Faculty of Aerospace Engineering

Longitudinal Flight Control by Reinforcement Learning

Online Adaptive Critic Design Approach to Altitude Control

Jun Hyeon Lee

November 26, 2019

Longitudinal Flight Control by Reinforcement Learning

Online Adaptive Critic Design Approach to Altitude Control

Master of Science Thesis

For obtaining the degree of Master of Science in Aerospace Engineering
at Delft University of Technology

Jun Hyeon Lee

November 26, 2019

Faculty of Aerospace Engineering · Delft University of Technology

Delft University Of Technology
Department Of
Control and Simulation

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance a thesis entitled **"Longitudinal Flight Control by Reinforcement Learning"** by **Jun Hyeon Lee** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: November 26, 2019

Readers:

Dr. G.C.H.E. de Croon

Dr. Ir. E. van Kampen

Dr. Ir. E. Mooij

B. Sun, MSc

Acronyms

ACD	Adaptive Critic Design
ADDHP	Action Dependent Dual Heuristic Programming
ADGDHP	Action Dependent Globalized Dual Heuristic Programming
ADHDP	Action Dependent Heuristic Dynamic Programming
ADI	Adaptive Dynamic Inversion
ADP	Approximate Dynamic Programming
AFC	Automatic Flight Control
AFCS	Automatic Flight Control System
ANN	Artificial Neural Network
DHP	Dual Heuristic Programming
DNN	Deep Neural Network
DP	Dynamic Programming
GDHP	Globalized Dual Heuristic Programming
GPI	Generalized Policy Iteration
HDP	Heuristic Dynamic Programming
IADP	Incremental Approximate Dynamic Programming
IBS	Incremental Back Stepping
IDHP	Incremental Dual Heuristic Programming
IHDP	Incremental Heuristic Dynamic Programming
INDI	Incremental Nonlinear Dynamic Inversion
LADP	Linear Approximate Dynamic Programming
LQR	Linear Quadratic Regulation
LTI	Linear Time Invariant
MC	Monte Carlo
MDP	Markov Decision Process
NDI	Nonlinear Dynamic Inversion
OLS	Ordinary Least Squares

RBF	Radial Basis Function
RL	Reinforcement Learning
RLS	Recursive Least Squares
SARSA	State-Action-Reward-State-Action
TD	Temporal Difference

Contents

Chapter 1

Introduction

The aerospace industry has grown in size and diversity to meet the increasing demand for air travel, recreational use, and industry applications. This has led to technological advancements for all subsystems within aircraft and spacecraft making the systems to be more complex and varied. Parallel to the growth and technological developments within the aerospace industry, a paradigm shift towards increased automation is seen throughout the world further reducing the need for human intervention for control tasks. Autonomous aircraft, sometimes operated close to or within densely populated area, places emphasis on the underlying aircraft flight control system to be more adaptive to adverse conditions such as gusts or system failure.

Most aircraft in operation today are controlled by gain scheduled PID controllers. Gain scheduling is required to allow PID controllers, which are linear controllers, to control a non-linear system such as an aircraft. The development procedure for such controllers include linearized aircraft system identification and controller parameter tuning at multiple operating points within the flight envelope. System identification processes at multiple operating points are typically known to be time consuming and expensive. Additionally, the controller fully depends on the identified model of the aircraft system and shows limited adaptability to unanticipated system dynamics change or large external disturbances. The state-of-the-art controllers found in some modern aircraft utilize Nonlinear Dynamic Inversion (NDI) controllers to eliminate the gain parameter tuning procedure found in gain scheduled PID controllers (D. Enns, Bugajski, Hendrick, & Stein, 1994; Durham, Bordignon, & Beck, 2016). However, this is done at the cost of increased dependence on the identified model of the aircraft. The necessary exhaustive system identification process poses limitations in unconventional aircraft flight controller development. Additionally, the economic burden of the system identification process motivates the development of an online adaptive control system with reduced dependence on a-priori knowledge of the system.

The field of Reinforcement Learning (RL) has received much attention over the recent years mainly owing to its series of successful demonstration of real world applications. A well known demonstration case is AlphaGo Zero winning 100-0 against the previous version of AlphaGo that had defeated the world champion in a game of Go with no expert or domain knowledge provided to the algorithm except for game rules (Silver et al., 2017). Reinforcement Learning can be seen as a computational framework in which an agent learns to derive an optimal policy from its surrounding environment through reinforcement signals based on its actions (Sutton & Barto, 2018). In terms of controller development, RL controllers can simplify the development process if the RL controller is able to derive a near optimal control law by interaction with the environment. Additionally, if the RL controller can accommodate a non stationary environment, an inherently adaptive controller can be developed as online learning during operation will change the control law based on the changed environment. To address the issue of increasing task automation, RL based algorithms have previously demonstrated their ability to make high level decisions from low level raw data as showcased in (Mnih et al., 2013) where an RL algorithm was trained to play Atari games with screen pixels used as input. As such, aircraft control problem formulation within the RL framework may effectively address the challenges presented to the aerospace industry in terms of online adaptive control system development.

Research focus is placed on the Adaptive Critic Design (ACD) RL algorithms to tackle the continuous online control task of an unknown nonlinear system (Werbos, 1977; Prokhorov & Wunsch, 1997; Wang, Zhang, & Liu, 2009; Lewis, Vrabie, & Vamvoudakis, 2012). Due to its explicit parametrized policy improvement and evaluation structure, online learning is facilitated. However, reformulation of the backwards recursive Bellman Equation in ACD translated to a need to maintain a model estimate of the system. In (Zhou, van Kampen, & Chu, 2018b), Incremental Dual Heuristic Programming (IDHP) has been introduced where fully online control without the need for an offline trained model estimation is possible. This has created an ongoing research and development process within TU Delft to implement a fully online RL controller for the Cessna Citation PH-LAB research aircraft. Previous implementation results of IDHP to the full scale Cessna Citation simulation model in (Heyer, 2019) has shown its ability to control a nonlinear system with sudden system dynamics change during operation. However, the use of PID controllers for outer loop attitude and altitude control along with perfect sensor assumption leaves room for further development. Therefore, this thesis work presents the process and implementation results obtained from establishing online RL longitudinal control of Cessna Citation simulation model to altitude control with measurement noise and atmospheric gusts.

1-1 Research Objective

The final goal, to which this thesis work aims to contribute to, is the implementation of online, adaptive reinforcement learning controller to the Cessna Citation PH-LAB research aircraft. As part of the flight controller development process, the controller must be validated on the high fidelity nonlinear Cessna Citation model first. Previous implementation cases of online RL controller to the high fidelity nonlinear Cessna Citation model exist. However, the implemented controller designs assume perfect measurements and no atmospheric disturbances have been accounted for. Additionally, only aircraft rate control has been established using RL controllers with aircraft attitude and heading controlled by PID controllers. The following research objective is presented taking the above points into account.

> The primary research objective is ***to implement a fully online and adaptive reinforcement learning altitude controller for the Cessna Citation***

model with increased simulation fidelity over current IDHP variant controller implementation to further close the gap between simulation and reality.

Under the research objective, 4 research questions have been formulated. Research questions 1 and 2 focus on the choice of the most favorable reinforcement learning controller framework and to propose a controller design to achieve Cessna Citation model altitude control under the presence of measurement noise and atmospheric gusts. Research questions 3 and 4 are concerned with the implementation of the suggested controller design and analysis.

RQ 1 **What is the most favorable reinforcement learning controller framework for Cessna Citation altitude control?**

RQ 2 **What improvements can be made to the most favorable reinforcement learning controller to establish altitude control for the Cessna Citation model?**

RQ 3 **How can the chosen online adaptive reinforcement learning controller design be implemented for the Cessna Citation model?**

RQ 4 **What are the effects of measurement noise and atmospheric gusts to the implemented IDHP controller tracking results?**

1-2 Report Overview

This thesis report consists of three parts. Part 1 contains the scientific article presenting the designed RL controllers, simulation environment, and simulation results. Part 2 contains both literature survey and preliminary analysis done prior to the main research. Literature survey establishes foundational knowledge on flight control and reinforcement learning as well as revealing the state-of-the-art reinforcement learning flight controllers. Based on findings from the literature survey, preliminary analysis is conducted as a feasibility check of the most favorable RL controller framework. Additional results from the thesis project are presented in Part 3. Finally, the report is concluded in Part 4 where the the above research questions are answered and recommendations for future research are given.

Part I

Scientific Paper

Online reinforcement learning for fixed-wing aircraft longitudinal control

Jun Hyeon Lee[*]
Delft University of Technology, The Netherlands

Reinforcement learning is used as a type of adaptive flight control. Adaptive Critic Design (ACD) is a popular approach for online reinforcement learning control due to its explicit generalization of the policy evaluation and the policy improvement elements. A variant of ACD, Incremental Dual Heuristic Programming (IDHP) has previously been developed that allows fully online adaptive control by online identification of system and control matrices. Previous implementation attempts to a high fidelity Cessna Citation model have shown accurate simultaneous altitude and roll angle reference tracking results with outer loop PID and inner loop IDHP rate controllers after an online training phase. This paper presents an implementation attempt to achieve full IDHP altitude control under the influence of measurement noise and atmospheric gusts. Two IDHP controller designs are proposed with and without the cascaded actor structure. Simulation results with measurement noise indicate that the IDHP controller design without the cascaded actor structure can achieve high success ratios. It is demonstrated that IDHP altitude control under measurement noise and atmospheric gusts are achievable under four flight conditions.

Nomenclature

n, m	=	number of states, number of actions
$\mathbf{x}, \mathbf{x}^r, \mathbf{u}$	=	aircraft states, reference aircraft states, aircraft inputs
$\mathbf{s}, \mathbf{a}$	=	reinforcement learning states, reinforcement learning actions
p, q, r	=	aircraft body rates: roll rate, pitch rate, yaw rate
V_{TAS}	=	aircraft true airspeed
α, β	=	angle of attack, sideslip angle
ϕ, θ, ψ	=	aircraft attitude angles: roll angle, pitch angle, yaw angle
x_E, y_E, h	=	aircraft earth reference frame positions
PLA_1, PLA_2	=	engine throttle settings
$\delta_e, \delta_a, \delta_r$	=	aircraft control surface deflection angles: elevator, aileron, rudder
r, c	=	reward, cost
κ	=	reward/cost normalizing term
$J\ \lambda$	=	cost-to-go function, cost-to-go function gradient
$\hat{J}, \hat{\lambda}$	=	cost-to-go function estimate, cost-to-go function gradient estimate
γ	=	discount factor
η_a	=	actor learning rate for baseline controller
η_{a1}, η_{a2}	=	actor 1 learning rate, actor 2 learning rate for cascaded controller
η_c	=	critic learning rate
A_{in}, C_{in}	=	actor input, critic input
γ_{RLS}	=	recursive least squares forgetting factor
$\mathbf{\Theta}, \mathbf{P}$	=	parameter matrix, covariance matrix
$K\ , \epsilon$	=	recursive least squares kalman gain, innovation error
$\mathbf{F}, \mathbf{G}$	=	system matrix, control matrix
$\hat{\mathbf{F}}, \hat{\mathbf{G}}$	=	system matrix estimate, control matrix estimate

[*]MSc. Student, Control & Simulation, Faculty of Aerospace Engineering, Delft University of Technology

I. Introduction

Within, and not limited to, the aerospace industry, there has been a clear trend towards increased automation where higher level tasks are relieved from human intervention. This trend is more apparent within the recreational sector of the industry where Unmanned Aerial Vehicles (UAVs) have gained popularity. A prime example of such increased level of automation is the "follow me mode" feature found in both professional and recreational UAVs demonstrating vision based control [1].

Although the level of autonomy of aircraft systems has been constantly increasing, the underlying attitude and heading flight controller design remains relatively unchanged. Most aircraft in operation to date rely on gain scheduled PID feedback controllers. System identification is essential at multiple operating points within the flight envelope at which controller gains must be tuned through an optimization process. Not only is the development costly, the designed controller show limitations in coping with unanticipated changes in system dynamics or large external disturbances. Nonlinear Dynamic Inversion(NDI) controllers found within some high performance aircraft in operation eliminate the gain scheduling process, but is yet dependent on identified model fidelity and online state measurements [2, 3].

When automated aircraft, such as UAVs, are readily available to the public and become more embedded in daily lives, the need for adaptive controllers is emphasized. Also from the manufacturer's perspective, the benefits of adaptive controllers are clear. Ability to maintain control throughout change in system dynamics due to system failure or large external disturbances such as gust are attractive features for an aircraft. Therefore it can be stated that there is an industry-wide demand for an adaptive flight controller.

Among nonlinear adaptive controller designs, Adaptive Dynamic Inversion (ADI) has been developed with the use of neural networks for online identification of plant inversion error [4] and online identification of control matrix triggered by fault detection [5]. Conceived by [6], Incremental Nonlinear Dynamic Inversion (INDI) based on angular acceleration feedback has shown successful simulation results for a T-tailed UAV model [7], helicopter model [8], and later successfully demonstrated for a flying wing UAV and a passenger aircraft [9, 10]. Applying similar measurement based controller design strategy seen in INDI, Incremental Backstepping (IBS) control method has also been developed [11].

As an alternative approach to adaptive flight control, Reinforcement Learning (RL) can be considered. Within the scope of control, RL can be defined as a process of deriving optimal control policy based on observations made from the environment [12]. Adaptive Critic Designs (ACDs) is a class of Dynamic Programming (DP) RL algorithms where the backwards recursive calculation of Bellman equation is redefined as an algorithm that steps forward in time [13]. ACD maintains separate parametrized actor and critic elements which respectively handles policy improvement and policy evaluation and can be categorized by the critic estimate: Heuristic Dynamic Programming (HDP), Dual Heuristic Dynamic Programming (DHP), and Globalized DHP (GDHP) [14–16]. The separate policy evaluation and improvement structure allows the ACD method to combine the benefits of critic-only and actor-only methods allowing facilitated online implementation [17]. For flight control, several implementation cases can be found. One of the first implementation cases for linear aircraft control simulation can be found in [18]. For nonlinear flight control, successful implementation cases have been reported for a full scale model of: missile [19, 20], fixed-wing aircraft [21–26], and helicopter [27].

Through online estimation of system and control matrices based on incremental measurements, Incremental HDP (IHDP) [24] and Incremental DHP (IDHP) [25] have demonstrated successful control of a nonlinear system without the need for an offline trained model. IDHP was then implemented to the full scale model of the Cessna Citation aircraft model for IDHP angular rate control under perfect sensor assumption [26]. The next step towards implementation in an actual Citation aircraft consists of removing the perfect sensor assumption and establishing longitudinal RL control for altitude tracking.

The contributions of this paper are: 1) Comparison of IDHP controller designs with and without cascaded actor network, 2) Measurement noise impact analysis to IDHP controller designs, 3) IDHP altitude control feasibility demonstration under measurement noise and atmospheric disturbances. Results from the implementation process of extending RL longitudinal control to altitude tracking while adding measurement noise and atmospheric disturbances are provided. The implementation process is divided into three stages. The first stage consists of batch simulation of the two designed IDHP controllers for longitudinal altitude tracking task of the full scale Cessna Citation model under perfect sensor assumption. In the second stage, measurement noise is added to the simulation environment and analysis of its impact is carried out with batch simulation results. The first and second stages combined can be understood as a process to determine a more suitable controller design for altitude tracking task. Finally, the third stage performs empirical analysis on the chosen controller design under light and moderate atmospheric gust scenarios for four flight conditions.

Section II defines the flight control problem as a Markov Decision Process (MDP) and presents the structure and the adaptation process of the IDHP controller. Section III describes the simulation environment and the design process of IDHP controllers are presented. Section IV presents the batch simulation results and analysis of designed controllers for the longitudinal altitude tracking task. Finally, V concludes the paper and provides recommendations for further analysis.

II. Reinforcement Learning Framework

In this section the RL framework is presented. First, continuous reference tracking task is redefined within the Reinforcement Learning (RL) Adaptive Critic Designs (ACDs) framework. Second, Incremental Dual Heuristic Programming(IDHP) is described regarding its algorithm procedure modification and overall structure.

A. Adaptive Critic Designs Problem

Flight control can be described as a process to achieve the primary goal of error minimization between the desired and the actual states of an aircraft, x_t^r and x_t. Reformulated as a MDP, the process can be represented by an agent occupying state $s_t \in \mathbb{R}^n$ at decision epoch t choosing action $a_t \in \mathbb{R}^m$ according to policy π to receive scalar reward $r_{t+1} \in \mathbb{R}$ and next state $s_{t+1} \in \mathbb{R}^m$ signal from the environment.

Let us define the reward to be negatively proportional to be the sum of squared error at decision epoch t. The reward function is shown in Eq. 1 where l is the number of tracked states and κ_j is a normalizing term for each tracked state s_j. As an alternative view used in this paper, the goal can be represented in terms of minimizing cost where cost is defined as negative reward shown in Eq. 2. Through minimizing the cumulative sum of cost the agent receives, the primary goal of flight control can be achieved.

$$\mathbf{r}_t = r(\mathbf{s}_t) = -\sum_{j=1}^{l} \kappa_j (\mathbf{s}_{t,j}^r - \mathbf{s}_{t,j})^2 \tag{1}$$

$$\mathbf{c}_t = -\mathbf{r}_t = \sum_{j=1}^{l} \kappa_j (\mathbf{s}_{t,j}^r - \mathbf{s}_{t,j})^2 \tag{2}$$

Within the ACD framework, cumulative cost minimization is achieved through the forward step calculation of the Bellman equation [13]. The non-negative cost-to-go function $J(s_t)$ is defined in the form of infinite horizon discounted model as shown in Eq. 3 where $\gamma \in [0, 1]$ represents the discount factor. The flight control problem thus becomes a problem of parametrizing a policy such that the cost-to-go function is minimized.

$$J(\mathbf{s}_t) = \sum_{i=t}^{\infty} \gamma^{i-t} \mathbf{c}_i \tag{3}$$

B. IDHP Agent Overview

The overall structure of the agent follows from [25] as initially conceived with minor changes to the algorithm procedure. Figure 1 shows a schematic of the IDHP agent interacting with an environment. The shaded blocks represent elements that belong to the agent and the white blocks represent elements within the environment. An overview of each element within the agent is given in this subsection.

1. IDHP Algorithm Procedure Modification

The IDHP agent presented in [25] shows that the critic is trained backwards in time. The actor is trained forwards in time, but requires the next time step critic estimations based on next time step states predicted by the incremental model. The advantage of such an adaptation scheme is that additional actor-critic adaptation cycles are possible during one decision epoch. The disadvantage is the reliance on the incremental model to predict the next time step states. This disadvantage is more pronounced in the presence of noise and disturbances that further decreases the reliability of the incremental model. Therefore a measurement based adaptation scheme as seen in [28] and previously implemented in [26] is employed.

Figure 1 shows the agent adaptation occurring over two measurement time steps where the forward pass through the critic is done once in each time step providing the current and next time step cost-to-go function gradients. The

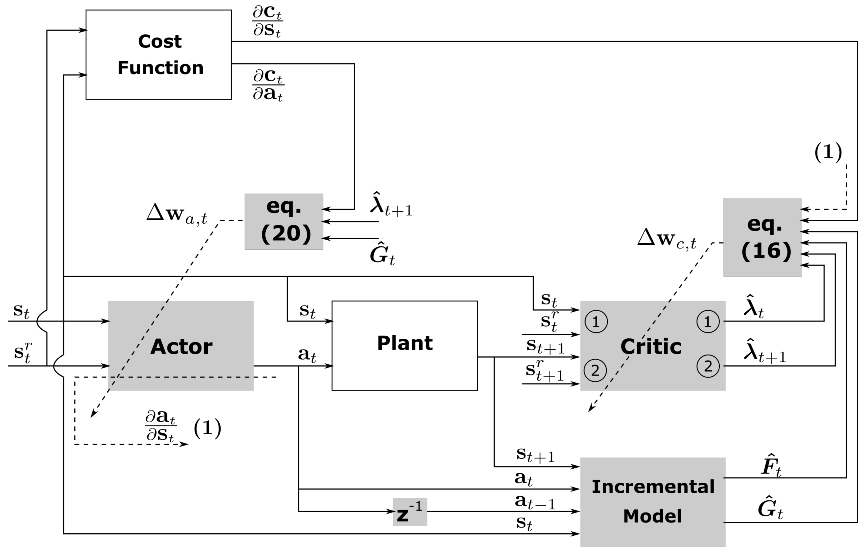

Cost Function
$\frac{\partial \mathbf{c}_t}{\partial \mathbf{s}_t}$
$\frac{\partial \mathbf{c}_t}{\partial \mathbf{a}_t}$
(1)
$\Delta \mathbf{w}_{a,t}$
eq. (20)
$\hat{\boldsymbol{\lambda}}_{t+1}$
$\hat{G}_t$
$\Delta \mathbf{w}_{c,t}$
eq. (16)
$\mathbf{s}_t$
$\mathbf{s}_t^r$
Actor
$\mathbf{a}_t$
Plant
$\mathbf{s}_{t+1}$
$\mathbf{s}_{t+1}^r$
Critic
$\hat{\boldsymbol{\lambda}}_t$
$\frac{\partial \mathbf{a}_t}{\partial \mathbf{s}_t}$ (1)
Incremental Model
$\hat{F}_t$
$\mathbf{a}_{t-1}$
$\mathbf{z}^{-1}$

Fig. 1 IDHP schematic diagram with measurement based adaptation paths

motivation for employing such an adaptation scheme is based on the assumption of small time steps and relatively slow changing environment dynamics which forms the fundamental basis for the use of incremental model in IDHP. If the decision epoch frequency is sufficiently high, then the advantage of reduced reliance on the incremental model outweighs the benefit of multiple adaptations during one epoch.

2. Incremental Model

IDHP utilizes an instantaneous linear system model identified through a first order Taylor expansion. Given sufficient sampling rate and relatively slow changing system dynamics the identified linear model at each time instance can be said to be time varying and adequately representative for use in ACDs [24, 25].

Consider a nonlinear discrete system where its time varying states are defined by Eq. 4. The first order Taylor expansion around t_0 gives Eq. 5. Setting $t_0 = t-1$, Eq. 5 becomes Eq. 6 where $\mathbf{F}_{t-1} = \frac{\partial f(\mathbf{s}_{t-1},\mathbf{a}_{t-1})}{\partial \mathbf{s}_{t-1}} \in \mathbb{R}^{n\times n}$ is the system matrix and $\mathbf{G}_{t-1} = \frac{\partial f(\mathbf{s}_{t-1},\mathbf{a}_{t-1})}{\partial \mathbf{a}_{t-1}} \in \mathbb{R}^{n\times m}$ is the control matrix. The incremental form of the Taylor expansion equation is shown in Eq. 7.

$$\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t) \in \mathbb{R}^n \tag{4}$$

$$\mathbf{s}_{t+1} \approx f(\mathbf{s}_{t_0}, \mathbf{a}_{t_0}) + \frac{\partial f(\mathbf{s},\mathbf{a})}{\partial \mathbf{s}}|_{\mathbf{s}_{t_0},\mathbf{a}_{t_0}}(\mathbf{s}_t - \mathbf{s}_{t_0}) + \frac{\partial f(\mathbf{s},\mathbf{a})}{\partial \mathbf{a}}|_{\mathbf{s}_{t_0},\mathbf{a}_{t_0}}(\mathbf{a}_t - \mathbf{a}_{t_0}) \tag{5}$$

$$\mathbf{s}_{t+1} \approx \mathbf{s}_t + \mathbf{F}_{t-1}(\mathbf{s}_t - \mathbf{s}_{t-1}) + \mathbf{G}_{t-1}(\mathbf{a}_t - \mathbf{a}_{t-1}) \tag{6}$$

$$\Delta \mathbf{s}_{t+1} \approx \mathbf{F}_{t-1}(\Delta \mathbf{s}_t) + \mathbf{G}_{t-1}(\Delta \mathbf{a}_t) \tag{7}$$

It has been shown that given the assumptions of high sampling frequency and relatively slow system dynamics, the change in system states can be approximated by previous incremental state and action measurements with $\mathbf{F}_{t-1}$ and $\mathbf{G}_{t-1}$ approximated by the incremental model. Due to algorithm procedure modification, the adaptation occurs after state measurements based on agent action output and does not rely on incremental state prediction. Therefore, $\mathbf{F}_t$ and $\mathbf{G}_t$ estimated by the incremental model with newly measured state information are utilized for the critic and the actor adaptation procedure. The structure of the system matrix $\mathbf{F}_t$ and control matrix $\mathbf{G}_t$ are given in Eq. 8 and Eq. 9.

$$\mathbf{F}_t = \begin{bmatrix} \frac{\partial s_{1,t}}{\partial s_{1,t}} & \frac{\partial s_{1,t}}{\partial s_{2,t}} & \cdots \\ \vdots & \ddots & \\ \frac{\partial s_{n,t}}{\partial s_{1,t}} & & \frac{\partial s_{n,t}}{\partial s_{n,t}} \end{bmatrix} \tag{8}$$

$$\mathbf{G}_t = \begin{bmatrix} \frac{\partial s_{1,t}}{\partial a_{1,t}} & \frac{\partial s_{1,t}}{\partial a_{2,t}} & \cdots \\ \vdots & \ddots & \\ \frac{\partial s_{n,t}}{\partial a_{1,t}} & & \frac{\partial s_{n,t}}{\partial a_{m,t}} \end{bmatrix} \tag{9}$$

3. Critic

The critic in an ACD structure is responsible for policy evaluation of the separated policy evaluation and improvement structure. The critic block within IDHP estimates the partial derivative of the cost-to-go function with respect to the states, $\lambda(\mathbf{s}_t) = \frac{\partial J(\mathbf{s}_t)}{\partial \mathbf{s}_t}$, and must be continuously differentiable. Therefore the critic can be defined as a differentiable $\lambda(\mathbf{s}, \mathbf{w}_c)$ parametrization.

The adaptation rule of the critic follows the one step temporal difference TD(0) gradient descent method. The TD error is defined in Eq. 10 where $\hat{\lambda}(\mathbf{s})$ represents the estimated partial derivative of the cost-to-go function under the current policy. The adaptation process aims to minimize the error function given in Eq. 11.

$$\mathbf{e}_{c,t} = \frac{\partial\left(\hat{J}(\mathbf{s}_t) - c_t - \gamma\hat{J}(\mathbf{s}_{t+1})\right)}{\partial \mathbf{s}_t} = \hat{\lambda}(\mathbf{s}_t) - \frac{\partial c_t}{\partial \mathbf{s}_t} - \gamma\hat{\lambda}(\mathbf{s}_{t+1})\frac{\partial \mathbf{s}_{t+1}}{\partial \mathbf{s}_t} \tag{10}$$

$$E_{c,t} = \frac{1}{2}\mathbf{e}_{c,t}^T \mathbf{e}_{c,t} \tag{11}$$

The critic weight update process can thus be defined in Eq. 12 where $\Delta\mathbf{w}_{c,t}$ is given in Eq. 13 with η_c as learning rate for the critic.

$$\mathbf{w}_{c,t+1} = \mathbf{w}_{c,t} + \Delta\mathbf{w}_{c,t} \tag{12}$$

$$\Delta\mathbf{w}_{c,t} = -\eta_c \frac{\partial E_{c,t}}{\partial\hat{\lambda}(\mathbf{s}_t)}\frac{\partial\hat{\lambda}(\mathbf{s}_t)}{\partial \mathbf{w}_{c,t}} = -\eta_c\left(\hat{\lambda}(\mathbf{s}_t) - \frac{\partial c_t}{\partial \mathbf{s}_t} - \gamma\hat{\lambda}(\mathbf{s}_{t+1})\frac{\partial \mathbf{s}_{t+1}}{\partial \mathbf{s}_t}\right)\frac{\partial\hat{\lambda}(\mathbf{s}_t)}{\partial \mathbf{w}_{c,t}} \tag{13}$$

Both $\hat{\lambda}(\mathbf{s}_t)$ and $\hat{\lambda}(\mathbf{s}_{t+1})$ are directly available from the critic through two separate runs with current and next time step critic inputs. The cost gradient vector $\frac{\partial c_t}{\partial \mathbf{s}_t}$ is obtained from the predefined cost function. The remaining term $\frac{\partial \mathbf{s}_{t+1}}{\partial \mathbf{s}_t}$ is calculated by Eq. 14 acknowledging the influences of both the current state and the current action to the next state.

$$\frac{\partial \mathbf{s}_{t+1}}{\partial \mathbf{s}_t} = \frac{\partial \mathbf{s}_{t+1}}{\partial \mathbf{s}_t} + \frac{\partial \mathbf{s}_{t+1}}{\partial \mathbf{a}_t}\frac{\partial \mathbf{a}_t}{\partial \mathbf{s}_t} \tag{14}$$

The term $\frac{\partial \mathbf{a}_t}{\partial \mathbf{s}_t}$ can be retrieved from the actor gradients. The partial derivatives $\frac{\partial \mathbf{s}_{t+1}}{\partial \mathbf{s}_t}$ and $\frac{\partial \mathbf{s}_{t+1}}{\partial \mathbf{a}_t}$ can be calculated using $\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t)$. Recalling the definition of system and control matrices, $\mathbf{F}_t = \frac{\partial f(\mathbf{s}_t,\mathbf{a}_t)}{\partial \mathbf{s}_t}$ and $\mathbf{G}_t = \frac{\partial f(\mathbf{s}_t,\mathbf{a}_t)}{\partial \mathbf{a}_t}$ can be defined resulting in $\frac{\partial \mathbf{s}_{t+1}}{\partial \mathbf{s}_t} \approx \hat{\mathbf{F}}_t$ and $\frac{\partial \mathbf{s}_{t+1}}{\partial \mathbf{a}_t} \approx \hat{\mathbf{G}}_t$. Therefore Eq. 14 can be reformulated into Eq. 15. The final critic weight update process is defined in Eq. 16

$$\frac{\partial \mathbf{s}_{t+1}}{\partial \mathbf{s}_t} \approx \hat{\mathbf{F}}_t + \hat{\mathbf{G}}_t\frac{\partial \mathbf{a}_t}{\partial \mathbf{s}_t} \tag{15}$$

$$\mathbf{w}_{c,t+1} = \mathbf{w}_{c,t} - \eta_c\left[\hat{\lambda}(\mathbf{s}_t) - \frac{\partial c_t}{\partial \mathbf{s}_t} - \gamma\hat{\lambda}(\mathbf{s}_{t+1})\left(\hat{\mathbf{F}}_t + \hat{\mathbf{G}}_t\frac{\partial \mathbf{a}_t}{\partial \mathbf{s}_t}\right)\right]\frac{\partial\hat{\lambda}(\mathbf{s})_t}{\partial \mathbf{w}_{c,t}} \tag{16}$$

4. Actor

While the critic is responsible for policy evaluation, the actor handles policy improvement completing the policy iteration procedure within ACDs. Given states s_t, the actor outputs action a_t according to the parametrized policy $\pi(\mathbf{s}, \mathbf{w}_a)$. The actor function approximator must also be differentiable.

The goal of the actor is to find a policy such that $J(\mathbf{s}_t)\,\forall\, t$ is minimized as shown in Eq. 17. This can then be reformulated into actor weight update process shown in Eq. 18 and Eq. 19.

$$arg\min_{\mathbf{a}_t} \frac{\partial J(\mathbf{s}_t)}{\partial \mathbf{a}_t} = arg\min_{\mathbf{a}_t} \frac{\partial\big(c_t + \gamma J(\mathbf{s}_{t+1})\big)}{\partial \mathbf{a}_t} \tag{17}$$

$$\mathbf{w}_{a,t+1} = \mathbf{w}_{a,t} + \Delta\mathbf{w}_{a,t} \tag{18}$$

$$\Delta\mathbf{w}_{a,t} = -\eta_a \left[\frac{\partial c_t}{\partial \mathbf{a}_t} + \gamma \frac{\partial J(\mathbf{s}_{t+1})}{\partial \mathbf{a}_t}\right] \frac{\partial \mathbf{a}_t}{\partial \mathbf{w}_{a,t}} = -\eta_a \left[\frac{\partial c_t}{\partial \mathbf{a}_t} + \gamma \hat{\lambda}(\mathbf{s})_{t+1} \frac{\partial \mathbf{s}_{t+1}}{\partial \mathbf{a}_t}\right] \frac{\partial \mathbf{a}_t}{\partial \mathbf{w}_{a,t}} \tag{19}$$

The $\frac{\partial c_t}{\partial \mathbf{a}_t}$ term can be obtained from the cost function if set to depend on action. $\hat{\lambda}(\mathbf{s})_{t+1}$ is estimated by the critic and $\frac{\partial \mathbf{s}_{t+1}}{\partial \mathbf{a}_t}$ is provided by incremental model output $\hat{\mathbf{G}}_t$. The actor weight update equation can therefore be finalized in Eq. 20.

$$\mathbf{w}_{a,t+1} = \mathbf{w}_{a,t} - \eta_a \left[\frac{\partial c_t}{\partial \mathbf{a}_t} - \gamma \hat{\lambda}(\mathbf{s})_{t+1} \hat{\mathbf{G}}_t\right] \frac{\partial \mathbf{a}_t}{\partial \mathbf{w}_{a,t}} \tag{20}$$

III. Environment Setup and Flight Controller Designs

In this section the simulation setup for the Cessna Citation aircraft longitudinal altitude reference tracking task is presented. First the environment setup including the Cessna Citation I simulation model, gust model, and the cost are given. Then, the structure of the incremental model, critic, and actor as implemented in the simulation are illustrated. Note that that two IDHP based controllers have been designed. One featuring cascaded actor network and one without.

A. Environment

The environment embodies all elements external to the decision making IDHP agent within the simulation. Software package developed within TU Delft combined with the gust model have been utilized to simulate Cessna Citation aircraft dynamics with external atmospheric disturbances as a black box model of the plant to be controlled. A cost function defining the objective of the IDHP controller is designed.

1. Cessna Citation Dynamics

The DASMAT software package, developed within TU Delft for high fidelity aircraft simulation, has been configured with Cessna Citation I aircraft specific parameters. In such configuration settings, DASMAT software package represents the Cessna Citation aircraft model including aerodynamic, propulsion, engine, and control actuator models and will henceforth be referred as the Citation model. The Citation model provides next time step aircraft states $\mathbf{x}_{t+1}$ given actuator command $\mathbf{u}_t$, throttle command $\mathbf{u}_{eng,t}$, and gust terms $\mathbf{u}_{g,t}$ where the current aircraft states $\mathbf{x}_t$ are maintained within the model. The model is simulated at 100 Hz assuming synchronous aircraft state $\mathbf{x}$ measurements. Figure 2 provides an overview of the input-output relationship of the model where the input and output terms are given in Eq. 21 to Eq. 24.

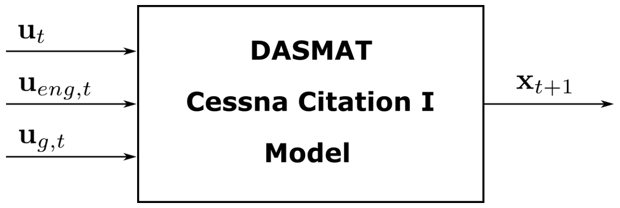


Fig. 2 DASMAT Cessna Citation model showing input and output variables

$$\mathbf{u} = [\delta_e,\ \delta_a,\ \delta_r] \tag{21}$$

$$\mathbf{u}_{eng} = [PLA_1,\ PLA_2] \tag{22}$$

$$\mathbf{u}_g = \left[\hat{u}_g,\ \alpha_g,\ \beta_g,\ \dot{\hat{u}}_g\bar{c}/V,\ \dot{\alpha}_g\bar{c}/V,\ \dot{\beta}_g,\ \hat{u}_{ga},\ \alpha_{ga}\right] \tag{23}$$

$$\mathbf{x} = [p,\ q,\ r,\ V_{TAS},\ \alpha,\ \beta,\ \phi,\ \theta,\ \psi,\ h,\ x_E,\ y_E] \tag{24}$$

The Citation model is initialized at steady flight trimmed states before controlled flight simulation. For full RL control during operation, it is clear that such trimmed initial condition needs to be removed. In [27] a separate neural network has been utilized to determine the initial trim commands. However, both controller designs have demonstrated its ability to control the system within operating regions considered in this research without the need to re-trim. Therefore, online trimming has not been considered in this research.

To analyze the longitudinal motion characteristic of the Citation model during altitude tracking, states q, α, θ, and h are treated as observable variables controlled with actuator command δ_e. Therefore the reinforcement learning states for altitude tracking can be summarized as shown in Eq. 25.

$$\mathbf{s} = [q,\ \alpha,\ \theta,\ h] \quad \mathbf{a} = [\delta_e] \tag{25}$$

The Citation model is provided including simple proportional feedback controllers for yaw damping and maintaining constant airspeed. For the purpose of this research, which focuses on the longitudinal motion control without airspeed control, both controllers provided within the model are utilized to minimize the influence of uncontrolled states within experiments.

To analyze controller design under the presence of measurement noise, the noise terms have been implemented in the form of zero mean Gaussian noise. Cessna Citation aircraft specific noise terms for states q, V_{TAS} and θ have been set following [29]. Because α is a relatively difficult state to measure as the measurement process involves a mechanical wind vane when compared to body rate and attitude angle states, a larger noise standard deviation value of $0.5°$ has been given. Perhaps the most important noise parameter for this research is the altitude noise standard deviation as it directly affects both critic and actor as input. No clear measurement noise statistics are available for the Cessna Citation and a standard deviation of 0.5 meters is assumed. The measurement noise parameters can be summarized in Table 1. It is to be noted that no error is given to the control state δ_e as measured δ_e is not utilized during IDHP control.

Table 1 Gaussian noise parameters

States	q **[rad/s]**	α **[rad]**	V_{TAS} **[m/s]**	θ **[rad]**	h **[m]**
Noise standard deviations	$6.325 \cdot 10^{-4}$	$8.73 \cdot 10^{-3}$	0.029	$6.325 \cdot 10^{-4}$	0.5

2. Gust model

As it can be seen in Figure 2, the Citation model allows gust terms as input. The gust terms are processed within the aerodynamic sub model of the Citation model. The aerodynamic force and moment coefficients due to turbulence are calculated and incorporated into calculating total aerodynamic forces and moments.

Only symmetrical gust terms are considered as any asymmetrical gust terms may interfere with longitudinal motion assessment and may even lead to failure when lateral motion is not controlled. The longitudinal gust terms provided to the Citation model can be summarized as: $\hat{u}_g$, α_g, $\dot{\hat{u}}_g\hat{c}/V$, and $\dot{\alpha}_g\hat{c}/V$.

The longitudinal Dryden gust model used in this paper is defined by the transfer functions shown in Eq. 26 and Eq. 27 to calculate the aforementioned symmetric gust terms. Above $609.6\,m$, Dryden gust model assumes isotropic gust and is completely defined by three variables: intensity σ_g $[m^2/s^2]$, scale length L_g $[m]$, and airspeed V_{TAS} $[m/s]$. σ_g needs to be continuously referenced from a lookup table based on gust intensity probability of exceedance. For the purpose of controller performance analysis to different gust intensities, σ_g will instead be set at a constant value representative of the probability of exceedance magnitudes corresponding to light and moderate gust scenarios.

$$H_{\hat{u}_g w}(s) = \frac{\sigma_g}{V_{TAS}} \sqrt{\frac{2L_g}{V_{TAS}}} \frac{1}{1 + \frac{L_g}{V_{TAS}} s} \tag{26}$$

$$H_{\hat{\alpha}_g w}(s) = \frac{\sigma_g}{V_{TAS}} \sqrt{\frac{L_g}{V_{TAS}}} \frac{1 + \sqrt{3}\frac{L_g}{V_{TAS}} s}{1 + \frac{L_g}{V_{TAS}} s} \tag{27}$$

Two different settings for the gust model are used within this paper, summarized in a list shown below. The parameters σ_g and L_g have been referenced from MIL-HDBK-1797 [30]. Scale length has been set to $L_g = 533.4m$ using the Dryden model for high altitude (above $609.6m$). σ_g is set at $\sigma_g = 0.836\, m^2/s^2$ and $\sigma_g = 5.946\, m^2/s^2$ each representing the probability of exceedance $P_{exc} = 10^{-1} \in [0, 1]$ "light gust" and $P_{exc} = 10^{-2} \in [0, 1]$ "moderate gust" for all simulated flight condition altitudes.

1) **No gust**: Zero gust input vector to the Citation model
2) **Light gust**: $\sigma_g = 0.84\, m^2/s^2$ and $L_g = 533.4\, m$
3) **Moderate gust**: $\sigma_g = 5.95\, m^2/s^2$ and $L_g = 533.4\, m$.

3. Cost function

For the task of altitude control, the only predetermined reference provided by the environment is h^r. Following the definition of cumulative cost previously stated in Eq. 2, the cost provided by the environment to the agent can be defined as a scalar term defined by Eq. 28 where κ_h is the scaling term. The scaling term $\kappa_h = 0.0001$ has been chosen to keep the magnitude of cost scalar comparable to cost-to-go function gradient estimates during controller adaptation.

$$c_t = \kappa_h (h_t^r - h_t)^2 \tag{28}$$

For the cascaded model, an additional cost signal based on θ is available due to the reference generated by the outer layer actor network in a cascaded actor network design. With $\kappa_\theta = 1$, Eq. 29 defines the cost function for the cascaded IDHP controller design.

$$c_t = \kappa_h (h_t^r - h_t)^2 + \kappa_\theta (\theta_t^r - \theta_t)^2 \tag{29}$$

B. Altitude Control with IDHP

The agent setup including parameter estimation methods and input/output states are discussed in this subsection. The IDHP agent design largely follows from [25] where the IDHP agent has been introduced. For the purpose of altitude tracking task, the state vector used within IDHP is appended with user generated altitude reference. Therefore the IDHP states and action vectors are redefined in Eq. 30:

$$\mathbf{s} = [q,\, \alpha,\, \theta,\, h,\, h^r], \qquad \mathbf{a} = [\delta_e] \tag{30}$$

1. Incremental Model

The incremental model is identified through Recursive Least Squares (RLS) parameter estimation method. RLS method has been chosen due to its incremental update rule beneficial for online parameter estimation reducing computational cost and alleviating issues arising from matrix inversion. Given states and actions in Eq. 25 the RLS update method can be defined.

Incremental state and action vectors are defined by $\Delta\mathbf{s}_t = \mathbf{s}_t - \mathbf{s}_{t-1}$ and $\Delta\mathbf{a}_t = \mathbf{a}_t - \mathbf{a}_{t-1}$. The goal is to estimate $\mathbf{F}_t$ and $\mathbf{G}_t$ after next time step states have been measured from the plant. The algorithm requires initial covariance matrix $\mathbf{P}_0 \in \mathbb{R}^{n+m \times n+m}$ and $\mathbf{\Theta}_0 \in \mathbb{R}^{n+m \times n}$ to be known. At each RLS algorithm epoch k, the regression vector $\mathbf{r}_k$ and measurement vector $\mathbf{y}_k$ are defined as shown in Eq. 31. The RLS Kalman gain can then be calculated using Eq. 32 with RLS forgetting factor followed by the Kalman error ϵ_k calculation in Eq. 33. Finally, $\hat{\mathbf{P}}_k$ and $\hat{\mathbf{\Theta}}_k$ can be calculated using Eq. 34 and Eq. 35. From $\hat{\mathbf{\Theta}}_k$, both $\hat{\mathbf{F}}_t$ and $\hat{\mathbf{G}}_t$ can be found according to Eq. 36.

$$\mathbf{r}_k = [\Delta\mathbf{s}_t,\, \Delta\mathbf{a}_t], \qquad \mathbf{y}_k = [\Delta\mathbf{s}_{t+1}] \tag{31}$$

$$K_k = \mathbf{P}_{k-1} \cdot \mathbf{r}_k^T \left[\mathbf{r}_k \cdot \mathbf{P}_{k-1} \cdot \mathbf{r}_k^T + \gamma_{RLS}\right]^{-1} \tag{32}$$

$$\boldsymbol{\epsilon}_k = \mathbf{y}_k - \mathbf{r}_k \cdot \hat{\boldsymbol{\Theta}}_{k-1} \tag{33}$$

$$\hat{\boldsymbol{\Theta}}_k = \hat{\boldsymbol{\Theta}}_{k-1} + K_k \cdot \hat{\boldsymbol{\epsilon}}_k \tag{34}$$

$$\hat{\mathbf{P}}_k = \mathbf{P}_{k-1} - K_k \cdot \mathbf{r}_k \cdot \mathbf{P}_{k-1} \tag{35}$$

$$\hat{\boldsymbol{\Theta}}_k = \begin{bmatrix} \hat{\mathbf{F}}_t^T \\ \hat{\mathbf{G}}_t^T \end{bmatrix} \tag{36}$$

The RLS estimation method shown above requires an initial $\mathbf{P}_0$ and $\boldsymbol{\Theta}_0$. The matrices are initialized as shown in Eq. 37. For $\mathbf{F}_0^T$ in $\boldsymbol{\Theta}_0$, the off diagonal elements are set as small non zero values to avoid negligible one step ahead λ_{t+1} contribution to the critic update process during early adaptation phase. The initial $\mathbf{G}_0$ found in $\boldsymbol{\Theta}_0$ have been given negative values with knowledge that positive elevator deflection results in negative change in chosen aircraft states. The final goal of this research is to contribute to developing an online flight controller where minimal knowledge of the controlled system is assumed. To avoid providing too much information to the initial parameter matrix, equal values have been chosen. As for the initial covariance matrix $\mathbf{P}_0$, a simple identity matrix has been given as no prior knowledge of parameter covariances is assumed.

The only tunable parameter in the RLS parameter estimation method is the forgetting factor $\gamma_{RLS} \in [0, 1]$ where higher γ_{RLS} leads to a more conservative update process retaining previous information. The forgetting factor is set to be 0.9 for all simulations conducted in this paper. This decision has been made based on the fact that a relatively stable $\mathbf{F}_t$ and $\mathbf{G}_t$ estimations are desirable for a stable cost-to-go function derivative estimation by the critic.

$$\boldsymbol{\Theta}_0 = \begin{bmatrix} 1 & 0.01 & 0.01 & 0.01 \\ 0.01 & 1 & 0.01 & 0.01 \\ 0.01 & 0.01 & 1 & 0.01 \\ 0.01 & 0.01 & 0.01 & 1 \\ -0.1 & -0.1 & -0.1 & -0.1 \end{bmatrix} \qquad \mathbf{P}_0 = \mathbf{I}_5 \tag{37}$$

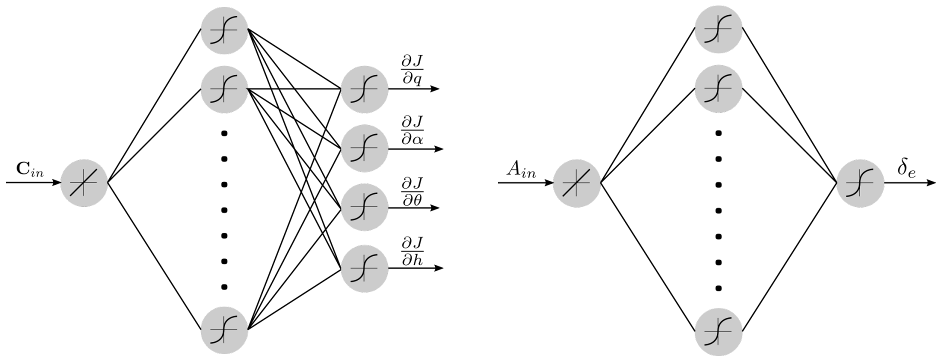
2. Critic and Actor Design

The critic and the actor within an IDHP agent are represented by fully-connected single layer neural networks and their weights are updated by means of back propagation. Figures 3b and 3a show the structure of the neural networks. The critic estimates the partial derivative of the cost-to-go function $\mathbf{J}$ with respect to states while the actor directly outputs the longitudinal control action δ_e.

Linear activation functions have been used for both critic and actor input layers. Hyperbolic tangent, or tanh, activation functions have been used for the hidden layer of both actor and critic neural networks. For the output layer, tanh activation function is chosen again for the critic while a scaled tanh activation function is chosen for the actor.

Hyperbolic tangent activation functions used for the hidden layers and the output layers show desirable qualities when compared to other activation functions. It is a differentiable nonlinear activation function and thus qualifies for use in both actor and critic neural networks. The activation function behaves similarly to the linear activation function until it reaches activation limits, to which it smoothly converges. This bounded property of the hyperbolic tangent activation function is especially useful as an actor output layer activation function. The output activation function dictates the overall shape of the control policy maintained by the actor. And the extent of its influence over the overall control policy is greater when the number actor network input are relatively low. Therefore, by using a scaled tanh activation function, a natural control policy boundary can be set within the physical limitations of the Cessna Citation aircraft.

For both critic and actor input C_{in} and A_{in}, scaled altitude tracking error $h_{scale} \cdot h_e = \kappa_h(h^r - h)$ is used where $\kappa_h = 0.0001$ is a normalizing term to keep the order of magnitude among states similar. For the altitude tracking task,



$\mathbf{C}_{in}$
$\frac{\partial J}{\partial q}$
$\frac{\partial J}{\partial \alpha}$
$\frac{\partial J}{\partial \theta}$
$\frac{\partial J}{\partial h}$
A_{in}
δ_e

(a) Critic neural network **(b) Actor neural network**

Fig. 3 Critic and actor structure

altitude error is the most important and relevant information. By only providing error term h_e the cost-to-go function derivative dimensions are reduced compared to providing all longitudinal states and error information to the critic while minimizing the loss of meaningful information necessary for obtaining a good cost-to-go function derivative estimate. A similar argument can be made about only providing h_e term to the actor. Although it is true that extra states may help the actor to form a more refined policy, the increased dimensionality will hinder learning speed and success ratio. This is commonly known as the curse of dimensionality [28].

The temporal difference forgetting factor $\gamma \in [0, 1]$ directly affects the learning process of the IDHP agent. If set too low, the agent becomes myopic and greedy [28]. Because only the altitude error information is used to estimate the cost-to-go function derivative, a relatively high forgetting factor of 0.9 is set. The number of neurons within the hidden layer is another important factor in ACD agent design. If set too low, the parametrization will be too coarse while a large number of neurons will increase computational complexity. Because of the relative simplicity of the actor and critic neural input, preliminary analysis has shown that the 10 hidden neurons are adequate for generalization.

3. Cascaded Actor Structure

A conventional cascaded PID controller structure utilizes an outer loop controller generating a reference signal for the successive inner loop controller to track. This form of controller structure is typically employed when a controlled state of the inner loop has a faster response in relation to the control input [31]. For fixed-wing aircraft systems, the cascaded controller architecture is implemented in the form of inner loop rate Stability Augmentation System (SAS) and outer loop Control Augmentation System (CAS) enclosed by an autopilot [32].

A cascaded actor network design method has been used for ACDs in several previous studies to explicitly structure the actor network using expert knowledge of the system [23, 25, 27]. Under the primary goal of establishing RL Citation altitude control, the cascaded actor network is implemented for IDHP controller design and compared to the design without. The schematic of the designed cascaded actor network is shown in Figure 4. As an intermediary state, θ has been chosen based on its relative ease of measurement when compared to α and lower signal to noise ratio when compared to q.

The use of cascaded actor network allows an additional θ_e signal to be used as critic input. However, this is not done for two reasons. First, preliminary analysis has shown that additional critic input term negatively affects the overall success ratio of the controller. Second, a more direct comparison with the baseline controller can be made when the information provided to the critic are the same among both designs.

4. Adaptive Learning Rates

The last tunable parameters for the IDHP agent are the critic and actor learning rates: η_a and η_c. These non-negative parameters govern the adaptation speed of the critic and the actor. As a relatively large learning rate after convergence to target may lead to divergence due to unnecessary adaptation, the learning rate are typically set to decrease with time [16, 27].

An alternative approach is the use of adaptive learning rates where the learning rates are tuned based on error measurements instead of decaying over time. Successful ACD controller implementation results have been reported in

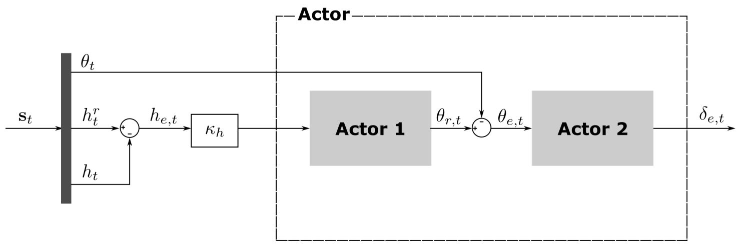


Fig. 4 Schematic of the cascaded actor for altitude tracking task

[33] and [25] using adaptive learning rates. The main advantage of such error based adaptive learning rates compared to time decaying learning rates is that the agent can re-adapt to changed environment with sufficient learning rate regardless of the amount of time that had passed.

Experiments conducted in this paper use error based adaptive learning rates. The learning rates are adjusted based on the Root Mean Squared Error (RMSE) of altitude to reference over the past 100 measurements which corresponds to 1 second in this simulation setting. If the RMSE of altitude is lower than h_{thresh}, learning rate decreases to η_{low}. Otherwise the learning rate is maintained at initial value η_{high}. This "on-off" strategy based on error threshold is less refined when compared to gradual adjustments seen in [33] and [25] but allows faster and stronger adaptation when environment changes are sudden. The parameters for the adaptive learning rate strategy used in the simulation are given in Eq. 38 for the baseline controller design and in Eq. 39 for the cascaded controller design. For the cascaded controller design, an additional θ_{thresh} is set for the past 10 measurements to adjust the learning rate of the second actor network.

$$\eta_{a,high} = [5,\ 10,\ 25], \quad \eta_{c,high} = [2,\ 5,\ 10], \quad \eta_{a,low} = 0.2, \quad \eta_{c,low} = 0.2, \quad h_{thresh} = 20m \tag{38}$$

$$\eta_{a1,high} = 25, \quad \eta_{a2,high} = 1, \quad \eta_{c,high} = 10, \quad \eta_{a,low} = 0.2, \quad \eta_{c,low} = 0.2, \quad h_{thresh} = 20m, \quad \theta_{thresh} = 2^\circ \tag{39}$$

IV. Results and Discussions

This section presents and discusses simulation results of 3 experiments. The first experiment is set up to compare the non-cascaded IDHP controller design (baseline controller) and the cascaded actor IDHP controller design (cascaded controller) in a perfect scenario where both measurement noise and disturbances are not present. The second experiment is designed to determine a controller design favorable for online longitudinal control of the Citation model in the presence of measurement noise. The third experiment is aimed to assess the performance of the chosen controller to external disturbances as well as measurement noise.

The altitude reference signal for experiments 1 and 2 is generated by a sinusoidal function with amplitude 250 meters and frequency 0.005 Hz to represent repeated climb and descent at approximately $1500 ft/min$. For experiment 3, a reference altitude consisting of climb, short cruise, descent, and cruise has been set to simulate a realistic scenario of cruise altitude increase shortly followed by cruise altitude correction.

Each controller design has been simulated for 300 independent runs at 100 Hz with no prior training. Through multiple batch simulations, it has been found that 300 independent runs are adequately representative of the controller performance for the chosen neural network weight initialization.

For experiments 1 and 2, the success conditions are defined as follows. The first sinusoidal period is named the transient period where the IDHP controller adapts its policy for reference tracking. The second sinusoidal period is defined as the steady phase where the controller should have converged to a stable near optimal policy tracking the reference altitude. Therefore the following success condition can be defined in Eq. 40 based on RMS error calculated in Eq. 41 where the number of time steps within steady phase is given by N_{steady}. Success condition threshold S_{thresh} has been set at 20 meters for tight success, 40 meters for loose success, and 100 meters for converging runs. This success condition structuring is a modified version of the tight and loose success condition structuring given in [15] for ACD comparison that allows controller comparison in terms of accuracy and convergence.

$$S_{thresh} > RMSE_{steady} \tag{40}$$

$$RMSE_{steady} = \sqrt{\frac{\sum_{i=0}^{N_{steady}} (h_i^r - h_i)^2}{N_{steady}}} \tag{41}$$

For experiment 3, the success condition has been defined based on the RMSE during final 30 seconds of the cruise phase. The run is considered successful if the calculated RMSE is within 40 meters of the reference altitude.

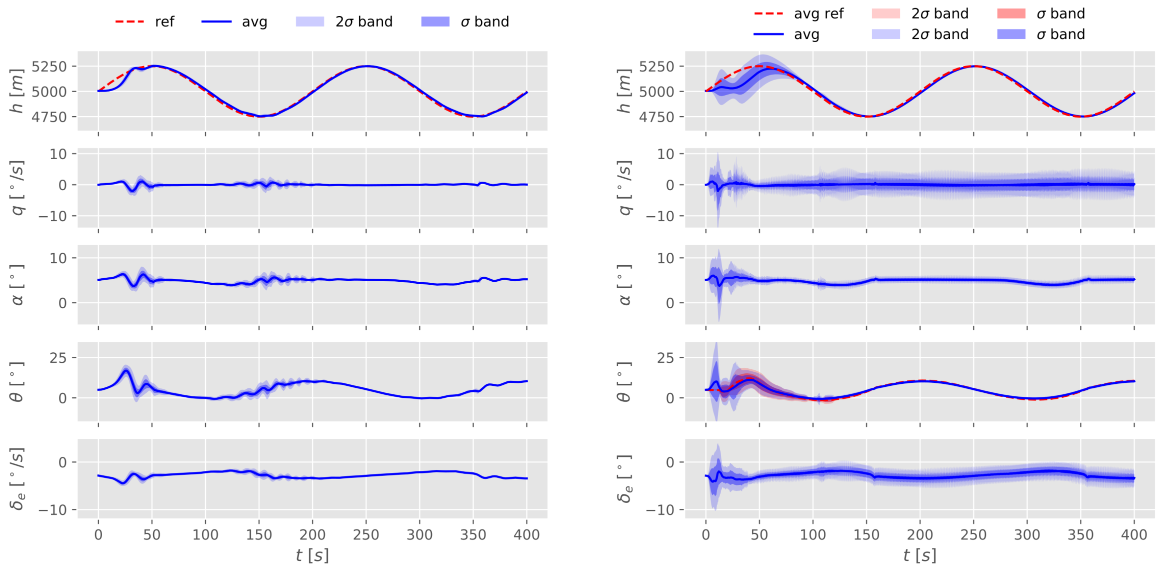
For online application of the IDHP controller, convergence speed to reference is also an important aspect of controller performance to consider. The measure of convergence speed is quantified in the form of rise time and is defined as the elapsed time before simulated Cessna Citation altitude is within 20 meters of the reference altitude.

A. Experiment 1: Comparison of Baseline and Cascaded Controller Performance Under Perfect Conditions

Experiment 1 is set up to compare the baseline and cascaded IDHP agent performance for the altitude tracking task. Tunable parameters have been kept constant among both controller designs. However, due to the fundamentally different actor structure, an additional actor learning rate $\eta_{a,2}$ needs to be defined for the cascaded controller.

The cascaded actor design utilizes an additional state θ for the second actor network. From preliminary analysis of baseline controller tracking task for θ, it has been found that the actor is able to quickly converge to the reference due to faster system dynamic relationship of state θ to δ_e. Additionally, it has been found that a lower learning rate for the inner actor is required to ensure stable learning process as the second actor network governs the the first actor network adaptation. Therefore the learning rates for the cascaded controller are set at: $\eta_{a,1} = 25$, $\eta_{a,2} = 1$, and $\eta_c = 10$. These learning rates have been found through empirical analysis around the reference learning rates obtained from [25]. Learning rates $\eta_a = 25$ and $\eta_c = 10$ have been chosen for the baseline controller in an effort to minimize learning rate difference between the baseline and the cascaded controllers.

Figures 5a and 5b compare the runs from batch simulation with σ and 2σ confidence bands assuming normally distributed states and input over independent runs. The number of runs satisfying converged success condition for each designs is also shown in figure caption.



(a) Baseline controller simulation results, 17/300 runs

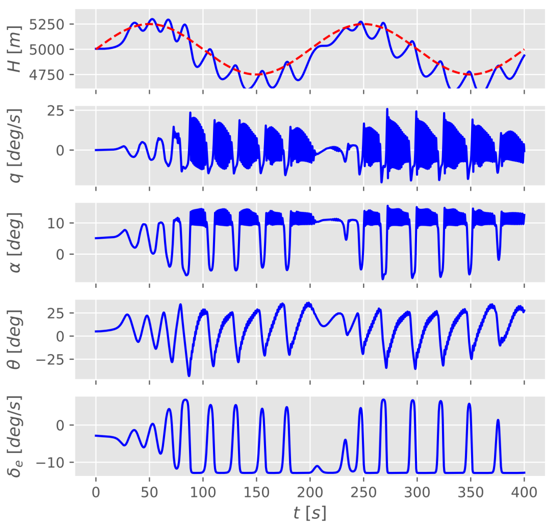
(b) Cascaded controller simulation results, 228/300 runs

Fig. 5 Comparison of converged runs under perfect conditions

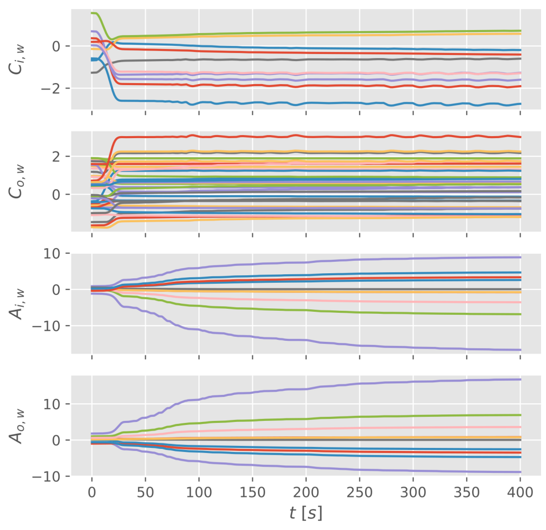
In Figure 5, both controllers show good tracking results with steady phase average RMSE of 10.9 meters for the baseline controller and 12.4 meters for the cascaded controller. One apparent difference between the controllers seen in Figure 5 is that the cascaded controller is slower to converge to the reference. This is due to the existence of a second actor network in the cascaded controller design. Intermediary pitch angle reference parametrization by the first actor and the tracking there of by the second actor requires additional time.

From Figure 5b it can be seen that the θ reference generated by the outer actor is almost instantaneously tracked by the inner actor network despite having a relatively low learning rate of 1. This can be explained by the fact that a faster angular dynamics response exist between the pitch angle and the elevator deflection when compared to altitude governed by slower translational dynamics. Therefore it can be concluded that among the successful runs, the cascaded controller was able to correctly exploit the expert knowledge provided by the cascaded actor design.

The baseline controller outperforms the cascaded controller in terms of rise time and overall accuracy. However, a significantly lower success ratio suggests that the baseline controller is unable to converge to a near-optimal policy. To investigate the cause of such low success ratio, a representative run is chosen from a set of failed runs and its critic and actor weight changes over time are shown in Figure 6.



(a) Simulation results of a representative failed run



(b) Critic and actor weights of a representative failed run

Fig. 6 Representative failed run of a baseline controller under perfect conditions

From Figure 6a it can be seen that although aggressive, the controller is tracking the overall trend of the reference altitude. Due to the aggressive elevator command generated by the actor, the aircraft reaches high α. At this high angle of attack, stall buffeting can be observed in the form of poorly damped oscillation in pitch rate that in turn manifests itself in α and θ as well. Although undesirable, the pitch rate oscillation is not the cause of the unsuccessful tracking as the increase in control policy aggressiveness can be observed before pitch rate oscillation occurs.

From critic weights progression seen in Figure 6b, no significant changes to λ parametrization have been made after initial adaptations between 0 to approximately 25 seconds. First meaningful actor weight adaptations follow subsequently to the critic weight gradient descent. It is evident that based on the large initial λ estimation provided by the critic and the large magnitude of cost during critic weights adaptation when the reference is yet to be tracked, the actor quickly adapts to an aggressive policy. The resultant overshoot is then aggressively corrected by the error-symmetric policy maintained due to the hyperbolic tangent output activation function of the actor. As it is clear that the problem originates from the fast initial policy iteration, it can be alleviated by slowing down the learning process by decreasing both η_a and η_c.

To demonstrate the effect of decreasing learning rates, two additional 300 independent batch simulations of the baseline controller with $\eta_a = 10$, $\eta_c = 5$ and $\eta_a = 5$, $\eta_c = 2$ have been made. The results obtained are summarized in Table 2 along with previously generated results. It can be seen that, as a direct result of lower learning rates, the steady phase RMSE value increase and rise time increase can be seen across all success conditions. However, an overall increase in success ratio by decreasing learning rates can be seen with the exception of the tight success condition for the lowest learning rate settings. This is mainly due to the time frame in which the success condition is defined in. Many runs with the lowest learning rate settings have shown continuous adaptation beyond the first reference period resulting in such result. Considering that the success ratio is the most important parameter determining controller reliability, an overall beneficiary effect can be seen by lowering the learning rates.

Two conclusions can be drawn through comparison of the baseline controller design to the cascaded controller design under perfect conditions. The first conclusion is that the cascaded controller design is able to accurately track the altitude reference signal when compared to the baseline controller. The second conclusion is that a longer convergence time can be expected for the cascaded actor design due to an additional state tracking found within the actor network.

Table 2 Experiment 1 batch simulation results under perfect conditions

		Baseline			**Cascaded**
		η_a = 25 η_c = 10	η_a = 10 η_c = 5	η_a = 5 η_c = 2	η_a = 25, 1 η_c = 10
Tight Success	**Success Ratio**	0.06	0.35	0.25	0.76
	Rise Time s	30.1	36.6	44.0	57.7
	S.P. Avg. RMSE m	10.9	14.9	18.0	12.4
Loose Success	**Success Ratio**	0.06	0.35	0.57	0.76
	Rise Time s	30.1	36.6	46.0	57.7
	S.P. Avg. RMSE m	10.9	14.9	19.9	12.4
Converged	**Success Ratio**	0.06	0.41	0.82	0.76
	Rise Time s	30.1	36.2	44.9	57.7
	S.P. Avg. RMSE m	10.9	26.8	42.3	12.4

B. Experiment 2: Controller Selection

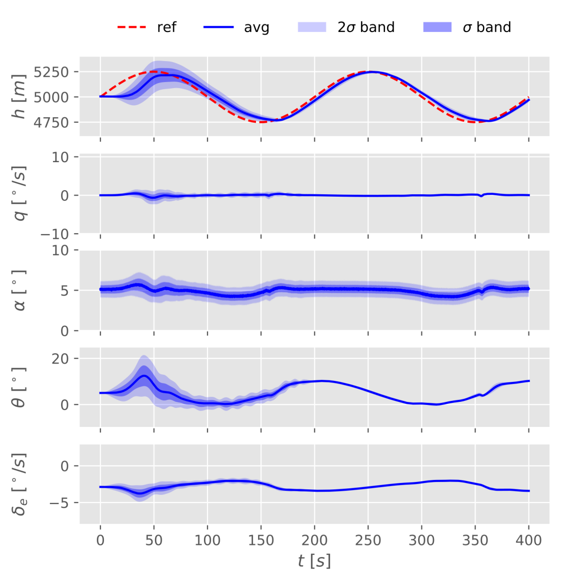
Two main characteristic differences have been observed for the baseline and the cascaded controller designs in experiment 1. As a step towards reducing the gap between simulation and reality, perfect sensor assumption is removed. This experiment has been set up to analyze characteristic differences between the two controller designs under measurement noise.

The baseline controller and the cascaded controller designs are simulated for 300 independent runs with Gaussian measurement noise. Simulation results with confidence bands σ and 2σ are shown for runs satisfying the converged success condition in Figure 7.

For the baseline controller design, comparing Figure 5a from previous experiment to Figure 7a, it can be seen that the addition of noise has a noticeable impact on the steady phase RMSE and rise time for either controller designs. Comparing Figure 5b to 7b for the cascaded controller design, steady phase RMSE is relatively unaffected while the rise time increases with the addition of measurement noise. Therefore it can be concluded that measurement noise negatively affects the overall learning speed of either controller designs but the cascaded controller design shows relative advantage in its ability to approximate a near optimal control policy.

Table 3 is provided to give an overview of the effect of measurement noise to both controller designs. An interesting result is the increase in success ratio seen for the baseline controller. An explanation for this phenomenon can be given by understanding the addition of measurement noise as a cause of slower learning by directly introducing uncertainty to critic, actor, and incremental model input and adaptation terms. From experiment 1, the simulation results of baseline controller with varying learning rates have shown that decreasing learning rates can increase success ratio at the cost of rise time and steady phase average RMSE. Similar effects are observed for the baseline controller by introducing measurement noise with the difference of greater success ratio increase. This greater success ratio increase can be explained by the fact that decreasing learning rate only affects the critic and actor update path while measurement noise affects the critic and actor input as well. The critic is discouraged from quickly converging to λ estimations that may lead to aggressive policy generation due to an overall slower learning caused by state uncertainty. Therefore it can be summarized that including measurement noise at given magnitudes to the simulation had an overall beneficial impact on the baseline controller by decreasing the overall learning process prohibiting aggressive policy generation.

While the introduction of measurement noise has a beneficial impact on the baseline controller success ratio at given learning rates, an overall decrease in success ratio for the cascaded controller design is observed. This is due


(a) **Baseline controller simulation results 279/300 runs**

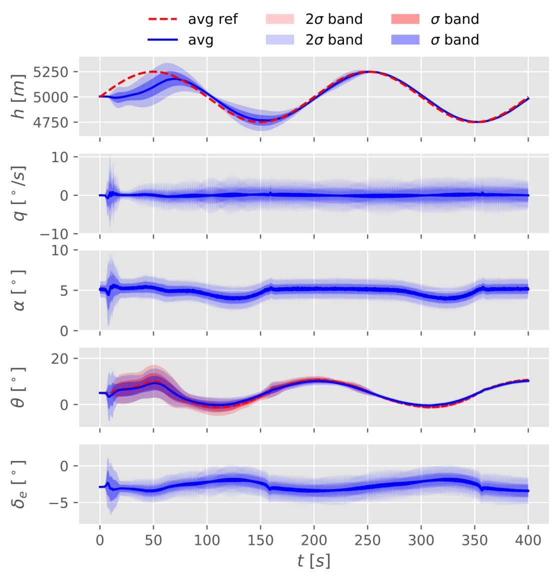

(b) **Cascaded controller simulation results 153/300 runs**

Fig. 7 Comparison of converged runs under measurement noise

to the relatively complex structure of the actor network seen in cascaded controller design. Not only is an additional state information provided to the actor network with added uncertainty, the overall control policy relies on both actor networks successfully generalizing dynamics of given states and control command. Additionally, aggressive policy generation, which can be alleviated by slower learning, is not an existing failure mode for the cascaded controller design. Therefore the addition of measurement noise negatively impacts the cascaded controller design to a greater degree when compared to the baseline controller design.

Table 3 Experiment 2 batch simulation results under measurement noise

		Baseline		**Cascaded**	
		$\eta_a = 25, \eta_c = 10$		$\eta_a = 25, 1, \eta_c = 10$	
		No Noise	**With Noise**	**No Noise**	**With Noise**
Tight Success	**Success Ratio**	0.06	0.00	0.76	0.51
	Rise Time s	30.1	42.3	57.7	69.4
	S.P. Avg. RMSE m	10.9	20.0	12.4	13.1
Loose Success	**Success Ratio**	0.06	0.81	0.76	0.52
	Rise Time s	30.1	56.8	57.7	69.7
	S.P. Avg. RMSE m	10.9	28.5	12.4	13.1
Converged	**Success Ratio**	0.06	0.93	0.76	0.53
	Rise Time s	30.1	60.5	57.7	70.7
	S.P. Avg. RMSE m	10.9	31.7	12.4	14.6

The results of this experiment show that both the baseline controller design and the cascaded controller design are feasible candidates to achieve near optimal control policy for altitude tracking task with varying success ratios. Through experiment 1 and experiment 2, it has been established that the baseline controller is able to achieve faster learning speed while the cascaded controller shows advantage in overall accuracy in the presence of measurement noise. Although the cascaded controller design may achieve higher success ratios through learning rate tuning, this is not pursued given the high success ratio already achieved by the baseline controller. Additionally, increased susceptibility to measurement noise and unique failure modes of incorrect θ reference generation adds to the disadvantage of the cascaded controller design. For the altitude control task, the baseline controller is a favorable controller design and will thus be chosen for further analysis under atmospheric gusts in experiment 3.

C. Experiment 3: IDHP Controller Performance Under Gust

In addition to measurement noise, atmospheric gusts are also expected during aircraft operation. To demonstrate the baseline controller performance under atmospheric disturbances, batch simulations of 300 independent runs at 4 different flight conditions have been performed under "light" and "moderate" gust scenarios.

The outline of the 4 flight conditions is presented in Table 4. The flight conditions have been chosen in the order of increasing dynamic pressure where aerodynamic damping effects and elevator effectiveness are expected to increase.

Table 4 Description of Flight Conditions used in Experiment 3

Flight Condition	Initial Altitude $[m]$	Initial Airspeed $[m/s]$
FC0	5000	90
FC1	2000	90
FC2	5000	140
FC3	2000	140

Figures 8 and 9 show the average timescale plots of all runs satisfying the success condition of $RMSE < 40m$ for the last 30 seconds of the simulation. The shaded regions represent the σ and 2σ bands among independent runs.

In both Figures 8 and 9, an overall decrease in success ratio can be observed when increasing gust intensities. This is a direct consequence of increased disturbance affecting the learning stability of the designed controller.

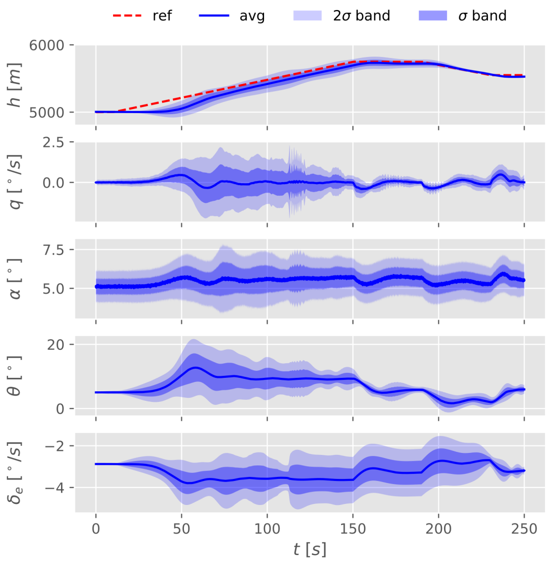
Another point of consideration is that the success ratio for FC2 and FC3 are typically lower than FC0 and FC1. This can be attributed to higher aircraft V_{TAS} for FC2 and FC3. Dynamic pressure is proportional to the square of airspeed. As elevator control effectiveness is proportional to dynamic pressure, elevator control effectiveness is in turn proportional to the square of airspeed. It has been previously established that aggressive control policy is the main failure mode for the baseline controller design. Increased control effectiveness amplifies such problem leading to the overall lower success ratio.

It is demonstrated that a relatively simple altitude error based IDHP design is capable of near optimal control policy generalization in the presence of measurement noise and atmospheric gusts for altitude tracking task. Considering that no prior online training was performed, further success ratio increase can be expected through online training phase design accompanied with learning rate tuning.

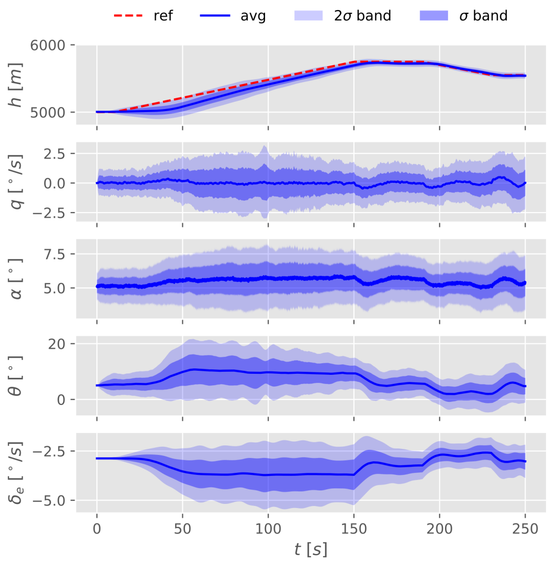
V. Conclusion and Recommendations

Two IDHP based controllers have been designed for the task of altitude reference tracking. The results have shown that the cascaded controller can characteristically generalize a more optimal control policy at the cost of slower learning speeds. Simulation results from experiments 1 and 2 showed the effects of adding measurement noise. The negative effects can be summarized by decreased cost-to-go function estimation performance and slower learning process due to state uncertainty. Such negative effects have impacted the cascaded controller design to a greater extent due to additional state information utilized by the cascaded controller. A beneficial effect of measurement noise on the learning process has also been observed. At certain learning rates, the baseline controller design saw an increase in the overall learning stability as state uncertainty repressed aggressive policy generation. Finally, experiment 3 has shown that the baseline controller is able to learn a near optimal altitude control policy in the presence of atmospheric disturbances and measurement noise. Overall, the findings within this research indicate that a relatively simple IDHP controller provided with only altitude tracking error to the actor and critic network can be sufficient for Cessna Citation aircraft altitude tracking task.

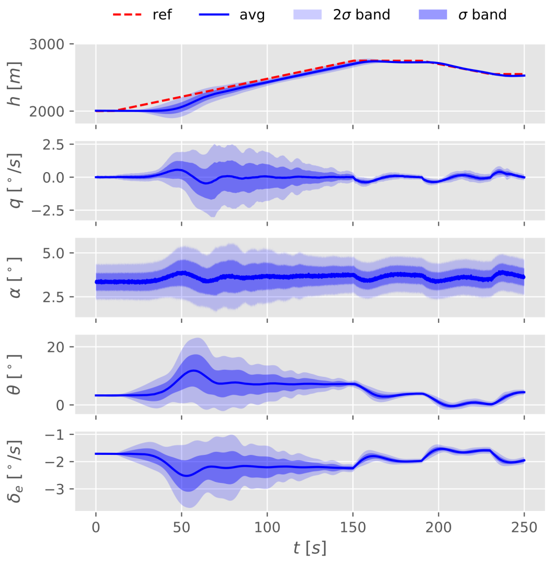
To increase the success ratio of the presented IDHP controller design, further research is needed in terms of in-depth measurement noise sensitivity analysis, critic estimation performance increase, and policy shaping. The first point relates to the fixed measurement noise parameters used for the simulation. Although the causality of success ratio increase has been identified, the exact relationship between success ratio and measurement noise amplitude remains



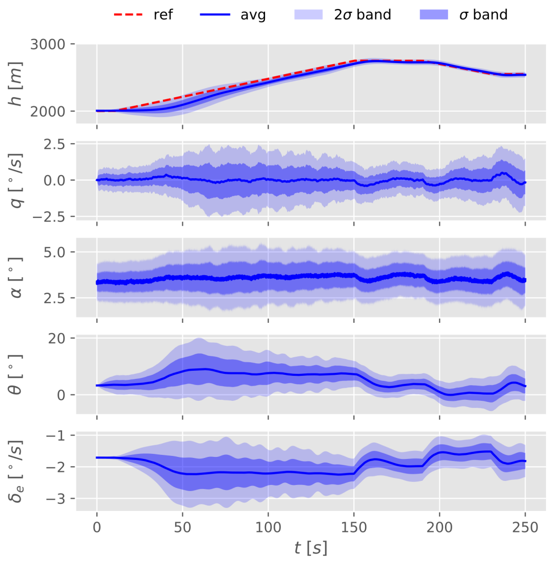
(a) FC0 light gust 283/300 runs



(b) FC0 moderate gust 248/300 runs

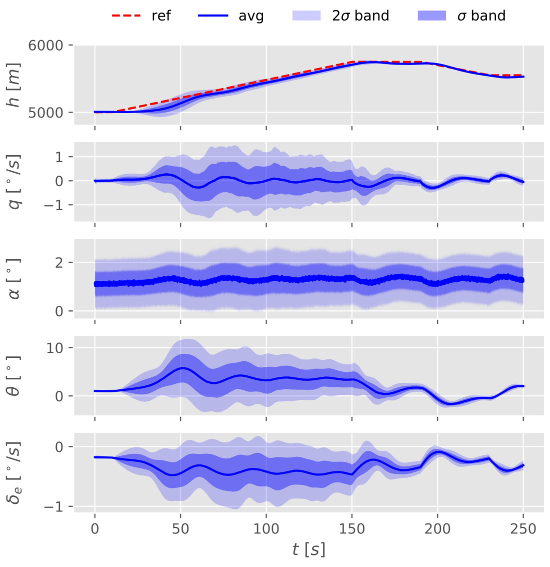


(c) FC1 light gust 270/300 runs

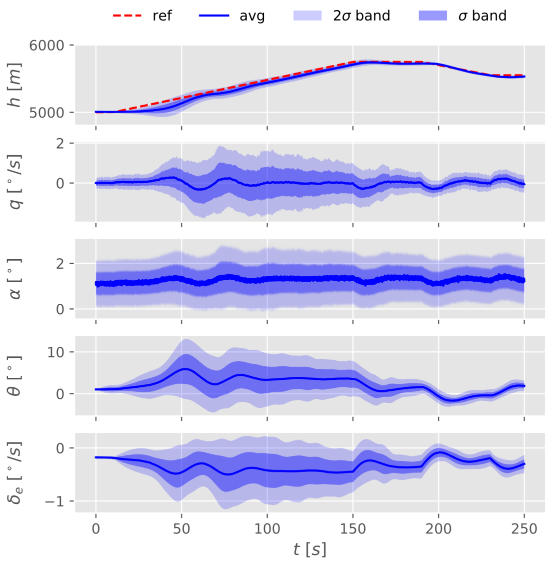


(d) FC1 moderate gust 257/300 runs

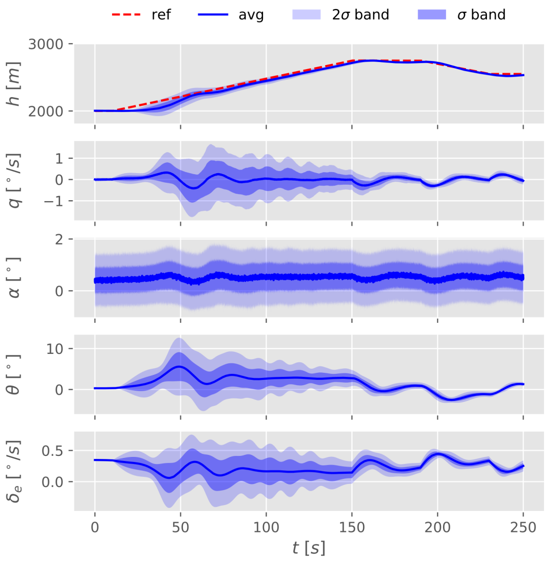
Fig. 8 Simulation results obtained with measurement noise and varying gust conditions for FC0 and FC1



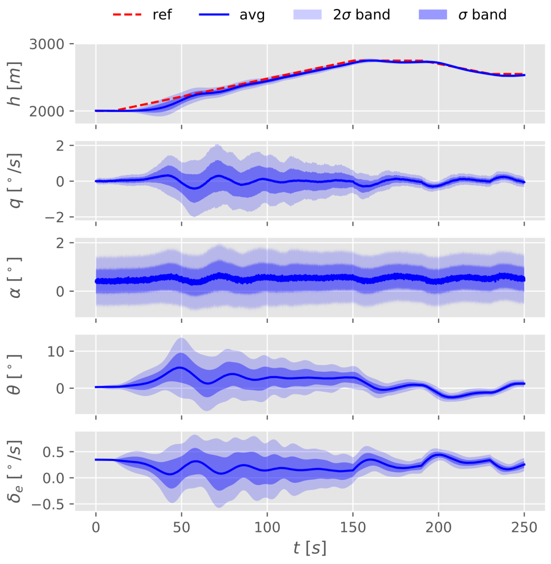
(a) FC2 light gust 226/300 runs



(b) FC2 moderate gust 206/300 runs



(c) FC3 light gust 222/300 runs



(d) FC3 moderate gust 202/300 runs

Fig. 9 Simulation results obtained with measurement noise and varying gust conditions for FC2 and FC3

unknown. Noise sensitivity analysis can help determine to which degree the measurement noise is beneficial to the learning process. The second point of further research is derived from the fact that some failed runs can be attributed to poor critic performance. To help stabilize critic estimation performance, a target critic previously utilized in [26, 34] can be considered. Although it is true that the overall convergence success ratio is largely dependent on critic estimation performance, the overall shape of the policy maintained by the actor ultimately leads to aggressive control policy from which destabilization occurs. This effect is more pronounced when a relatively simple policy is maintained. To alleviate aggressive policy generation, choosing a different activation function with a smoother gradient for the output layer can be considered. For future research, a sigmoidal actor output activation function offset by the trim point is suggested.

References

[1] Teuliere, C., Eck, L., and Marchand, E., "Chasing a moving target from a flying UAV," *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2011, pp. 4929–4934. https://doi.org/10.1109/IROS.2011.6048050.

[2] Enns, D., Bugajski, D., Hendrick, R., and Stein, G., "Dynamic inversion: an evolving methodology for flight control design," *International Journal of Control*, Vol. 59, No. 1, 1994, pp. 71–91. https://doi.org/10.1080/00207179408923070.

[3] Durham, W., Bordignon, K. A., and Beck, R., *Aircraft Control Allocation*, John Wiley & Sons, Ltd, Chichester, UK, 2016. https://doi.org/10.1002/9781118827789.

[4] Brinker, J., and Wise, K., "Nonlinear simulation analysis of a tailless advanced fighter aircraft reconfigurable flight control law," *Guidance, Navigation, and Control Conference and Exhibit*, American Institute of Aeronautics and Astronautics, Reston, Virigina, 1999. https://doi.org/10.2514/6.1999-4040.

[5] Doman, D. B., and Ngo, A. D., "Dynamic Inversion-Based Adaptive/Reconfigurable Control of the X-33 on Ascent," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 2, 2002, pp. 275–284. https://doi.org/10.2514/2.4879.

[6] Bacon, B., and Ostroff, A., "Reconfigurable flight control using nonlinear dynamic inversion with a special accelerometer implementation," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, American Institute of Aeronautics and Astronautics, Reston, Virigina, 2000. https://doi.org/10.2514/6.2000-4565.

[7] Sieberling, S., Chu, Q. P., and Mulder, J. A., "Robust Flight Control Using Incremental Nonlinear Dynamic Inversion and Angular Acceleration Prediction," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 6, 2010, pp. 1732–1742. https://doi.org/10.2514/1.49978.

[8] Simplício, P., Pavel, M. D., van Kampen, E., and Chu, Q. P., "An acceleration measurements-based approach for helicopter nonlinear flight control using incremental nonlinear dynamic inversion," *Control Engineering Practice*, Vol. 21, No. 8, 2013, pp. 1065–1077. https://doi.org/10.1016/j.conengprac.2013.03.009.

[9] Smeur, E. J. J., Chu, Q. P., and de Croon, G. C. H. E., "Adaptive Incremental Nonlinear Dynamic Inversion for Attitude Control of Micro Air Vehicles," *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 3, 2016, pp. 450–461. https://doi.org/10.2514/1.G001490.

[10] Grondman, F., Looye, G., Kuchar, R. O., Chu, Q. P., and Van Kampen, E. J., "Design and Flight Testing of Incremental Nonlinear Dynamic Inversion-based Control Laws for a Passenger Aircraft," *2018 AIAA Guidance, Navigation, and Control Conference*, American Institute of Aeronautics and Astronautics, Reston, Virginia, 2018. https://doi.org/10.2514/6.2018-0385.

[11] Acquatella, P., van Kampen, E. J., and Chu, Q. P., "Incremental backstepping for robust nonlinear flight control," *Proceedings of the EuroGNC 2013, 2nd CEAS Specialist Conference on Guidance, Navigation and Control*, 2013, pp. 1444–1463.

[12] Lewis, F. L., and Vrabie, D., "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circuits and Systems Magazine*, Vol. 9, No. 3, 2009, pp. 32–50. https://doi.org/10.1109/MCAS.2009.933854.

[13] Powell, W. B., *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, 2$^{\text{nd}}$ ed., John Wiley & Sons, Inc., Princeton, 2011.

[14] Werbos, P. J., "Advanced forecasting methods for global crisis warning and models of intelligence," *General Systems Yearbook*, Vol. 22, 1977, pp. 25–38.

[15] Prokhorov, D. V., Santiago, R. A., and Wunsch, D. C., "Adaptive critic designs: A case study for neurocontrol," *Neural Networks*, Vol. 8, No. 9, 1995, pp. 1367–1372. https://doi.org/10.1016/0893-6080(95)00042-9.

[16] Si, J., and Wang, Y. T., "On-line learning control by association and reinforcement," *IEEE Transactions on Neural Networks*, Vol. 12, No. 2, 2001, pp. 264–276. https://doi.org/10.1109/72.914523.

[17] Grondman, I., Busoniu, L., Lopes, G. A. D., and Babuska, R., "A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, Vol. 42, No. 6, 2012, pp. 1291–1307. https://doi.org/10.1109/TSMCC.2012.2218595.

[18] Balakrishnan, S. N., and Biega, V., "Adaptive-critic-based neural networks for aircraft optimal control," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 4, 1996, pp. 893–898. https://doi.org/10.2514/3.21715.

[19] Han, D., and Balakrishnan, S. N., "State-constrained agile missile control with adaptive-critic-based neural networks," *IEEE Transactions on Control Systems Technology*, Vol. 10, No. 4, 2002, pp. 481–489. https://doi.org/10.1109/TCST.2002.1014669.

[20] Lin, C. K., "Adaptive critic autopilot design of bank-to-turn missiles using fuzzy basis function networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 35, No. 2, 2005, pp. 197–207. https://doi.org/10.1109/TSMCB.2004.842246.

[21] Ferrari, S., and Stengel, R. F., "An adaptive critic global controller," *Proceedings of the American Control Conference*, Vol. 4, 2002, pp. 2665–2670. https://doi.org/10.1109/ACC.2002.1025189.

[22] Ferrari, S., and Stengel, R. F., "Online Adaptive Critic Flight Control," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 5, 2004, pp. 777–786. https://doi.org/10.2514/1.12597.

[23] van Kampen, E. J., Chu, Q. P., and Mulder, J. A., "Continuous Adaptive Critic Flight Control Aided with Approximated Plant Dynamics," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, American Institute of Aeronautics and Astronautics, Reston, Virigina, 2006. https://doi.org/10.2514/6.2006-6429.

[24] Zhou, Y., Van Kampen, E. J., and Chu, Q. P., "Incremental Model Based Heuristic Dynamic Programming for Nonlinear Adaptive Flight Control," *Proceedings of the International Micro Air Vehicles Conference and Competition 2016*, Beijing, China, 2016.

[25] Zhou, Y., van Kampen, E. J., and Chu, Q. P., "Incremental model based online dual heuristic programming for nonlinear adaptive control," *Control Engineering Practice*, Vol. 73, 2018, pp. 13–25. https://doi.org/10.1016/j.conengprac.2017.12.011.

[26] Heyer, S., "Reinforcement Learning for Flight Control," Master thesis, Delft University of Technology, 2019.

[27] Enns, R., and Si, J., "Helicopter trimming and tracking control using direct neural dynamic programming," *IEEE transactions on neural networks*, Vol. 14, No. 4, 2003, pp. 929–39. https://doi.org/10.1109/TNN.2003.813839.

[28] Sutton, R. S., and Barto, A. G., *Reinforcement Learning: An Introduction*, 2nd ed., MIT Press, Cambridge, Massachusetts, 2018.

[29] van 't Veld, R., Van Kampen, E. J., and Chu, Q. P., "Stability and Robustness Analysis and Improvements for Incremental Nonlinear Dynamic Inversion Control," *2018 AIAA Guidance, Navigation, and Control Conference*, American Institute of Aeronautics and Astronautics, Reston, Virginia, 2018. https://doi.org/10.2514/6.2018-1127.

[30] *MIL-HDBK- 1797: Flying Qulities of Piloted Aircraft*, U.S. Department of Defense, 1997.

[31] King, M., *Process Control*, John Wiley & Sons, Ltd, Chichester, UK, 2016. https://doi.org/10.1002/9781119157779, URL http://doi.wiley.com/10.1002/9781119157779.

[32] Stevens, B. L., Lewis, F. L., and Johnson, E. N., *Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems*, 3rd ed., John Wiley & Sons, Inc, Hoboken, NJ, USA, 2015. https://doi.org/10.1002/9781119174882.

[33] Ni, Z., He, H., and Wen, J., "Adaptive learning in tracking control based on the dual critic network design," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 24, No. 6, 2013, pp. 913–928. https://doi.org/10.1109/TNNLS.2013.2247627.

[34] Kroezen, D., "Online Reinforcement Learning for Flight Control," Master thesis, Delft University of Technology, 2019.

Part II

Preliminary Research

Preliminary Research Content Overview

Preliminary research consists of two chapters. Chapter II Literature Survey establishes the foundation on aircraft control and RL. Throughout the literature survey, challenges in implementing a reinforcement learning controller for the aircraft control task are identified and analyzed. The most suitable RL controller framework is identified and the chapter is concluded with a state-of-the-art review of such controller frameworks previously implemented for the aircraft control task.

Chapter 3 Preliminary Analysis presents the implementation of the most suitable RL controller framework for the reduced Cessna Citation short period model. Simulation on the reduced model is conducted as a feasibility check and as means to identify the points of improvement for the controller framework. The chapter contains a description of the model and the RL controller design. This is followed by a presentation of the simulated results and discussion concluding the chapter.

Chapter 2

Literature Survey

2-1 Aircraft Control

Aircraft control can typically be divided among three domains according to the body reference frame fixed at the aircraft center of gravity as depicted in Figure 2-1. Longitudinal control is controlled with a pitching moment M about the y-axis. Pitch acceleration $\dot{q}$ is induced by such moment in a nose-up positive direction. Similarly, lateral control and directional control are controlled with moment about the x and z-axis respectively. The moment for lateral control L creates the roll acceleration $\dot{p}$ and the moment for directional control N generates yaw acceleration $\dot{r}$. These three domains of control are mainly achieved through aerodynamic control effectors where the camber or the incidence of a larger aerodynamic surface are changed (Durham et al., 2016).

An additional fourth domain is considered which is the aircraft thrust (propulsion) control. As the name suggest, propulsive control effector is utilized to achieve control. Depending on the aircraft configuration it can be used to create linear thrust T along the x-axis of the aircraft, create directional moment N by means of asymmetric thrust in a multi-engine configuration, or create pitching moment M if thrust vectoring is available. In the case of Cessna 550 Citation II, the aircraft under consideration, thrust vectoring is unavailable and thrust control is primarily used for speed control by varying thrust T only.

Various maneuvers are done with the combination of control in the aforementioned four domains of aircraft control. For example, a coordinated turn requires lateral control to instigate and maintain the turn, directional control to maintain zero side slip, longitudinal control to maintain altitude, and finally thrust control to maintain velocity. The example given above only mentions the primary purpose of each control domain during such maneuver. In reality, there are many more correlated variables to be controlled as aerodynamic coupling exists between forces and disturbances among each axis of control. External disturbances also have to be taken into account such as wind or sudden gusts. The goal of aircraft control can be summarized to the control of complex dynamics of the aircraft system in the presence of stochastic external disturbances while retaining stringent comfort and safety regulations.

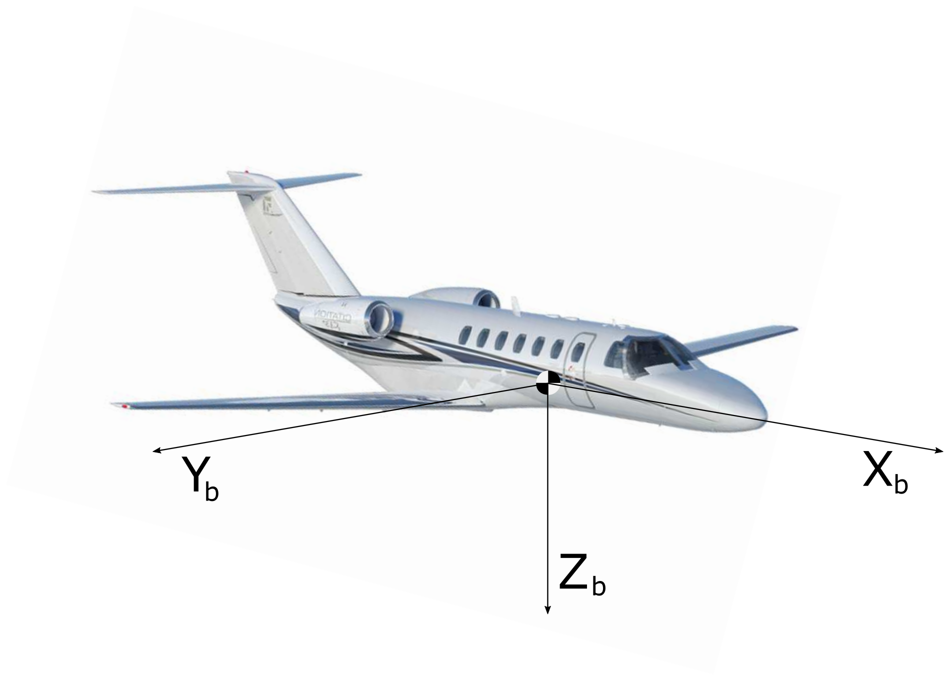


Figure 2-1: Aircraft body reference frame shown with Cessna Citation

2-1-1 Challenges and current limitations in Automatic flight control

Many modern aircraft typically employ multiple control effectors, at a number larger than the control inceptors present in the cockpit. Modern aircraft control can be marked by the use of fly-by-wire system where optical cables are used to transmit electrical signals to actuators instead of direct mechanical linkages. In the past, a complicated mechanical system that interconnects the control effectors(ganging) were employed (Stevens, Lewis, & Johnson, 2015). With the use of fly-by-wire system, the control law shifted from such control effector ganging to Automatic Flight Control (AFC) where the pilot's input is translated directly to specific responses from control effectors by a computer.

The advantages of fly-by-wire system are clear. The control system itself is much lighter in weight. And with the Automatic Flight Control System (AFCS), the reliance on the aircraft's bare response characteristics has been alleviated (Stevens et al., 2015). However, heavier dependence on the automatic flight control system means heavier dependency on the aircraft model unless model free control method is utilized. Many AFCS on-board aircraft today rely on gain scheduling method where the system is identified and linearized at multiple points within the flight envelope and combined to form a controller. This directly translates to time and economical burden during aircraft development. Additionally, if an unanticipated flight parameter alteration or environment change that exceeds anticipation occurs during flight(such as aircraft damage or sudden gust), the control effectiveness may degrade rapidly as the control laws previously defined are not applicable anymore. This can be seen as a fundamental limitation of the method where control laws are derived directly from a set flight envelope.

Current state-of-the-art AFCS in operation for manned aircraft is dynamic inversion (Durham et al., 2016). Consider Equation 2-1 describing the aircraft dynamics with nonlinear function $f(x(t))$ and linear control effectiveness $bu(t)$. Equation 2-2 represents desired dynamic response $\dot{x}_{des}$ and required control input $u^*(t)$ with approximation of model $\hat{f}(\cdot) \approx f(\cdot)$, $\hat{b} \approx b$, and sensor measurements $\hat{x}(t) \approx x(t)$ which can be rearranged into a control law shown in Equation 2-3. Substituting 2-3 into 2-1 yields Equation 2-4. Assuming perfect approximation and measurements, Equation 2-5 is derived from Equation 2-3 and Equation 2-6 is derived from 2-4. In essence, with a good model and accurate measurements, the control input required for desired dynamic response can be obtained by Equation 2-5.

$$\dot{x} = f(x(t)) + bu(t) \tag{2-1}$$

$$\dot{x_{des}} = \hat{f}(\hat{x}(t)) + \hat{b}u^*(t) \tag{2-2}$$

$$u^*(t) = \frac{1}{\hat{b}}(\dot{x}_{des} - \hat{f}(\hat{x}(t))) \tag{2-3}$$

$$\dot{x} = f(x(t)) + bu^*(t) = f(x(t)) + \frac{b}{\hat{b}}(\dot{x}_{des} - \hat{f}(\hat{x}(t))) \tag{2-4}$$

$$u^*(t) = \frac{1}{b}(\dot{x}_{des} - f(x(t))) \tag{2-5}$$

$$\dot{x} = \dot{x}_{des} \tag{2-6}$$

Dynamic inversion tackles the problem of nonlinear systems and effectively circumvents the need for gain scheduling. However, as it can be immediately noticed from the above equations, dynamic inversion is reliant on an accurate model and accurate sensor measurements. In case of model mismatch, dynamic inversion suffers from performance degradation. Therefore it can be said that the state-of-the-art control method in operation still does not address the issue of off line identified model dependency. In its basic form without an online controller retuning procedure, dynamic inversion is not an adaptive controller.

Main points of consideration for the improvement of existing controllers are: reducing or eliminating the dependency of aircraft control system on an off line identified model, and enabling the aircraft control system to adapt to changing environment or system. Numerous research has been done especially in the field of adaptive control to address the aforementioned points. Adaptive Dynamic Inversion (ADI) method has been developed primarily with the use of neural networks for online identification of plant inversion error or the model to derive control laws from. Notable example of the first approach where the plant inversion error is identified is the simulation of ADI on X-36 aircraft in various damage conditions (Brinker & Wise, 1999), which was later successfully demonstrated. An approach where the controller effectiveness matrix was identified online was shown by (Doman & Ngo, 2002) for the X-33 aircraft in ascent flight phase simulation. A more recent control approach is shown in (Sieberling, Chu, & Mulder, 2010) and (Simplício, Pavel, van Kampen, & Chu, 2013) where an Incremental Nonlinear Dynamic Inversion (INDI), conceived by (Bacon & Ostroff, 2000), was simulated for a T-tailed UAV and a helicopter respectively. INDI was later tested on a micro aerial vehicle in (Smeur, Chu, & de Croon, 2016). By using angular acceleration feedback warranted by a reformulation of rotational dynamic equations into an incremental form, INDI scheme removed the need for a baseline model and therefore reduced the model dependency. Another branch of adaptive controller design is adaptive backstepping where backstepping technique and function approximation for online adaptation are used in conjunction. A popular adaptive backstepping approach, command filtered adaptive backstepping, applied on a flying wing UAV simulation is shown in (Farrell, Sharma, & Polycarpou, 2005). Simulation results of a highly coupled maneuver with constrained states showed good control performance while retaining robustness. To reduce model dependency, incremental backstepping has been developed in (Acquatella, van Kampen, & Chu, 2013) where incremental control action has been used in combination with the backstepping approach. Simulation results on a longitudinal control task of a nonlinear model have shown comparable results.

The common trait shared by different adaptive control approaches is their (varying degrees of) model dependency. An alternative approach is to define the control problem within the RL framework. In essence, RL is "learning from interaction with the environment". As the control law may be derived through interaction with the environment, a favorable achievement of reducing or even eliminating model dependency and increasing adaptability can be expected. Therefore in the following section, the use of RL controllers is explored for the development of an online adaptive aircraft control system.

2-2 Reinforcement Learning Foundation

Reinforcement learning is a branch of machine learning where an agent learns from its surrounding environment through reward signals given by carrying out actions to interact with the environment. By carrying out different actions, an agent's goal is to find a way to act in the environment that would result in highest cumulative reward. Reinforcement learning differs from other approaches to machine learning, supervised learning and unsupervised learning, in that it neither requires a training set of labeled examples nor does it try to find structure hidden in collections of unlabeled data (Sutton & Barto, 2018).

In its most basic form, reinforcement learning framework consists of the following elements: environment, agent, state, action, and reward as shown in Figure 2-2. An environment is where the agent is situated in and interacts with to receive next state and reward based on its action. This scalar reward is generated by a reward function responsible for assigning an agent's action with a single numerical reward which serves as a direct and immediate benchmark of an agent's action. An agent is an active decision making element which determines which actions to take according to its policy. An agent's policy determines its actions based on the current state of the agent, past experience, and some element of exploration. An agent's policy is shaped by a value function. Apart from the reward that enables immediate evaluation of action that results in a certain state, we introduce a concept of value which evaluates the action in the long run. The value of a state includes possible future states and their rewards that ensue after an action is taken to bring an agent to a certain state. Thus it can be seen as a cumulative expected reward of a state. Value function makes up an essential building block within the reinforcement learning framework and its structure directly impacts the performance of reinforcement learning frameworks.

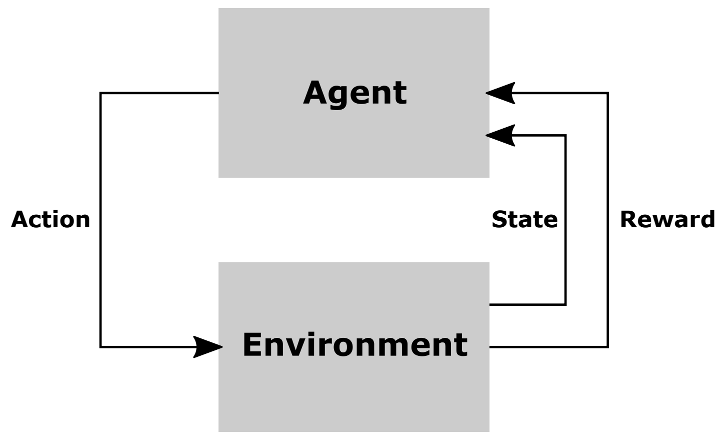


Figure 2-2: Reinforcement learning framework

An agent aims to maximize the cumulative reward it receives by choosing actions of greatest value. Therefore, an actor's policy π incorporates exploitation of known actions that generate maximum cumulative reward while trying different actions that may result in even greater cumulative reward. This idea of a tradeoff between exploitation and exploration is an important concept to reinforcement learning. If an agent purely exploits actions that result in greatest value based on previous observations, it may converge on a suboptimal policy whereas if an agent purely explores the states without exploiting previously established value of states, it may fail with a high probability. Therefore the algorithm must find balance between exploitation and exploration in order for the agent to find an optimal policy.

2-2-1 Optimality

An agent's goal is to achieve an optimum policy. A question then arises as to how optimality should be defined. In the field of reinforcement learning, optimality is directly linked to maximizing the expected reward V^π. Thus optimality can be defined by a mathematical model to calculate the cumulative expected reward from rewards received at time step r_t. Three optimality models are discussed in (Kaelbling, Littman, & Moore, 1996): finite-horizon model, infinite-horizon discounted model, and average-reward model.

Finite-horizon model of optimality as shown in Equation 2-7 models optimality by the sum of expected reward from current time step to the next h time steps. From the structure of the model, it is clear that the length of time steps h is used to adjust the forecast horizon of the model. Therefore, this model is well suited to problems where the remaining time steps are known. Otherwise the choice of h may have an adverse effect to the optimal policy choice with delayed rewards as the horizon length may not capture important future rewards.

$$V^\pi(s) = E\Big(\sum_{t=0}^{h} r_t \mid s_t = s\Big) \tag{2-7}$$

Infinite-horizon discounted model introduces the concept of discount factor γ by which future rewards are discounted while the considered timeline is now infinite. This model of optimality is defined as shown in Equation 2-8. Compared to the finite-horizon model, the infinite-horizon model is able to account for a longer timeline albeit the future rewards are geometrically discounted. In (Kaelbling et al., 1996), the author states that the discount factor bounds the infinite sum and makes the inifinite-horizon discounted model more mathematically tractable than the finite-horizon model leading to its popularity. However, if the chosen discount factor is too low, only rewards from near future states will have meaningful contribution. This would make the chosen optimal policy to be myopic and greedy and lead to poor performance. In (Kober, Bagnell, & Peters, 2013), the author states that this unstable optimal control law is evident even for linear quadratic regulation problems and is the reason why the discounted models are frequently inadmissible in robot control.

$$V^{\pi}(s) = E\Big(\sum_{t=0}^{\infty} \gamma^t r_t \mid s_t = s\Big) \tag{2-8}$$

Average-reward model takes the average of all subsequent future reward. Equation 2-9 defines the model. The model has neither a fixed time horizon nor a discount factor that needs to be hand tuned. However, a distinct disadvantage to this model is that the initial rewards gained in near future are masked by the long term average rewards. Therefore a policy that would generate greater initial reward cannot be distinguished from a policy that wouldn't (Kaelbling et al., 1996).

$$V^{\pi}(s) = \lim_{h\to\infty} E\Big(\frac{1}{h}\sum_{t=0}^{h} r_t \mid s_t = s\Big) \tag{2-9}$$

With regards to the optimal aircraft control problem, it is clear that the finite-horizon model of optimality have significant disadvantages. Apart from the fact that there is no obvious terminal state with high reward from which the model forecast horizon can be derived from, using the finite-horizon model may lead to poor controller adaptivity or even instability. The model introduces unnecessary additional ambiguity of having to manually set the forecast horizon. This would mean that different maneuvers require different forecast horizon depending on the time scale of its dynamics, and in the case of a change in environment where the dynamics are slowed, the agent may not be able to guarantee control at all.

Infinite-horizon discounted model and the average-reward model are both infinite-horizon models and are more well suited for the optimal aircraft control problem. As mentioned before both models have their drawbacks. For the infinite-horizon discounted model, there is an additional hand tuned parameter, discount factor γ, that could result in poor performance if set too low. For the average-reward model, the model is unable to distinguish initial large rewards as they get masked by the long term reward. In (Kober et al., 2013), the author favors the average-reward model for robot control as long term rewards are more valued than initial large rewards. However when considering aircraft optimal control, where reward signals are often dependent on tracking error at each time step, such myopic behavior from low discount factor may even be favored even at the cost of long term sub-optimality.

2-2-2 Markov Decision Process

From the optimality model definitions it is clear that the future rewards need to be taken into account to achieve optimality. An important concept to reinforcement learning problems is that the agent's action not only impacts the immediate reward it receives, but also the next state which in turn would impact the subsequent rewards. Another important point of consideration is that in some reinforcement learning problems, choosing a sequence of actions with insignificant immediate rewards may lead to a large sum of rewards in the distant future and thus achieving optimality. To achieve optimality, often a tradeoff of immediate and delayed rewards needs to be made while considering the dependence of future rewards on current action decision. Markov Decision Process (MDP) are a formal framework on which this class of problems can be defined in.

MDPs are a mathematically idealized form of the reinforcement learning problem for which precise theoretical statements can be made (Sutton & Barto, 2018). MDPs consider systems that have Markovian property, which can be summarized by the system's dependency to the past only through the current state and action selected. Following the definition given by (Puterman, 1994), MDPs can be defined by a collection of elements: decision epochs T, set of states S, set of allowable actions A_s, reward function $R_t(s,a)$, and transition probability function $P_t(\cdot \mid s,a)$.

$$\{T, S, A_s, P_t(\cdot \mid s,a), R_t(s,a)\}$$

Decision epochs T, where its elements are denoted by t, are points of time when decisions are made by an agent that can be classified in four ways: finite or infinite and continuous or discrete. In a continuous case, decisions can be made at either: all decision epochs, event based random points of time, or at predetermined points of time. In discrete time case, time is divided in periods where a decision epoch marks the beginning of a period. Decision epoch in continuous case can be represented by $T = [0, N]$ and in discrete time case $T = 1, 2, ..., N$ where $N \leq \infty$. In the finite case N would be a finite integer.

At each decision epoch t an agent receives some representation of the environment. This representation is called a state s from a set of possible environment states denoted by S. At each state s, an agent can select its action a from a set of allowable actions in that state denoted by A_s. Both sets can be either continuous or discrete and finite or infinite.

An agent receives a reward $R_t(s,a)$ and the resultant transition probability function $P_t(\cdot \mid s,a)$ from selecting action $a \in A_s$ in state s at decision epoch t can be found. The reward can be: a lump sum received at a fixed or random time prior to the next decision epoch, accrued continuously throught the current period, a random quantity that depends on the system state at the subsequent decision epoch, or a combination of the listed forms (Puterman, 1994). In the case where the reward depends on the state of the next epoch, the reward can be represented by a following Equation 2-10.

$$R_t(s,a) = \sum_{s_{t+1} \in S} R_t(s_t, a_t, s_{t+1}) P_t(s_{t+1} \mid s,a) \tag{2-10}$$

2-2-3 Optimal Value and Policy Functions

Policy determines an agent's behavior in terms of navigating through the state space. An optimal policy is a policy that would allow the agent to follow a sequence of actions through the state space that maximizes the expected reward V^{π}. Or in the case of minimizing the cost instead of maximizing reward, the term *cost-to-go* is used instead of *expected reward* denoted by J. In this section, optimal value function and optimal policy functions are defined using the infinite-horizon discounted model and the average-reward models for optimality. These optimal value functions and policy functions are used as fundamental base for which various reinforcement learning algorithms are developed (Kaelbling et al., 1996).

Consider the infinite-horizon discounted model of optimality in Equation 2-8 where the expected infinite sum of rewards are discounted by γ. Optimal value function is simply the expected infinite discounted sum of reward as the agent executes the optimal policy from state s. Following the optimal policy, the expected reward is maximized. Thus, the optimal value function can be defined as shown in Equation 2-11.

$$V^*(s) = \max_{\pi}(V^{\pi}) = \max_{\pi} E\Big(\sum_{t=0}^{\infty} \gamma^t r_t\Big) \tag{2-11}$$

For evaluation of Equation 2-11, *optimality equations* or *Bellman equations* are used. As stated in (Puterman, 1994), the solutions of these equations correspond to optimal value functions and provide basis for optimal policy determination. The Bellman equations for the infinite discounted model are defined in 2-12.

$$V^*(s) = \max_{a} \Big(R(s,a) + \gamma \sum_{s' \in S} P(s',a,s)V^*(s')\Big), \forall s \in S \tag{2-12}$$

The Bellman equations above states that the value of a state s is the next step reward and the sum of discounted expected value of the next state given the action a according to optimal policy. Following from the Bellman equations, the optimal policy can be defined as the action that would result in optimal value.

$$\pi^*(s) = arg\max_{a} \Big(R(s,a) + \gamma \sum_{s' \in S} P(s',a,s)V^*(s')\Big) \tag{2-13}$$

Instead of the state value function, action value functions can also be considered where the value of taking action a in state s is measured following policy π. In this case, the action value function is defined as shown in Equation 2-14. Then the optimal action value function can be defined as in Equation 2-15 and can be evaluated using the Bellman equations as shown in Equation 2-16. Finally, optimal policy can be defined as shown in Equation 2-17.

$$Q^{\pi}(s,a) = E\Big(\sum_{t=0}^{\infty} \gamma^t r_t \mid s_t = s, a_t = a\Big) \tag{2-14}$$

$$Q^*(s,a) = \max_{\pi}(Q^{\pi}) \tag{2-15}$$

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s',a,s) \max_{a'} Q^*(s',a'), \forall s \in S \tag{2-16}$$

$$\pi^*(s) = arg \max_{a} Q^*(s,a) \tag{2-17}$$

Now the average-reward model of optimality is considered. Recall that the goal is to find the optimal policy that would result in the highest averaged expected reward in this case. As stated in (Sutton & Barto, 2018), for most cases it is adequate to use the average reward per time step as a benchmark to compare policies. The expected reward can then be calculated by taking the difference between the reward received and the average reward per time step. This is also known as differential return and the value function can then be derived as shown in 2-18 where $\bar{R}$ is the maximum true average reward. For the optimal policy function shown in Equation 2-19, the derivation is analogous to the discounted infinite-horizon case.

$$V^*(s) = \max_{a} \Big(R(s,a) - \bar{R} + \sum_{s' \in S} P(s',a,s) V^*(s') \Big) \tag{2-18}$$

$$\pi^*(s) = arg \max_{a} \Big(R(s,a) - \bar{R} + \sum_{s' \in S} P(s',a,s) V^*(s') \Big) \tag{2-19}$$

2-2-4 Fundamental Model-Based and Model-Free Solution Techniques

Optimal value functions and policy functions have been defined for two different infinite-horizon optimality models. Three main classes of fundamental algorithms are presented to calculate the optimal policy function: Dynamic Programming, Monte Carlo Methods, and Temporal Difference Learning. The main distinction that can be made among these classes of algorithms is the a-priori knowledge of the model and the use of bootstrapping. In this section, in-depth details of the three main solution technique variants are not discussed. Instead, central ideas to respective techniques and relevant concepts such as Generalized Policy Iteration (GPI), Direct/Indirect-RL, and On/Off-Policy methods are discussed.

Dynamic Programming: Model-based solution technique

In a MDP setting, a model refers to the state transition function P_t and the reward function R_t. Dynamic Programming (DP) is a class of algorithms that can compute optimal policies given that such a perfect model is known (Wiering & van Otterlo, 2014). Immediately it can be seen that the use of DP is limited due to its perfect model requirement. In most real world RL problems, the model is not known in advance which naturally directs one towards the use of model-free techniques. However, DP provides a theoretical foundation for other model-free solution techniques which can be seen as trying to achieve the same goal with less computation and without the assumption of a perfect model of the environment (Sutton & Barto, 2018). Two main variants of the DP algorithm are discussed along with GPI, and the concept of bootstrapping.

Policy iteration is the first DP variant and consists of two iterative phases: policy evaluation and policy improvement. Policy evaluation phase iteratively updates the value function for each state following an arbitrary policy until the successive value function difference is small enough. In other words, policy evaluation searches for V^{π} of fixed policy π for all states. On the other hand, policy improvement phase searches for a superior policy π'. This is done by iteratively selecting an arbitrary action a in a state and thereafter following the current policy for all states until value is maximized for all states. The iterative chain of policy iteration can be seen below where π_n refers to n^{th} policy and V_{π_n} refers to the value function found under n^{th} policy. Each iteration of policy is guaranteed to be better than the previous policy and for finite MDP where state and action space is finite, convergence to optimal policy is guaranteed for a finite number of iterations.

$$\pi_0 \xrightarrow{Evaluation} V_{\pi_0} \xrightarrow{Improvement} \pi_1 \xrightarrow{Evaluation} V_{\pi_1} \xrightarrow{Improvement} \cdots \xrightarrow{Improvement} \pi^* \xrightarrow{Evaluation} V^*$$

$$V_{\pi_{n+1}}(s) \geq V_{\pi_n}(s), \quad \forall n \in \mathbb{Z} \quad and \quad \forall s \in S$$

A major drawback of the policy iteration algorithm is that multiple sweeps are necessary through the state space during the policy evaluation phase until convergence to V_{π} is achieved. Policy evaluation phase of policy iteration method can be seen as its own exhaustive iterative search over the state space. Value iteration algorithm on the other hand does not wait for full convergence of the value function and stops the process after one update, truncating the evaluation process. In addtion, the value iteration algorithm combines this truncated evaluation process with policy improvement into one update rule. In fact, the policy evaluation phase and the value iteration algorithm are identical except that the maximum of value function is found over all actions. While the policy iteration algorithm directly searches for an optimal policy by changing the policy through the policy improvement phase, the value iteration algorithm indirectly searches for an optimal policy via the optimal value function (Kaelbling et al., 1996).

$$V_{t=0} \rightarrow V_{t=1} \rightarrow V_{t=2} \rightarrow V_{t=3} \rightarrow V_{t=4} \rightarrow V_{t=5} \rightarrow \cdots \rightarrow V^*$$

Two main approaches in DP have been presented. Although the policy iteration and value iteration algorithm differ in terms of exact structure and the extent to which the state space is evaluated, a core idea exists. This core idea in DP is the interaction between the policy evaluation and improvement processes to find an optimal policy, referred to as the *Generalized Policy Iteration(GPI).*

In (Sutton & Barto, 2018), the author describes the evaluation and improvement processes in GPI to be both competing and cooperating. It can be easily understood that changing the policy destabilizes the pre-established state value function in that they are no longer correct. Similarly, once the value function is updated for the changed policy, the policy is often found to be suboptimal. Thus the evaluation and improvement processes can be seen as competing in that changing either the policy or value function makes the other no longer satisfactory. However, through the iterative interaction of these two competing processes, both the value

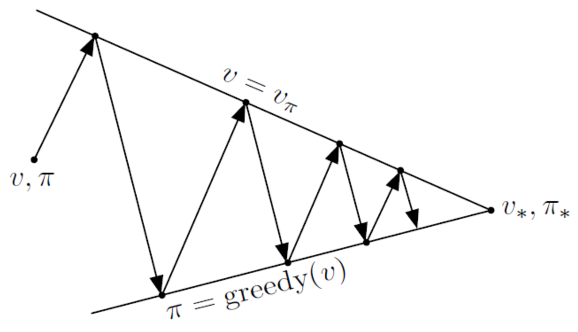


Figure 2-3: Figure showing GPI with alternating evaluation and improvement processes. From Sutton, R.S. (2018)

function and the policy reaches optimality as shown in Figure 2-3 with simplified dimensions and geometry.

In both policy and value iteration algorithms, the successive value estimations are based on previous value estimations. This is known as bootstrapping which is the inductive process of basing subsequent updates on an existing previous estimates instead of relying exclusively on actual rewards and returns. Immediately it can be understood that as the state space becomes large and generalization by function approximation is required, basing estimates on top of estimates may lead to biased or incorrect state value representations. However, the benefits in terms of computational and data efficiency through the use of bootstrapping are great especially in the case of large state space. Additionally, bootstrapping can allow faster learning by learning to exploit the state property, by recognizing a state upon returning to it (Sutton & Barto, 2018).

Monte Carlo Methods: Model-free solution technique

In the case where a-priori knowledge of the model is unavailable, two approaches can be taken. One option is to learn the model through interaction and using such model to derive optimal policy. This is known as indirect reinforcement learning. Another option, known as direct reinforcement learning, is to derive the optimal policy directly without creating a model. In the scope of control, the difference between direct and indirect adaptive control can be summarized as shown below (Kaelbling et al., 1996).

- Direct RL: Learn a controller without learning a model
- Indirect RL: Learn a model, use this model to derive a controller

Monte Carlo (MC) Methods belong to direct reinforcement learning category. Unlike DP, Monte Carlo methods do not need a-priori knowledge of a model and do not bootstrap. This stems from the fact that Monte Carlo methods only require experience of the states previously visited, actions taken, and rewards received. From this experience through performing rollouts under current policy, Monte Carlo methods are able to estimate values by averaging sample returns received.

As stated before, Monte Carlo methods are direct RL methods where a model is not learned. The problem then rises in terms of how an optimal policy can be derived from state-value pairs as one-step ahead lookup is no longer possible. Therefore in Monte Carlo methods, it is beneficial to consider state-action values denoted by $Q(s,a)$. To obtain an optimal policy, Monte Carlo methods follow the GPI scheme where now the approximate policy function and state-action value function are iteratively improved.

$$\pi_0 \xrightarrow{Evaluation} Q_{\pi_0} \xrightarrow{Improvement} \pi_1 \xrightarrow{Evaluation} Q_{\pi_1} \xrightarrow{Improvement} \cdots \xrightarrow{Improvement} \pi^* \xrightarrow{Evaluation} Q^*$$

When visited state-action pairs are considered instead of visited states in Monte Carlo methods, both the state and the action taken from that state need to be recorded to be considered as experience. This leads to the problem of many state-action pairs left unexplored and ultimately leads to an incomplete value estimation. To solve this problem of maintaining exploration, two solutions are available: exploring starts, and the use of stochastic policy. Exploring starts is done by setting the start of an episode in a state-action pair where all state-action pairs have the probability to be chosen as the start. However, in application to optimal control of an aircraft where the starting state-action pairs cannot be chosen arbitrarily, the assumption of exploring start may not be valid. Another method is only considering stochastic policies where all states and actions have a probability of being chosen. This leads to the off-policy method where two policies are maintained: one that becomes the optimal policy and another that is exploratory.

Temporal Difference Methods: Model-free solution technique

Both the Temporal Difference (TD) methods and the MC methods are model-free direct RL methods in that no a-priori knowledge of model is needed and a model is not learned to derive a controller. However, while Monte Carlo methods perform complete roll-outs of current policy from which the state-action values can be derived, temporal difference methods can update state values per time step. This immediate state value update is possible by value estimation based only on immediate reward received and the estimated value of next state. And like the DP method, TD methods utilize bootstrapping where value estimates are based on previous value estimates. TD methods can be seen as a mix of both DP and MC methods.

A key idea behind TD methods is the use of temporal difference errors for updating the value function. Temporal difference error refers to the difference between the previous estimate of the value function and the current estimate of the value function by only taking the current state value, immediate reward, and one time step ahead state value into account. The temporal difference error for $TD(0)$ is defined in Equation 2-20 taken from (Sutton & Barto, 2018) where γ is the discount factor. Note that the number between the brackets in $TD(0)$ defines the number of time steps that are taken into account which in this case is one time step ahead. The update rule based on this TD error is shown in Equation 2-21 where α is the learning rate. For convergence, the learning rate α has to be decreased every iteration (Wiering & van Otterlo, 2014).

$$\delta_t \doteq R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \tag{2-20}$$

$$V(S_t) \leftarrow V(S_t) + \alpha\delta_t \tag{2-21}$$

To find an optimal policy, the GPI is utilized again. Similar to the MC method, the TD method is applied for the evaluation phase of the GPI and faces the same difficulties of tradeoff between exploration and exploitation in terms of policy improvement. There are three main approaches to tackle the policy improvement process: off-policy Q-learning method, on-policy SARSA method, and an on-policy ACD method that maintains policy and value function separate.

The advantages of $TD(0)$ method over MC method are clear. Due to its simple incremental one-step ahead update rule based on experience, $TD(0)$ methods can be naturally adapted in an on-line, continuous problems while requiring smaller computational power and memory. The use of temporal difference error can be applied to utilize the model as in DP and can be extended to a multi-step case which allows bridging to the MC method. The simplicity and versatility of TD method allows its application to numerous optimal control tasks as it will be shown in the following sections.

The above equations define the one-step TD update rule. What should be noted is that taking more than one time step into account is also possible. First method to take more than one time step into account is called n-step TD method. This simply means that n-step rewards are observed to update the value function instead of one-step reward. The n-step TD method serves as an intermediate method between $TD(0)$ method that bases its update on one reward and the MC method that bases its update on rewards received until the end of an episode. Another method is $TD(\lambda)$ that uses eligibility traces with trace-decay parameter $\lambda \in [0, 1]$. A fading short-term weight vector, eligibility trace, is kept parallel to the long-term weight vector and the long-term weight that participates in the learning process. With the use of such eligibility trace, continuous learning can take place instead of waiting for n-step reward observation while reducing the reducing the number of n feature vectors to a single eligibility trace vector (Sutton & Barto, 2018). (Tesauro, 1992) is a well known paper showcasing one of the first successful application of TD(λ) reinforcement learning to the game of Backgammon. The author concludes that the TD net with zero built in knowledge outperforms other identical networks trained on a massive data base of human expert knowledge. And when the same temporal difference reinforcement learning approach was taken with built-in features by adding hidden layers and utilizing longer training times, the level of play was comparable to a human expert.

2-2-5 SARSA

State-Action-Reward-State-Action (SARSA) derives its name from the algorithm sequence of occupying state s, choosing action a, observing reward r, observing the next state s', and choosing action a'. Introduced by (Rummery & Niranjan, 1994) with neural networks as means of function approximation for action value function, SARSA is an on-policy algorithm following the GPI scheme while using TD methods for the evaluation part. Instead of considering state value function V^π, action value function Q^π is considered where the value of choosing action a at state s is found. As for the structure of its update rule, a modified version of the TD update rule previously defined in 2-21 is used. The new TD update rule in shown below in Equation 2-22.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \tag{2-22}$$

In the update rule, α is the learning rate to determine how much of the previously learned action value function is retained while γ is the discount factor to determine how much of the immediate reward will be considered.

For the tabular case with the assumption that all state action pairs are tried infinitely, convergence was proven for the one step SARSA(0) algorithm in (Singh, Jaakkola, Littman, & Szepesvári, 2000) following two different learning policies. However the tabular representation of the action value function has clear limitations as the size of state space increases. To combat this, many function approximation methods have been used incorporated with SARSA algorithm to generalize the action value function. Because SARSA is an on-policy reinforcement learning algorithm, it doesn't fall under the deadly triad where instability and divergence may occur when combining function approximation, bootstrapping, and off-policy training as described in (Sutton & Barto, 2018). Therefore SARSA has an advantage over Q-learning, addressed later in Subsection 2-2-6, in application cases with large state spaces where function approximation is a necessity.

Notable application cases of SARSA algorithm for large state spaces include: online $SARSA(\lambda)$ algorithm combined with sparse coarse-coded function approximator(CMAC) showing convergence when applied to various classical continuous state space control tasks such as 2D grid world, puddle world, and mountain car (Sutton, 1996), (Stone, Sutton, & Kuhlmann, 2005) where episodic SMDP $SARSA(\lambda)$ with variable eligibility trace λ and linear tile-coding function approximation have shown superior results when compared to hand coded policy for the task of keepaway in RoboCup simulated soccer with sensor/actuator noise, and a more recent application case in (Zhao, Haitao Wang, Kun Shao, & Zhu, 2016) where deep convolutional neural network is combined with SARSA update rule and experience replay for video games where comparable and sometimes higher cumulative reward was achieved to the Q-learning counterpart.

2-2-6 Q-learning

Introduced by (Watkins, 1989), Q-learning is an off-policy reinforcement learning algorithm utilizing TD method for state-action value function approximation. The author has later proven that the Q-learning algorithm converges for the case of discrete action values as long as all actions are repeatedly sampled at all states (Watkins & Dayan, 1992). Because it is an off-policy RL algorithm, it is able to estimate the optimal value and policy independently of the policy being followed. This addresses the issue of exploration. The update rule for the Q-learning algorithm is presented in Equation 2-23.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \tag{2-23}$$

Same as the SARSA algorithm, α denotes the learning rate and γ is the discount factor to determine how much of the immediate reward will be considered. From the action value function found from Equation 2-23, the policy can be updated following Equation 2-24.

$$\pi'(S_t) = arg \max_a Q(S_t, a) \tag{2-24}$$

Many extensions and application cases of the Q-learning algorithm are available. The original Q-learning algorithm utilizes the 1-step $TD(0)$ update rule. The idea to implement n-step $TD(\lambda)$ was already brought up by (Watkins, 1989) since the algorithm's debut. This has been done in (Peng & Williams, 1994) using eligibility traces. Because of this, the insensitivity of maintaining optimal action value estimate regardless of the action being taken had been lost. However, it has been demonstrated that the algorithm performs better for most cases especially for cases where the state space is continuous or large. The author argues that when the state space is large enough, the use of multi-step information propagation is a necessity. (Cichosz & Mulawka, 1995) has shown that implementing $TD(\lambda)$ update rule with the use of eligibility traces is computationally heavy for large state spaces. The author instead presents a more generalized and computationally lighter Truncated Temporal Difference (TTD) procedure that shows significantly faster learning speed when compared to the $TD(0)$ Q-learning algorithm.

With an identical motivation as the SARSA algorithm presented before, Q-learning has been used in conjunction with various function approximators to combat large state spaces. One of the earlier attempts of combining function approximator with the Q-learning algorithm is the use of Cerebellar Model Articulation Controller (CMAC) to represent the state action value function in (Atkeson & Santamaria, 1997). Neural fields have been used in (Gross, Stephan, & Krabbes, 1998) to generate a two-dimensional topology of a continuous state action space. (Glorennec & Jouffe, 1997) explored the use of fuzzy inference system where both the action and the action value functions are stored. In addition to faster learning rate when compared to feed forward neural networks, exploitation of prior knowledge can be achieved through fuzzy rule manipulation. In (Gaskett, Wettergreen, & Zelinsky, 1999), a wire-fitted neural network was employed to generate a continuous state action space where a single feed-forward neural network is coupled with a moving average interpolator. Some of the more impressive recent results came from the use of convolutional neural network to estimate state action value function in what is called deep Q-learning algorithm shown in (Mnih et al., 2013). It has been demonstrated that the deep Q-learning algorithm is able to play seven Atari games with total rewards far exceeding any other algorithms including SARSA and in some games a human controller.

Although some impressive results were demonstrated for SARSA and Q-learning algorithms, there exists some fundamental limitations for either algorithms. Both SARSA and Q-learning algorithms do not have an explicit policy improvement element making them *critic only* methods. The policy is typically found through a computationally expensive optimization procedure for every state occupied which forces the action space to be discretized and thus preventing the true optimum policy to be found (Grondman, Busoniu, Lopes, & Babuska, 2012).

2-3 Function Approximation

The three fundamental RL solution techniques (DP, MC, TD) all rely on look-up tables to store value for each states or state-action pairs. For RL problems where the state-space is limited and discrete, look-up tables are a valid method of information storage. However for real world RL problems where the state dimensionality is high and/or continuous, as in the case of low-level aircraft control, the computational complexity grows exponentially as the number or state variables increase. This is referred to as the *curse of dimensionality*, first introduced in (Bellman, 1957). Initially the term was often reserved as the reason why Dynamic Programming cannot be used (Powell, 2011). However, (Sutton & Barto, 2018) argues that the curse of dimensionality is an inherent problem of RL problems with large state-space. Therefore it will henceforth be treated as such instead of as a problem specific to DP.

(Sutton & Barto, 2018) states that the key issue is generalization in which a meaningful approximation needs to be generated from samples of a much larger subset. Function approximation is a method of generalization that allows approximation of a desired function via the use of samples. Theoretically, any function approximation can be combined with an existing RL algorithms to tackle RL problems with large and/or continuous state-space. This is possible as it is assumed that in a large, smooth state space, similar states are expected to have similar values and similar optimal actions (Kaelbling et al., 1996). In the case of aircraft dynamics state space where no discontinuity can be expected apart from the hard limit on altitude(the ground), this assumption is practically valid. If sufficient generalization can be achieved with the use of function approximation, the curse of dimensionality can be alleviated.

It is clear that function approximation is needed for most RL problems including the low-level aircraft control problem. However, new points of consideration arise when function approximation is used such as: proof of convergence, (near)optimality, and consistency (Busoniu, Babuska, De Schutter, & Ernst, 2010). Additionally, the complexity of function approximation method is directly related to the computational complexity of the RL algorithm. Therefore, the choice of function approximation method is a non-trivial task. The following subsections list and investigate some of the most commonly used function approximation methods.

2-3-1 Overview of Function Approximation Methods

Two categories of function approximation methods are parametric and non-parametric function approximation methods. Parametric function approximation methods aim to approximate the target function by adjusting some form of weight vector parameters to a predefined linear or nonlinear function through supervised learning. Commonly known function approximation methods of Recursive Least Squares (RLS) method and Artificial Neural Network (ANN) are examples of parametric function approximation methods. On the other hand, non-parametric function approximation methods are not limited in form to a fixed parametrized class of functions (Sutton & Barto, 2018). Non-parametric function approximation methods can typically be defined as memory-based function approximation methods. Compared to parametric function approximators, its advantage is that it is highly flexible and thus is favored in cases where data is scarce. However, the dependence on memory means that for real world applications its ability to alleviate the curse of dimensionality is limited. This is especially troublesome when non-parametric function approximators are used for online reinforcement learning as new data are received continuously throughout the entire lifetime

(Busoniu et al., 2010). In this overview of function approximation methods, only parametric function approximation methods are considered.

2-3-2 Linear Parametric Function Approximation

Also known as linear in the parameter function approximators, linear parametric function approximation methods employ linear weights that have to be learned. Compared to nonlinear parametric function approximators, its main advantages are the ease of theoretical property analysis of the resulting RL algorithm, efficient data handling, and lower computational cost. The ease of theoretical property analysis has lead to convergence proofs for many RL algorithms when utilizing linear parametric function approximation. The use of polynomial in combination with least squares fitting is an effective and popular approach. Many variants exist owing to different methods of least squares techniques. RLS method is discussed in particular as it is the most suitable method for online implementation as incoming data is used to update the existing model. (Xu, He, & Hu, 2002) presents the use of RLS in $TD(\lambda)$ algorithm and subsequently applied it for critic training within actor critic method. The authors have shown superior efficiency for both $TD(\lambda)$ algorithm and the actor critic algorithm compared to conventional methods using least squares for the application case of cart pole balancing problem. A more relevant use cases of RLS are given in (Zhou, van Kampen, & Chu, 2017), (Zhou, Van Kampen, & Chu, 2016), and (Zhou et al., 2018b) where the author uses RLS for the online incremental system model identification. Another example of linear parametric function approximation is coarse coding. Coarse coding, with tile coding or CMAC as its variant, partitions the state space by a set of overlapping receptive fields where the size and shape and the number of receptive fields can be adjusted to determine the nature of generalization (Sutton & Barto, 2018). Due to its binary feature vectors, the computation of approximated function is straightforward and simple. An example of successful real life applications case is given in (Stone et al., 2005). Tile coding was used to generalize a large state space with hidden and uncertain states in combination with $SARSA(\lambda)$ algorithm for Robocup soccer keepaway. Additionally, (Degris, Pilarski, & Sutton, 2012) has shown the practicality of tile coding with multiple actor critic algorithm variants for various real time continuous state action space problems. Radial Basis Function (RBF) can be seen as a form of coarse coding such that the state space is generalized by overlapping Radial Basis Functions(RBFs). However, a key difference is that with RBFs, the feature vector is not binary and instead represented in terms of intervals. Due to this key difference, the approximated feature is smoothly varied and differentiable but also adds computational complexity when compared to coarse coding with binary feature vector (Sutton & Barto, 2018). In (Kretchmar & Anderson, 1997), a comparison of the use of CMAC(tile coding) and RBFs as function approximators as part of Q-learning algorithm for the mountain car problem is presented. Results show that RBFs is able to achieve higher steady state performance when compared to CMAC.

2-3-3 Nonlinear Parametric Function Approximation

Unlike linear parametric function approximation represented by many different methods, the nonlinear parametric function approximation can mainly be represented by Artificial Neural Networks (ANNs). In its simplest functional form, and the most widely used, a single layer feed forward ANN is constructed with an input layer, one hidden layer, and an output layer. Feed forward in this case meaning that no connection is made such that the node output affects input. (Cybenko, 1989) demonstrated that a single hidden layer ANN with a finite number of sigmoidal activation functions can approximate a continuous function to any degree of accuracy. Apart from the choice of activation functions and the number of nodes in each layer, single hidden layer feed forward ANN is straightforward to implement with guaranteed global approximation power. To train the ANN, back propagation algorithm is commonly employed where alternating forward and backward passes through the ANN are performed. In the forward pass, the activation of each nodes are calculated with current weights. In the backward pass, the error of the current estimate is calculated and passed backwards in the form of partial derivatives to update the weights accordingly. The existence of weight gradients are especially useful for when ANNs are used to estimate the value function via TD error calculation or improve policy according the policy-gradient algorithm. Many successful results have been obtained using single hidden layer feed forward ANN as a function approximator in reinforcement learning. There exists reinforcement learning problems with complex target function that require a hierarchical structure of lower level abstractions (Sutton & Barto, 2018). Deep Neural Network (DNN) employ multiple hidden layers with varying structures to achieve such form. However, in the context of low level aircraft control where the target function of reference signal is relatively straight forward, a single hidden layer feed-forward ANN may suffice.

As previously listed under linear parametric function approximation methods, RBF Network (RBFN) can be transformed to a nonlinear function approximation method by allowing the centers and widths of the RBFs to be adjusted. The main difference between the ANN and the RBFN is that RBFNs utilize local representation while ANNs use distributed representation of the state space (Alpaydin, 2014). Local representation means that for a given input, only units in the relevant local neighborhood are active. Distributed representation on the other hand means that all units within the approximator are active for an input. This provides the advantage of RBFNs over ANNs in that its training speed is faster as only local weight adjustments need to converge. However the distinct disadvantage of RBFNs become evident as the dimensionality of the input layer increases as many more local units are needed when compared to ANNs (Alpaydin, 2014). Therefore it can be stated that although the RBFN typically demonstrate faster convergence, it is less well suited to alleviate the curse of dimensionality than the ANN as input state dimension increases.

2-4 Approximate Dynamic Programming

As previously stated, Dynamic Programming methods aim to solve the discounted infinite optimality problem defined in Equation 2-8 by recursive backwards calculation of Equation 2-12. Due to the curse of dimensionality, the increase in state dimensions pose a significant problem for such backwards calculation of the Bellman equation. Approximate Dynamic Programming (ADP), has been developed with the goal of tackling the curse of dimensionality.

The fundamental idea behind ADP is how the backwards recursive calculation of Bellman equations can be redefined as an algorithm that steps forward through time (Powell, 2011). Two problems arise when considering such conversion to step forward in time. The first problem rises from the fact that now backwards calculation of value function and subsequent optimal policy calculation is unavailable. Therefore, an approximation of value function need to be used to determine optimal policy. The second problem is that the transition probability function is unknown. In other words, the state transition based on the action taken through value approximation is unknown. This problem is resolved by approximating a model of a system which can either be identified before or during the reinforcement learning process.

First a branch of ADP called Linear Approximate Dynamic Programming (LADP) demonstrating efficient optimality problem solving for a special case is briefly introduced. Second, another approach to ADP called ACD that can be applied to a broader set of problems is introduced and analyzed. Finally, an overview of ACD algorithm applied for various control problems are presented.

2-4-1 Linear Approximate Dynamic Programming

When the system is Linear Time Invariant (LTI), an efficient ADP method exists called the LADP. Before introducing the algorithm, a concept of cost function is introduced within the scope of ADP. The goal of reinforcement learning to maximize reward can directly be translated to the goal of optimal control to minimize the control cost (Khan, Herrmann, Lewis, Pipe, & Melhuish, 2012). Consider a discrete nonlinear system shown in Equationeq:disnonlinsys. The control cost to be minimized can be represented in the form of cost function J as shown in Equation 2-26 with U representing the utility function. Notice that the cost function and the infinite hoziron discounted optimality model previously defined in 2-8 are essentially identical. The only difference is that the cost function needs to be minimized. Then the derivation of optimal cost at time k shown in Equation 2-27 and the optimal control action $u^*(k)$ shown in Equation 2-28 according to Bellman are straightforward.

$$x(k+1) = F\Big(x(k), u(k), k\Big) \tag{2-25}$$

$$J\Big(x(i), i\Big) = \sum_{k=i}^{\infty} \gamma^{k-i} U\Big(x(k), u(k), k\Big) \tag{2-26}$$

$$J^*(x(k)) = \min_{u(k)} \Big(U(x(k), u(k)) + \gamma J^*(x(k+1))\Big) \tag{2-27}$$

$$u^*(k) = arg \min_{u(k)} \Big(U(x(k), u(k)) + \gamma J^*(x(k+1))\Big) \tag{2-28}$$

LADP applies to Linear Quadratic Regulation (LQR) problems when the system is defined by linear differential equations and the cost function is a quadratic function. In this special case, the optimal control law can be obtained for every time step k in the form shown in Eq. 2-29 where the gain matrices L_k is defined as algebraic Riccati Eq. 2-30 (Bertsekas, 1995).

$$u^*(x) = Lx \tag{2-29}$$

$$L = -(B'KB + R)^{-1} B'KA \tag{2-30}$$

Essentially the LADP method provides an explicit optimal control solution in the form of linear feedback gain matrix without the need for a model. However, as the problem definition states, the system to be controlled must be LTI. In many real world control problems, systems are often nonlinear and time varying with an aircraft as an prime example of such systems. (Zhou et al., 2017) and (Zhou, van Kampen, & Chu, 2018a) present a variant of LADP called Incremental Approximate Dynamic Programming (IADP) method aimed to reduce the LADP limitation of inapplicability to nonlinear and time varying systems by the use of linearized incremental system model identified by a least squares method. The results show that after training, the policy is able to successfully control a nonlinear system for disturbance rejection and reference following tasks. Despite the results achieved with IADP algorithm, an offline policy training is still a necessity as the ADP algorithm is a critic only algorithm. Additionally, the reliance on quadratic cost function restricts the generality of the method. Therefore the algorithm is not fully online, shows difficulties handling time varying systems, and its applicability to general control problems limited.

2-4-2 Adaptive Critic Design

All algorithms considered so far have been critic-only algorithms such as Q-learning, SARSA, and LADP where policy is found through an optimization procedure at each state occupied. A different approach to ADP is presented where both the actor and critic elements are implemented each explicitly handling policy improvement and evaluation. The advantage of separate policy improvement and evaluation blocks is that real time policy change is facilitated. In many control tasks, real time policy change, or control law change, is desirable as the environment surrounding the agent may change. In the case of aircraft control such environment change can be altitude change where the air pressure/temperature difference calls for different control input to achieve optimum control. A more extreme example of environment change in aircraft control task may be when the aircraft is damaged, fundamentally changing the aircraft dynamics and requiring a radically different control law to be applied.

(Werbos, 1977) presented an idea of including an explicit “action model” and coined the terms Heuristic Dynamic Programming (HDP) and Dual Heuristic Programming (DHP). The main difference between the HDP and the DHP is the error minimized by the critic networks. In HDP the critic network aims to estimate the cost function J while in DHP the critic network aims to estimate the derivatives of J with respect to state vector that may also include control vector. A third variant of the ACD, Globalized Dual Heuristic Programming (GDHP) has a critic network that aims to approximate both J and its derivatives. In this subsection HDP, DHP, and GDHP are presented along with their action dependent variants Action Dependent Heuristic Dynamic Programming (ADHDP), Action Dependent Dual Heuristic Programming (ADDHP), and Action Dependent Globalized Dual Heuristic Programming (ADGDHP). Note that the ACDs presented below will all assume that ANNs are used as function approximator for actor, critic, and model blocks as they were originally envisioned with ANNs.

HDP

ACDs are actor critic algorithms. HDP maintains a separate function approximation blocks for policy evaluation and improvement as outlined by GPI procedure. Elements within the ACD architecture responsible for policy evaluation and improvement are respectively named actor and critic. The differences between HDP, DHP, GDHP and their AD variants can be outlined by critic output, how both actor and critic are trained, and the use of a system model.

The critic within HDP outputs cost function J at each timestep and aims to minimize the TD error. TD error was previously defined in Eq. 2-20 and is reformulated below in terms of utility function U and cost function J in Eq. 2-31. The subsequent error function is shown in Eq. 2-32 and the gradient descent update rule for ANN weights is given in Eq. 2-33. The actor on the other hand takes state information at each timestep and outputs control action for the next timestep u_{t+1}. The actor aims to generate action such that the cost function is minimized. Because the generated control action affects the next time step states and subsequently the cost function, training the actor requires the derivative vectors $\frac{\partial J_{t+1}}{\partial x_{t+1}}$ and $\frac{\partial x_{t+1}}{\partial u_t}$. $\frac{\partial J_{t+1}}{\partial x_{t+1}}$ can be found from the critic directly and $\frac{\partial x_{t+1}}{\partial u_t}$ requires a model to be estimated.

$$e_t = J_t - U_t - \gamma J_{t+1} \tag{2-31}$$

$$\Delta E_{C,t} = \frac{1}{2} e_t^2 \tag{2-32}$$

$$\Delta W_{C,t} = -\eta \frac{\partial E_{C,t}}{\partial J_t} \frac{\partial J_t}{\partial W_{C,t}} \tag{2-33}$$

DHP

The main difference between the HDP and the DHP is that the critic in DHP aims to estimate the gradient of cost function J with respect to states denoted by λ shown in Eq. 2-34. The TD error to be minimized in this case is shown in Eq. 2-35. Using λ, Eq. 2-35 can be rearranged to 2-36. The error function is the same as shown in Eq. 2-32 and the new critic weight update rule is shown in Eq. 2-37.

$$\lambda_t = \frac{\partial J_t}{\partial x_t} \tag{2-34}$$

$$e_t = \frac{\partial Jt}{\partial x_t} - \frac{\partial U_t}{\partial x_t} - \gamma \frac{\partial J_{t+1}}{\partial x_t} \tag{2-35}$$

$$e_t = \lambda_t - \frac{\delta U_{t+1}}{\delta x_t} - \gamma \lambda_{t+1} \frac{\delta x_{t+1}}{\delta x_t} \tag{2-36}$$

$$\Delta W_{C,t} = -\eta \frac{\partial E_{C,t}}{\partial \lambda_t} \frac{\partial \lambda_t}{\partial W_{C,t}} \tag{2-37}$$

From Eq. 2-36, the second term can be simply calculated from the utility function chosen. λ_t and λ_{t+1} are directly calculated by the critic. However the state transition function, $\frac{\partial x_{t+1}}{\partial x_t}$ from the third term in Eq. 2-36, is needed. The critic training is more complicated in DHP as the state transition function $\frac{\delta x_{t+1}}{\delta x_t}$ needs to be found through two different pathways. The state transition function can be calculated according to Eq. 2-38. The first term on the right hand side can be directly obtained from the model. The second term requires both the model and the actor to be obtained.

$$\frac{\delta x_{t+1}}{\delta x_t} = \frac{\partial x_{t+1}}{\partial x_t} + \frac{\partial x_{t+1}}{\partial u_t}\frac{\partial u_t}{\partial x_t} \tag{2-38}$$

Due to a more complicated critic training procedure, the computational burden is increased for DHP when compared to HDP. However, by allowing the critic to approximate the gradient of the cost function J, many researchers have found that DHP outperforms HDP for various tasks. Among them, (Venayagamoorthy, Harley, & Wunsch, 2002) compared the performance of HDP and DHP implementations for the task of turbo generator voltage regulation task. The author states that the superior performance of DHP when compared to HDP is because the derivatives of the cost function J, λ, is the most important information for optimal trajectory search. This is explicitly found in DHP while in HDP, this information is indirectly obtained through backpropagation potentially resulting in an unsmooth λ. (Si & Wang, 2001) state that by explicit approximation of λ in DHP, the potential of additional error build up through time by backpropagation in HDP can be reduced.

GDHP

GDHP can be seen as a combination of the critic blocks in both HDP and DHP. In GDHP, the critic simultaneously estimates both the cost function J and the cost function gradient λ. Eq. 2-39 shows the critic weight update rule for GDHP showing two separate adaptation signals with learning rates η_1 and η_2 with $E_{C1,t}$ and $E_{C2,t}$ representing error functions previously defined for HDP and DHP. As described in (Prokhorov & Wunsch, 1997), the original GDHP architecture implements an additional critic network in parallel to a standard HDP style critic network. The standard HDP style critic network handles adaptation signal 1 while the second parallel critic network produces adaptation signal 2 to be used in the standard HDP style critic network using the approximated cost function J and all hidden neuron states from the first critic network as input.

$$\Delta W_{C,t} = -\eta_1 \frac{\partial E_{C1,t}}{\partial J_t}\frac{\partial J_t}{\partial W_{C,t}} - \eta_2 \frac{\partial E_{C2,t}}{\partial \lambda_t}\frac{\partial \lambda_t}{\partial W_{C,t}} \tag{2-39}$$

As previously mentioned, (Venayagamoorthy et al., 2002) suggests that the gradient of cost function J is the most important information to optimal control. However, this statement does not discount the importance of the cost function J itself. By evaluating both the cost function J and its gradient, the performance of GDHP is expected to exceed that of either HDP or DHP. This is evidenced in (Prokhorov, Santiago, & Wunsch, 1995) and (Prokhorov & Wunsch, 1997) where an empirical comparison study was performed for the aircraft auto landing task which showed GDHP outperforming both HDP and DHP. However, the results show that both DHP and GDHP are considerably better when compared to HDP while GDHP only shows marginal performance gain when compared to DHP. This raises a question of whether the added computational complexity of GDHP can be justified by its marginal performance gain over DHP.

Action Dependent variants: ADHDP, ADDHP, ADGDHP

All of the previously mentioned HDP, DHP, and GDHP have model dependency for either actor training, critic training or both. Action dependent ACDs aim to remove or alleviate such model dependency by directly connecting the actor block output as critic block input. As a result of such structure, the model dependency for the actor in all of AD variants have been removed. For ADHDP in particular, the dependence on the model is completely removed as the HDP critic does not require a model in the first place. For ADDHP and ADGDHP, the model dependency is not completely removed as the model is still necessary to calculate the derivative term required for critic training. Because ADHDP is the only ACD that removes model dependency completely, it is discussed more in detail.

In (van Kampen, Chu, & Mulder, 2006), HDP and ADHDP have been tested for F16 longitudinal control task. Empirical results show that with offline training, HDP outperforms ADHDP for controlling both the baseline F16 model and the F16 model with online altered dynamics to which the ACDs had to adapt to. (Prokhorov & Wunsch, 1997) shows similar results for the aircraft auto-landing task where HDP reported more “tight successes” than ADHDP.

2-4-3 Summary of ACDs

Table 2-1 outlines all ACDs that have been introduced so far. It shows the actor model dependence removed for AD variants. In particular, it can be seen that model dependence is completely removed for ADHDP.

Table 2-1: Comparison summary of ACD

	Critic			**Actor**		
	Input	**Output**	**Model Dependency**	**Input**	**Output**	**Model Dependency**
HDP	x	J	X	x_t	u_t	O
DHP	x	$\frac{\partial J}{\partial x}$	O	x_t	u_t	O
GDHP	x	$J, \frac{\partial J}{\partial x}$	O	x_t	u_t	O
ADHDP	x, u	J	X	x_t	u_t	X
ADDHP	x, u	$\frac{\partial J}{\partial x}, \frac{\partial J}{\partial u}$	O	x_t	u_t	X
ADGDHP	x, u	$J, \frac{\partial J}{\partial x}, \frac{\partial J}{\partial u}$	O	x_t	u_t	X

Comparison studies have shown that the performance of ADHDP is lower than HDP and even lower when compared to DHP and GDHP. General insight of the ACDs can be achieved that as the ACDs become more advanced and computationally complex in the order of ADHDP, HDP, DHP, and GDHP, an increase in performance can also be expected. The main problem associated with ACDs can also be identified. In their original forms, all ACDs except for ADHDP requires a model of the system. This model dependence introduces additional uncertainty to the algorithm rendering them susceptible to noise and external disturbances that can ultimately lead them to be unstable. Additionally, this problem of model dependence introduces the need to train the model for the ACD to be able to derive optimal control law. For complex and nonlinear systems, such as an aircraft, offline model identification is key to ensure control stability. In other words, for complex nonlinear systems, offline model training is often required or recommended.

2-5 Application Cases of ACD to Aircraft Control

ACDs have been applied to various control problems with its ease of online policy improvement when compared to critic-only RL algorithms. One of the earlier ACD real time application case that demonstrates such capability is shown in (Benbrahim, Doleac, Franklin, & Selfridge, 1992) where a physical ball balancing task was completed with real time learning. Results showed steady increase of steps until failure until trial number 100 where no subsequent failures were observed. In this section a detailed presentation and analysis of ACDs applied to aircraft control are given.

2-5-1 DHP 4-State Longitudinal Aircraft Model Control, 1996

The research presented in (Balakrishnan & Biega, 1996) show one of the first attempts to apply ACD for not only aircraft control but for feedback control of a system. The structure of the ACD used in the research follows the form of standard DHP where the gradient of cost function λ is approximated by the critic network. The ANN structure used in this research is a feedforward ANN with sigmoidal activation functions. The task of LTI 4-state aircraft longitudinal control of maintaining trimmed aircraft states with varying initial states, the results show that the DHP is able to generate a near optimum control law. As there is no mention of model network training, it is assumed that a perfect model had been used for this research. The problem setting presented in this research is limited. There are many other factors that need to be taken into consideration for application to actual aircraft control. First, a perfect model of the aircraft which is required for DHP is non existent and an approximate model needs to be derived. Second, the aircraft is a time varying system. Third, the aircraft is a highly nonlinear and coupled system. Fourth, the measurement noise needs to be taken into account. Fifth, different sampling times for different sensors need to be taken into account. Finally, external disturbances such as wind and gusts need to be taken into account. However, despite the limited scope of the research conducted in (Balakrishnan & Biega, 1996) that prevents the results to prove that DHP can be used for Aircraft optimal control, it needs to be considered that this research mostly serves as an indication that DHP shows potential to converge to near optimal control law when acting as feedback controller. Therefore, the results have value in that it serves as a feasibility study of DHP for aircraft control.

2-5-2 DHP Missile Model Path Finding, 2002

(Han & Balakrishnan, 2002) presents the use of offline trained DHP for optimum path finding of a nonlinear longitudinal missile model. The problem setting is that of guidance control where the start and end states are bounded and the time to end state needs to be minimized.

The missile is fired at the speed of varying initial Mach numbers and the optimum missile angle of attack time history that minimizes the time to end state, position behind its flight path at a minimum Mach number of 0.8, is found. The results show that with varying initial conditions, the missile was successfully controlled to the end state. Although the research presented in (Han & Balakrishnan, 2002) is not directly relevant to the goal of this thesis in that it solves a path finding problem rather than low level aircraft control problem, the results presented in this research shows the applicability and feasibility of ACD to aircraft path finding problem. Relating to the development of ACD controller for the Cessna Citation aircraft, this research demonstrates the possibility of a full RL ACD controller development to control both the flight trajectory and trajectory tracking.

2-5-3 DHP Full Scale Aircraft Model Control, 2002

(Ferrari & Stengel, 2002) presents the results of one of the first implementation of ACD to control a full, nonlinear, six-degree-of-freedom simulation of an aircraft. The ACD implemented is DHP with a known set of LTI models at various operating points to provide one step ahead state prediction to both the actor and critic networks. Single hidden layer ANNs have been used for both the actor and critic blocks. Both the actor and critic network weights are initialized for different operating points. The initialization procedure is derived by using the knowledge of LTI control law at different operating points. As for online actor critic training, a variant of the resilient backpropagation algorithm (RPROP) is used to alleviate the negative impact of large neural networks that are required for full aircraft simulation. (Ferrari & Stengel, 2002) show accelerated convergence at low gradient areas and slowed search speed when local minima are missed. The implemented algorithm follows the general structure of DHP but mainly differs in that actor block is split into three ANNs with two ANNs forming an "actor" block and one ANN forming a supplementary actor block all contributing to generate action signal. The layout of the DHP architecture implemented in (Ferrari & Stengel, 2002) is shown in Figure 2-4 with NN_A representing the combination of two ANNs and NN_F representing the supplementary actor network.

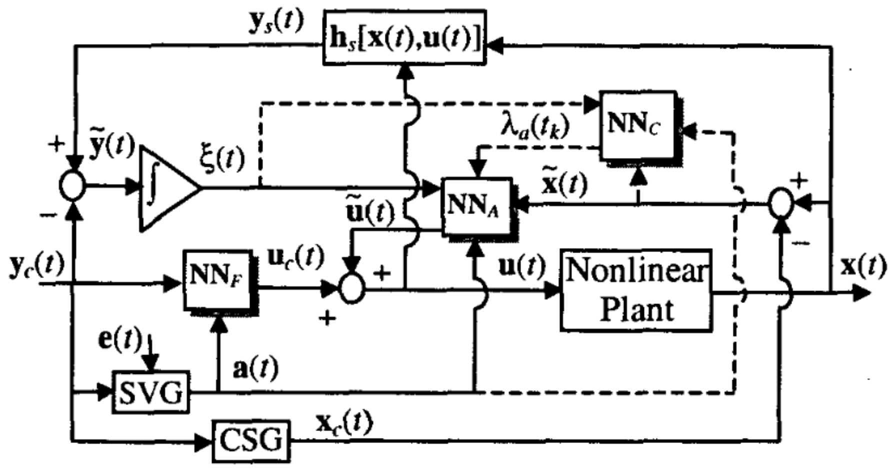


Figure 2-4: DHP architecture showing separate actor blocks. From Ferrari, S. (2012)

The results of flight control simulation show that both the stability and performance of the implemented DHP architecture is good. In comparison to an equivalent PI neural network controller that have been initialized with the same weights, the implemented algorithm showed superior performance. The ANN weight initialization, split actor ANN architecture, and RPROP back propagation algorithm used to train the networks have been implemented to manage the negative effect of a complex nonlinear aircraft system. For the goal of implementing an ADP for the Cessna Citation model, such results are directly comparable and adaptable. However, the weight initialization method requires linearized model of the system at different operating points and thus requires a-priori knowledge of the system. Therefore it can be said that the weight initialization method is of less value considering the real life implementation of the algorithm. Also note that the work presented in (Ferrari & Stengel, 2002) assumes perfect measurements and no external disturbances.

2-5-4 ADHDP Full Scale Apache Model Control, 2003

(R. Enns & Si, 2003) show implementation of an ADP variant called Direct Neural Dynamic Programming (DNDP) for the simulated trajectory tracking task of a validated nonlinear model of an Apache helicopter. Simulations have been performed with and without turbulence and step gusts generated by Dryden model. The actor networks' weights and critic network weights have been randomly initialized. DNDP can be classified as ADHDP as the actor block outputs are directly connected as critic block inputs. The advantage of such architecture as explained by the authors are twofold. One is that the benefits system cross couplings are taken into account for control action generation as the need to split the control system into multiple actors is not necessary. Another is an advantage inherent to ADHDP in that no system model is required. The results of various tracking task simulations at different turbulence conditions show that the implemented DNDP algorithm is able to control the simulated complex nonlinear system.

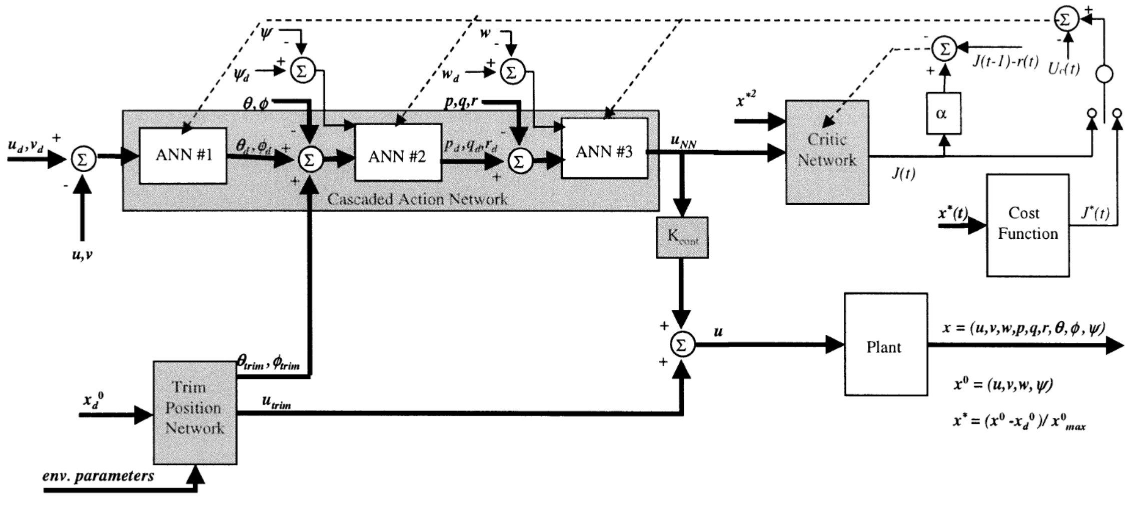


Figure 2-5: DNDP architecture showing the cascaded actor networks and a separate trim network. From Enns, R. (2003)

For the implementation of DNDP, the authors included two major features that differentiate their work from a baseline ADHDP architecture to accommodate the complex, nonlinear, and highly cross coupled system that has to be controlled. The first feature is the cascaded actor networks that can be seen in Figure 2-5 taken from (R. Enns & Si, 2003). Instead of utilizing one actor ANN taking a large number of states to generate multiple control actions, the cascaded actor networks each take a relevant subset of states to output the reference trajectory for the following actor network. The cascaded structure directly follows from the cascaded controllers seen in conventional helicopter controller architecture. The advantages of such cascaded actor networks as described by the authors are increased learning speed and that the system's physical relationships and dependencies can be exploited by the actor networks. The first advantage is probably due to the increased overall learning efficiency by decreased number of hidden layer weights for each actor networks allowed as a more explicit relationship between each actor inputs and outputs can be drawn. The second feature that differentiates this implementation of DNDP is the offline trained trim network that is embedded into the DNDP architecture as shown in Figure 2-5. As the authors describe in (R. Enns & Si, 2003), "the ability to use the NDP to trim the helicopter is paramount to successfully applying DNDP to the tracking control problem". All of the features described above can be seen as embedding expert knowledge into RL algorithm.

2-5-5 DHP Full Scale Aircraft Model Adaptive Control, 2004

(Ferrari & Stengel, 2004) can be seen as an extension of research conducted in (Ferrari & Stengel, 2002). The implemented DHP architecture is kept the same with the same training procedure. While there are no changes made to the implemented DHP itself, multiple case studies are available in (Ferrari & Stengel, 2004) where the implemented DHP has been simulated for different maneuvers.

Case 1 presented in the article is identical to that of (Ferrari & Stengel, 2002) showing the controlled aircraft response to a $5\,\text{deg}\,\gamma$ step command and $30\,\text{deg}\,\phi$ step command. Case 2 studies the aircraft response to a large $-70\,\text{deg}$ turn command to study the feasibility of the implemented DHP controller when a conventional decoupled controllers would be compromised due to significant coupled dynamics present during the maneuver. The simulation model has not been validated for this maneuver as the post-stall aerodynamic effects had not been modeled. Nevertheless the implemented DHP controller showed successful results of being able to track the desired banked turn while a conventional decoupled controller fails to do so, showing gyration from which it is unable to recover from. Case 3 studies the ability of the implemented DHP controller to adapt in case of an emergency. In an approach and landing maneuver, multiple control failures are simulated including engine failure and stuck stabilator/rudder. In the recovery phase, engine thrust available returns to 50%, the stabilator fully operational, and rudder still stuck. The simulated responses of DHP controller show a more efficient usage of available thrust by reducing angle of attack oscillations while showing overall better damping of roll and heading angle oscillations. Case 4 aims to simulate the DHP controller's ability to handle model mismatch. Parameters of the simulated aircraft are changed from what the model used in DHP had been trained for with an overall decrease of control effectiveness and stability coefficients. The time response from the small angle variation simulation show DHP controller's performance matching that of a linear controller derived from a perfect model while showing improvement over linear controller derived from the offline training model (prior to change).

While the perfect sensor assumption and offline training phase are still present for the DHP implemented in (Ferrari & Stengel, 2004), it can accommodate highly coupled dynamics the controlled full aircraft model. Therefore it can be stated that the unique design features available in this implementation of DHP, as discussed in Subsection 2-5-3, can be considered for implementation when designing the ACD controller for the Cessna Citation model.

2-5-6 Full Scale F16 Model Longitudinal Flight Control, 2006

(van Kampen et al., 2006) investigates the differences between two ACD variant controllers HDP and ADHDP through implementation to a simulated longitudinal F16 model. The implemented HDP and ADHDP both include cascaded actor networks similar to those seen in (R. Enns & Si, 2003). Additionally the actor networks are split in two at the highest level to separate thrust and pitch angle controllers. A separate predefined trim control input block is maintained as well. The control architecture is shown in Figure 2-6 taken from (van Kampen et al., 2006). Apart from the control structure, all other design features for both ADHDP and HDP are maintained as close to the original design as the research conducted in (van Kampen et al., 2006) is for the purpose of comparison of two ACDs. Both the critic and actor ANNs are fully connected single hidden layer ANNs with tanh activation functions. The implemented HDP architecture maintains a global model that is pre-trained offline before the offline actor-critic training procedure and updated online. During the global model pre-training procedure, both the actor and critics blocks are set constant.

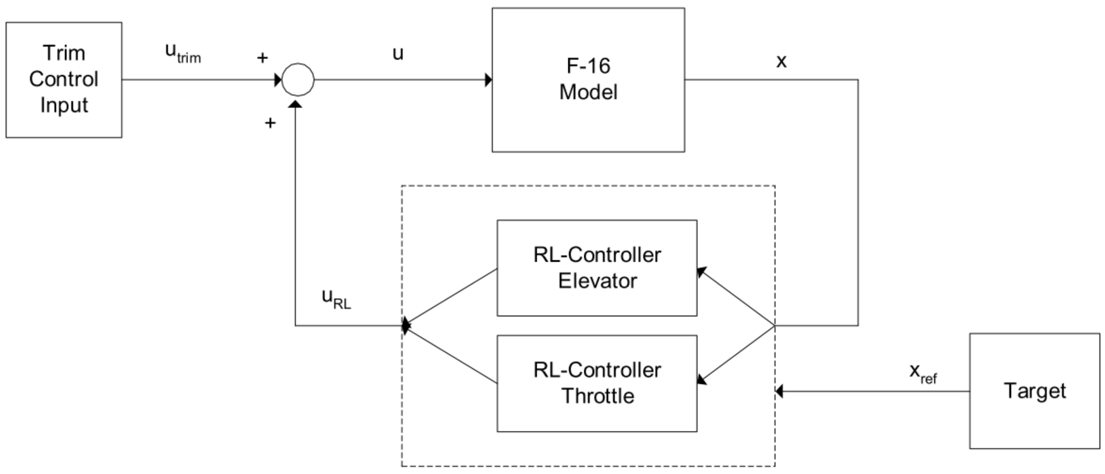


Figure 2-6: Control architecture showing separate pitch and thrust controllers. From Van Kampen, E. (2006)

Both the implemented ADHDP and HDP controllers have been simulated for the task of reference pitch angle tracking task while maintaining constant velocity. Multiple experiments have been performed for such task. To investigate the robustness of each designs, the first experiment compares the time series response data when plant dynamics are altered during controller operation. The second experiment compares the effect of sensor noise to each controller implementations by introducing noise to the pitch angle sensor measurements. The final experiment aims to investigate the effective control range of each controllers by changing the initial altitude and airspeed from the initial trimmed values to which the controllers have been trained for. The results show that HDP controller outperforms ADHDP controller for all experiments under all tested conditions except for the second experiment. For the first experiment, it has been demonstrated that by keeping an explicit global model of the system, HDP controller learning method is less complex with less interdependency between the actor and critic blocks. Therefore it shows better robustness to changing system dynamics during operation. For the second experiment conducted to investigate the effect of noisy signals, it has been shown that HDP is less capable of coping with noisy measurements when compared to ADHDP. The author states this result is again due to the fact that an explicit representation of a model is used which introduces an additional source of error. The final experiment results showed significant difference between ADHDP and HDP controllers where the ADHDP controller became unstable while the HDP controller was able to maintain a steady error.

The conclusion of the research conducted in (van Kampen et al., 2006) is that mainly due to the explicit representation of the system model, HDP is able to outperform ADHDP. When considering the results found in this research to the scope of online ACD controller implementation to the Cessna Citation model, the need to train the system model offline is not desirable. On the other hand, although the model is not required, the relatively low performance shown by ADHDP is also undesirable.

2-5-7 Adaptive Critic Designs for Discrete-Time Zero-Sum games With Application to H_∞ Control, 2007

The research provided in (Al-Tamimi, Abu-Khalaf, & Lewis, 2007) cannot be seen as implementation of HDP and DHP controller for the purpose of aircraft control. But rather it needs to be understood as a presentation of forward in time RL algorithms following HDP and DHP structure that are able to solve the Riccati equation of discrete time H_∞ optimal control problems. And for the purpose of such presentation, a linear F16 model is used to compare the formulated HDP and DHP algorithms. Although practical design features that helps the implementation ACDs in aircraft optimal control are not presented, general insight to HDP and DHP can be gained through the results provided in (Al-Tamimi et al., 2007). The results show that the DHP critic network converges to the Riccatti equation solution faster than that of HDP due to the availability of vector gradient information rather than scalar information. Therefore for the purpose of implementing ACD for the control task of the Cessna Citation model, the results of this research should be understood as that H_∞ control formulation is possible using ACD and that DHP performs better than HDP in such formulation.

2-5-8 Online 2 State Longitudinal Aircraft Model Control Using Incremental Heuristic Dynamic Programming, 2016

Thus far, all ACD implementations presented require an offline trained model unless ADHDP had been implemented which does not require a model. ADHDP has a relative poor performance when compared to other model dependent ACDs (Prokhorov et al., 1995; Prokhorov & Wunsch, 1997; van Kampen et al., 2006). Therefore it appears that there is a tradeoff between model independence and performance.

(Zhou et al., 2016) presents a variant of HDP that replaces the global model with an incremental model motivated by successful implementation of IADP controller in (Zhou et al., 2017) and (Zhou et al., 2018a). The incremental model structure has previously been successfully implemented to non RL aircraft controllers in the form of INDI controller as shown in simulated results of (Sieberling et al., 2010), (Simplício et al., 2013), and later in actual UAV flight testing in (Smeur et al., 2016). Additionally, Incremental Back Stepping (IBS) controller has been developed and have shown success in simulation (Acquatella et al., 2013).

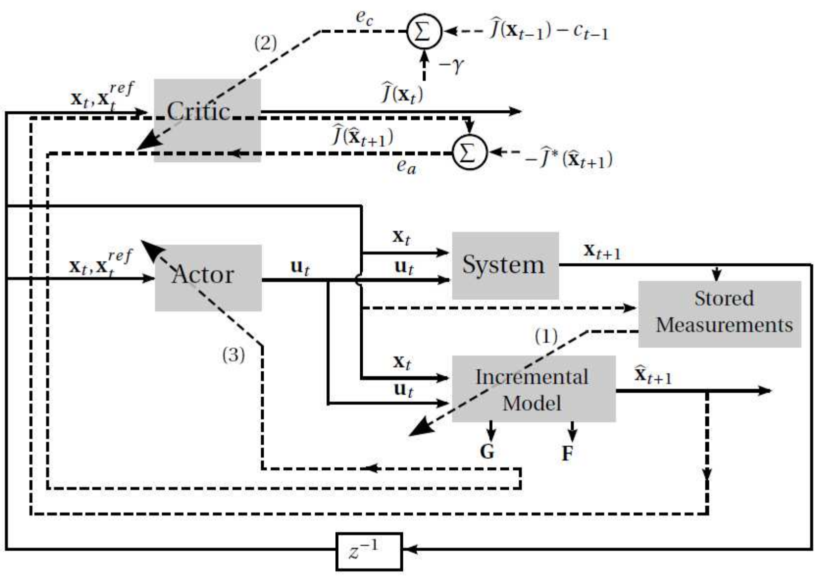


Figure 2-7: IHDP architecture showing forward paths and backpropagation paths. From Zhou, Y. (2016)

The Incremental Heuristic Dynamic Programming (IHDP) controller architecture is shown in Figure 2-7 taken from (Zhou et al., 2016). The figure shows that the global model typically seen in HDP has been replaced by an incremental model block. Incremental model stores linearized incremental models of the nonlinear system. The advantage of such structure is that offline training of the global model is no longer required as the incremental model can be identified online. Additionally, computational cost can be alleviated as a computationally expensive nonlinear function approximator such as ANN is no longer required. The resulting IHDP algorithm is therefore a fully online ACD while the computational complexity is reduced thanks to the use of Ordinary Least Squares (OLS) function approximation for the incremental model when compared to the original HDP algorithm.

The performance of IHDP is demonstrated in comparison to a baseline HDP algorithm for the angle of attack reference tracking task using a nonlinear simplified missile model. The actor block in both IHDP and HDP controllers are structured in cascaded actor networks form as shown in (R. Enns & Si, 2003). The HDP controller did not have an offline training phase. Figure 2-8 clearly shows that the IHDP was able to predict system states faster and more accurately than a global model ANN warranting its applicability in a fully online learning setting. Additional simulation results demonstrated that the IHDP controller converges faster and shows significantly lower trajectory tracking errors for varying initial conditions.

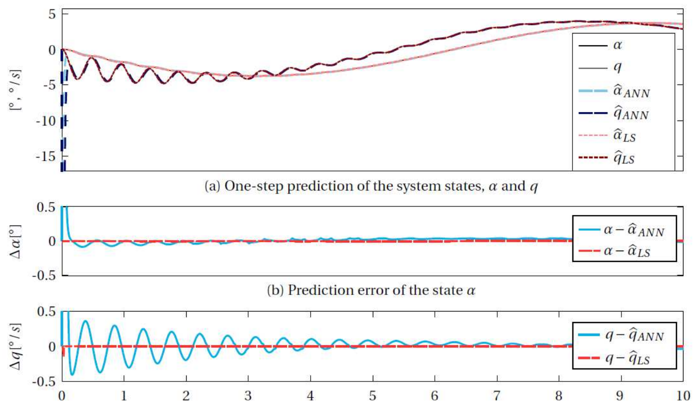


Figure 2-8: Time series simulation results comparing untrained HDP and IHDP. From Zhou, Y. (2016)

2-5-9 Online 2 State Longitudinal Aircraft Model Control Using Incremental Dual Heuristic Programming, 2018

As an extension of the incremental model implementation to HDP algorithm shown in (Zhou et al., 2016), incremental model was implemented to replace the global system model for DHP algorithm in (Zhou et al., 2018b). The resulting IDHP architecture is shown in Figure 2-9. Change has been made to the function approximation method for the identification of the incremental model where OLS method was replaced by RLS method. By utilizing RLS as incremental model approximator, the dependence on persistent excitation has been reduced when compared to the piecewise OLS method used in (Zhou et al., 2016). Additionally, the natural recursive nature of RLS method does not introduce additional computational complexity.

Results from IDHP controller implementation to control a nonlinear simplified missile model for the task of trajectory tracking show that IDHP outperforms DHP overall. For the task of reference tracking, IDHP showed significantly faster convergence to reference and lower overall tracking error. When tested for online adaptability by introducing sudden change in system dynamics, IDHP again showed quick adaptation and stable tracking while DHP failed to control the missile. To further validate the robustness of IDHP, the author implemented sensor measurement noise to both states (no noise was added to the elevator deflection term). The results showed that although low amplitude high frequency oscillations can be found from simulated angle of attack time series data, the overall tracking accuracy is well maintained.

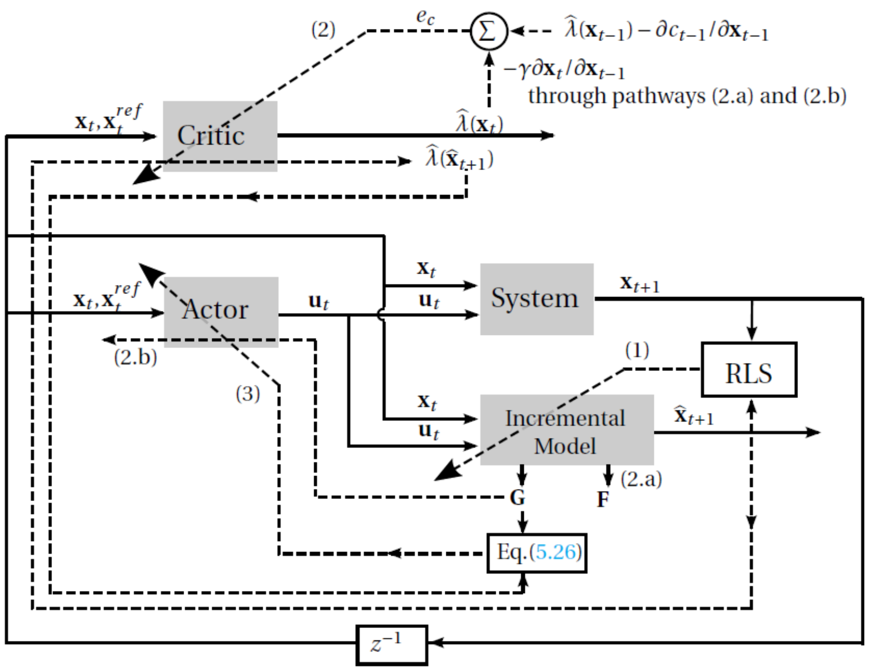

(2)
e_c
Σ
$\hat{\lambda}(\mathbf{x}_{t-1}) - \partial c_{t-1}/\partial \mathbf{x}_{t-1}$
$-\gamma \partial \mathbf{x}_t / \partial \mathbf{x}_{t-1}$
through pathways (2.a) and (2.b)
$\mathbf{x}_t, \mathbf{x}_t^{ref}$
Critic
$\hat{\lambda}(\mathbf{x}_t)$
$\hat{\lambda}(\hat{\mathbf{x}}_{t+1})$
$\mathbf{x}_t, \mathbf{x}_t^{ref}$
Actor
$\mathbf{u}_t$
$\mathbf{x}_t$
$\mathbf{u}_t$
System
$\mathbf{x}_{t+1}$
(2.b)
(3)
(1)
RLS
$\mathbf{x}_t$
$\mathbf{u}_t$
Incremental Model
$\hat{\mathbf{x}}_{t+1}$
G
F
(2.a)
Eq.(5.26)
z^{-1}

Figure 2-9: IDHP architecture showing both forward paths and backpropagation paths. From Zhou, Y. (2018)

When comparing IHDP to IDHP, the author found that the results are consistent with findings demonstrated in (Prokhorov et al., 1995), (Prokhorov & Wunsch, 1997), and (Venayagamoorthy et al., 2002), where DHP showed better overall performance when compared to HDP. IDHP outperformed IHDP overall in terms of success rate, overall accuracy, and wider initial conditions allowed.

2-5-10 ACD Application Case Overview

After reviewing multiple ACD implementations for the aircraft control task, two main limitations of the current state of ACD aircraft controllers can be identified. The first limitation is that there are no fully online ACD aircraft controllers that have been successfully implemented to the high fidelity nonlinear model of an aircraft. Fully online ACD mainly refers to the need to train the model ANN offline. When considering ACD controller implementation to an actual aircraft. This directly translates to the need to train the ACD controller with an aircraft model offline. This is problematic as an accurate model of an aircraft is costly to obtain as multiple flight tests throughout the flight envelope are required. The closest implementation case of ACD that resembles a fully online ACD controller is demonstrated in (R. Enns & Si, 2003) where ADHDP had been used to control a high fidelity nonlinear model of an Apache helicopter. However, due to the use of a offline trained trim network incorporated within the ACD controller which the authors describe as "paramount to achieving success", prevents the implemented algorithm to be fully online. The second limitation is the robustness of the ACD controllers implemented for aircraft control. In aircraft control, control divergence directly leads to possible hazardous situations. As most implementations of ACD controllers have been applied to a simplified aircraft models, no conclusive remarks can be made about the success rate of the algorithms presented when implemented on a high fidelity model or an actual aircraft as the number of states and increase along with error. However, each ACD controller implementations have shown design choices that contribute improve the overall performance and robustness. Design choices such as cascaded actor networks shown in (R. Enns & Si, 2003), (van Kampen et al., 2006), (Zhou et al., 2016), and (Zhou et al., 2018b) or RRPROP backpropagation algorithms utilized in (Ferrari & Stengel, 2002) and (Ferrari & Stengel, 2004) are good examples of such attempts.

As for the choice of ACD that is the most suitable for implementation to the full Cessna Citation model, the following line of reasoning is presented. The ACDs are evaluated in terms of their potential for fully online learning, adaptiveness to the changing environment, and stability that demonstrates the controller's ability to converge to the reference and not diverge thereafter. Of the ACD variants implemented as aircraft controllers, only IHDP and IDHP show potential for full online controller implementation. ACD controllers are fundamentally adaptive controllers with ACD variants differing only in terms of the extent to which it can adapt. (Zhou et al., 2018b) show that IDHP achieve higher online adaptability and stability along with higher accuracy, efficiency, and learning capability. Therefore it can be concluded that IDHP is the most suitable controller candidate for implementation in the high fidelity Cessna Citation model.

2-6 State-of-the-art research concerning IDHP implementation to Cessna Citation

(Heyer, 2019) presents an implementation of a IDHP variant framework to a high fidelity 6 DOF Cessna Citation model with both major and minor changes to the IDHP framework originally proposed in (Zhou et al., 2018b). The author points out the inherent learning instability present in the original IDHP framework which is unacceptable for its application for aircraft control. This leads to the major change made to the baseline IDHP framework: a separate target critic network. The additional target critic network with soft target updates, inspired from the works of (Lillicrap et al., 2015), is aimed to improve learning stability by restricting the target value of the critic to change slowly. The effect of this change is moving the unstable problem of learning the value function closer to the case of supervised learning where robust solutions exist (Lillicrap et al., 2015). This approach differs from the dual-critic HDP implementation done in (Ni, He, & Wen, 2013) where a second reference critic network was employed for internal goal representation and to help approximate the cost-to-go function in detail. As for the ANN setup, a classical single hidden layer fully connected ANNs with tanh activation functions have been used for the actor and critic blocks while a combination of tanh and linear activation functions have been used for the target critic network. To implement actuator limitation, the actor output layer functions have been scaled accordingly to respective actuators. For the RL reward setup, the author uses the standard cost minimization model upper bounded by zero and calculated by the reference error scaled by a weight matrix. For the incremental model implementation, a standard RLS update process has been employed. The training strategy is identical to (Si & Wang, 2001) where the algorithm simply waits until next time step measurements are available for the update instead of utilizing the prediction made by the model. Of the available engine/actuator dynamics and the sensor models from the high fidelity Cessna Citation model apart from its main model, only the former engine/actuator dynamics have been included. Finally, the author utilizes two separate IDHP frameworks for the longitudinal and latitudinal control with respective outer loops controlled by a standard PID controller.

Empirical analysis of time series response data obtained under different initial flight conditions for the pitch and roll rate reference tracking is conducted. The time series response data obtained over 100 separate runs with initial V_{TAS} of $90m/s$ and altitude of $2000m$ are presented in the form of minmax plots in Figure 2-10. It shows that good overall tracking performance is obtained except for pitch rate tracking when the reference pitch rate changes rapidly. The author reports that a success rate of 100% had been achieved for 100 separate simulation runs under 4 different initial flight conditions.

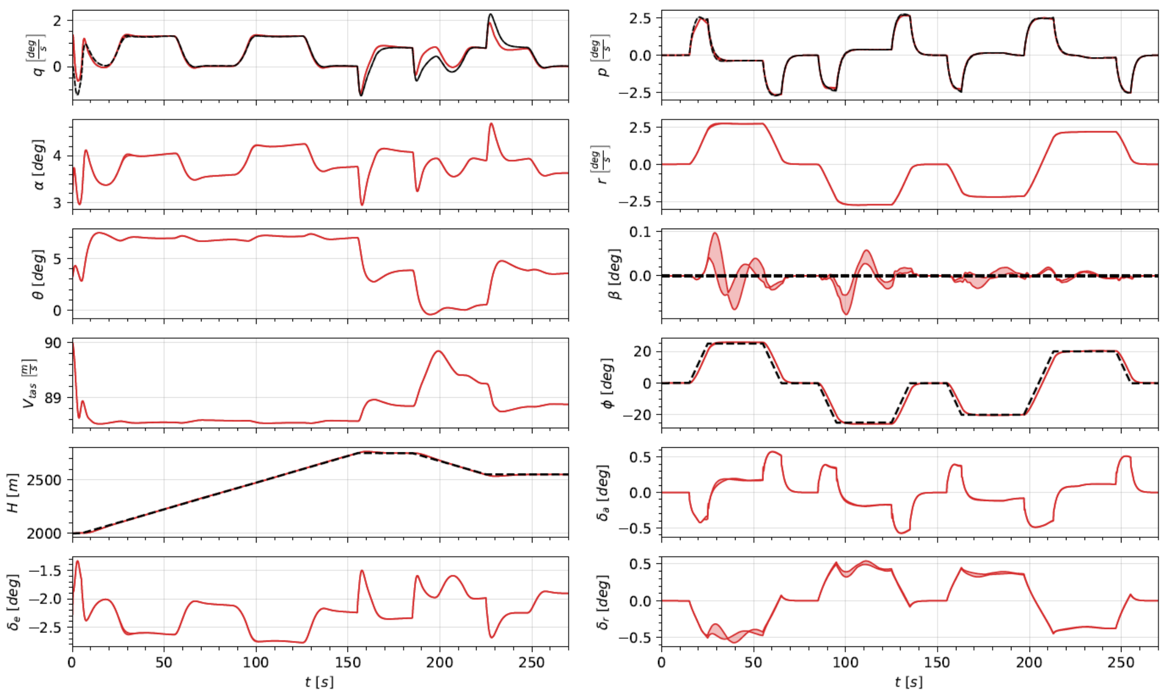


Figure 2-10: Time response data of 100 separate runs for reference tracking task with initial conditions of V_{TAS} of $90m/s$ and $H = 2000m$. From Heyer, S. (2019)

Another approach of implementing IDHP controller framework for the Cessna Citation model control is shown in (Kroezen, 2019). Identical to the research completed in (Heyer, 2019), the author does not incorporate any sensor noise or actuator delay into the simulation and uses a target critic. For the design layout of the implemented actor-critic network, longitudinal and latitudinal actors are separated for respective aircraft rate control. The structure differs from the completely separated longitudinal and latitudinal rate control layout shown by (Heyer, 2019). Because the critic is shared by both longitudinal and lateral actor networks, more motion couplings are anticipated to be represented by the controller. What is noteworthy however is the use of Rectified Linear units (ReLu) hidden layer activation functions for all actor/critic neural networks. The author states that unlike the standard sigmoid functions, ReLu functions do not suffer from vanishing gradients with the additional benefit of reduced computational complexity due to it being a linear function. However, as the author himself states, the network needs to be larger and deeper to mitigate the discontinuity seen at the network output. This is due to the inherent characteristic of ReLu activation function. ReLu function boasts sparse activation adding to its lighter computational load. However, once the activation functions reach zero, it will be stuck and become unresponsive to any future gradient descent. Therefore, more neurons need to be added to mask the discontinuity. This also means that its advantage of reduced computational complexity is diminished by the shear increase in the number of activation functions required. Regarding to its other benefit that ReLu functions do not suffer from vanishing gradients, this also means that the activation functions are not bounded. Unbounded activation functions can ultimately destabilize the entire actor critic network and should be avoided if possible.

In (Kroezen, 2019), simulation results are available for the task of altitude and bank angle reference tracking at 4 different initial airspeed and altitude flight conditions. The results showed an average of 90% success rate for all flights conditions. Comparison study has also been done to compare the IDHP controller performance with and without a target critic network. Results showed that higher success rate can be achieved with the addition of a target critic network for all flights conditions.

Both (Heyer, 2019) and (Kroezen, 2019) have contributed to advanced the continuing research of implementing IDHP to the Cessna Citation aircraft. Common finding between the two research is that a target critic can be implemented to provide a slower, but a more robust policy update. In (Kroezen, 2019), the effect of such target critic in implementation to the Cessna Citation model has been demonstrated showing increased success rates. Different approaches have been taken as to where the separation of longitudinal and latitudinal motions and what activation functions are used within the actor/critic artificial neural networks. The effect of these differences need to be analyzed to further the research of implementing IDHP to the Cessna Citation aircraft. Additionally, both implementations utilized PID controllers for outer loop control to generate inner loop rate reference. Therefore it cannot be considered that the either IDHP controller implementations are fully online. Two concluding remarks can be made from the analysis of state-of-the-art IDHP controller implementations to high fidelity Cessna Citation model. First remark is that the crucial next step is removing the perfect sensor/actuator measurements assumption as well as incorporating external disturbances to the simulation. As it is clear that sensor/actuator measurement bias, noise, delay, and asynchronous sampling time are to be anticipated in real life testing, research needs to be conducted to ensure that the IDHP controller is able to cope with these adverse conditions. Second remark is that both IDHP implementations cannot be considered to be fully online as both controller structures include PID outer loop control that require offline tuning. Therefore it can be said that the performance of the IDHP controller has only been validated for the Cessna Citation model rate control without sensor noise or external disturbances.

Chapter 3

Preliminary Analysis

3-1 Introduction to Preliminary Analysis

At the end of the literature survey, it has been found that Incremental Dual Heuristic Dynamic Programming IDHP is the most favorable baseline design architecture for developing an online, adaptive, and stable RL controller for the Cessna Citation model simulation. This preliminary analysis serves as a feasibility check for the implementation of IDHP controller. A simplified Cessna Citation model is used as a testbed for the IDHP controller development. The results presented in the preliminary analysis are meant to provide an indication of the online stability and performance of IDHP controller as well as identify the implemented controller's shortcomings. Discussion of obtained results will generate points of consideration for applying IDHP controller when the simulation fidelity is increased.

3-2 Simulation Environment

Environment in a RL setting refers to all other elements apart from the agent which in this case is the IDHP controller. This section presents and justifies the simulated environment.

3-2-1 Short Period LTI Cessna Citation model

To show feasibility of IDHP controller to control an aircraft, a reduced model of the Cessna Citation aircraft is utilized as the plant to be controlled. The reduced Cessna Citation model, henceforth referred to as reduced model or plant, contains the longitudinal short period dynamics of the Cessna Citation aircraft. It has been linearized, and is time invariant. An aircraft is a highly nonlinear system. Although the reduced model is LTI, aircraft dynamics can be defined by a smooth state space where no abrupt changes in dynamics occur, an LTI model is deemed to represent the aircraft dynamics as long as the states are within relative closeness trim point. Additionally, the simplicity of the reduced model allows quick controller development in terms of verification and parameter tuning.

The reduced model is derived from (Mulder et al., 2013) and its state space form is shown in Eq. 3-1. The reduced model is derived from the Cessna Ce500 Citation in cruise condition at $V = 59.9m/s$. As it can be seen in Eq. 3-1, the reduced short period model contains two states and one control action: angle of attack α, pitch rate q, and elevator deflection δ_e.

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} -0.7341 & 0.9744 \\ -1.4537 & -1.5562 \end{bmatrix} \begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} -0.0887 \\ -6.6327 \end{bmatrix} \delta_e \tag{3-1}$$

The IDHP controller is an online RL algorithm. Therefore within the simulation, the reduced model is utilized to represent the plant to be controlled. No a-priori knowledge is assumed or utilized about the plant except for its inputs and outputs.

3-2-2 Simulated Control Task

The reduced model is used to represent the plant to be controlled. The simulated control task is an angle of attack reference tracking task where reference angle of attack to be followed is generated according to Eq. 3-2. The amplitude of the reference signal is 10 deg and its period is 20 seconds. A sinusoidal reference signal is chosen as it continuously and smoothly explores the α state space. The simulation is performed at 50 Hz.

$$\alpha_{ref} = \frac{10\pi}{180} sin(0.1\pi t) \tag{3-2}$$

Pitch rate tracking is expected to generate better performance as the aircraft dynamics dictates that a more direct relationship can be drawn between the pitch rate and the angle of attack. However in this simulated control task, only the angle of attack is tracked for two reasons. First reason is to demonstrate that the IDHP controller is able to control the reduced model when a more indirect task is presented. Second reason is to roughly compare the performance of implemented IDHP controller to those performed in previous studies utilizing cascaded actor networks or pitch rate reference tracking.

3-3 IDHP Implementation

In this section each elements within the implemented IDHP controller architecture are presented. IDHP as presented in (Zhou et al., 2018b) is used as reference architecture and research performed in (van Kampen et al., 2006) is used for specific design choices including reward shaping and actor/critic input/output choices.

3-3-1 IDHP baseline architecture

The baseline IDHP controller architecture overview is given in Figure 3-1 with each elements showing only its inputs and outputs. Black lines represent input or output of current timestep variables and red lines represent one step ahead variables. The general structure illustrates that current timestep states are sent to the actor block to determine control action which is then sent to the plant to obtain one step ahead states while being sent to the utility function block to calculate the utility gradient. One step ahead states obtained from the plant are then used to calculate the cost function gradient λ_{t+1} in the critic block and update the incremental model of the plant along with the control input. Additionally, note that the critic takes both the current states and the one step ahead states to calculate λ_t and λ_{t+1} separately in two forward calculations.

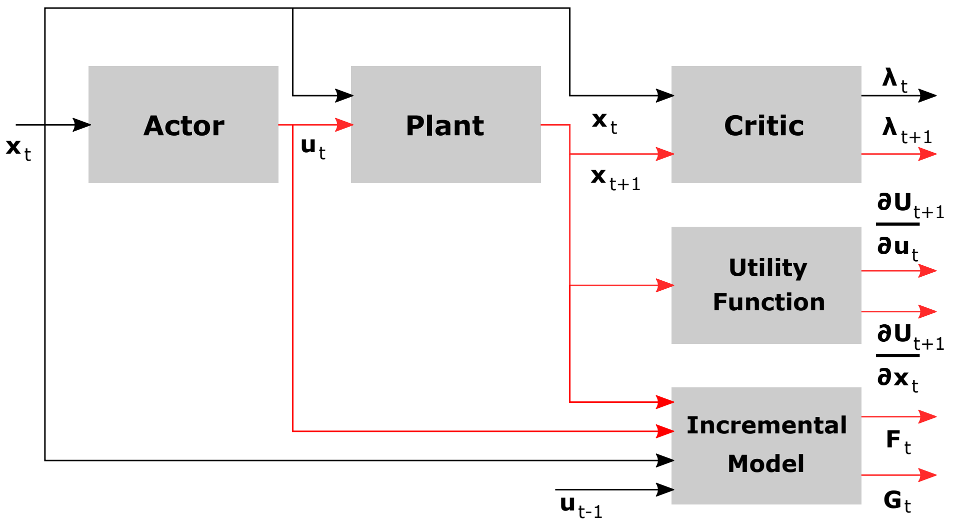


Figure 3-1: IDHP architecture showing input output relationships of all elements

3-3-2 Incremental Model

The main feature within IDHP architecture that distinguishes it from other DHP architectures is the use of an incremental model replace the need for a global plant model in DHP. Given sufficient sampling rate and relatively slow varying system dynamics, the current identified linearized model can be said to be time varying and is adequately representative for use in DHP (Zhou et al., 2016, 2018a).

Consider a discrete system defined by Eq. 3-3 where f is a system dynamics function. Eq. 3-4 shows that when the sampling rate is high enough, the nonlinear system dynamics around $\mathbf{x}_t$ can be approximated by the first-order Taylor expansion where $\mathbf{F}_{t-1} = \frac{\partial f(\mathbf{x}_{t-1},\mathbf{u}_{t-1})}{\partial \mathbf{x}_{t-1}}$ is the system transition matrix and $\mathbf{G}_{t-1} = \frac{\partial f(\mathbf{x}_{t-1},\mathbf{u}_{t-1})}{\partial \mathbf{u}_{t-1}}$. Eq. 3-4 can be rearranged to an incremental form as shown in Eq. 3-5.

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) \tag{3-3}$$

$$\mathbf{x}_{t+1} \approx \mathbf{x}_t + \mathbf{F}_{t-1}(\mathbf{x}_t - \mathbf{x}_{t-1}) + \mathbf{G}_{t-1}(\mathbf{u}_t - \mathbf{u}_{t-1}) \tag{3-4}$$

$$\Delta\mathbf{x}_{t+1} \approx \mathbf{F}_{t-1}(\Delta\mathbf{x}_t) + \mathbf{G}_{t-1}(\Delta\mathbf{u}_t) \tag{3-5}$$

RLS method is used to efficiently approximate **F** and **G** matrices. OLS method needs to perform matrix inversion at every time step which is costly and unreliable if there are not enough excitation in the signals. The latter is the problem of persistent excitation which is addressed by implementing RLS method which only needs to perform matrix inversion once and therefore efficiently and reliably updates the approximation. Note that the "Incremental Model" block seen in Figure 3-1 represents the constantly updated incremental model with RLS parameter estimator included.

The parameter matrix Θ structure and regression matrix A structure for RLS identification are given as implemented. First, **F** and **G** matrices are defined as implemented within this preliminary analysis in Eqs. 3-6 and 3-7. The parameter matrix $\hat{\Theta}$ can be constructed using **F** and **G** matrices as shown in Eq. 3-8. The regression matrix A is constructed as shown in Eq. 3-9 with each column representing a measurement series of state and input increments. With the parameter matrix and regression matrix defined, RLS approximation method implemented in this preliminary analysis following the standard procedure with a forgetting factor of $\mu = 0.8$ and an initial $\hat{\Theta}$ estimation performed with 6 measurements.

$$\mathbf{F}_{t-1} = \begin{bmatrix} \frac{\partial \alpha_{t-1}}{\partial \alpha_{t-1}} & \frac{\partial \alpha_{t-1}}{\partial q_{t-1}} \\ \frac{\partial q_{t-1}}{\partial \alpha_{t-1}} & \frac{\partial q_{t-1}}{\partial q_{t-1}} \end{bmatrix} \tag{3-6}$$

$$\mathbf{G}_{t-1} = \begin{bmatrix} \frac{\partial \alpha_{t-1}}{\partial \delta_{e,t-1}} \\ \frac{\partial q_{t-1}}{\partial \delta_{e,t-1}} \end{bmatrix} \tag{3-7}$$

$$\mathbf{\Theta}_{t-1} = \begin{bmatrix} \mathbf{F}_{t-1}^T \\ \mathbf{G}_{t-1}^T \end{bmatrix} \tag{3-8}$$

$$\mathbf{A} = \begin{bmatrix} \Delta\boldsymbol{\alpha} & \Delta\mathbf{q} & \Delta\boldsymbol{\delta}_e \end{bmatrix} \tag{3-9}$$

3-3-3 Utility Function

The goal of any reinforcement learning controller is to minimize the cost function J. In the case of IDHP, this goal is represented by the critic approximating the gradient of the cost function λ_t defined in Eq. 3-10 and the actor adapting its policy by moving towards the path of minimizing cost function. The critic approximates the true cost function gradient by minimizing the TD error defined in Eq. 3-11. Eq. 3-11 can be rearranged to Eq. 3-12 with $\lambda_{t+1} = \frac{\partial J(x_{t+1})}{\partial x_t}$ and $\lambda_t = \frac{\partial J(x_t)}{\partial x_t}$.

$$\lambda_t = \frac{\partial J_t}{\partial x_t} \tag{3-10}$$

$$e_c = \frac{\partial (J_t - U_t - \gamma J_{t+1})}{\partial x_t} \tag{3-11}$$

$$e_c = \lambda_t - \frac{\partial U_t}{\partial x_t} - \gamma \lambda_{t+1} \frac{\partial x_{t+1}}{\partial x_t} \tag{3-12}$$

The Utility function block as seen in Figure 3-1 calculates the partial derivative of the utility function with respect to the states required for the critic block training as can be seen in Eq. 3-12. In any RL optimal control task, utility function shaping is an important design choice that dictates the performance of the RL controller. In this implementation of IDHP controller for reference angle of attack tracking task, a simple utility function is defined as a gain multiplied error function between the reference angle of attack and the simulated angle of attack. The motivation for such utility function is that the IDHP implementation in this preliminary analysis only tracks the angle of attack reference. The chosen utility function is shown in Eq. 3-13 with a constant κ. The gain κ can be used to modify the aggressiveness of the control policy. With the utility function defined, $\frac{\partial U_t}{\partial x_t}$ can be simply calculated using Eqs. 3-14 and 3-15.

$$U_t = \kappa \Big(\alpha_{ref,t} - \alpha_t \Big)^2 \tag{3-13}$$

$$\frac{\partial U_t}{\partial \alpha_t} = -2\kappa \Big(\alpha_{ref,t} - \alpha_t \Big) \tag{3-14}$$

$$\frac{\partial U_t}{\partial q_t} = 0 \tag{3-15}$$

3-3-4 Critic

The Critic function block's purpose is to estimate the cost function gradient. An ANN function approximator is used with its structure shown in Figure 3-2. It is a fully connected one hidden layer ANN with linear input and output activation functions and tanh hidden layer activation functions. The ANN is initialized with random weights: $\mathbf{W}_{c,init} \subset [-0.1, 0.1]$. Such interval has been chosen for initial critic weights to minimize the effect of initial critic weights choice on the approximated cost function gradient. The tanh activation functions used for the hidden layer show some desirable qualities when compared to other activation functions. It is a differentiable nonlinear activation function that behaves similarly as the linear activation function except when it reaches $y = -1$ or $y = 1$ activation to which it smoothly converges to. This bounds each activation functions alleviating the effect of activation functions to ANN divergence. Additionally, when compared to the similarly shaped sigmoidal activation function bounded and converging to 0 and 1 and therefore outputs zero when a large negative input is given, there is a lower chance that the activation function becomes stuck at 0 activation.

The input to the critic ANN is the error between the reference angle of attack and the simulated angle of attack as shown in Eq. 3-16. Although it is possible to use states as additional inputs to the critic ANN, results from (van Kampen et al., 2006) and (Kroezen, 2019) showed that for the reduced model, simply the error signal would suffice as input to the critic block.

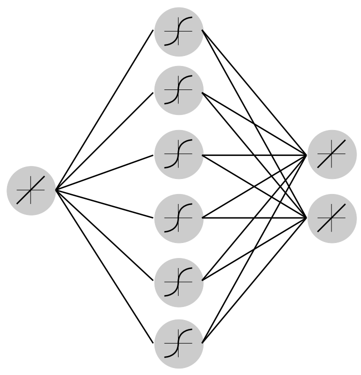


Figure 3-2: Critic Neural Network Structure

$$c_{in} = \alpha_{ref} - \alpha \tag{3-16}$$

Critic is trained using the TD error $\mathbf{e}_{c,t}$ for the DHP critic previously defined in Eq. 3-12. λ_t and λ_{t+1} can be directly obtained as the output of critic at different time steps. $\frac{\partial U_t}{\partial x_t}$ term is also directly calculated by the utility function block. The only unknown term $\frac{\partial x_{t+1}}{\partial x_t}$ can be represented as a summation of two partial derivatives of the system with respect to states and input as shown in Eq. 3-17. This can then be rearranged to Eq. 3-18 using $\mathbf{F}$ and $\mathbf{G}$ matrices previously defined in Eqs. 3-6 and 3-7. Because both $\mathbf{F}_t$ and $\mathbf{G}_t$ can been approximated by the incremental model, the only remaining unknown is the $\frac{\partial u_t}{\partial x_t}$ term which can be obtained from the actor ANN.

$$\frac{\partial x_{t+1}}{\partial x_t} = \frac{\partial f(\mathbf{x}_t, \mathbf{u}_t)}{\partial x_t} + \frac{\partial f(\mathbf{x}_t, \mathbf{u}_t)}{\partial u_t}\frac{\partial u_t}{\partial x_t} \tag{3-17}$$

$$\frac{\partial x_{t+1}}{\partial x_t} = \mathbf{F}_t + \mathbf{G}_t \frac{\partial u_t}{\partial x_t} \tag{3-18}$$

With all terms necessary to calculate the TD error $e_{c,t}$ known, the critic ANN can be trained by minimizing the error measure defined in Eq. 3-19. Therefore the following weight update rule shown in Eq. 3-20 can be derived where $\Delta W_{C,t}$ is defined in Eq. 3-21 with η_c as the learning rate parameter.

$$E_{c,t} = \frac{1}{2}\mathbf{e}_{c,t}^T \mathbf{e}_{c,t} \tag{3-19}$$

$$\mathbf{W}_{c,t+1} = \mathbf{W}_{c,t} + \Delta\mathbf{W}_{c,t} \tag{3-20}$$

$$\Delta\mathbf{W}_{c,t} = -\eta_c \frac{\partial E_{c,t}}{\partial \lambda_t}\frac{\partial \lambda_t}{\partial W_{c,t}} = -\eta e_{c,t}\frac{\partial \lambda_t}{\partial W_{c,t}} = -\eta \mathbf{e}_{c,t}^T \frac{\partial \lambda_t}{\partial W_{c,t}} \tag{3-21}$$

3-3-5 Actor

The actor block stores the policy of the IDHP controller mapping states with respect to the action to be taken. Ultimately, the actor's aim is to store optimal action that would lead to cost function minimization. Similar to the critic, the actor is represented by a fully connected ANN with one input node, one hidden layer, and one output node. Identical to the critic, the input to the actor ANN is the error between the reference angle of attack and the simulated angle of attack. The ANN structure for the Actor block is shown in Figure 3-3. The actor ANN weights are initialized randomly in $\mathbf{W}_{c,init} \subset [-0.1, 0.1]$. The interval choice for actor weight initialization is to minimize the effect of initial weights on the derived control action. The main difference between the critic and the actor ANNs is that a scaled tanh activation function has been used for the output layer. This is to ensure that the generated action stays bounded by physical limitation of the elevator while penalizing extreme actions from being generated.

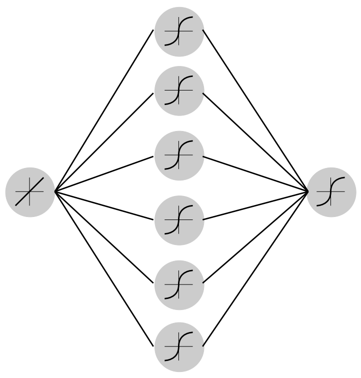


Figure 3-3: Actor Neural Network structure

In order to train the actor ANN to generate optimal action, the optimal control action defined previously in Eq. 2-28 can be utilized. Using this definition, the actor weight gradient vector shown in Eq. 3-22 can be derived. This weight gradient is then used for the standard gradient descent weight update rule defined in Eq. 3-23 for the actor ANN. Observing Eq. 3-22, it can be seen that the $\frac{\partial U_t}{\partial \mathbf{u}_t}$ term can be directly found from the utility function block which in this case is 0 as the utility function does not depend on the input. λ_{t+1} is directly estimated by the critic block and has already been used for critic training. The remaining unknown term $\frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{u}_t}$ can be numerically estimated as shown in Eq. 3-24. An approximation of $\mathbf{G}_t$ is available from the incremental model. The actor training procedure has been explained in detail. However it needs to be noted that as the author points out in (van Kampen et al., 2006), because only an approximation $\mathbf{G}_t$ is available from the incremental model, any errors introduced during incremental model identification will affect actor network.

$$\Delta \mathbf{W}_{a,t} = -\eta_a \Big(\frac{\partial U_t}{\partial \mathbf{u}_t} + \gamma \lambda_{t+1} \frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{u}_t} \Big) \frac{\partial \mathbf{u}_t}{\partial W_{a,t}} \tag{3-22}$$

$$\mathbf{W}_{a,t+1} = \mathbf{W}_{a,t} + \Delta \mathbf{W}_{a,t} \tag{3-23}$$

$$\frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{u}_t} \approx \frac{\partial f(\mathbf{x}_t, \mathbf{u}_t)}{\partial \mathbf{u}_t} = \mathbf{G}_t \tag{3-24}$$

3-3-6 IDHP schematic

With each blocks present within the IDHP architecture described in detail, a complete IDHP controller schematic is presented in Figure 3-4. For critic backpropagation, Eq. 3-12 showing critic error signal can be reformulated to Eq. 3-25. Similarly, actor backpropagation weight gradient Eq. 3-22 can be reformulated to Eq. 3-26.

$$e_c = \lambda_t - \frac{\partial U_t}{\partial x_t} - \gamma\lambda_{t+1}\Big(\mathbf{F}_t + \mathbf{G}_t\frac{\partial u_t}{\partial x_t}\Big) \tag{3-25}$$

$$\Delta\mathbf{W}_{a,t} = -\eta_a\Big(\frac{\partial U_t}{\partial \mathbf{u}_t} + \gamma\lambda_{t+1}\mathbf{G}_t\Big)\frac{\partial \mathbf{u}_t}{\partial W_{a,t}} \tag{3-26}$$

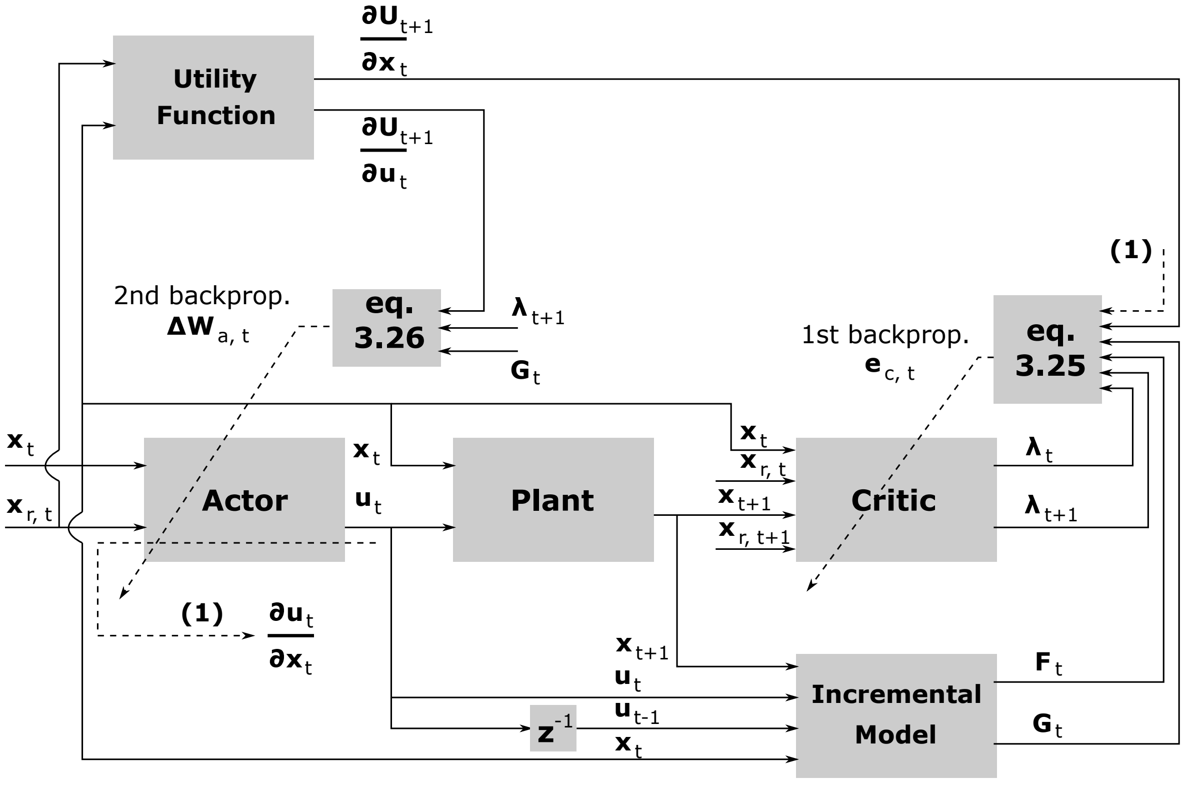


Figure 3-4: Detailed IDHP controller schematic

3-4 Results

IDHP controller has been developed for the angle of attack reference tracking task. This section is divided into two main parts and additional results obtained. In the first main part, the results obtained from a single representative run for the period of 200 seconds are presented and discussed. In the second main part, the results obtained from 500 independent runs of the implemented IDHP controller are presented and analyzed. Additionally, a comparison of results obtained from the preliminary analysis to results obtained by another is presented.

3-4-1 Results from a single run

Results obtained from a single representative run is presented in Figures 3-5 to 3-15. As it will be shown later, a single run is adequately representative of the overall characteristic of the implemented IDHP controller.

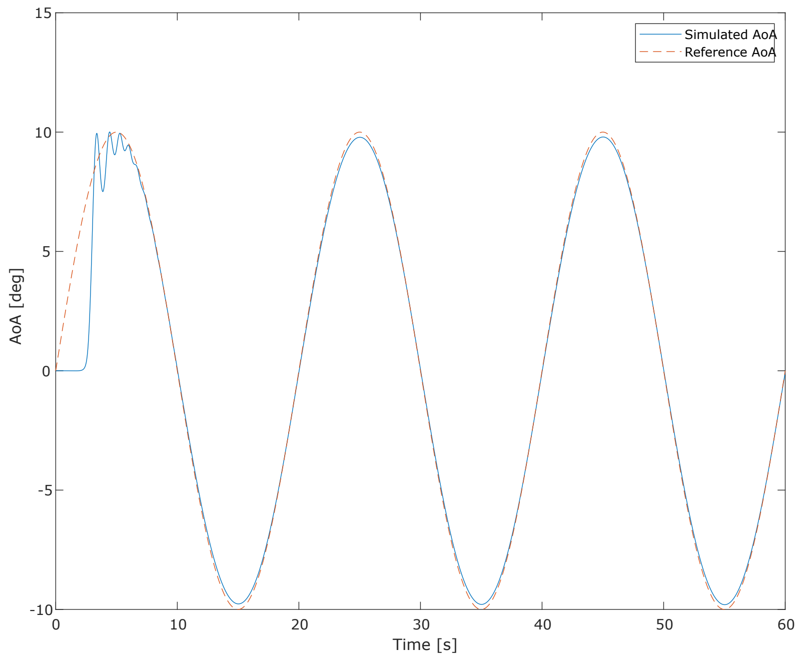


Figure 3-5: Simulated angle of attack signal of controlled system with reference angle of attack signal

Figure 3-5 shows the angle of attack reference tracking performed by the implemented IDHP controller. As it can be seen, the controller begins to track the reference signal after 2.5 seconds has passed. This is due to the fact that both critic and actor training are initialized at $[-0.1, 0.1]$ and initial gradient descent takes some time to diverge away from the initial weights. This can be clearly seen in Figures 3-12 to 3-15 where the input and output ANN weights of both critic and actor ANNs only begin to show divergence away from initial weights at around 1 second and 1.5 second respectively. The time differene between the critic and actor divergence can be explained by the fundamental property of GPI where a cost function J is utilized to determine optimal policy. It should be understood that the cost function J, of which its gradient is estimated by the critic, needs to be approximated first before the actor can find the optimal policy.

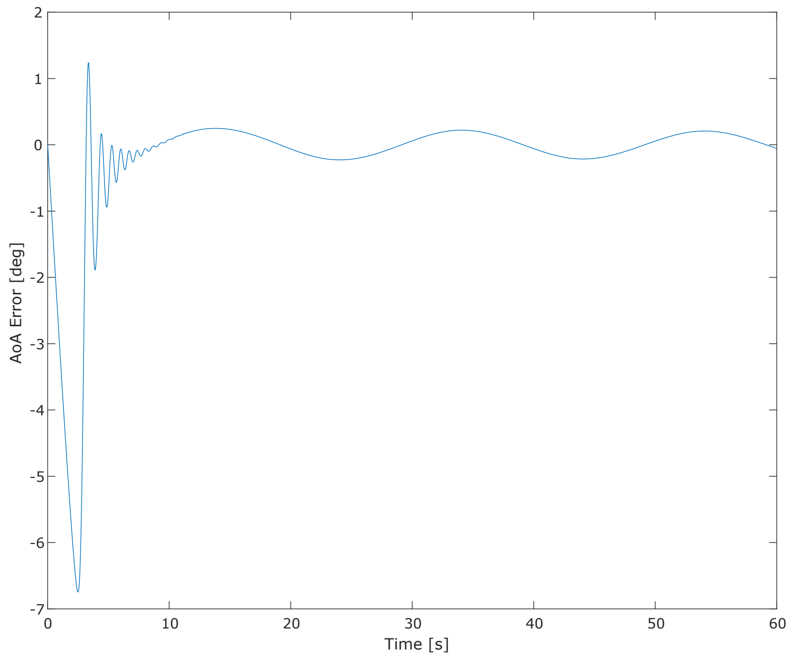


Figure 3-6: Angle of attack error of simulated simulated signal to the reference signal

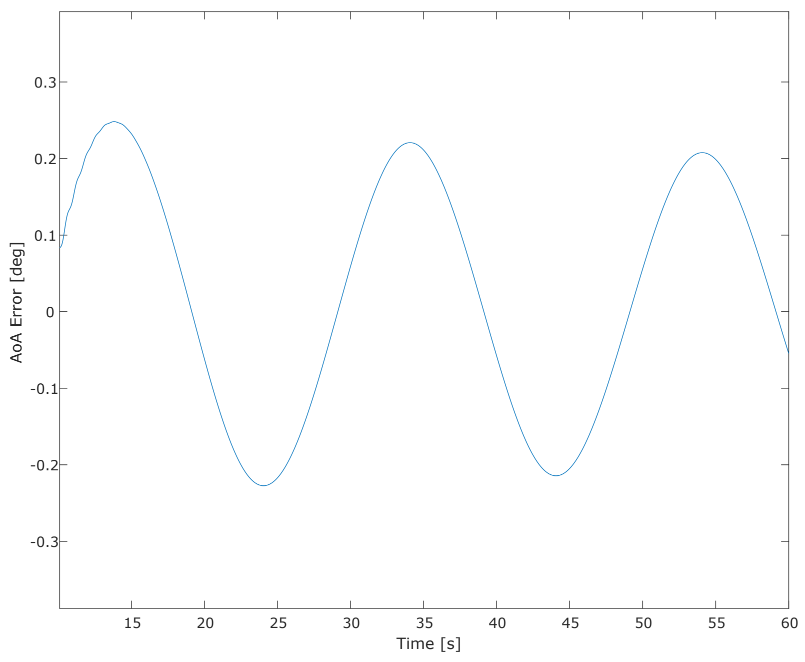


Figure 3-7: Angle of attack error of simulated simulated signal to the reference signal shown from 10 seconds

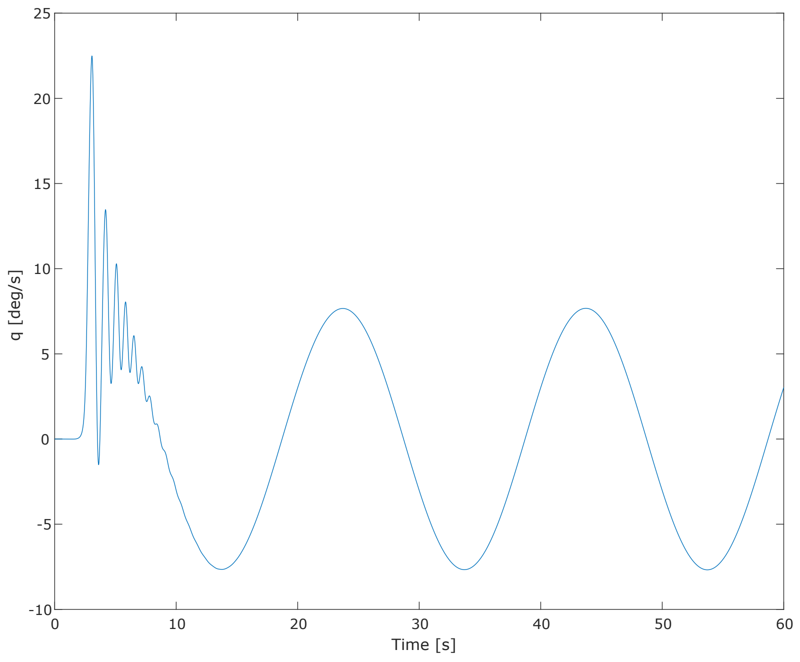


Figure 3-8: Simulated pitch rate signal of controlled system

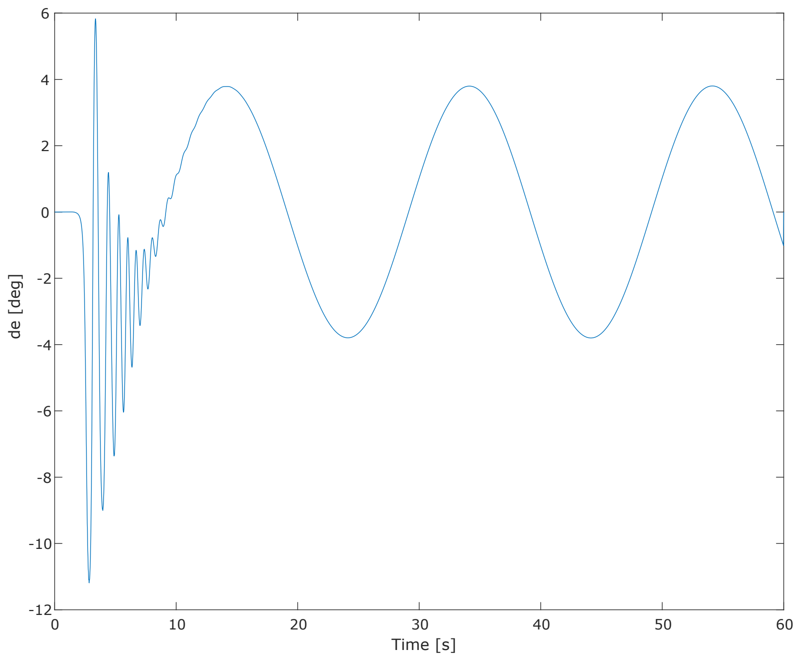


Figure 3-9: Simulated elevator deflection signal of controlled system

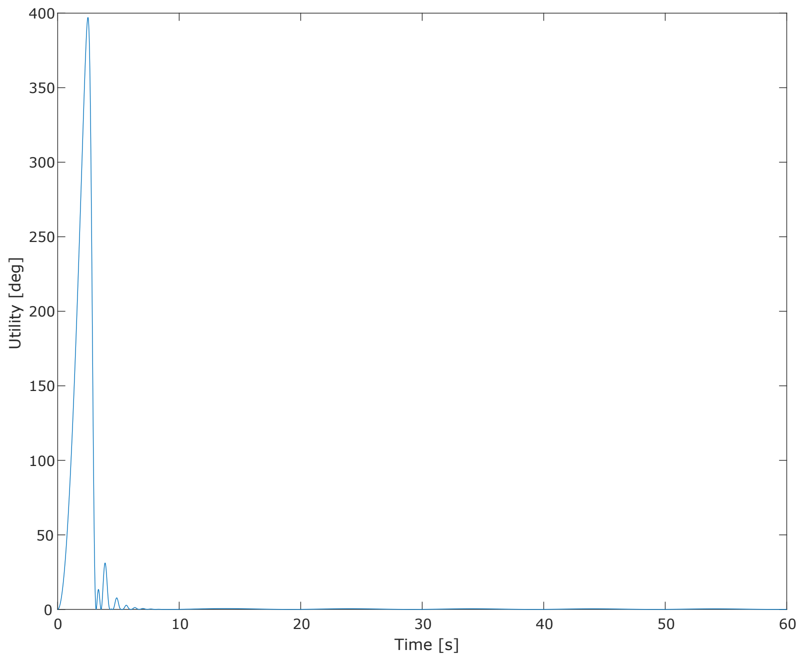


Figure 3-10: Utility over time

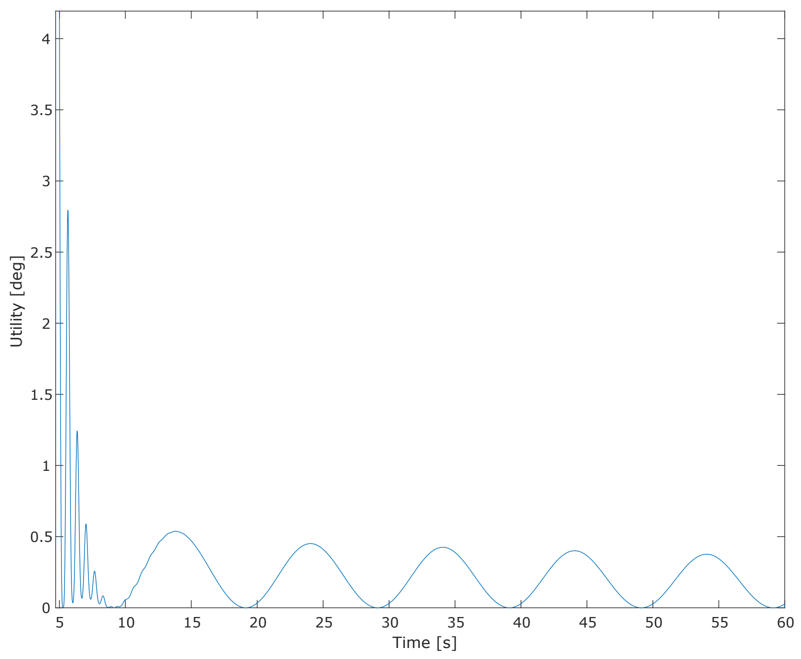


Figure 3-11: Utility over time after 5 seconds

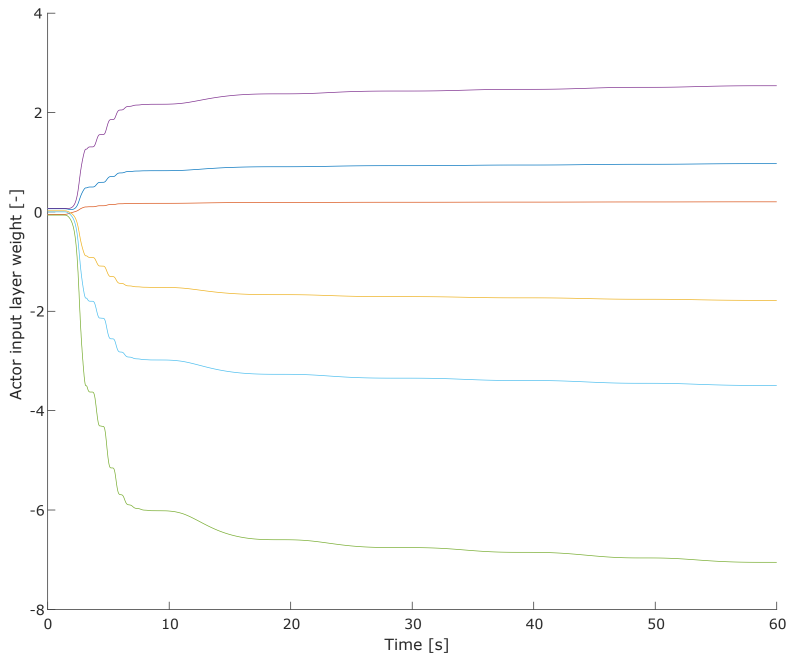


Figure 3-12: Actor ANN input layer weights progression

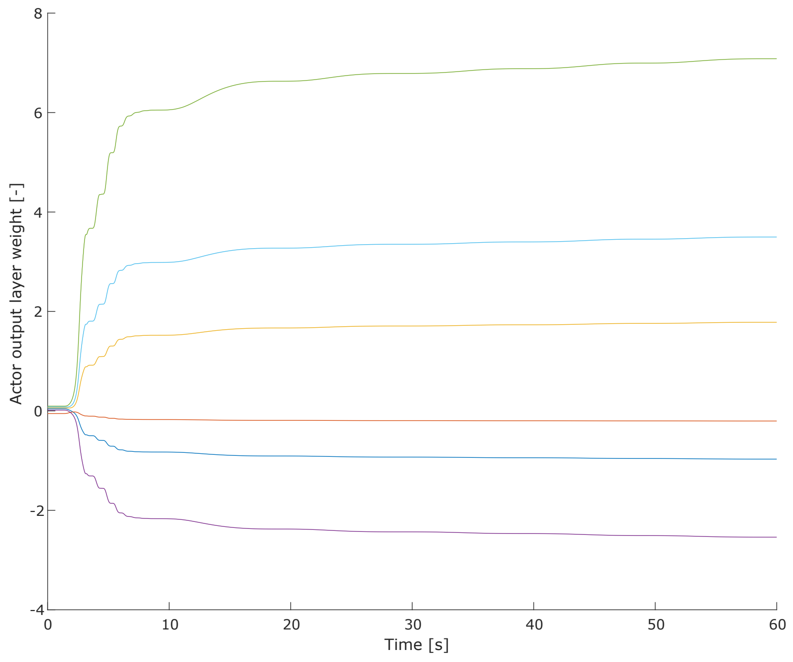


Figure 3-13: Actor ANN output layer weights progression

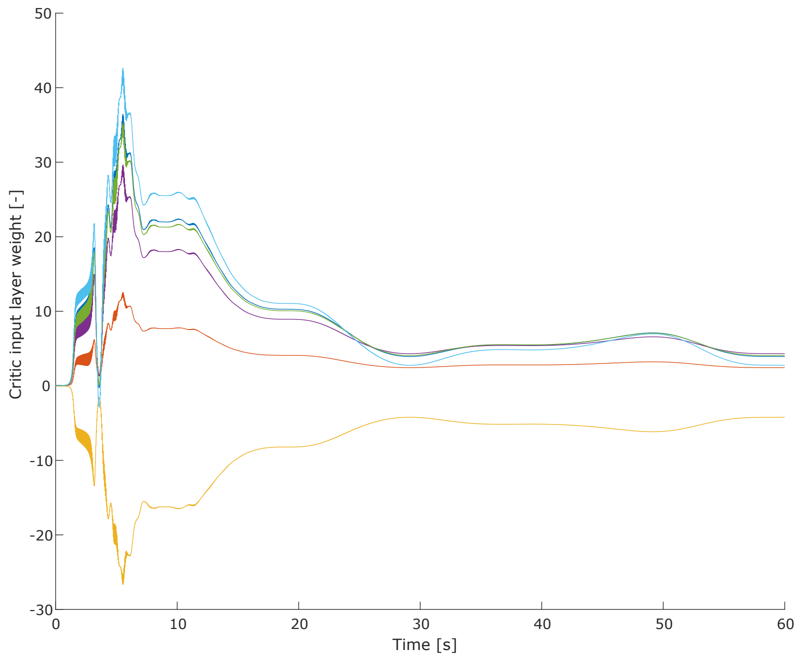


Figure 3-14: Critic ANN input layer weights progression

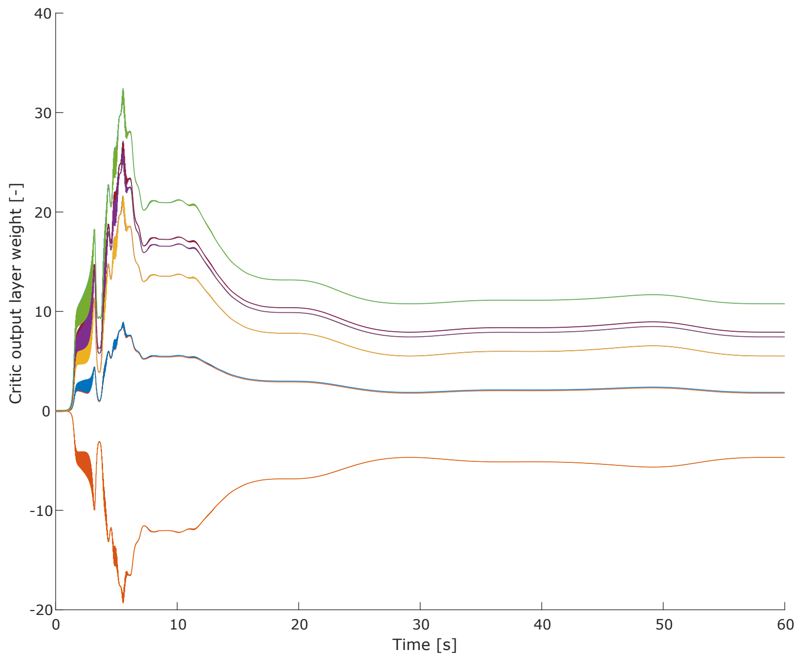


Figure 3-15: Critic ANN output layer weights progression

As for the overall accuracy of the IDHP controller, it can be seen from Figure 3-5 that after 2.5 seconds when the controller begins to track the reference signal, the policy found by the actor is aggressive as it tries to minimize the error and initially overshoots. The simulated α shows diminishing oscillatory behavior around the reference after the initial overshoot. Smooth reference tracking is visible after the inital oscillatory behavior over the first reference α peak with marginal steady state error seen at each peaks thereafter. This suggests that at least a near optimum policy has been found which is improved over time as evidenced by the decreasing error signal shown in Figure 3-7.

A point that needs to be addressed from the results obtained is that the simulated pitch rate shown in Figure 3-6 during the initial convergence to the reference oscillates violently. The cause of this can be explained by the gain parameter given to calculate the utility at each timestep. If the gain parameter is too small, then the learning may not occur or slow learning is seen as the training signal for the critic is too small to let it effectively diverge away from the initial weights. Additionally, low gain parameter also increases the steady state error after convergence.

A much closer dynamic relationship can be drawn between the pitch rate and the elevator input when compared to the angle of attack and the elevator input. As only the angle of attack reference signal error is provided to the actor and no information is given in terms of pitch rate, the rapid oscillation of the pitch angle during convergence can be explained by the actor network's tendency to over-evaluate the angle of attack error. If a pitch rate reference can be provided as well, at the cost of increased actor network size and complexity, the resulting pitch rate oscillations are expected to be reduced. Previous works such as (R. Enns & Si, 2003), (van Kampen et al., 2006), (Zhou et al., 2016), (Zhou et al., 2018b) have utilized the knowledge of the aircraft system dynamics to develop a cascaded actor network structure. Essentially by decomposing one large actor network to a cascaded small actor networks, a more direct relationship between the states and the input can be established. Implementing such cascaded actor network structure could be an efficient way to develop a more informed controller which can reduce undesirable behavior such as oscillations.

3-4-2 Empirical analysis

To demonstrate the implemented IDHP controller's stability and overall performance, an empirical analysis is performed. The empirical analysis performed based on results obtained from 500 independent simulation runs with 100 runs each at different initial angle of attack of the plant: $[-3\,\text{deg}, -1.5\,\text{deg}, 0\,\text{deg}, 1.5\,\text{deg}, 3\,\text{deg}]$. The results are presented in Figures 3-16 to 3-19.

From Figure 3-16, it can be seen that for all initial α, the simulated α quickly converges towards zero because the actor block has not been trained yet and therefore outputs near zero elevator deflection command for the initial 1.5 seconds. Then as the actor weights diverge away from the initial weights, it can be observed that an overall similar tracking pattern is present over the 500 independent runs. The large elevator command to quickly reduce the α_{error} at it can be seen in Figure 3-19 followed by diminishing oscillatory command as the reference was overshot. The steady state error illustrated in Figure 3-18 show comparably lower error variance when compared to the initial error variance seen during the first α_{ref} peak in Figure 3-17 evidencing that similar near optimal policies have been found during all

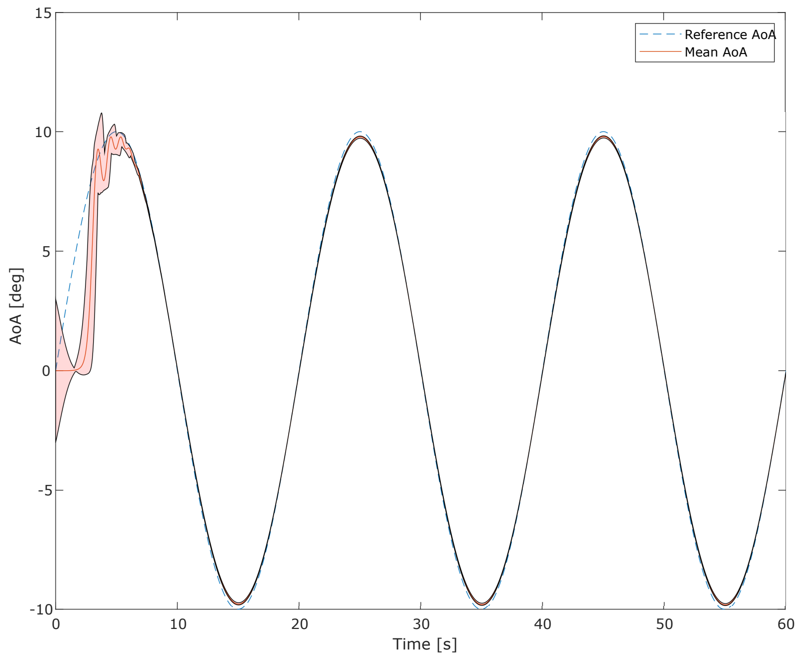


Figure 3-16: Plot of minimum, mean, and maximum simulated angle of attack signal of controlled system with reference angle of attack signal

independent runs. Assuming that the angle of attack response is in its transient phase for the first 10 seconds and in steady state from then on, the RMS α error during the transient state was found to be $1.9 \cdot 10^{-3} rad$ and steady state RMS error was found to be $7.8 \cdot 10^{-6} rad$. The RMS error has decreased significantly further supporting the argument that the IDHP controller is able to find a near optimal control policy.

It can be stated that the implemented IDHP controller is able to converge to a stable near optimal policy within 10 seconds of the simulation at $50Hz$ without offline training or offline model identification. This conclusion is supported by the fact that all 500 independent simulations did not exhibit any divergent behavior. However, it needs to be noted that the plant used for this preliminary analysis is a nonlinear 2 state 1 input LTI system where both the actor and critic network sizes were small. It can be anticipated that as the size of the ANNs grow to accommodate a larger state space, the time needed to converge to a near optimal policy will increase and may even result in a wrong policy. Another point of consideration is the angle of attack convergence to zero seen within 1.5 seconds. The cause for this had been given as near zero initial actor ANN weights which translate to small elevator deflection resulting in zero angle of attack response. While this may hold for a dynamically stable system and can be exploited for stable online training procedure before the actual tracking maneuver begins, it would certainly lead to diverging system response for a dynamically unstable system.

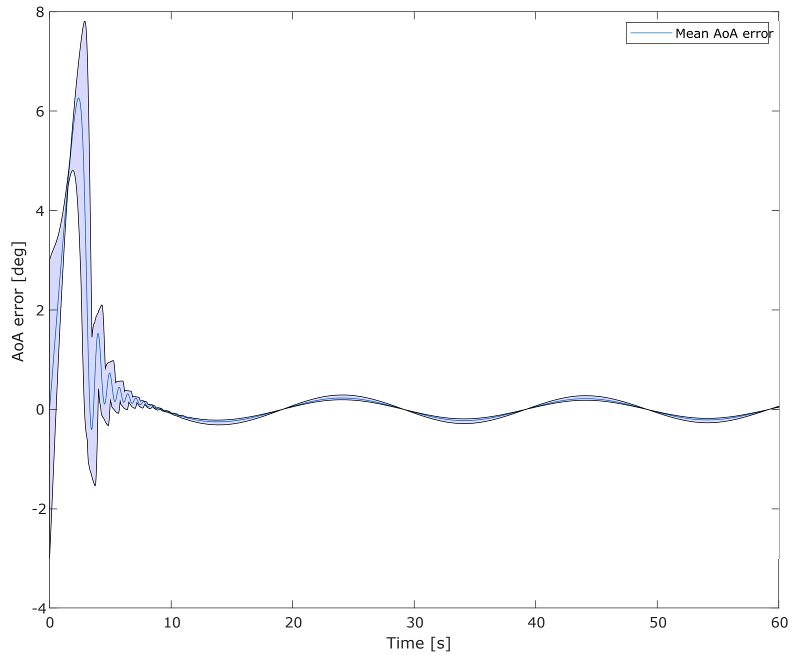


Figure 3-17: Plot of minimum, mean, and maximum simulated angle of attack error signal of controlled system

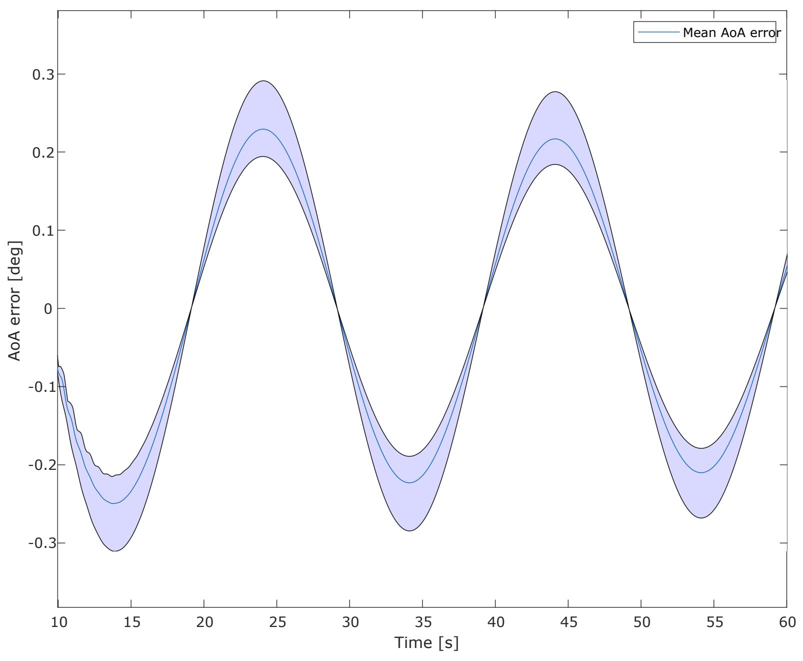


Figure 3-18: Plot of minimum, mean, and maximum simulated angle of attack error signal of controlled system shown from 10 seconds

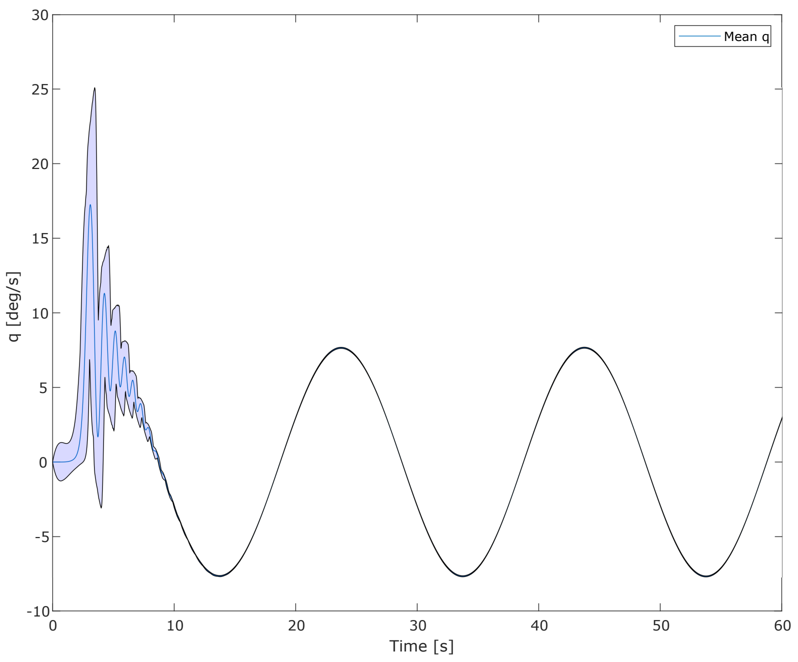


Figure 3-19: Plot of minimum, mean, and maximum simulated pitch rate signal of controlled system

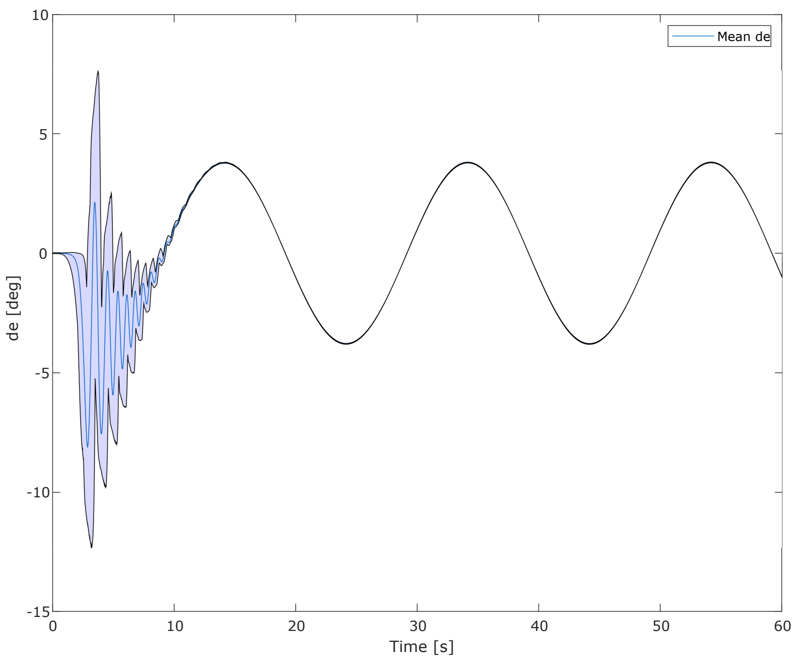


Figure 3-20: Plot of minimum, mean, and maximum simulated elevator deflection signal of controlled system

3-5 Preliminary Analysis Conclusion

In this preliminary analysis the simulation setting, implemented IDHP controller architecture and results obtained from simulation are presented and discussed. The implemented IDHP controller has been simulated to control the LTI short period model of the Cessna Citation aircraft for a given 10 deg amplitude 20 seconds period reference angle of attack signal with a varying initial angle of attack at $[-3\,\text{deg}, -1.5\,\text{deg}, 0\,\text{deg}, 1.5\,\text{deg}, 3\,\text{deg}]$. The results obtained from 500 independent runs have shown that significantly lower RMS tracking error of $7.8 \cdot 10^{-6} rad$ is observed during the steady state phase of the simulation when compared to the transient phase RMS tracking error of $1.9 \cdot 10^{-3} rad$ with no recorded instances of divergence. It can be concluded that although the angle of attack reference signal is given, which has weaker dynamic relation to the elevator deflection compared to the pitch rate, IDHP showed reliable stability and performance for the given tracking task.

Part III

Additional Results and Discussions

Chapter 4

Results obtained from Varying Learning Rates

A summary of results obtained from batch simulations with varying learning rates has been presented in the paper. In this section, time scale plots of simulation results are presented with an analysis of learning rate impact on IDHP baseline controller. Figures 4-1 to 4-4 show the obtained results from 300 independent runs with shaded regions representing σ and 2σ bands assuming normal distribution among independent runs.

It can be seen that increasing learning rates result in an increased reference convergence speed at the cost of decreased success ratio. Additionally, controller with higher learning rates show that a more optimal control policy is obtained in a shorter time frame when compared to the controller with lower learning rates. Initial prediction suggested that direct control policy between elevator deflection and altitude cannot be established by the baseline IDHP controller due to relatively slow aircraft translational motion dynamics. Considering that the main failure mode is aggressive policy generation that leads to oscillations around the reference altitude, the designed baseline IDHP controller is able to parametrize a control policy for Cessna Citation altitude tracking task. Decreasing learning rates can help suppress such aggressive policy generation by slowing the learning process as a whole.

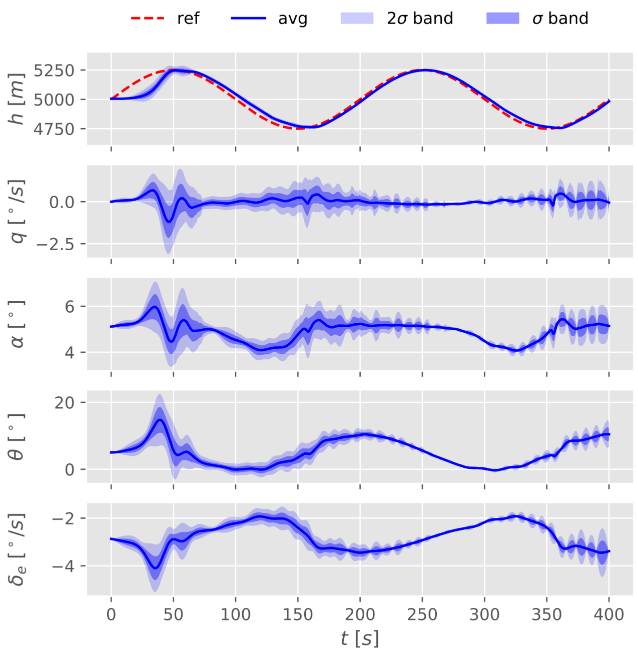


Figure 4-1: Simulation results under perfect conditions and learning rates $\eta_a = 5$ and $\eta_c = 2$ 131/300 runs

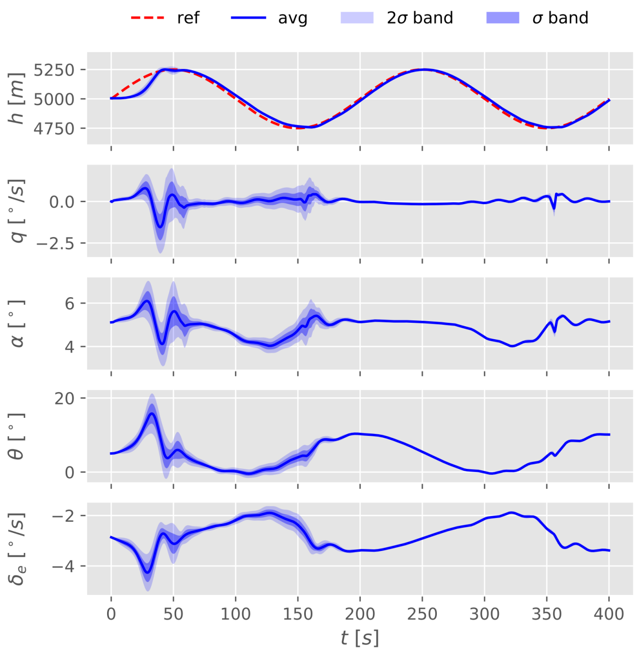


Figure 4-2: Simulation results under perfect conditions and learning rates $\eta_a = 10$ and $\eta_c = 5$ 105/300 runs

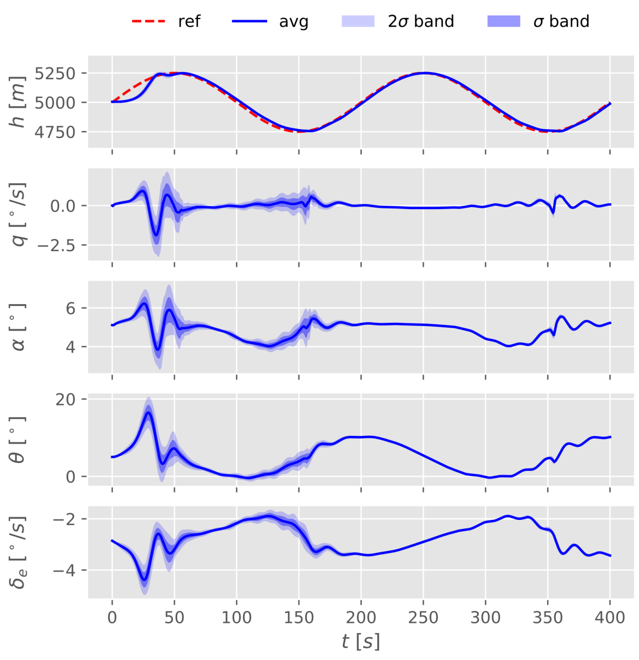


Figure 4-3: Simulation results under perfect conditions and learning rates $\eta_a = 17$ and $\eta_c = 7$ 40/300 runs

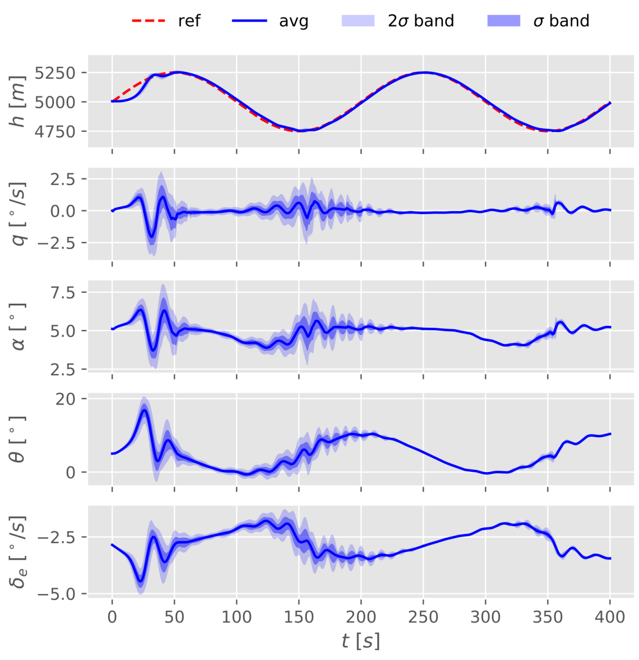


Figure 4-4: Simulation results under perfect conditions and learning rates $\eta_a = 25$ and $\eta_c = 10$ 17/300 runs

Chapter 5

Effect of Additional Critic Input for Cascaded Controller Design

In order to allow a more relative comparison between the designs, the cascaded IDHP controller design presented in the paper uses only altitude tracking error as critic input. However, pitch angle tracking error is used within the cascaded actor network. Therefore the critic is provided with disproportionate amount of information when compared to the actor. A common design choice is providing equal input vector to both critic and actor within ACD designs. Therefore in this section, a new cascaded controller design providing equal state information to both critic and actor is presented. Figures 5-1 show the critic network used within the scientific paper and Figure 5-2 shows the structure of critic network used within this section with additional critic input. Altitude scaling term $\kappa_h = 0.0001$ is identical to the value used in the scientific paper and $\kappa_\theta = 1$ is chosen.

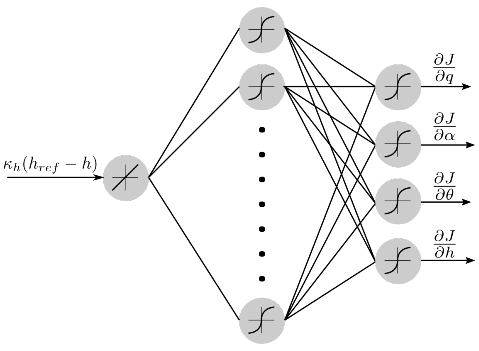

$\kappa_h(h_{ref} - h)$
$\frac{\partial J}{\partial q}$
$\frac{\partial J}{\partial \alpha}$
$\frac{\partial J}{\partial \theta}$
$\frac{\partial J}{\partial h}$

Figure 5-1: Critic network structure used within scientific paper with only sclaed altitude error as input

Figures 5-3 to 5-6 show batch simulation results from 300 independent runs satisfying converged success condition of altitude average $RMSE$ less than 100 meters in the second reference signal period. From the results, it is evident that providing the critic with additional state information has a negative impact. From Figures 5-3 and 5-4, slower convergence to

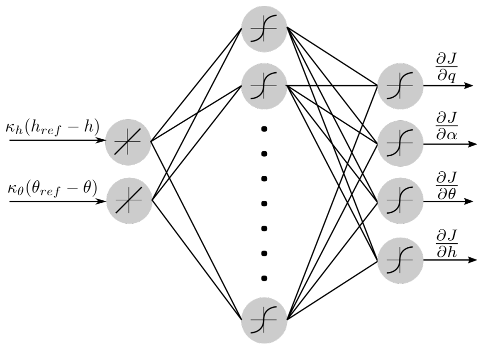

$\kappa_h(h_{ref} - h)$
$\kappa_\theta(\theta_{ref} - \theta)$
$\frac{\partial J}{\partial q}$
$\frac{\partial J}{\partial \alpha}$
$\frac{\partial J}{\partial \theta}$
$\frac{\partial J}{\partial h}$

Figure 5-2: Critic network structure with additional scaled θ error input term

reference and lower success ratio can be observed. At given learning rates, it can be concluded that the negative effect of increased critic complexity outweighs the benefits from additional relevant information provided to the critic. The consequences of added critic complexity is amplified when measurement noise are added to state observations. Figure 5-5 shows significantly slower convergence to altitude reference and lower success ratio when compared to Figure 5-6. No conclusive remarks can be made due to the lack of noise sensitivity analysis and learning rate sensitivity analysis. However, it is clear that providing all available relevant information to the critic is not necessarily beneficial.

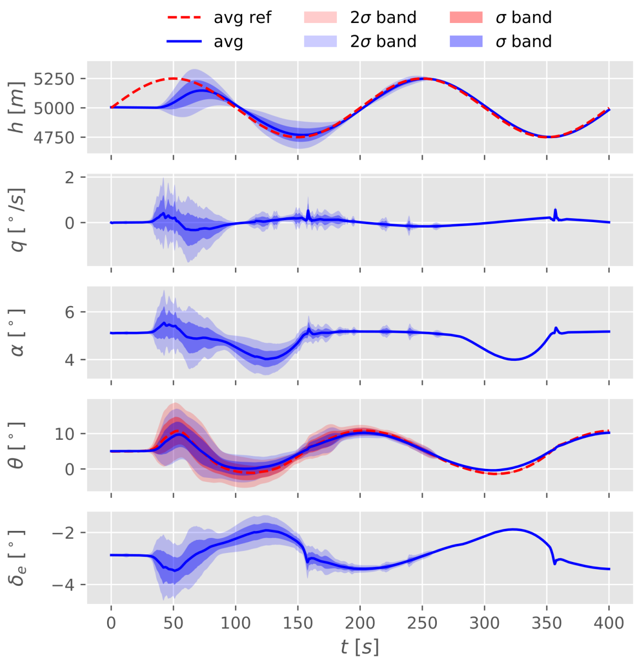


Figure 5-3: Simulation results obtained from cascaded controller with altitude and pitch angle errors given as critic input under perfect conditions 156/300 runs

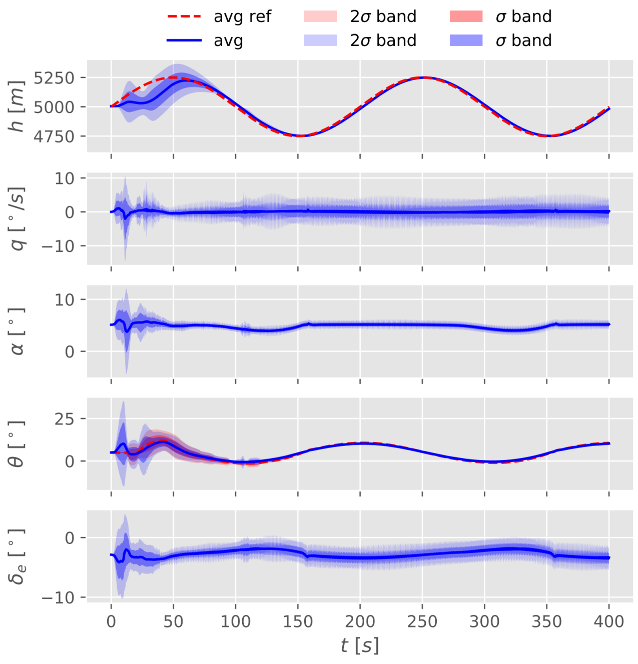


Figure 5-4: Simulation results obtained from cascaded controller with only altitude error given as critic input under perfect conditions 228/300 runs

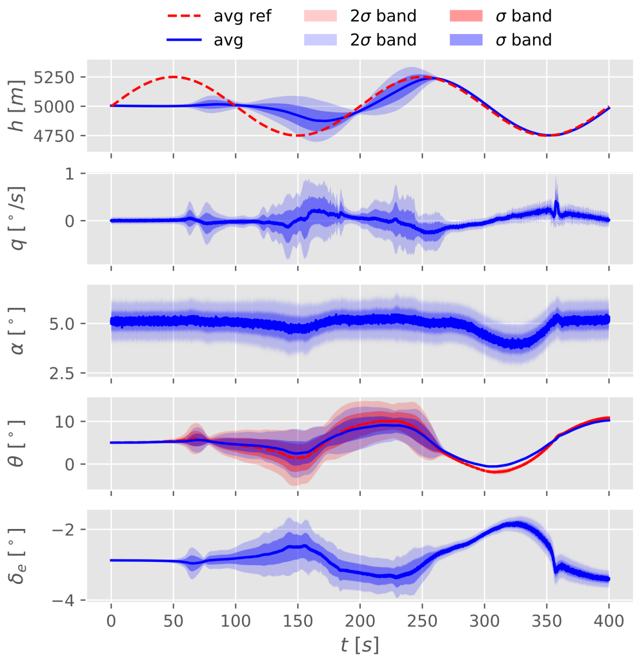


Figure 5-5: Simulation results obtained from cascaded controller with altitude and pitch angle errors given as critic input under measurement noise 31/300 runs

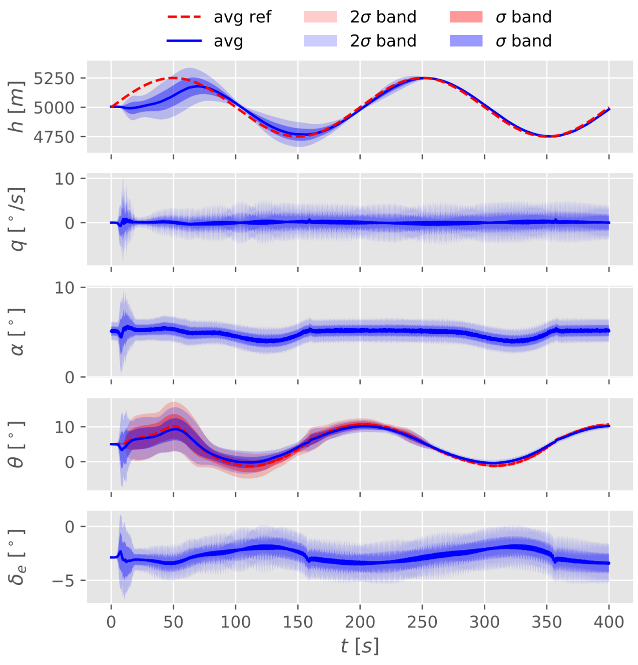


Figure 5-6: Simulation results obtained from cascaded controller with only altitude error given as critic input under measurement noise 153/300 runs

Chapter 6

Cascaded IDHP Controller Simulation Results Under Atmospheric Gusts

The baseline controller had been chosen as a more favorable design for altitude tracking task. This decision has been made on the basis of its lower sensitivity to measurement noise and its relative simplicity. However, the results have also shown that the cascaded controller design is a valid choice based on its consistent near optimal policy parametrization. Therefore the cascaded controller design as presented in the paper is simulated under atmospheric gusts. The simulation conditions are identical to experiment 3. Note that the results shown in this section are generated with the cascaded controller design with only altitude error as critic input.

Figures 6-1 to 6-8 show the batch simulation results satisfying success condition of altitude average $RMSE$ less than 40 meters in the last 30 seconds of the simulation run. From altitude response of the Cessna Citation model, it can be immediately noticed that increased gust intensity reduces the variance among independent runs. For FC0 and FC1, the success ratio has decreased with increased gust intensity while the opposite effect is seen for FC2 and FC3. (R. Enns & Si, 2003) showed that the addition of external disturbances does not have a straight forward effect of decreased learning performance but can also introduce an element of exploration beneficial to the learning process. From obtained results showing decreased variance in altitude response with increased gust intensity, it is evident that such element of exploration had helped the IDHP controller to converge to a near optimal control policy faster. Reduced success ratio seen for FC0 and FC1 with increased gust intensity suggests that the external disturbance can destabilize the learning process. For FC2 and FC3, increasing the gust intensity has an overall beneficial effect of increasing success ratio. The main difference between FC0/1 and FC2/3 are controller effectiveness dependent mainly on airspeed. The addition of external disturbances in flight conditions where aggressive policy generation is more likely is beneficial. Similar results of increased success ratio have been observed for the baseline controller with the addition of measurement noise. Therefore it can be summarized that the addition of external disturbances can have both positive and negative impact on the controller performance.

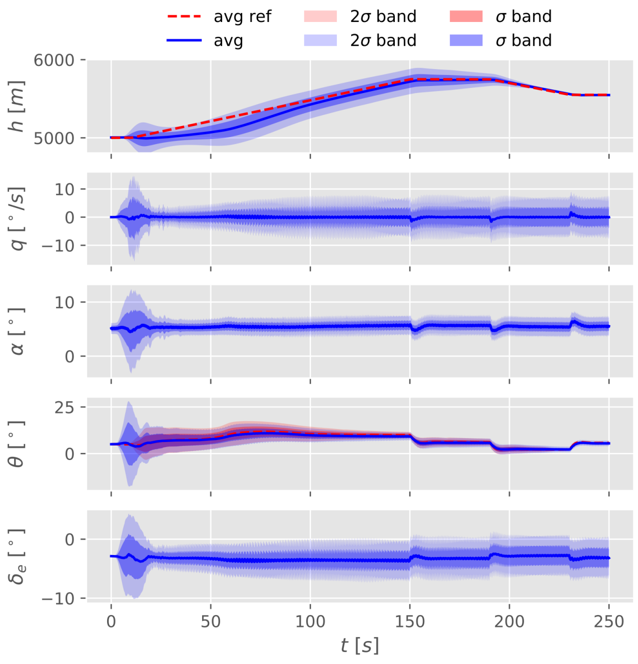


Figure 6-1: Cascaded Controller simulation results under measurement noise and light gust conditions in FC0 223/300 runs

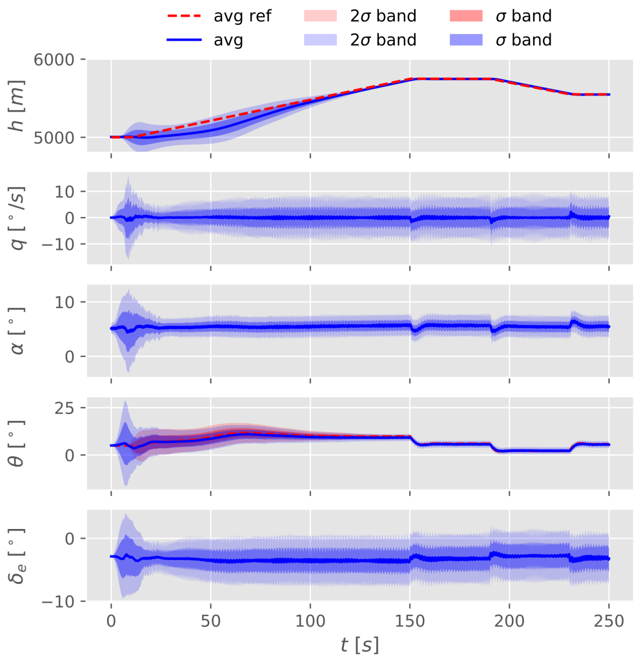


Figure 6-2: Cascaded Controller simulation results under measurement noise and moderate gust conditions in FC0 209/300 runs

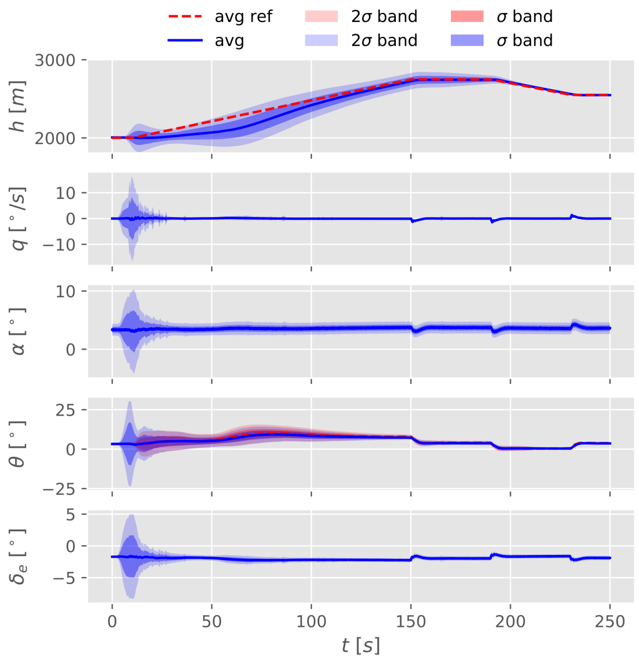


Figure 6-3: Cascaded Controller simulation results under measurement noise and light gust conditions in FC1 228/300 runs

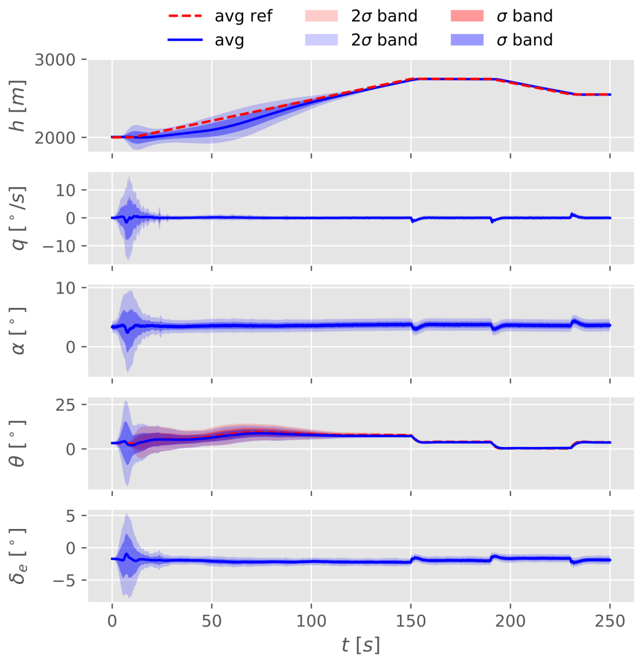


Figure 6-4: Cascaded Controller simulation results under measurement noise and moderate gust conditions in FC1 215/300 runs

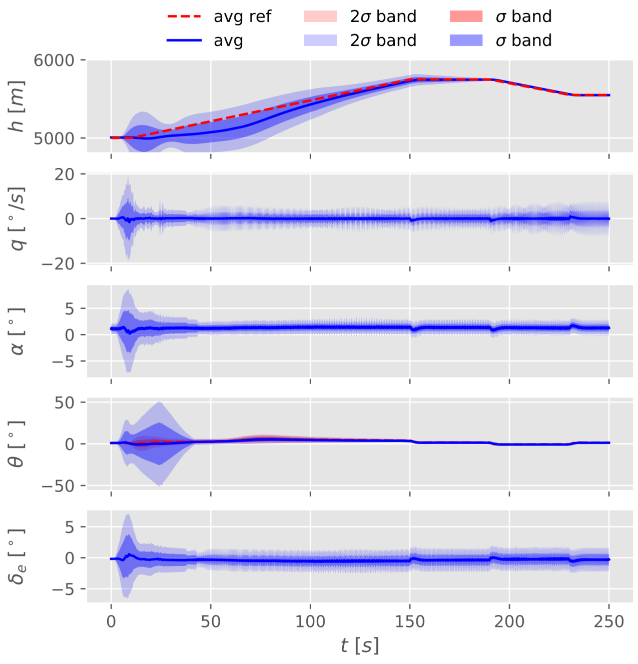


Figure 6-5: Cascaded Controller simulation results under measurement noise and light gust conditions in FC2 226/300 runs

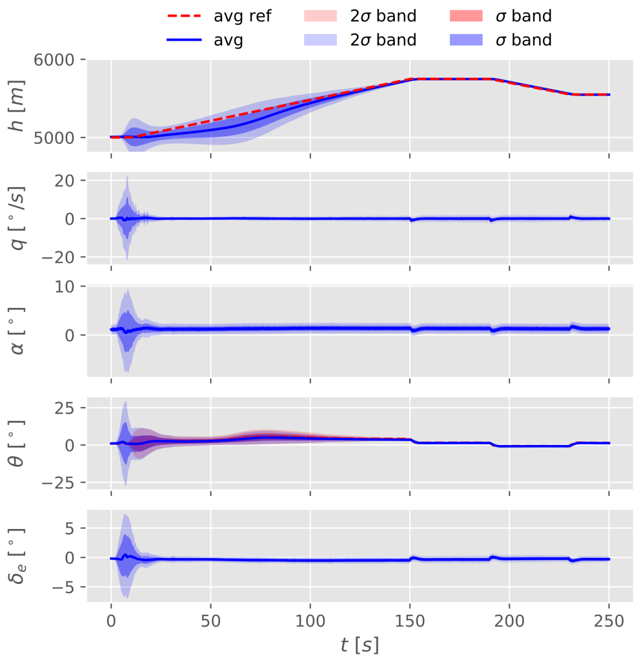


Figure 6-6: Cascaded Controller simulation results under measurement noise and moderate gust conditions in FC2 237/300 runs

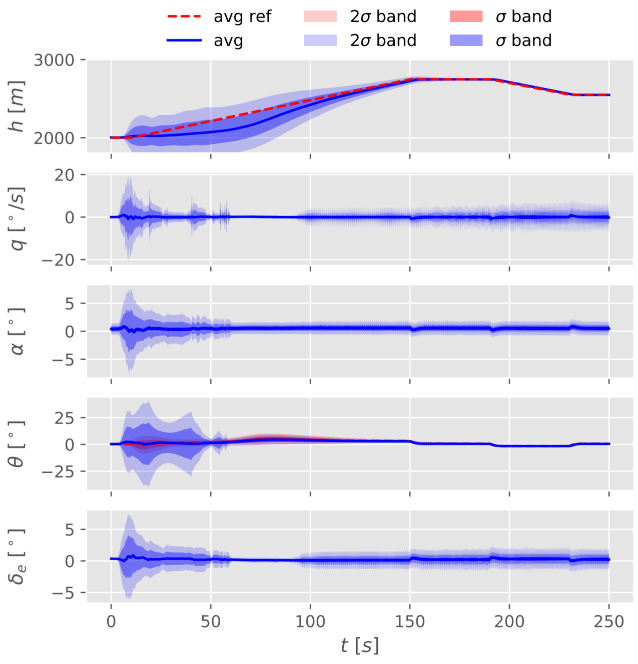


Figure 6-7: Cascaded Controller simulation results under measurement noise and light gust conditions in FC3 215/300 runs

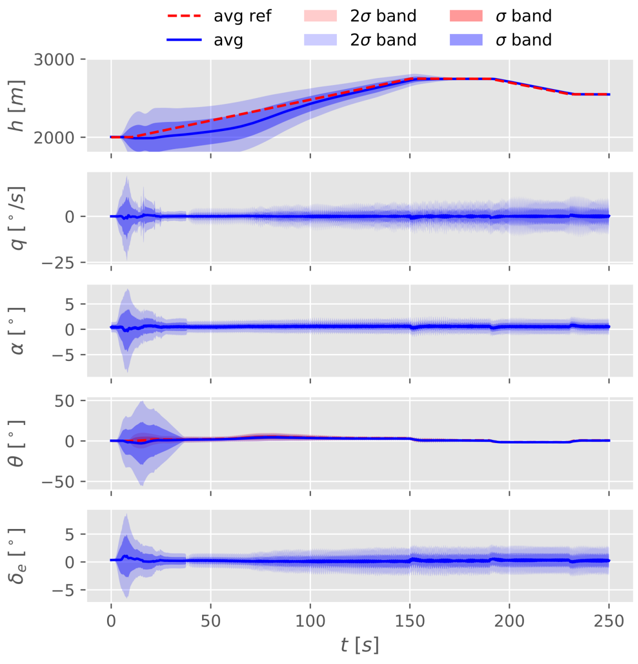


Figure 6-8: Cascaded Controller simulation results under measurement noise and moderate gust conditions in FC3 230/300 runs

To investigate characteristic differences controller designs under external disturbances, a representative run is chosen from results generated by each designs under moderate gust condition at FC0. Figure 6-9 shows the Cessna Citation model response controlled with a baseline controller and Figure 6-10 shows the response when controlled by a cascaded controller design.

The timescale response plots show that the cascaded controller design is clearly superior in controlling the Cessna Citation aircraft under external disturbances. This is due to the explicit control policy that is maintained within the cascaded controller design between θ and δ_e. As direct control of aircraft rotational dynamics is established, the controller is quicker to respond to atmospheric turbulence. Therefore it can be concluded that in absence of lateral motion, the cascaded controller is not required to establish altitude control but shows superior atmospheric disturbance rejection.

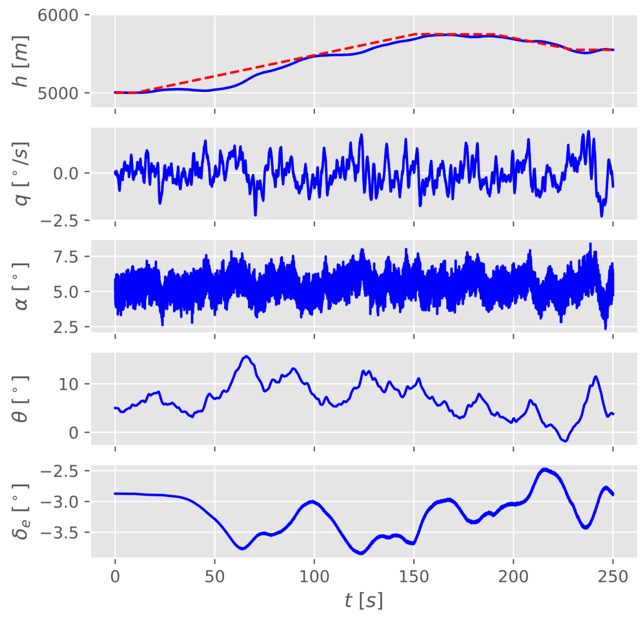


Figure 6-9: Representative baseline controller run under measurement noise and moderate gust conditions in FC0

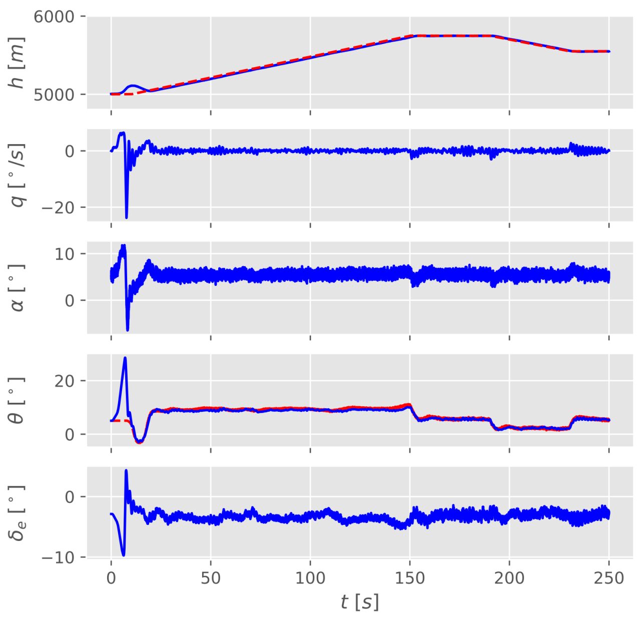


Figure 6-10: Representative cascaded controller run under measurement noise and moderate gust conditions in FC0

Part IV

Conclusion and Recommendations

Chapter 7

Conclusion

As an alternative approach to traditional adaptive controllers, reinforcement learning controller has been proposed to develop an inherently adaptive controller. To contribute towards the ongoing research and development process to implement an online, adaptive reinforcement learning controller to the Cessna Citation PH-LAB aircraft, the following research objective has been formulated.

> ***Implement an online and adaptive reinforcement learning altitude controller for the Cessna Citation model with increased simulation fidelity over current IDHP variant controller implementation to further close the gap between simulation and reality.***

Under the research objective, 4 research questions have been set up which have been answered through preliminary and main research phases.

RQ 1 **What is the most favorable reinforcement learning controller framework for fully online Cessna Citation altitude control?**

Adaptive Critic Designs (ACDs) have been identified to be the most prominent RL flight controller framework method due to their natural online applicability and explicit representation of a policy improvement element and a policy evaluation element. Review of state-of-the-art aircraft ACD controllers has identified the the main limitation of ACD controllers. In its original formulation, ACD controller requires an offline trained model preventing them from becoming a fully online controller. However, Incremental Heuristic Dynamic Programming (IHDP) and Incremental Dual Heuristic Dynamic Programming (IDHP) frameworks were found to be capable of full online operation. Simulation results have shown that IDHP outperforms IHDP in terms of tracking error, control success rate, and adaptability (Zhou et al., 2018b). Previous implementation cases of IDHP rate control of the Cessna Citation have shown good reference tracking results as well (Heyer, 2019; Kroezen, 2019). Therefore, IDHP was identified to be the most suitable RL controller framework for the Cessna Citation model altitude control task.

RQ 2 What changes can be made to the most favorable reinforcement learning controller to establish altitude control for the Cessna Citation model?

Relatively slower dynamic relationship between the aircraft altitude and the elevator deflection exist when compared to pitch rate where previous successful implementations have been demonstrated. This has lead to the hypothesis that a larger and more specified policy with additional actor inputs may be required to establish altitude control. Therefore, a cascaded actor network design has been proposed as a possible solution to provide structure to the increased policy dimension using expert knowledge of the aircraft system (R. Enns & Si, 2003; van Kampen et al., 2006; Zhou et al., 2018b). Inferior incremental model estimations is expected with the addition of measurement noise and atmospheric gusts. Therefore a measurement based training strategy is proposed to alleviate incremental model dependency of the IDHP controller. The main strategy is to wait for next time step measurements from Cessna Citation model instead of using estimated states generated by the incremental model (Sutton & Barto, 2018; Heyer, 2019). Additionally, a high-low adaptive learning rate is proposed that allows high learning rates when tracking errors are large and low learning rates when tracking errors are small to alleviate weight divergence due to continued learning.

RQ 3 How can the chosen online adaptive reinforcement learning controller design be implemented for the Cessna Citation model?

Using the DASMAT software package with Cessna Citation I specific settings, a high fidelity Cessna Citation model simulated at 100 Hz to represent the system. Using answers to research question 2, two IDHP controller designs have been created with and without the cascaded actor network. For the purpose of altitude IDHP control, only the longitudinal states obtained from the Cessna Citation model has been utilized. To limit lateral motion interference to longitudinal motion, constant trimmed input has been given to the Cessna Citation model rudder and aileron deflection angles. Although true airspeed is considered to be a longitudinal state, it has not been utilized as IDHP controller state and is controlled to be constant by the internal PID auto-throttle controller within the model. Therefore, the designed IDHP controllers utilize pitch rate, pitch angle, angle of attack, and altitude as reinforcement learning states with altitude error from reference given as input to both actor and critic. The elevator deflection angle has been used as action state. The measurement noise to reinforcement learning states have been modeled as Gaussian white noise. Finally, two atmospheric gust scenarios have been modeled as constant light and moderate intensity gusts according to the Dryden model.

RQ 4 What are the effects of measurement noise and atmospheric gusts to the implemented IDHP controller tracking results?

Simulation results without measurement noise have shown that both IDHP controller designs are able to establish Cessna Citation altitude control. When simulated with measurement noise, it has been shown that the addition of measurement noise can have a beneficial effect on the overall success ratio of the baseline controller without cascaded actor network. Introduction of measurement noise decreases the learning speed of the controller due to state uncertainty, but ultimately increases the success ratio by repressing aggressive control policy parametrization. For the cascaded controller design utilizing additional state information within the actor network, the negative impact of state uncertainty decreased the overall success ratio. To demonstrate that Cessna Citation aircraft altitude tracking is possible with measurement noise and atmospheric gust, 4 flight conditions for simulation has been determined. The 4 flight conditions for which the Cessna Citation altitude tracking tasks are demonstrated in have been set at combinations of airspeed [$90m/s$, $140m/s$] and altitudes [$2000m$, $5000m$]. The reference altitude signal has been designed to simulate cruise at initial altitude, climb to cruise, and descent to cruise at varying altitudes. Simulation results without prior training at each flight conditions have shown that altitude tracking is achievable for both light and moderate gust scenarios. Increasing gust intensity resulted in an overall decrease in success ratio and learning speed at all flight conditions for the baseline controller as a direct consequence of uncertain action-cost relationship. However, simulation results of the cascaded controller design at FC2 and FC3 have shown an increase in success ratio with increased gust intensity. This suggests that atmospheric gusts introduce an element of exploration to the reinforcement learning controller that may benefit the controller.

With all the research questions answered, the following conclusion for this thesis project can be drawn. By utilizing an IDHP framework, two online reinforcement learning altitude controllers have been designed and implemented to the Cessna Citation model. It has been demonstrated that direct altitude control can be established with a relatively simple IDHP controller using altitude error as input and cost. By introducing measurement noise and atmospheric gusts to the simulated environment, the simulation fidelity has been increased over previous IDHP variant controller implementation. Additionally, simulation results have shown that introduction of measurement noise and atmospheric gusts can have both beneficial and negative impact on the overall performance of the IDHP controller. The research in this thesis project contributes to the overall goal of implementing an online RL flight controller to the Cessna Citation PH-LAB aircraft by providing two valid IDHP controller designs to establish online RL altitude control. This in turn contributes to exploring RL control as an alternative adaptive flight control design.

Chapter 8

Recommendations

Two IDHP controllers have been designed and implemented to the high fidelity Cessna Citation model to establish RL altitude control. Although the research conducted in this thesis project have increased the simulation fidelity by adding measurement noise and atmospheric gusts to the simulation environment, lateral motion has not been considered. Additionally, assumptions regarding measurement noise characteristics, implemented design choices, and overall controller stability are points of further development and research. Therefore the following list of recommendations for future research are presented.

- Controller validation with lateral motion

The results of this thesis research have shown that RL altitude control can be established for the high fidelity Cessna Citation model. However, the simulations have been performed under relatively stable lateral motion as the rudder and aileron inputs have been kept constant at initial trim points. To further validate IDHP altitude control, simulation under varying lateral motion is a crucial next step. After validation, expanding RL control to simultaneous longitudinal and lateral motion can be considered. Furthermore, a comparison of baseline and cascaded controller under lateral motion should also be performed.

- Noise sensitivity analysis

The results have shown that for given parameters, the addition of measurement noise is beneficial for the overall increase in convergence ratio. Although the causality of success rate increase has been established, the exact relationship between success rate and measurement noise amplitude remains unknown. For future work seeking to increase convergence ratio, noise sensitivity analysis can help determine to which degree the measurement noise is beneficial to the learning process.

- Gradual adaptive learning rates

The adaptive learning rate scheme used in this research is based on the tracking error high-low adaptation rule. For a more continuous and gradual adaptive learning rates, readers are referred to (Ni et al., 2013). Additionally, increasing the error threshold over time can help reduce network over-training due to convergence to non-optimal policy while retaining adaptive control to sudden large changes in the environment.

- Critic estimation performance increase

Some failed runs can be attributed to poor λ estimation by the critic. As a general rule, critic estimation performance is critical to the success of the designed controller. To further increase convergence properties of the designed controllers, target critic previously implemented in (Heyer, 2019) can be considered.

- Policy shaping through actor output layer activation function choice

Although it is true that the overall convergence success rate is largely dependent on critic estimation performance, the overall shape of the policy maintained by the actor ultimately leads to aggressive control policy from which destabilization occurs. This effect is more pronounced when a relatively simple policy is maintained. To alleviate the problem without adding policy dimension, choosing a different activation function for the output layer can be considered with a smoother gradient. For future research, a sigmoidal actor output activation function offset by the trim point is suggested.

Appendix A

Preliminary Analysis IDHP parameter setting

Various parameters can affect the overall performance and stability of the implemented IDHP controller. In Table A-1, an overview of the parameters used for the preliminary analysis simulation are given.

Parameter	Value	Unit	Description
dt	0.02	[s]	Simulation time step
ω	20	[s]	α_{ref} period
L	6	[dt]	Length of time steps used for initial incremental model estimation
μ	0.8	[N/A]	RLS forgetting factor
γ	0.6	[N/A]	IDHP discount rate
N_a	6	[N/A]	Number of hidden layer nodes for actor ANN
N_c	6	[N/A]	Number of hidden layer nodes for critic ANN
η_a	0.1	[N/A]	Learning rate actor ANN
η_c	0.01	[N/A]	Learning rate critic ANN

Table A-1: Table of parameters used for IDHP implementation for the short period Cessna Citation model

Bibliography

Acquatella, P., van Kampen, E., & Chu, Q. P. (2013). Incremental backstepping for robust nonlinear flight control. In *Proceedings of the eurognc 2013, 2nd ceas specialist conference on guidance, navigation and control* (pp. 1444–1463).

Alpaydin, E. (2014). *Introduction to Machine Learning* (3rd ed.). MIT Press.

Al-Tamimi, A., Abu-Khalaf, M., & Lewis, F. L. (2007, feb). Adaptive critic designs for discrete-time zero-sum games with application to H_∞ control. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *37*(1), 240–247. doi: 10.1109/TSMCB.2006.880135

Atkeson, C., & Santamaria, J. (1997). A comparison of direct and model-based reinforcement learning. In *Proceedings of international conference on robotics and automation* (Vol. 4, pp. 3557–3564). IEEE. doi: 10.1109/ROBOT.1997.606886

Bacon, B., & Ostroff, A. (2000, aug). Reconfigurable flight control using nonlinear dynamic inversion with a special accelerometer implementation. In *Aiaa guidance, navigation, and control conference and exhibit.* Reston, Virigina: American Institute of Aeronautics and Astronautics. doi: 10.2514/6.2000-4565

Balakrishnan, S. N., & Biega, V. (1996, jul). Adaptive-critic-based neural networks for aircraft optimal control. *Journal of Guidance, Control, and Dynamics*, *19*(4), 893–898. doi: 10.2514/3.21715

Bellman, R. (1957). *Dynamic Programming.* Princeton: Princeton University Press.

Benbrahim, H., Doleac, J., Franklin, J., & Selfridge, O. (1992). Real-time learning: a ball on a beam. In *[proceedings 1992] ijcnn international joint conference on neural networks* (Vol. 1, pp. 98–103). IEEE. doi: 10.1109/IJCNN.1992.287219

Bertsekas, D. (1995). *Dynamic Programming and Optimal Control: Volume I.* Belmont, Massachusetts: Athena Scientific.

Brinker, J., & Wise, K. (1999, aug). Nonlinear simulation analysis of a tailless advanced fighter aircraft reconfigurable flight control law. In *Guidance, navigation, and control conference and exhibit.* Reston, Virigina: American Institute of Aeronautics and Astronautics. doi: 10.2514/6.1999-4040

Busoniu, L., Babuska, R., De Schutter, B., & Ernst, D. (2010). *Reinforcement Learning and Dynamic Programming Using Function Approximators* (1st ed.). CRC Press.

Cichosz, P., & Mulawka, J. J. (1995). Fast and Efficient Reinforcement Learning with Truncated Temporal Differences. In *Machine learning proceedings 1995* (pp. 99–107). Elsevier. doi: 10.1016/B978-1-55860-377-6.50021-9

Cybenko, G. (1989, dec). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, *2*(4), 303–314. doi: 10.1007/BF02551274

Degris, T., Pilarski, P. M., & Sutton, R. S. (2012, jun). Model-Free reinforcement learning with continuous action in practice. In *2012 american control conference (acc)* (pp. 2177–2182). IEEE. doi: 10.1109/ACC.2012.6315022

Doman, D. B., & Ngo, A. D. (2002, mar). Dynamic Inversion-Based Adaptive/Reconfigurable Control of the X-33 on Ascent. *Journal of Guidance, Control, and Dynamics*, *25*(2), 275–284. doi: 10.2514/2.4879

Durham, W., Bordignon, K. A., & Beck, R. (2016). *Aircraft Control Allocation.* Chichester, UK: John Wiley & Sons, Ltd. doi: 10.1002/9781118827789

Enns, D., Bugajski, D., Hendrick, R., & Stein, G. (1994). Dynamic inversion: an evolving methodology for flight control design. *International Journal of Control*, *59*(1), 71–91. doi: 10.1080/00207179408923070

Enns, R., & Si, J. (2003, jul). Helicopter trimming and tracking control using direct neural dynamic programming. *IEEE transactions on neural networks*, *14*(4), 929–39. doi: 10.1109/TNN.2003.813839

Farrell, J., Sharma, M., & Polycarpou, M. (2005, nov). Backstepping-Based Flight Control with Adaptive Function Approximation. *Journal of Guidance, Control, and Dynamics*, *28*(6), 1089–1102. doi: 10.2514/1.13030

Ferrari, S., & Stengel, R. F. (2002). An adaptive critic global controller. *Proceedings of the American Control Conference*, *4*, 2665–2670. doi: 10.1109/ACC.2002.1025189

Ferrari, S., & Stengel, R. F. (2004). Online Adaptive Critic Flight Control. *Journal of Guidance, Control, and Dynamics*, *27*(5), 777–786. doi: 10.2514/1.12597

Gaskett, C., Wettergreen, D., & Zelinsky, A. (1999). Q-Learning in Continuous State and Action Spaces. In N. Foo (Ed.), *Advanced topics in artificial intelligence* (pp. 417–428). Berlin, Heidelberg: Springer Berlin Heidelberg.

Glorennec, P., & Jouffe, L. (1997). Fuzzy Q-learning. In *Proceedings of 6th international fuzzy systems conference* (Vol. 2, pp. 659–662). IEEE. doi: 10.1109/FUZZY.1997.622790

Grondman, I., Busoniu, L., Lopes, G. A. D., & Babuska, R. (2012, nov). A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *42*(6), 1291–1307. doi: 10.1109/TSMCC.2012.2218595

Gross, H.-M., Stephan, V., & Krabbes, M. (1998). A neural field approach to topological reinforcement learning in continuous action spaces. In *1998 ieee international joint conference on neural networks proceedings. ieee world congress on computational intelligence (cat. no.98ch36227)* (Vol. 3, pp. 1992–1997). IEEE. doi: 10.1109/IJCNN.1998.687165

Han, D., & Balakrishnan, S. N. (2002). State-constrained agile missile control with adaptive-critic-based neural networks. *IEEE Transactions on Control Systems Technology*, *10*(4), 481–489. doi: 10.1109/TCST.2002.1014669

Heyer, S. (2019). *Reinforcement Learning for Flight Control* (Master Thesis). Delft University of Technology.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement Learning : A Survey. *Journal of Artificial Intelligence Research*, *4*, 237–285. doi: 10.1561/2200000049

Khan, S. G., Herrmann, G., Lewis, F. L., Pipe, T., & Melhuish, C. (2012). Reinforcement learning and optimal adaptive control: An overview and implementation examples. *Annual Reviews in Control*, *36*(1), 42–59. doi: 10.1016/j.arcontrol.2012.03.004

Kober, J., Bagnell, J. A., & Peters, J. (2013, sep). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, *32*(11), 1238–1274. doi: 10.1177/0278364913495721

Kretchmar, R. M., & Anderson, C. W. (1997). Comparison of CMACs and radial basis functions for local function approximators in reinforcement learning. *IEEE International Conference on Neural Networks - Conference Proceedings*, *2*, 834–837. doi: 10.1109/ICNN.1997.616132

Kroezen, D. (2019). *Online Reinforcement Learning for Flight Control* (Master Thesis). Delft University of Technology.

Lewis, F. L., Vrabie, D., & Vamvoudakis, K. G. (2012). Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *IEEE Control Systems*, *32*(6), 76–105. doi: 10.1109/MCS.2012.2214134

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., ... Wierstra, D. (2015, sep). Continuous control with deep reinforcement learning. *Foundations and Trends® in Machine Learning*, *2*(1), 1–127. doi: 10.1561/2200000006

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013, dec). Playing Atari with Deep Reinforcement Learning. *NIPS Deep Learning Workshop 2013*, 1–9.

Mulder, J. A., Van Staveren, W. H. J. J., Van Der Vaart, J. C., De Weerdt, E., De Visser, C. C., In 't Veld, A. C., & Mooij, E. (2013). *Lecture Notes AE3202 Flight Dynamics.* Delft: Delft University of Technology.

Ni, Z., He, H., & Wen, J. (2013). Adaptive learning in tracking control based on the dual critic network design. *IEEE Transactions on Neural Networks and Learning Systems*, *24*(6), 913–928. doi: 10.1109/TNNLS.2013.2247627

Peng, J., & Williams, R. J. (1994). Incremental Multi-Step Q-Learning. In *Machine learning proceedings 1994* (Vol. 22, pp. 226–232). Elsevier. doi: 10.1016/B978-1-55860-335-6.50035-0

Powell, W. B. (2011). *Approximate Dynamic Programming: Solving the Curses of Dimensionality* (2nd ed.). Princeton: John Wiley & Sons, Inc.

Prokhorov, D. V., Santiago, R. A., & Wunsch, D. C. (1995). Adaptive critic designs: A case study for neurocontrol. *Neural Networks*, *8*(9), 1367–1372. doi: 10.1016/0893-6080(95)00042-9

Prokhorov, D. V., & Wunsch, D. C. (1997). Adaptive Critic Designs. *IEEE Transactions on Neural Networks*, *8*(5), 997–1007.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (1st ed.). New York: John Wiley & Sons, Inc.

Rummery, G. A., & Niranjan, M. (1994). *On-Line Q-Learning Using Connectionist Systems* (Tech. Rep.). Cambridge: Cambridge University Engieering Department.

Si, J., & Wang, Y. T. (2001). On-line learning control by association and reinforcement. *IEEE Transactions on Neural Networks*, *12*(2), 264–276. doi: 10.1109/72.914523

Sieberling, S., Chu, Q. P., & Mulder, J. A. (2010). Robust Flight Control Using Incremental Nonlinear Dynamic Inversion and Angular Acceleration Prediction. *Journal of Guidance, Control, and Dynamics*, *33*(6), 1732–1742. doi: 10.2514/1.49978

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, *550*(7676), 354–359. doi: 10.1038/nature24270

Simplício, P., Pavel, M. D., van Kampen, E., & Chu, Q. P. (2013). An acceleration measurements-based approach for helicopter nonlinear flight control using incremental nonlinear dynamic inversion. *Control Engineering Practice*, *21*(8), 1065–1077. doi: 10.1016/j.conengprac.2013.03.009

Singh, S., Jaakkola, T., Littman, M. L., & Szepesvári, C. (2000). Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, *38*(3), 287–308.

Smeur, E. J. J., Chu, Q., & de Croon, G. C. H. E. (2016, mar). Adaptive Incremental Nonlinear Dynamic Inversion for Attitude Control of Micro Air Vehicles. *Journal of Guidance, Control, and Dynamics*, *39*(3), 450–461. doi: 10.2514/1.G001490

Stevens, B. L., Lewis, F. L., & Johnson, E. N. (2015). *Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems* (3rd ed.). Hoboken, NJ, USA: John Wiley & Sons, Inc. doi: 10.1002/9781119174882

Stone, P., Sutton, R. S., & Kuhlmann, G. (2005, sep). Reinforcement Learning for RoboCup Soccer Keepaway. *Adaptive Behavior*, *13*(3), 165–188. doi: 10.1177/105971230501300301

Sutton, R. S. (1996). Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding. *Advances in neural information processing systems*, 1038–1044.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). Cambridge, Massachusetts: MIT Press.

Tesauro, G. (1992). Practical Issues in Temporal Difference Learning. In J. Moody, S. Hanson, & R. P. Lippmann (Eds.), *Advances in neural information processing systems 4* (pp. 259–266). Morgan-Kaufmann.

van Kampen, E., Chu, Q. P., & Mulder, J. A. (2006, aug). Continuous Adaptive Critic Flight Control Aided with Approximated Plant Dynamics. In *Aiaa guidance, navigation, and control conference and exhibit.* Reston, Virigina: American Institute of Aeronautics and Astronautics. doi: 10.2514/6.2006-6429

Venayagamoorthy, G., Harley, R., & Wunsch, D. (2002, may). Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol of a turbogenerator. *IEEE Transactions on Neural Networks*, *13*(3), 764–773. doi: 10.1109/TNN.2002.1000146

Wang, F.-y., Zhang, H., & Liu, D. (2009, may). Adaptive Dynamic Programming: An Introduction. *IEEE Computational Intelligence Magazine*, *4*(2), 39–47. doi: 10.1109/MCI.2009.932261

Watkins, C. J. C. H. (1989). *Learning from delayed rewards* (PhD Thesis). King's College.

Watkins, C. J. C. H., & Dayan, P. (1992, may). Q-learning. *Machine Learning*, *8*(3-4), 279–292. doi: 10.1007/BF00992698

Werbos, P. J. (1977). Advanced forecasting methods for global crisis warning and models of intelligence. *General Systems Yearbook*, *22*, 25–38.

Wiering, M., & van Otterlo, M. (2014). *Reinforcement Learning: State-of-the-art.* Berlin: Springer Publishing Company, Incorporated.

Xu, X., He, H. G., & Hu, D. (2002). Efficient reinforcement learning using recursive least-squares methods. *Journal of Artificial Intelligence Research*, *16*, 259–292.

Zhao, D., Haitao Wang, Kun Shao, & Zhu, Y. (2016, dec). Deep reinforcement learning with experience replay based on SARSA. In *2016 ieee symposium series on computational intelligence (ssci)* (pp. 1–6). IEEE. doi: 10.1109/SSCI.2016.7849837

Zhou, Y., Van Kampen, E.-J., & Chu, Q. P. (2016). Incremental Model Based Heuristic Dynamic Programming for Nonlinear Adaptive Flight Control. In *Proceedings of the international micro air vehicles conference and competition 2016.* Beijing, China.

Zhou, Y., van Kampen, E.-J., & Chu, Q. (2018a). Incremental Approximate Dynamic Programming for Nonlinear Adaptive Tracking Control with Partial Observability. *Journal of Guidance, Control, and Dynamics*, *41*(12), 2554–2567. doi: 10.2514/1.g003472

Zhou, Y., van Kampen, E.-J., & Chu, Q. P. (2017). Nonlinear Adaptive Flight Control Using Incremental Approximate Dynamic Programming and Output Feedback. *Journal of Guidance, Control, and Dynamics*, *40*(2), 493–500. doi: 10.2514/1.G001762

Zhou, Y., van Kampen, E. J., & Chu, Q. P. (2018b). Incremental model based online dual heuristic programming for nonlinear adaptive control. *Control Engineering Practice*, *73*, 13–25. doi: 10.1016/j.conengprac.2017.12.011