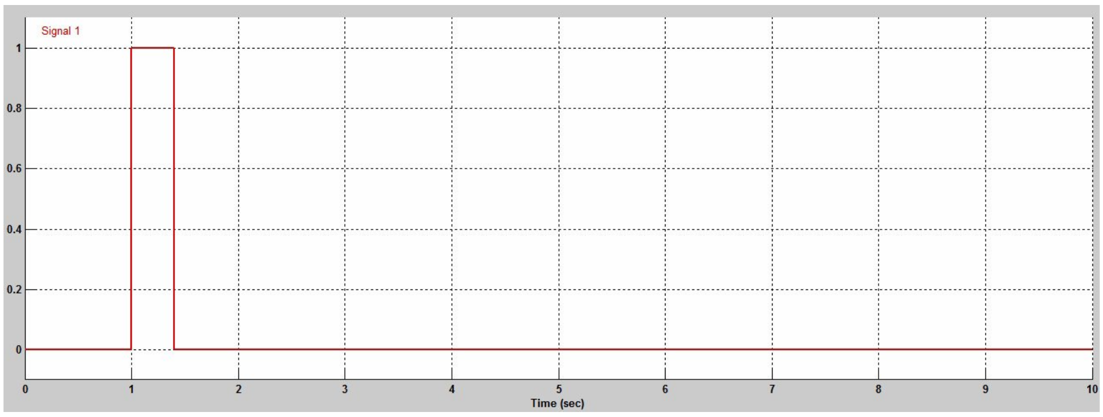


Figure 4.14: Impulse disturbance

rate was as in Figure 4.15 when the disturbance was injected in the aileron, Figure 4.16 when injected in the rudder and Figure 4.17 when injected in both aileron and rudder inputs. The response under an impulse disturbance in the aileron input shows the system under the RL controller goes out of hand and deviates completely. This is because the roll damping due to aileron becomes unstable and the limited aileron control is insufficient to stop it. The disturbance at the rudder

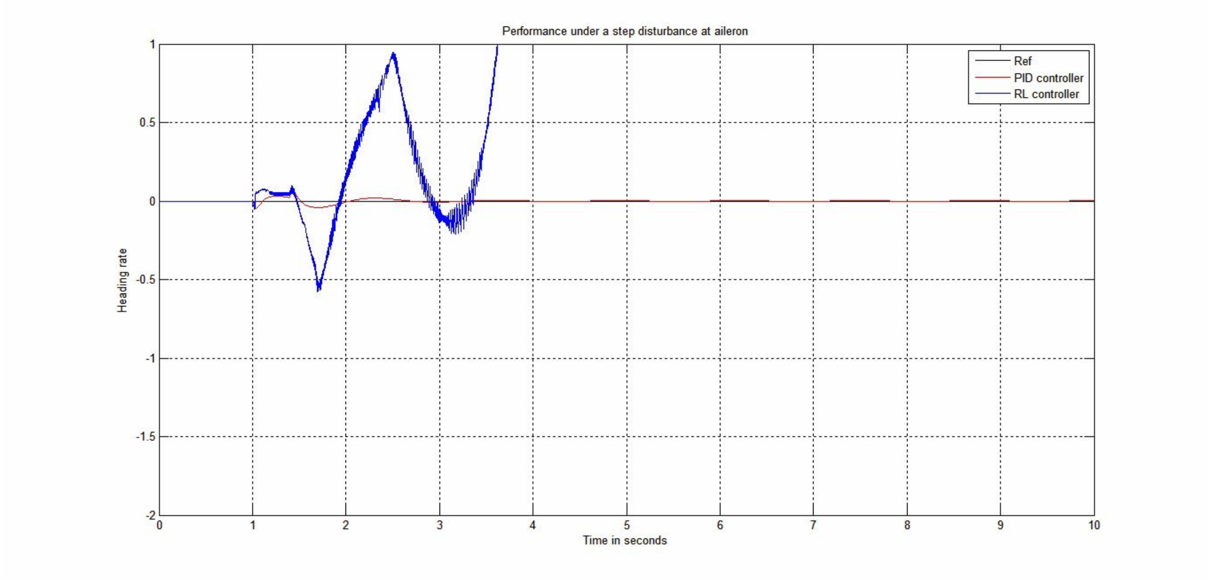


Figure 4.15: Response under an impulse disturbance at aileron

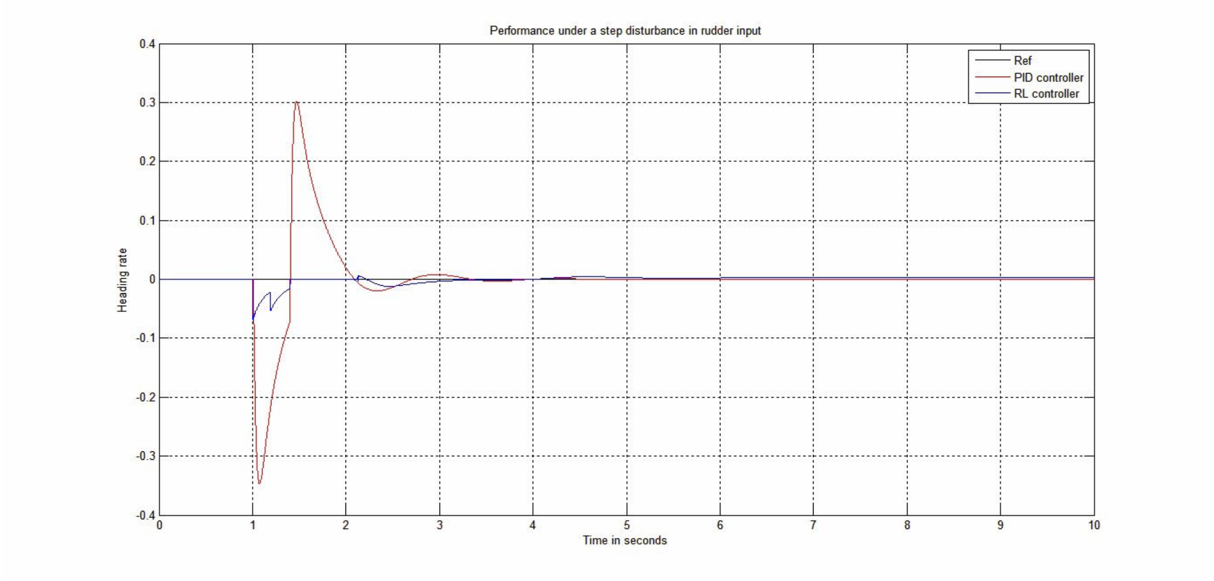


Figure 4.16: Response under an impulse disturbance at rudder

is suppressed well and kept within the target reference. The reference controller response shows a pronounced disturbance showing at the output as Figure 4.16 shows. The bad response of due to a disturbance in the aileron as observed in Figure 4.15 is compensated for by the rudder action; which according to [5] is the primary control for a spin recovery, when the disturbance is injected in both inputs as Figure 4.17 shows.

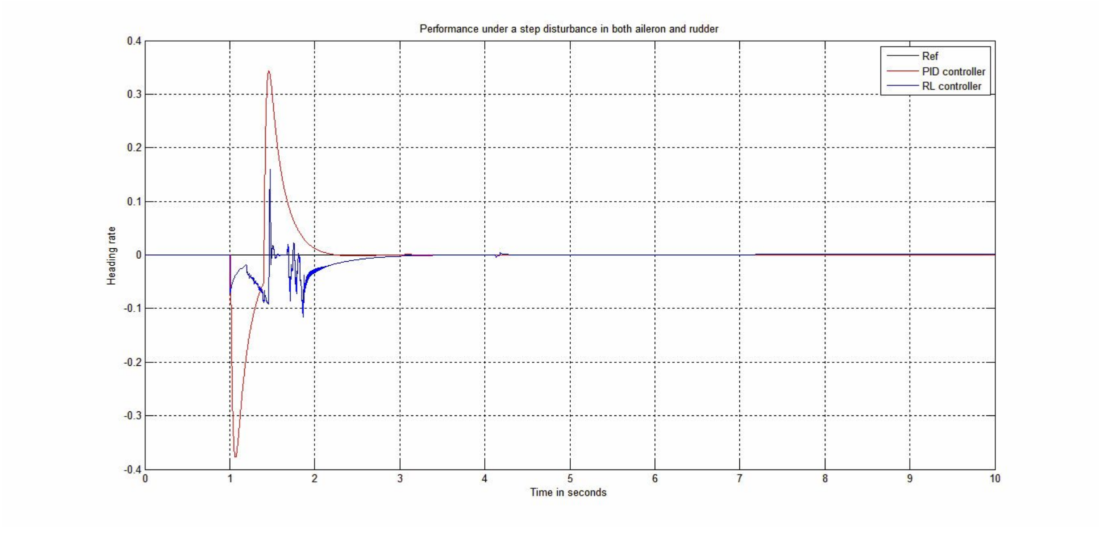


Figure 4.17: Response under an impulse disturbance at aileron and rudder

Chirp signal Disturbance

A chirp signal is a linearly frequency-swept cosine signal at time instance defined by the time array t. It is defined as

$y = chirp(t, f_0, t_1, f_1)$

where f_o is the instanteneous frequency at time 0, and f_1 is the instanteneous frequency at time t_1. The signal generated for this experiment was as in Figure 4.18, where the parameters were $f_0 = 0Hz$, $f_1 = 30Hz$, $t_1 = 4$ seconds and the

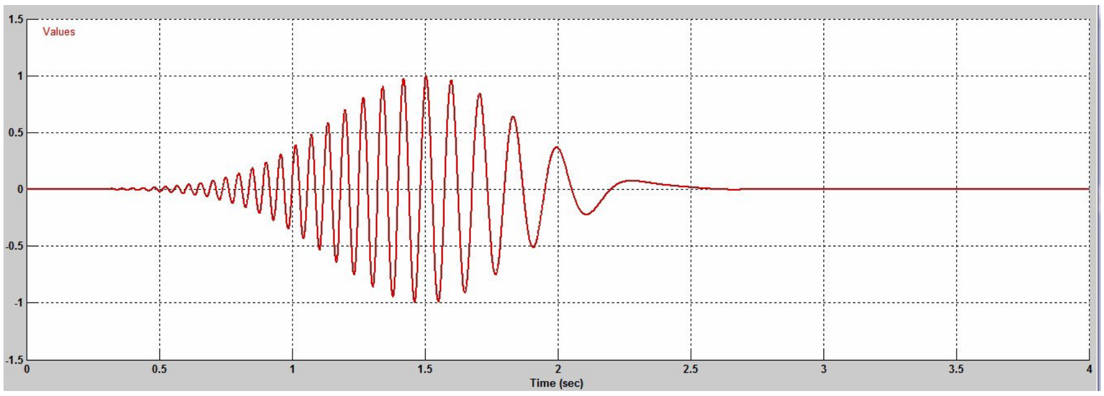


Figure 4.18: Chirp disturbance

function was linearly scaled down using the function $\exp^{-(2t-3)^2}$ so as to achieve that response. The response of the RL controller in comparison with that of the reference PID controller was as shown in Figure 4.19 to Figure 4.21.

The behavior under this disturbance in the rudder input is notable, as Figure

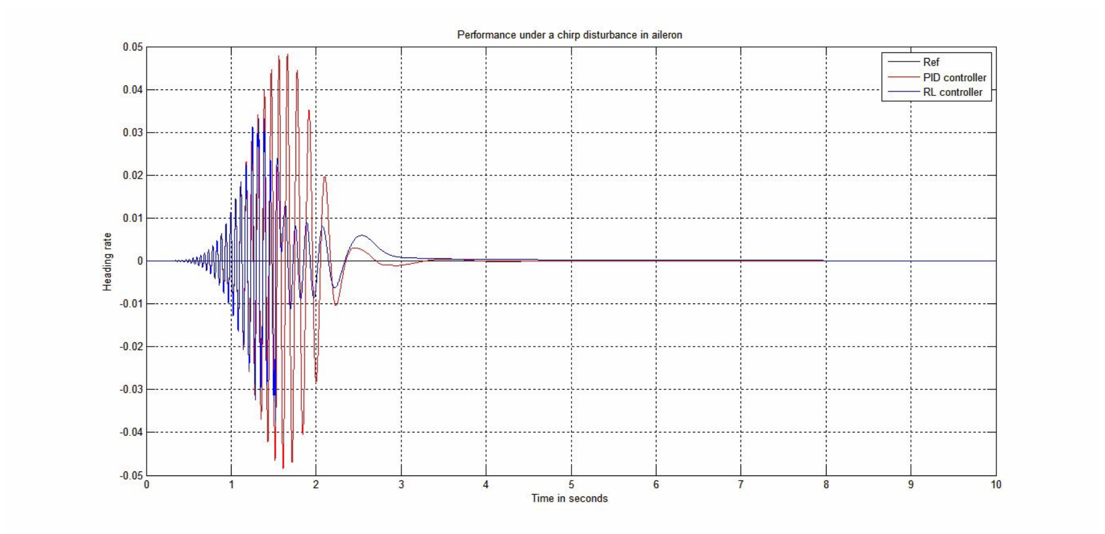


Figure 4.19: Response chirp disturbance in aileron

4.20 shows some new form of periodic oscillation after initially minimizing the chirp disturbance. The periodic oscillation kicks in after the chirp disturbance was minimized at both inputs as shown in Figure 4.21, but was compensated

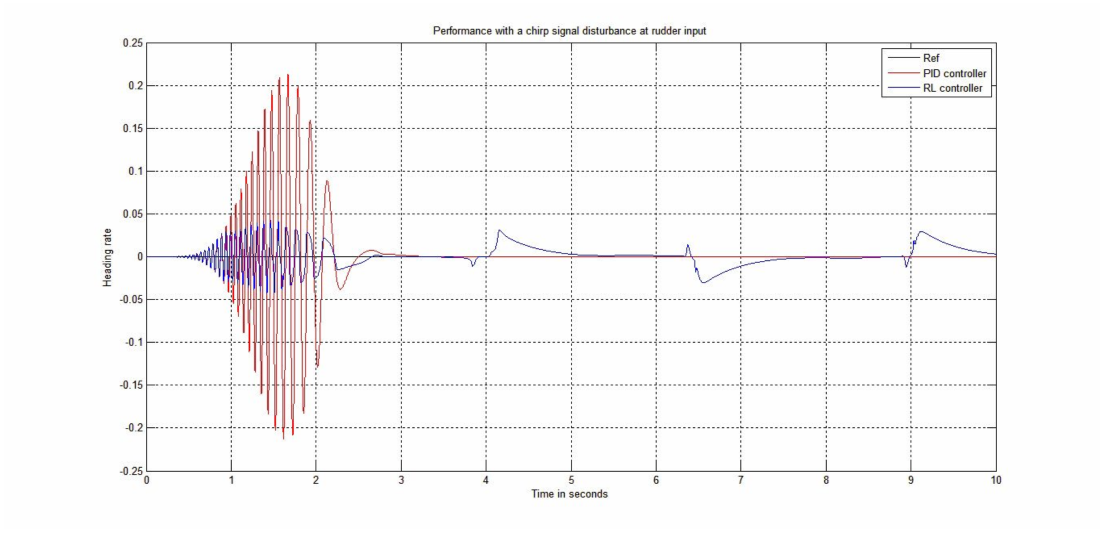


Figure 4.20: Response chirp disturbance in rudder

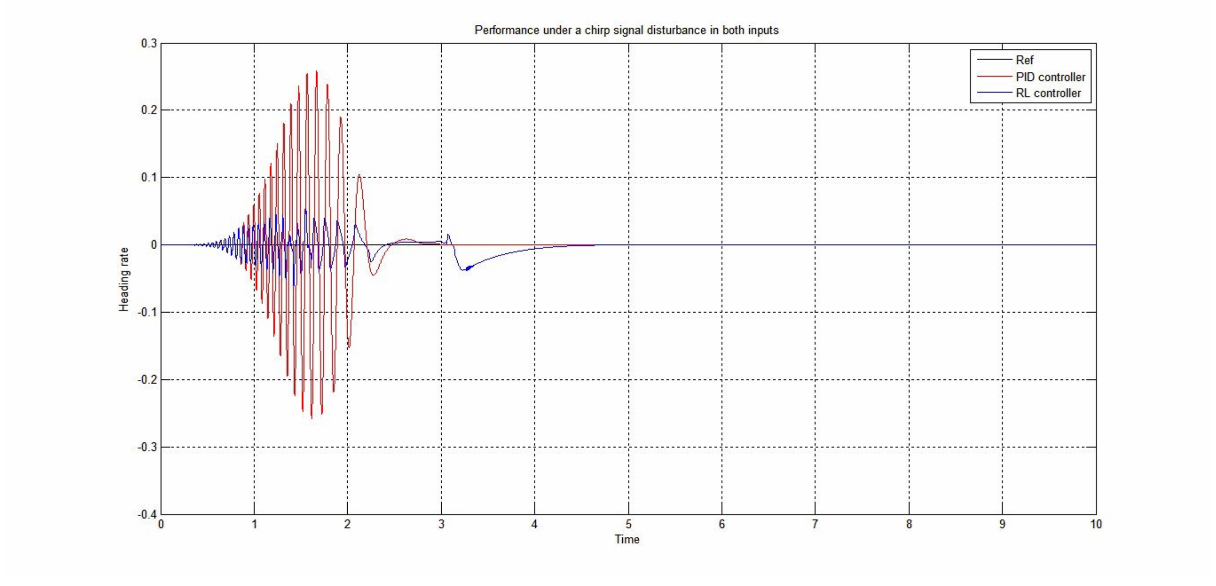


Figure 4.21: Performance chirp disturbance in aileron and rudder

quite well by the other input Figure 4.19 which did not experience the additional periodic disturbance when the same disturbance was injected into it.

Random Disturbance

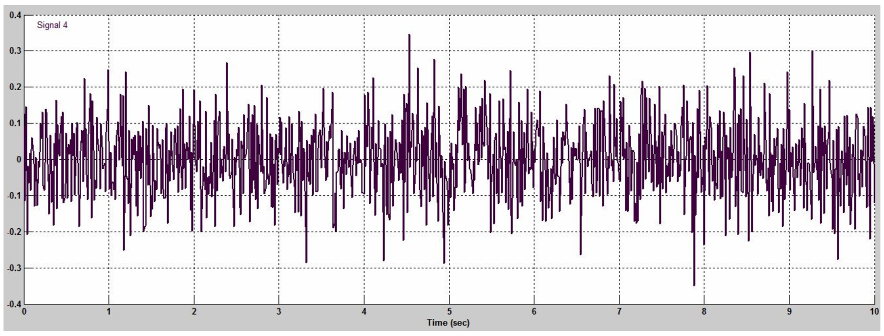


Figure 4.22: Random disturbance

A random Gaussian signal with a mean of 0 rad and 0.01 rad^2 variance as was used by [20] was generated as shown in Figure 4.22 .

The response achieved was as shown in Figure 4.23 when the random signal was introduced as a disturbance in the aileron input. The PID reference controller

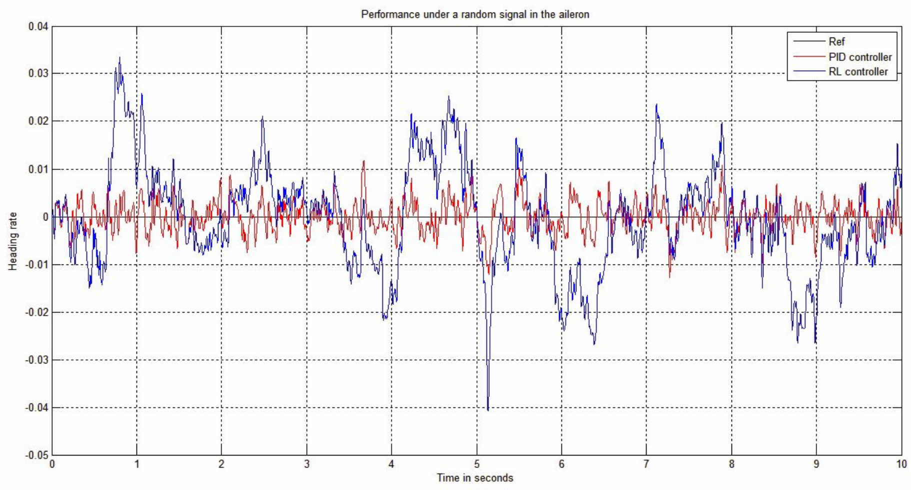


Figure 4.23: Response random disturbance in aileron

performed better. The RL controller response was poor due the fact that the modelled system as was evidenced in the singular value decomposition will have low gain at high frequency. Therefore, the system is unable to effectively generate high gains to minimize a high frequency disturbance. The effect of the random

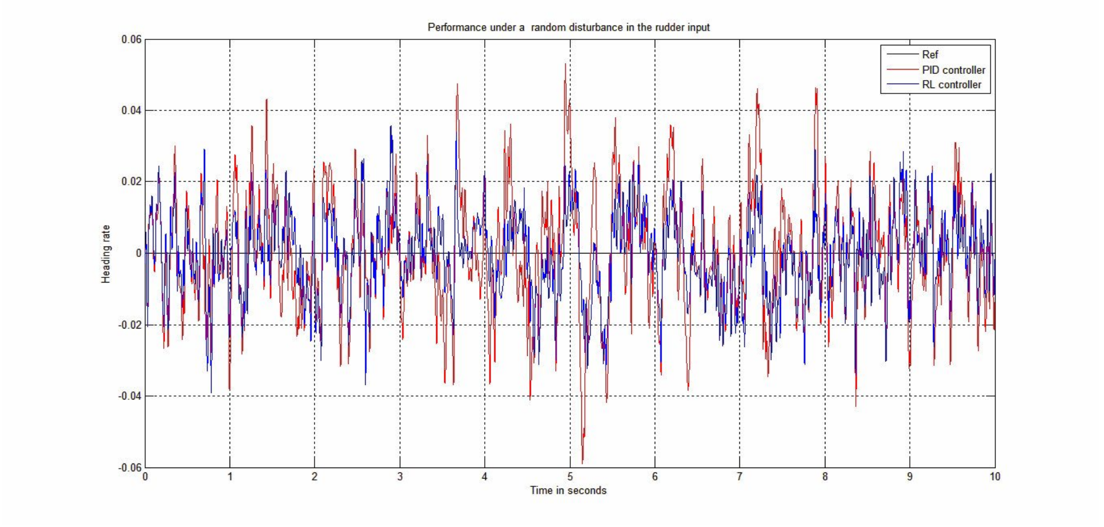


Figure 4.24: Performance random disturbance in rudder

disturbance when injected into the rudder input is a shown in Figure 4.24. The disturbance is successfully managed with bounds. The disturbance in both inputs is held within bounds as Figure 4.25 shows.

4.4.2 Performance under disturbances on UAV model in X-Plane

Different wind disturbances were introduced in the X-Plane platform. Initially a cross wind of 5 knots (approx. 2.57 m/s) and 345° wind direction, 2° shear direction was introduced and the tracking ability of the controller under this condition was as in Figure 4.26.

It can be seen that, the RL controller is able to counter the effects of this lateral

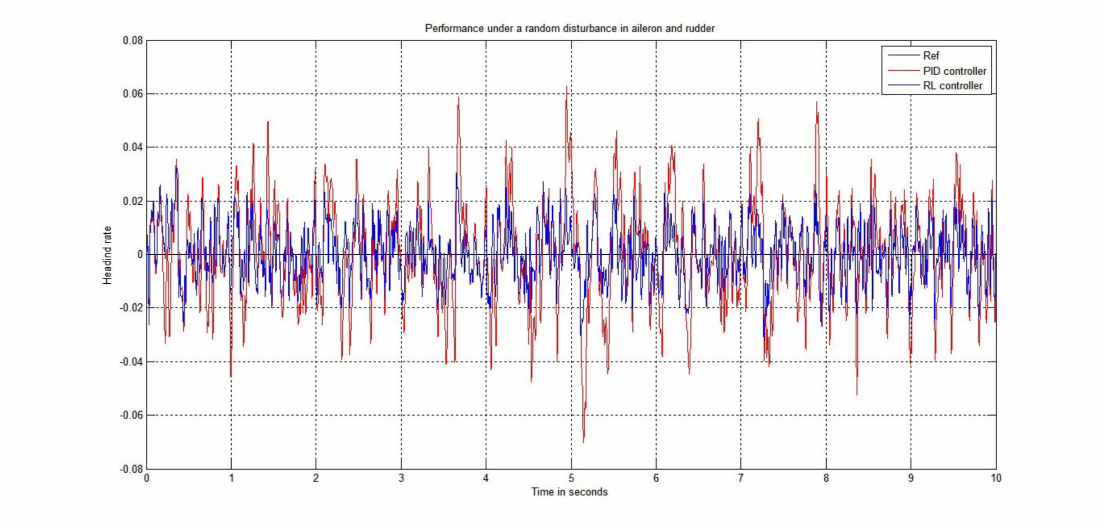


Figure 4.25: Performance random disturbance in aileron and rudder

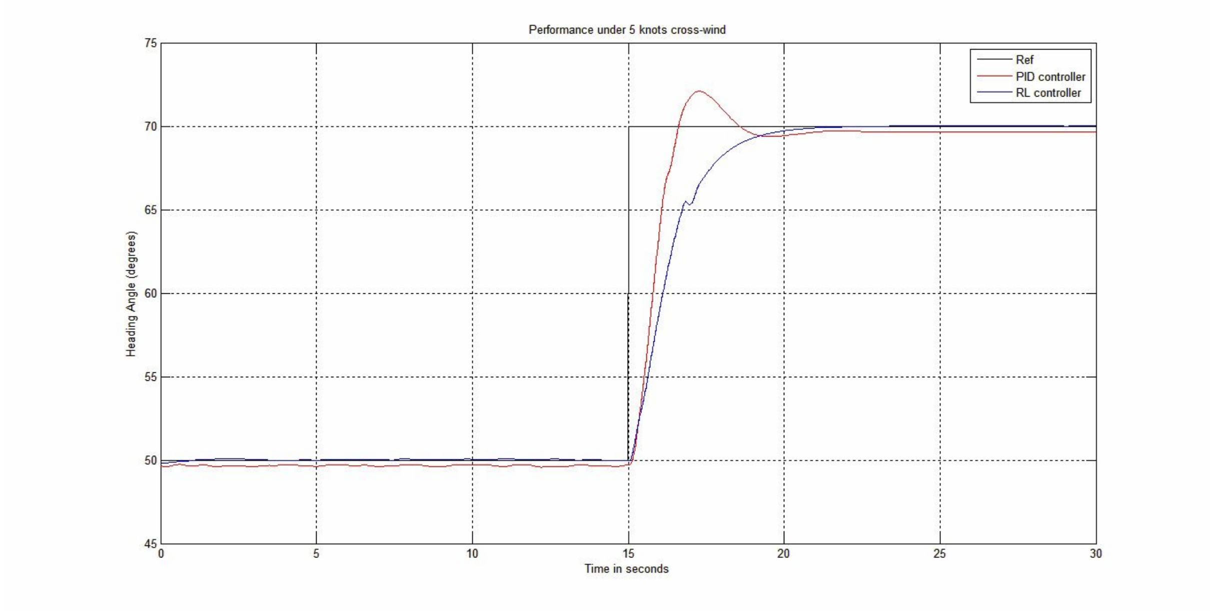


Figure 4.26: Response under 5 knots cross wind

cross wind as time passes. The initial stages shows that there is some slight deviation from the reference due to the disturbance but after a while, the UAV is able to track well the reference heading. The performance of PID is inferior as compared to that of the RL controller. Also, the PID controller exhibits a small steady-state/offset error.

A cross wind of 10 knots (approx. 5.14 m/s) and 345° wind direction, 2° shear direction was introduced and the tracking ability of the controller under this condition was as shown in Figure 4.27. As in the previous case, the RL controller

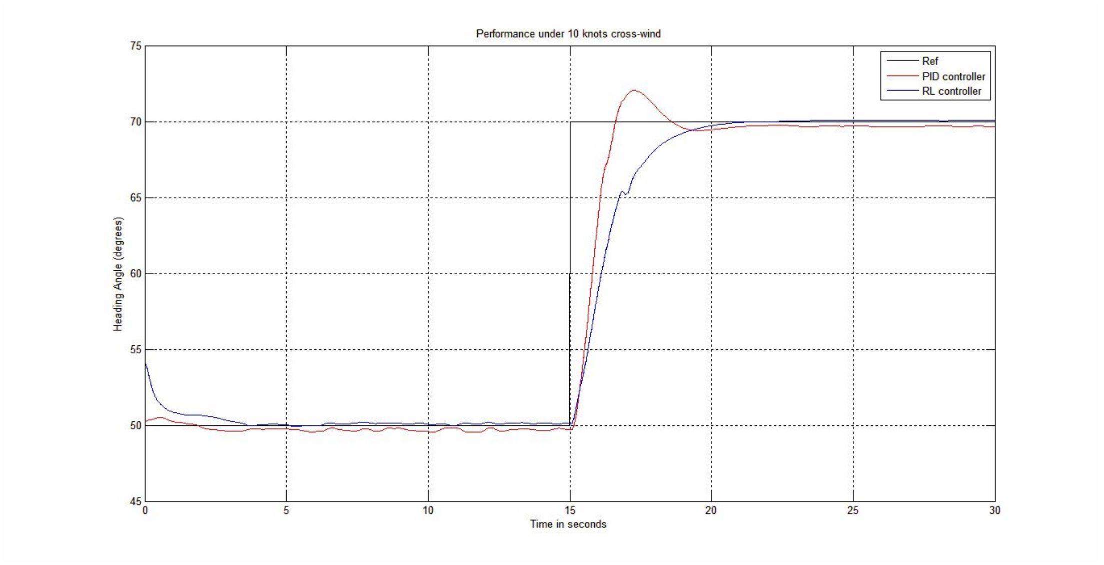


Figure 4.27: Response under 10 knots cross wind

performed better.

A lateral cross wind of 20 knots (approx. 10.3 m/s) and 345° wind direction, 5° shear direction was introduced and Figure 4.28 the response of the controllers. As can be deduced, the ripples from the PID controller are bigger.

A 40 knots (approx. 20.3 m/s) wind, 345° wind direction and 10° shear direction was introduced as Figure 4.29 shows the graphs achieved. This represented severe

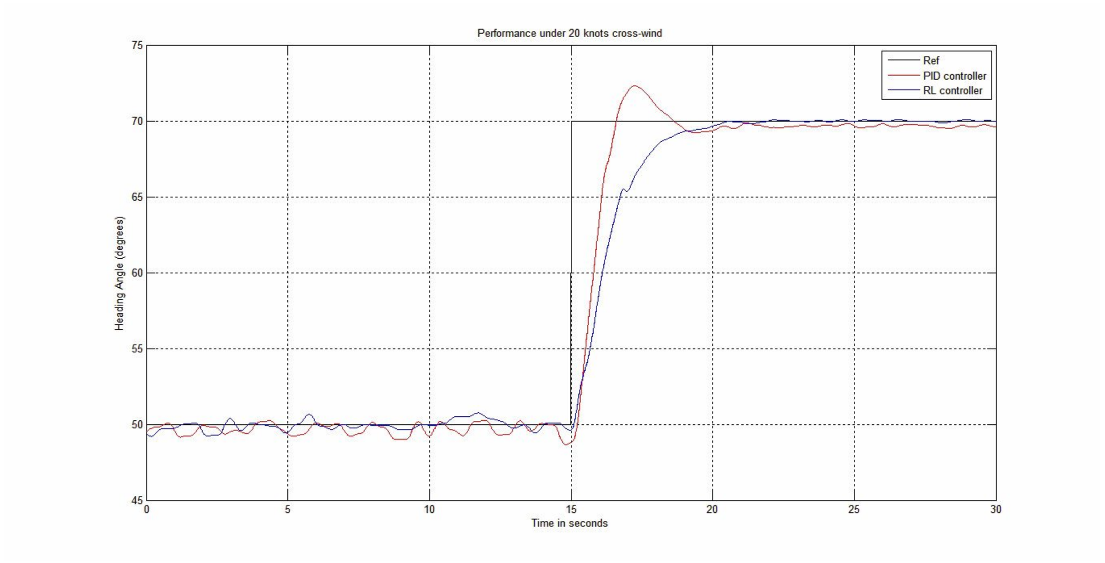


Figure 4.28: Response under 20 knots cross wind

wind conditions for a small UAV. The tracking ability of the controller under this condition showed big deviations from the reference heading for both controllers. The offset error in this response is hidden, since the ripples are bigger and one can not clearly distinguish the offset error from the disturbance effect. This is due to the deviations achieved from the wind disturbance are huge thus making the UAV to oscillate over and below the target reference angle as it tries to reduce the disturbance. A similar response was reported in [51] where the roll angle response was being investigated under 50 knots wind gust.

Turbulence

A turbulence of a setting 1 in X-Plane was introduced on a separate flight regime. This includes a cross-wind of about 20 knots, wind shear of 5 knot and a 10° shear speed as set by the X-Plane simulator. This represents extreme turbulent conditions for a small UAV and the tracking performance was as in Figure 4.30. As it can be seen, the response of the UAV under this weather condition is

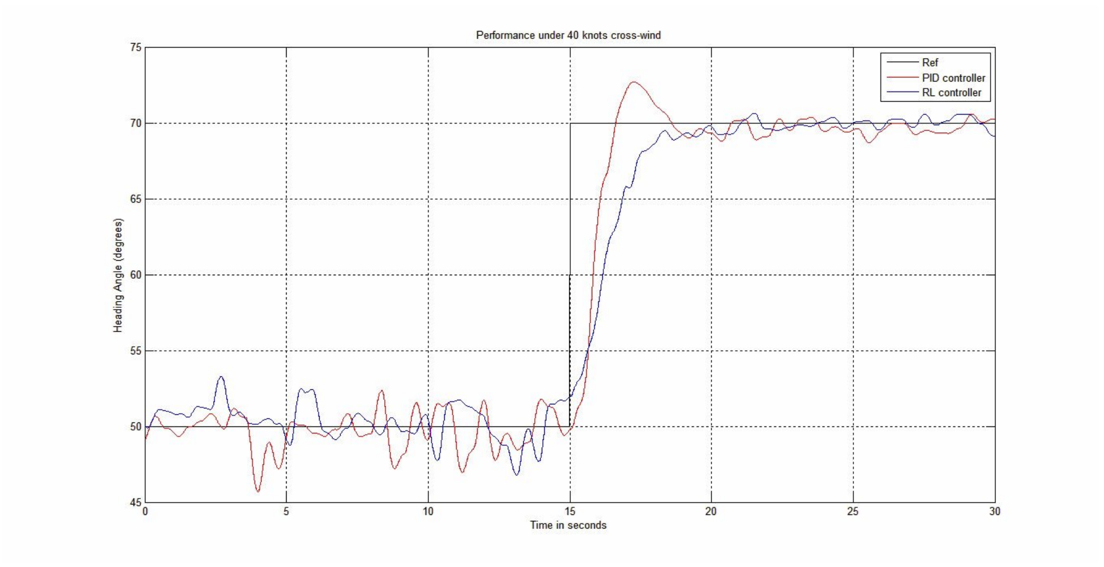


Figure 4.29: Response under 40 knots cross wind

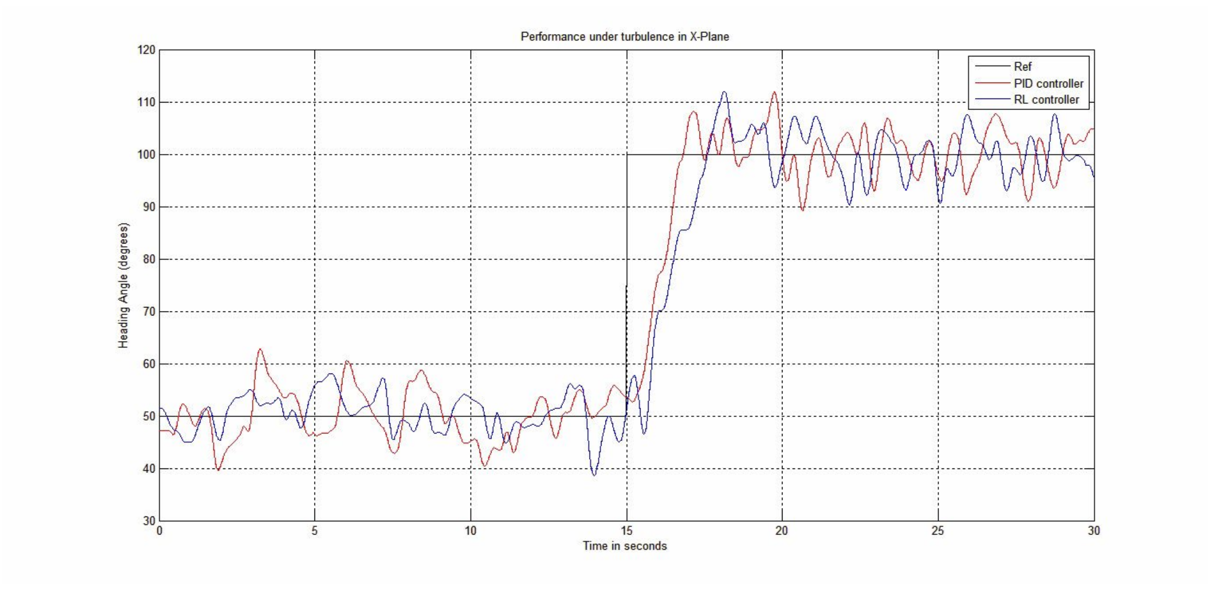


Figure 4.30: Performance under turbulence

poor. The UAV heads in the right direction but with a lot of overshoots and oscillations about the desired reference position. This movement is not desirable in a small UAV as it might lead to structural damage to the body fuselage. Similar responses under turbulence were obtained by [48] where smaller UAVs were swept away under turbulence. The main reason for this behaviour is as put forth by [52][48], that sensor and actuator dynamics do not scale with aircraft dynamics hence a main limitation in lateral response for UAVs in turbulent conditions. In [53], it is argued that for significant oscillatory disturbances, a UAV never sees enough smooth air to dampen out because of the short distance between the CoG and the vertical stabilizer. In the research [51], whereby the effect of a lateral wave turbulence of a give span and length was considered on wings from zero to large wing spans. It was found that only wings with a span of about equal to the turbulent wave length will experience large roll effects. In view of this, [52] states that this shortcoming can be addressed through structural aerodynamic design.

PID Offset error

The PID responses have a steady state error. For the mathematical model this steady state error was measured to be $0.0002°$ in the yawing rate which is an improvement from $0.0006°$ that was reported in [24]. Also in [23], a steady state error of 0.05% was reported on the heading angle. In the X-Plane platform, this steady state error averages to about $0.2485°$ on the heading angle. The increase in the error is considered to account for measurement noise and those non-parametric quantities that cannot be quantified but contribute to flying qualities e.g. humidity, temperature, atmospheric texture that increase with increase in atmospheric disturbance. This is supported by [54], where it is stated that the nominal operating point of PID may be unstable in certain cases like in wind disturbances. The RL controller is able to cater for this, as it uses adaptation techniques by using value functions from each state-action pair hence capture the effect on the UAV motion.

4.5 Summary

An analysis of the UAV model is done, to determine the properties. A reference controller using PID controllers is presented which forms the basis of comparison with the designed controller. MATLAB/SIMULINK simulations tests using the mathematical model were carried out and they demonstrate that the reinforcement learning controller possess better reference tracking ability. This also is observed when the controller is used as a real time controller for a UAV model in X-Plane. Then disturbances were introduced in both simulations; for the mathematical model and the results were good other than responses from the aileron whereby the cause of the response has been explained. For real disturbances as set in X-Plane, the simulation results showed better tracking as compared to the response of the PID reference controller, such that even under extreme turbulence, the UAV oscillates about the target reference.

CHAPTER FIVE

CONCLUSION

5.1 Conclusion

UAVs are the future of aerial vehicles due to their numerous application in different areas. Due to their size and hence dynamics, they pose serious control challenges of dealing with varying disturbances as they endeavour to accomplish a given mission. The UAV was modelled as a two player differential game against nature. On investigation, it showed that the open loop system unstable. A PID controller was developed using interactive tuning in MATLAB to track a reference and achieve closed loop stability. A cascade system was used and it successfully introduced stability in the closed loop system, and achieved stable tracking of a reference. The PID controller's results compared well with results from other researchers, and thus was used as a reference controller.

A simple adaptive controller based on reinforcement learning for active disturbance rejection in the modelled fixed wing UAV was developed. A game theoretic method coupled with *SARSA*, a reinforcement learning technique was applied in coming up with the adaptive controller. Simulation tests were carried out using the UAV mathematical model. These were compared to those of the reference controller, and the adaptive controller simulation results were better.

Real time simulations using the adaptive controller on a UAV model in X-Plane were carried out. The reinforcement learning controller showed a 9.6% improvement in rise time and a reduction of about 50.7% in the overshoot for a step heading change as was compared with the reference controller. For consecutive step heading changes in X-Plane, the adaptive controller was able to eliminate the overshoot completely as is evidenced in the results. Further simulations tests in X-Plane were carried out under different wind disturbances. The simulations showed better performance in tracking the reference heading using the adaptive controller in comparison with the reference controller.

Simulation results have shown that the adaptive controller based on reinforcement learning performed better in terms of tracking the commanded input even in the presence of disturbances than a well tuned PID controller which was used as the reference controller, using both the mathematical model and real time simulations on a UAV model in X-Plane. It can therefore be concluded that, the designed adaptive controller can perform better in real life than the already existing PID controllers in UAV autopilots as is evident from the simulations.

5.2 Further Work

In this work, the general nonlinear model of a UAV is decoupled into longitudinal and lateral directional modes. The lateral mode is used in designing the controller. Further research should aim at trying to investigate if the longitudinal mode can contribute in compensating for the deviations incase of turbulent conditions.

Also, this work has not addressed the issue of exploration versus exploitation which is a central tenant to reinforcement learning. The adaptive controller only exploited the value-function and new states were found by inferring favourable value-function values. Further research should address exploration while also striking a balance with real-time control.

REFERENCES

[1] R. Yanushevsky, *Guidance of Unmanned Aerial Vehicles*, Boca Raton: CRC Press, 2011.

[2] M. Bento “Unmanned Aerial Vehicles: An Overview”, Working Papers, Feb. 2008. Article.

[3] Q. Abdullah.(2014).*Classification of the Unmanned Aerial Systems*[Online]. Available:http://www.e-education.psu.edu/geog597g/node/5.

[4] X. Hua, Y. Ruyi, Y. Jianqiang, F. Guoliang and J. Fengshui, “Disturbance Rejection in UAV’s Velocity and Altitude Control: Problems and Solutions,” Proceedings of the 30th Chinese Control Conference, 2011.

[5] M. Cook, *Flight Dynamics Principles*, 2nd ed. Oxford, UK: Elsevier Aerospace Engineering Series, 2007.

[6] Y. Shin, A. Calise and M. Motter, “Adaptive Autopilot Designs for an Unmanned Aerial Vehicles,” Proceedings of AIAA Guidance, Navigation and Control Conference and Exhibit, 2005.

[7] R. Button, A. Barto and R. Williams, “Reinforcement learning is Direct Adaptive Optimal Control”, Proceedings of the American Control Conference, Boston, 1991.

[8] F. Lewis, D. Vrabie and V. Syrmos, *Optimal Control*, John Wiley and Sons Inc., 3rd Edition, 2012.

[9] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction.* Cambridge, MA: MIT Press, 2005.

[10] M. Harmon, L. Baird and H. Klopf, "Reinforcement Learning Applied to a Differential Game," Adaptive Behaviour, 1995.

[11] P. Straffin, *Game Theory and Strategy*, Mathematical Association of America, 1993.

[12] A. Bressan, "Noncooperative Differential Games: A Tutorial," Penn State University, 2010.

[13] J. Sheppard, "Co-learning in differential games," Machine Learning, 1998.

[14] Y. Ishiwaka, T. Sato and Y. Kakazu, "An approach to the pursuit problem on a heterogenous multiagent system using Reinforcement Learning," Robotics and Autonomous Systems, 2003.

[15] S. Givigi, H. Schwartz and X. Lu, "A reinforcement learning adaptive fuzzy controller for differential games," Journal of Intelligent and Robotic Systems, 2010.

[16] S.Yasini, A. Karimpour, M. Sistani and H. Modares, "Online concurrent reinforcement learning algorithm to solve two-player zero-sum games for partially unknown nonlinear continuous-time systems, "International Journal of Adaptive Control and Signal Processing, 2014.

[17] J. Hans, "Auto-disturbance rejection control and its Applications," Control and Decision, 1998.

[18] Z. Qing and G. Zhiqiang, “On Practical Applications of Active Disturbance Rejection Control,” Proceedings of the 29th Chinese Control Conference, Beijing, 2010.

[19] Z. Yu and G. Li, “Unmanned Helicopter Autopilot Design Based on Active Disturbance Rejection Control, ” Advanced Science and Technology Letters, 2014.

[20] H. Ferreira, R. Baptista, J. Ishihara and G. Borges, “Disturbance Rejection in a Fixed Wing UAV using Nonlinear H∞ State feedback,” 9th IEEE International Conference on Control and Automation (ICCA), Santiago, Chile, 2011.

[21] C. Yang and C. Kung, “Nonlinear H∞ flight control of general six degree-of-freedom motions,” Proceedings of the American Control Conference, 2000.

[22] D. Kingston, R. Beard, T. McLain, M. Larsen, and W. Ren, “Autonomous Vehicle Technologies for Small Fixed Wing UAVs”, American Institute of Aeronautics and Astronautics, 2003.

[23] A. Mansoor, R. Hamza and A. Zofishan, “Heading control of a Fixed wing UAV using Alternate control surfaces”, IEEE, Dec., 2012.

[24] T. Espinoza, A. Dzul and M. Llama, “Linear and Nonlinear controllers Applied to Fixed-Wing UAVs”, International Journal on Advanced Robotic Systems, 2013.

[25] V. Nagababu and M. Prabhudas , “A Fuzzy logic strategy on attitude controlling of Longitudinal autopilot for better disturbance rejection”, International Journal of Engineering Research and Technology, Dec., 2013.

[26] D. Stojcsics, “Fuzzy controller for small size Unmanned Aerial Vehicles”, 10th IEEE International Symposium on Applied Machine Intelligence and Informatics, Herl’any, Slovakia, 2012.

[27] J. Spletzer and C. Montella , “Reinforcement Learning for Autonomous Dynamic Soaring in shear winds”, IEEE/RSJ International Conference, Sept., 2014.

[28] A. Geramifard, J. Redding, N. Roy and J. How, “UAV Cooperative Control with Stochastic Risk Models”, American Control Conference, San Francisco, USA, 2011.

[29] A. brezoescu, T. Espinonza, R. Lozano and P. Castillo, “Adaptive Trajectory Following for a Fixed-Wing UAV in Presence of Crosswind”, Journal on Intelligent Robotic System, 2013.

[30] H. Castaneda, O. Salas-Pena and J. Leon-Morales, “Robust Autopilot for a Fixed Wing UAV using Adaptive Super Twisting Technique”, PHYSCON, San Luis Potosi, Mexico, 2013.

[31] J. Zhang, Q. Li, N. Cheng and B. Liang, “ Path-following control for fixed-wing unmmaned aerial vehicles based on a virtual target”, Journal of Aerospace Engineering, 2012.

[32] A. Calise, Y. Shin and M. Motter , “Application of Adaptive Autopilot designs for an Unmanned Aerial Vehicle”, American Institute of Aeronautics and Astronautics, 2005.

[33] R. Nelson, *Flight Stability and Automatic Control*, 2nd ed. Singapore: McGraw Hill, 1998.

[34] T. McLain and R. Beard, *Small Unmanned Aircraft; Theory and Practice*, 1st ed. Princeton, New Jersey: Princeton University Press, 2012.

[35] J. Yi, G. Fan, H. Xiong and F. Jing, “Anti-crosswind Autolanding of UAVs based on Active Disturbance Rejection Control”, AIAA Guidance, Navigation, and Control Conference, Ontario, 2010.

[36] Y. Chai, “Synthesis and Validation of Flight Control for UAV”, University of Minnesota, Minnesota, PhD Thesis Dec. 2009.

[37] D. Kleinman, “On an Iterative Technique for Ricatti Equation Computations”, IEEE Transactions on Automatic Control, 1968.

[38] W. Aangenent, D. Kostic, A. Jager, G. Molengraft and M. Steinbuch, “Data based Optimal Control”, Proceedings of the America Control Conference, Portland, USA, 2005.

[39] M. Abu-Khalaf and F. Lewis, “Neurodynamic programming and Zero-sum games for constrained Control Systems”, IEEE Transactions on Neural Networks, 2008.

[40] Y. Feng, D. Anderson and M. Rotkowitz, " A game theoretic Algorithm to compute local stabilizing solutions to HJI equations in nonlinear H_∞ Control", Automatica, 2009.

[41] H. Geering, *Optimal Control with Engineering Applications* Berlin: Springer, 2007.

[42] J. Etele, "Overview of Wind Gust Modelling with Application to Autonomous Low-level UAV Control", Defence R and D Canada, Ottawa, DRDC Ottawa CR 2006-221, Nov., 2006.

[43] I. Landau, R. Lozano, M. M'Saad and A. Karimi, *Adaptive Control: Algorithms, Analysis and Applications*, 2nd ed.: Springer, 2011.

[44] H. Bou-Ammar, H. Voos and W. Ertel, "Controller Design for Quadrotor UAVs Using Reinforcement Learning", IEEE International Conference on Control Applications, Yokohama, Sept., 2010.

[45] J. Santamaria, R. Sutton and A. Ram, "Experiments with Reinforcement Learning in Problems with Continuous State and Action Spaces", Unpublished.

[46] A. Meyer, *X-Plane 10: Operation Manual.*, 2012.

[47] R. Lucio and O. Neusa, "UAV Autopilot Controllers Test Platform Using Matlab/Simulink and X-Plane", 40th ASEE/IEEE Frontiers in Education Conference, Washington DC, 2010.

[48] K. Zhukov, V. Vyshinsky and J. Rohacs , "Effects of atmospheric turbulence on UAV", IFFK, Budapest, 2014.

[49] P. Lissaman, "Effects of Turbulence on Bank Upsets of Small Flight Vehicles", 47th AIAA Aerospace Sciences Meeting, Orlando, Florida, 2009.

[50] B. Aliyu, A. Petinrin and J. Adewumi, "PID COntrol DEsidn of Sideslip Angle fo a Fixed Wing Mini-UAV", Advances in Research,2016.

[51] A. Bittar, O. Nuesa and H. Figueiredo, " Hardware in the Loop simulation with X-Plane of Attitude Control of a SUAV Exploring Atmospheric Conditions", Journal of Intelligent Robotic Systems, 2014.

[52] W. Pisano and D. Lawrence, "Control Limitations of Small Unmanned Aerial Vehicles in Turbulent Environments", AIAA Guidance, Navigation and Control Conference, Chicago, Illinois, 2009.

[53] D. Lundstrom and P. Krus, "Testing of Atmospheric Turbulence Effects on the Performance of Micro Air Vehicles", International Journal of Micro Air Vehicles, 2012.

[54] H. Chao, Y. Cao and Y. Chen, " Autopilots for Small Fixed-Wing Unmanned Aerial Vehicles: A Survey", IEEE International Conference on Mechatronics and Automation, Habrin, China, 2007.

[55] L. Ribeiro and N. Oliveira, "UAV Autopilot Controllers Test Platform Using Matlab/Simulink and X-Plane," 40th ASEE/IEEE Frontiers in Education Conference, Session S2H-1, 2010.

APPENDICES

Appendix A: Math Model Derivation

The equilibrium of the moments acting about the three body axes (x, y, z) form the three moments equations as;

$$\begin{aligned} L &= \dot{p}I_{xx} - \dot{r}I_{xz} - pqI_{xz} + qr(I_{zz} - I_{yy}) \quad - rolling \quad motion \\ M &= \dot{q}I_{yy} + pr(I_{xx} - I_{zz}) + (p^2 - r^2)I_{xz} \quad - pitching \quad motion \\ N &= \dot{r}I_{xz} - \dot{p}I_{xz} + pq(I_{yy} - I_{xx}) + qrI_{xz} \quad - yawing \quad motion \end{aligned}$$

where (p, q, r) are the angular velocity components along the (x, y, z) axes, and $I_{xx}, I_{xz}, I_{yy}, I_{zz}$ are moments of inertia in the respective axes. By definition, (u, v, w) and (p, q, r) are small quantities such that terms involving products and squares of this terms are insignificantly small and may be ignored. Therefore, above equations reduce to

$$\begin{aligned} I_{xx}\dot{p} - I_{xz}\dot{r} &= L \\ I_{yy}\dot{q} &= M \\ I_{zz}\dot{r} - I_{xz}\dot{p} &= N \end{aligned}$$

The components of total force (X, Y, Z) acting on a rigid body are;

$$\begin{aligned} m(\dot{U} - rV + qW) &= X \\ m(\dot{V} - pW + rU) &= Y \\ m(\dot{W} - qU + pV) &= Z \end{aligned}$$

where m is the total mass of the body and (U, V, W) are the velocity components. Further development of equations of motion requires that the terms on the right hand side of above equation adequately describes the disturbing forces. The disturbing force to be considered are wind disturbances labelled with subscript d in preceding equation.

$$\begin{aligned} m(\dot{U} - rV + qW) &= X_g \quad - Longitudinal \quad motion \\ m(\dot{V} - pW + rU) &= Y_g \quad - Lateral \quad motion \\ m(\dot{W} - qU + pV) &= Z_g \quad - Longitudinal \quad motion \end{aligned}$$

In the presence of wind disturbances, the atmosphere is moving relative to the earth. The equations of motion have to be modified due to the fact aerodynamic forces are functions of relative motion between aircraft and atmosphere, and not the inertial velocities. Let u be the aircraft speed, and u_g be the wind gust speed in that direction, then aircraft speed with respect to atmosphere is $u_a = u - u_g$. After rearrangement by noting that $W_e = V_o \sin\theta_e$, the lateral directions equations including disturbances become;

$$\begin{aligned} m(\dot{V} + rU) &= Y_{ae} + \mathring{Y}_v(v - v_g) + \mathring{Y}_p(p - p_g) \\ &\quad + \mathring{Y}_r(r - r_g) + mg\Psi\cos\theta_e + \mathring{Y}_{\delta a}\delta a + \mathring{Y}_{\delta r}\delta r \\ I_{xx}\dot{p} - I_{xz}\dot{r} &= L_{ae} + \mathring{L}_v(v - v_g) + \mathring{L}_p(p - p_g) \\ &\quad + \mathring{L}_r(r - r_g) + \mathring{L}_{\delta a}\delta a + \mathring{L}_{\delta r}\delta r \\ I_{zz}\dot{r} - I_{xz}\dot{p} &= N_{ae} + \mathring{N}_v(v - v_g) + \mathring{N}_p(p - p_g) \\ &\quad + \mathring{N}_r(r - r_g) + \mathring{N}_{\delta a}\delta a + \mathring{N}_{\delta r}\delta r \end{aligned}$$

where the coefficients $(\dot{Y}, \dot{L}, \dot{N})$ are aerodynamic stability derivatives, δ_a and δ_r are the aileron and rudder control inputs for the lateral motion. It is also noted that under a steady trimmed flight, Y_{ae}, L_{ae} and N_{ae} are all zero.

$$m(\dot{v}+rU_e) = \mathring{Y}_v V + \mathring{Y}_p P + \mathring{Y}_r r + mg\cos\theta_e \phi + \mathring{Y}_{\delta_a}\delta_a + \mathring{Y}_{\delta_r}\delta_r - \mathring{Y}_v V_g - \mathring{Y}_p p_g - \mathring{Y}_r r_g$$

$$I_{xx}\dot{p} - I_{xz}\dot{r} = \mathring{L}_v v + \mathring{L}_p p + \mathring{L}_r r + \mathring{L}_{\delta_a}\delta_a + \mathring{L}_{\delta_r}\delta_r - \mathring{L}_v V_g - \mathring{L}_p p_g - \mathring{L}_r r_g$$

$$I_{xx}\dot{r} - I_{xz}\dot{p} = \mathring{N}_v v + \mathring{N}_p p + \mathring{N}_r r + \mathring{L}_{\delta_a}\delta_a + \mathring{N}_{\delta_r}\delta_r - \mathring{N}_v v_g - \mathring{N}_p p_g - \mathring{N}_r r_g$$

$$\dot{\phi} = p$$

Rearranging and putting the above in state-space format, reduces to

$$\begin{bmatrix} m(\dot{v}+rU_e) \\ I_x\dot{p} - I_{xz}\dot{r} \\ I_x\dot{r} - I_{xz}\dot{p} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \mathring{Y}_p & \mathring{Y}_v & \mathring{Y}_r & mg\cos\theta_e \\ \mathring{L}_p & \mathring{L}_v & \mathring{L}_r & 0 \\ \mathring{N}_p & \mathring{N}_v & \mathring{N}_r & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ v \\ r \\ \phi \end{bmatrix} + \begin{bmatrix} \mathring{Y}_{\delta_a} & Y_{\mathring{delta}_r} \\ \mathring{L}_{\delta_a} & L_{\mathring{delta}_r} \\ \mathring{N}_{\delta_a} & N_{\mathring{delta}_r} \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} - \begin{bmatrix} \mathring{Y}_v & \mathring{Y}_p & \mathring{Y}_r \\ \mathring{L}_v & \mathring{L}_p & \mathring{L}_r \\ \mathring{N}_v & \mathring{N}_p & \mathring{N}_r \end{bmatrix}$$

Making the necessary substitutions including $\beta = v/u_e$, then re-arranging we get

$$\begin{bmatrix} \dot{\beta} \\ \dot{p} \\ \dot{r} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} Y_p & Y_\beta & Y_r - 1 & mg\cos\theta_e \\ L_p & L_\beta & L_r & 0 \\ N_p & N_\beta & N_r & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ \beta \\ r \\ \phi \end{bmatrix} + \begin{bmatrix} Y_{\delta_a} & Y_{\delta_r} \\ L_{\delta_a} & L_{\delta_r} \\ N_{\delta_a} & N_{\delta_r} \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix}$$

$$- \begin{bmatrix} Y_\beta & Y_p & Y_r \\ L_\beta & L_p & L_r \\ N_\beta & N_p & N_r \end{bmatrix} \begin{bmatrix} \beta_g \\ p_g \\ r_g \end{bmatrix}$$

Lateral motion derivatives functions

$$
\begin{aligned}
Y_\beta &= C_{y_\beta}\frac{QS}{m} \qquad (m/s^2)\\
Y_p &= C_{y_p}\frac{QSb}{2mu_o} \qquad (m/s)\\
Y_r &= C_{y_r}\frac{QSb}{2mu_o} \qquad (m/s)\\
N_\beta &= C_{n_\beta}\frac{QSb}{I_x x} \qquad (s^{-2})\\
N_p &= C_{n_p}\frac{QSb^2}{2I_{xx}u_o} \qquad (s^-1)\\
N_r &= C_{n_r}\frac{QSb^2}{2I_{xx}u_o} \qquad (s^-1)\\
L_\beta &= C_{l_\beta}\frac{QSb}{I_x x} \qquad (s^{-2})\\
L_p &= C_{l_p}\frac{QSb^2}{2I_{xx}u_o} \qquad (s^-1)\\
L_r &= C_{l_r}\frac{QSb^2}{2I_{xx}u_o} \qquad (s^-1)\\
Y_{\delta a} &= C_{y_{\delta a}}\frac{QS}{m} \qquad (m/s^2)\\
Y_{\delta r} &= C_{y_{\delta r}}\frac{QS}{m} \qquad (m/s^2)\\
N_{\delta a} &= C_{n_{\delta a}}\frac{QSb}{I_{xx}} \qquad (s^2)\\
N_{\delta r} &= C_{n_{\delta r}}\frac{QSb}{I_{xx}} \qquad (s^-2)\\
L_{\delta a} &= C_{l_{\delta a}}\frac{QSb}{I_{xx}} \qquad (s^-2)\\
L_{\delta_r} &= C_{l_{\delta r}}\frac{QSb}{I_{xx}} \qquad (s^-2)
\end{aligned}
$$

Appendix B: Codes

FeedforwardNet.m

It implements a normal feedforward neural network

```
function [outp,op]  = TrainNeuralNet(Error1,Error2)

%Description : Feedforword Neural Network

%Function for calculation of feedforward two−layer

%ANN with 3 tanh hidden units and a linear /(tansig) Output

%#codegen


global W V xz

Error1 = Error1/180; Error2 = Error2/180;

Output_of_HiddenLayer = coder.nullcopy(xz);

output = coder.nullcopy(zeros(2,1));

Input = [Error1;Error2];

%Multiply the Input with the synaptic weight matrix W.

Input_of_HiddenLayer =W * Input;

coder.extrinsic('tanh');

%Calculate the Output of the Hidden Layer using tanh function

Output_of_HiddenLayer = tanh(Input_of_HiddenLayer);

xz = Output_of_HiddenLayer;

%Multiply the Output of the Hidden Layer with the synaptic weight

% matrix V
```

```
Input_of_OutputLayer = V' * Output_of_HiddenLayer;

coder.extrinsic('tansig','elliot2sig','awgn','purelin');

% Calculate the Output of the Output Layer,use tansig Function

Output_of_OutputLayer = tansig(Input_of_OutputLayer);

output = Output_of_OutputLayer;

outp= output(1,:);

op= output(2,:);

end
```

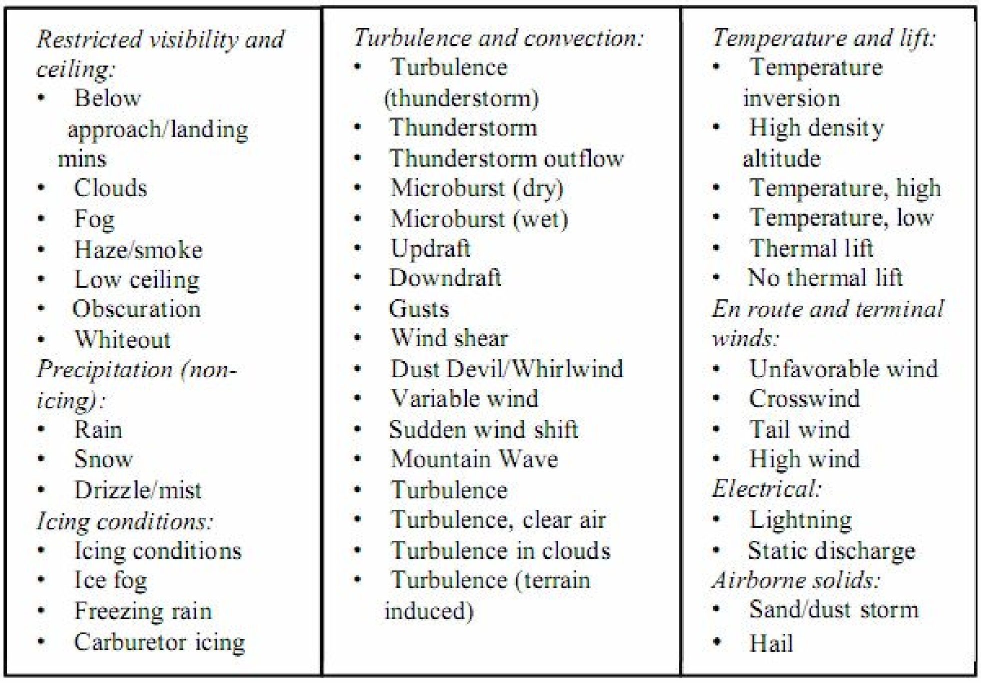
BackPropagation.m

It implements Back propagation algorithm

```
function BackProp(outp,u_opt,op,u_opt1)
%Description : It Implements backpropagation algorithm and updates th
%#codegen


global W V xz
delta_W = coder.nullcopy(zeros(3,2));
delta_V = coder.nullcopy(zeros(3,1));
del_W = coder.nullcopy(zeros(3,1));
Input = [u_opt;u_opt1];
UU=[u_opt;u_opt1]; UU1=[outp;op];
delta_V = xz*(UU - UU1)';
del_W = ([1;1;1] - (xz).^2).*(V*(UU - UU1));
delta_W = del_W*Input';
alpha=.6;
% Updating the neural weights
W = W+alpha.*delta_W;
V = V+alpha.*delta_V;
end
```

Appendix C: Weather Factors



Restricted visibility and ceiling:
- Below approach/landing mins
- Clouds
- Fog
- Haze/smoke
- Low ceiling
- Obscuration
- Whiteout

Precipitation (non-icing):
- Rain
- Snow
- Drizzle/mist

Icing conditions:
- Icing conditions
- Ice fog
- Freezing rain
- Carburetor icing

Turbulence and convection:
- Turbulence (thunderstorm)
- Thunderstorm
- Thunderstorm outflow
- Microburst (dry)
- Microburst (wet)
- Updraft
- Downdraft
- Gusts
- Wind shear
- Dust Devil/Whirlwind
- Variable wind
- Sudden wind shift
- Mountain Wave
- Turbulence
- Turbulence, clear air
- Turbulence in clouds
- Turbulence (terrain induced)

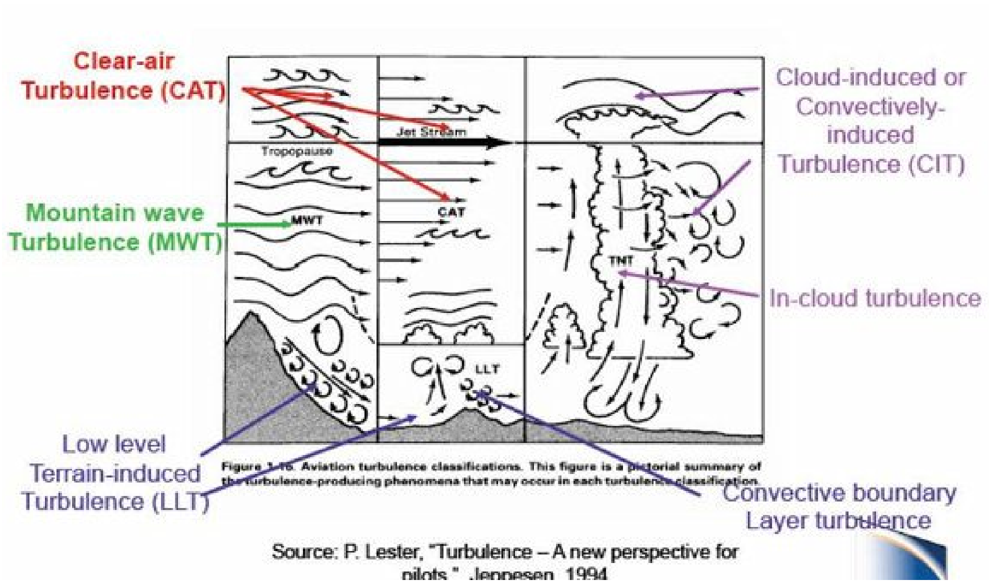
Temperature and lift:
- Temperature inversion
- High density altitude
- Temperature, high
- Temperature, low
- Thermal lift
- No thermal lift

En route and terminal winds:
- Unfavorable wind
- Crosswind
- Tail wind
- High wind

Electrical:
- Lightning
- Static discharge

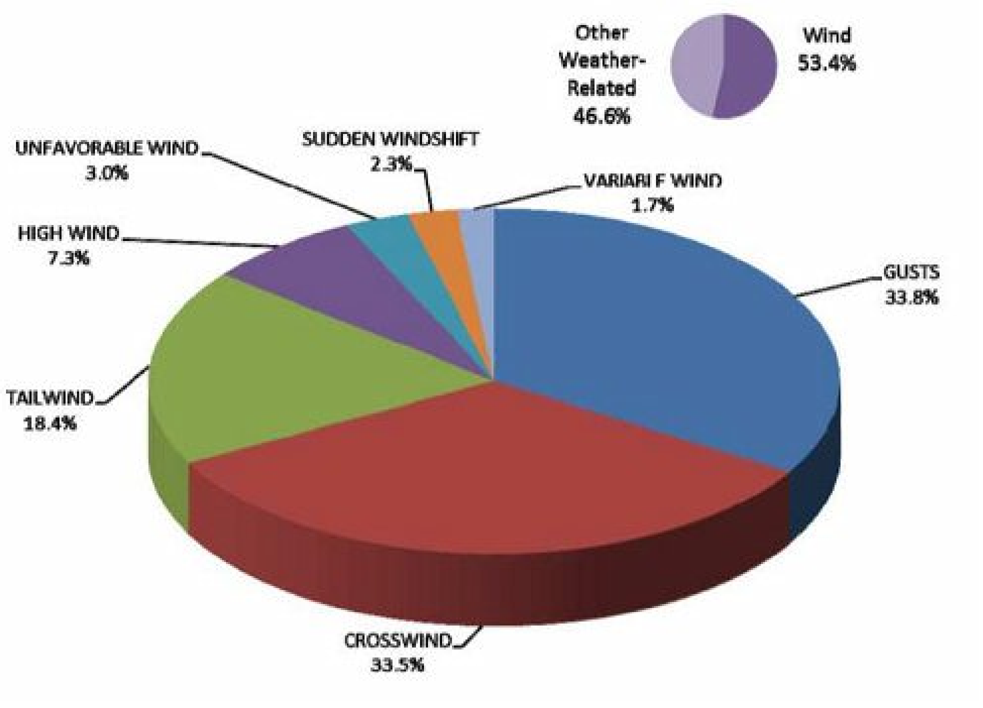
Airborne solids:
- Sand/dust storm
- Hail

NTSB factors and weather categories [48]



Clear-air Turbulence (CAT)
Jet Stream
Tropopause
Mountain wave Turbulence (MWT)
MWT
CAT
TNT
LLT
Cloud-induced or Convectively-induced Turbulence (CIT)
In-cloud turbulence
Low level Terrain-induced Turbulence (LLT)
Convective boundary Layer turbulence
Figure 1.16. Aviation turbulence classifications. This figure is a pictorial summary of the turbulence-producing phenomena that may occur in each turbulence classification.
Source: P. Lester, "Turbulence – A new perspective for pilots," Jeppesen, 1994

Different air turbulence phenomena [48]



NTBS weather citations in aircraft accidents[48]