CRANFIELD UNIVERSITY

DEWI H. BUDIARTI

DEVELOPMENT OF MODEL FREE FLIGHT CONTROL SYSTEM USING DEEP DETERMINISTIC POLICY GRADIENT (DDPG)

CRANFIELD UNIVERSITY
SCHOOL OF AEROSPACE

PhD Thesis
Academic Year: 2015 - 2019

Supervisor: Prof Antonios Tsourdos
Associate Supervisor: Dr. Hyo-Sang Shin
September 2019

CRANFIELD UNIVERSITY

CRANFIELD UNIVERSITY
SCHOOL OF AEROSPACE

PhD Thesis

Academic Year 2015 - 2019

Dewi H. Budiarti

DEVELOPMENT OF MODEL FREE FLIGHT CONTROL SYSTEM USING DEEP DETERMINISTIC POLICY GRADIENT (DDPG)

Supervisor: Prof Antonios Tsourdos
Associate Supervisor: Dr. Hyo-Sang Shin
September 2019

This thesis is submitted in partial fulfilment of the requirements for the degree of PhD

ABSTRACT

Developing a flight control system for a complete 6 degree-of-freedom for an air vehicle remains a huge task that requires time and effort to gather all the necessary data. This thesis proposes the use of reinforcement learning to develop a policy for a flight control system of an air vehicle. This method is designed to be independent of a model but it does require a set of samples for the reinforcement learning agent to learn from.

A novel reinforcement learning method called Deep Deterministic Policy Gradient (DDPG) is applied to counter the problem with large and continuous space in a flight control. However, applying the DDPG for multiple action is often difficult. Too many possibilities can hinder the reinforcement learning agent from converging its learning process.

This thesis proposes a learning strategy that helps shape the way the learning agent learns with multiple actions. It also shows that the final policy for flight control can be extracted and applied immediately for a flight control system.

Keywords:

reinforcement learning, flight control, deep deterministic policy gradient, learning strategy

ACKNOWLEDGEMENTS

All praise belongs to Allah Subhanahu wa Ta’ala, who is All Knowing and All Seeing. Laa Ilaha illallah Muhammad Rasulullah. I bear witness that there is no God but Allah and that Muhammad is His messenger. It is He and only He that I run to for every hardship and it is due to Him that I achieve what I have achieve today.

I thank Allah for the chance to pursue knowledge under the guidance of Prof Antonios Tsourdos and Dr. Hyo-Sang Shin. Their supervision and patience has broadened my horizon to the topics that I am pursuing. I extend my gratitude to both of them and also Dr. Chang-Hun Lee for all his help and discussion.

I want to thank my husband, Supriana Suwanda, for his love, motivation and support. Without you, this adventure would less likely enter my mind. My lovely children, Mikaila and Gabriel. Mikaila, you have grown into a mature young adult. It amazes me how much you’ve grown, both inside and out. In syaa Allah you will grow into a sholehah. Gabriel, you have learned so much and I see more and more of yourself every day. In syaa Allah you will grow into a sholeh gentleman indeed.

Thank you to RISTEK-Pro who has sponsored me to pursue a higher education and to the institution where I work at, Agency for the Assessment and Application of Technology (BPPT).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

LIST OF ABBREVIATIONS

BF	Basis Function
DPG	Deterministic Policy Gradient
DQN	Deep Q Network
FOC	Fixed Operating Condition
IT	Information Technology
LfD	Learn from Demonstration
LQR	Linear Quadratic Regulator
MC	Monte Carlo
MDP	Markov Decision Process
Q	Policy
RL	Reinforcement Learning
SARSA	State Action Reward (next) State (next) Action
TD	Temporal Difference
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
VOC	Variating Operating Condition
VTOL	Vertical Take-Off and Landing
x_{cg}	Centre of gravity
x_{cp}	Centre of pressure

1 INTRODUCTION

1.1 Background

In the last decade, there is a growing interest in developing a flight control system with a limited dynamic model or even without a dynamic model altogether. The development of unconventional designs in UAVs (Unmanned Aerial Vehicle) such as multirotor or tailless fixed wing UAV (flying wing) makes it more enticing because generating a dynamic model of it can be complex and cost too much.

Figure 1-1 Example of Unconventional UAV

Another interest for not using a dynamic model is the growing trend of developing an aircraft with unusual flight envelope. These aircrafts can change their wind axes during flight such as V-22 Osprey and fixed wing VTOL (Vertical Take-Off and Landing) UAV.

Figure 1-2 Example of aircraft with changing wind axes

Also emerging are new aircraft that has an unusual flight envelope due to added unconventional control surface such as thrust vectoring and additional propeller. Thrust vectoring are commonly used to advance the manoeuvrability of a fighter aircraft. While additional propeller are also known to be added to give a UAV a VTOL capability without changing the wind axes of the aircraft.

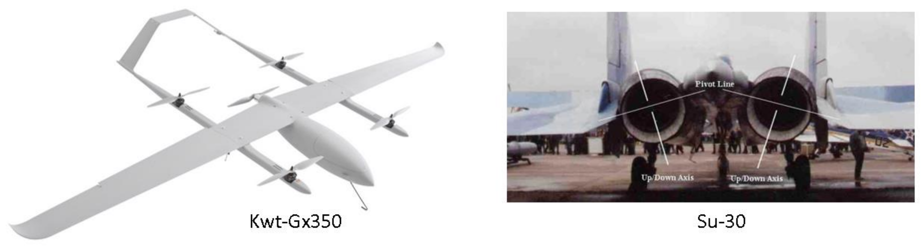


Figure 1-3 Example of aircrafts with unconventional control surface

Reinforcement learning is a data based learning methodology [1]. The training, learning, is performed by exploring the situation to develop awareness or a sense of improved behaviour.

This method mimics the way an animal learns about its environment. It receives a positive feedback when it gives the appropriate response to a situation and it receives a negative feedback when it gives the inappropriate response instead. This can eliminate most or even all of the time-consuming process to derive an almost accurate dynamic model of an air vehicle.

An advantage to this method is that it might be able to stumble to the optimal response to a situation instead of just finding the best response within a certain limit defined for dynamic modelling.

Another aspect of reinforcement learning is that the control system may also be more robust to changes of circumstance. Because of its ability to determine an action based on the actions taken from other situation. In theory, this presents a possibility of the system adjusting itself when a fault or a change in air vehicle characteristic occurred. Therefore, not only could it control an aircraft in normal condition, it can also adapt to certain changes occurring during flight.

However, a recurring problem has often occurred in previous studies due to the nature of the flight data. This is essential, as the reinforcement learning trains with it to develop the flight control system. The nature of the flight data is continuous, high dimensional and sometimes stochastic. This applies to both the attitude (state) of the air vehicle and the control surface (action).

Reinforcement learning works by mapping certain condition (state) with certain action. It explores every state and every action and exploits the pairs that generates the most reward [1]. Similar to that of animals, it explores its environment by trial and error. Each lesson learned are stored so it determines which area are safe, or even advantageous (food, water), and which area are to be avoided (cliffs). Just like the animals, there are circumstances where it will utilize its earned knowledge and there are circumstances where it will explore. This is the advantage of reinforcement learning. It learns when to explore more and when to exploit more. It learns to balances between exploration and exploitation in each circumstance it faces.

In an aircraft, the state can be defined as the various flight variables and the actions can be defined as the control surfaces available. When the state and/or the action are continuous, then the number of mapping will grow exponentially. This is what is called the “curse of dimensionality” problem[1].

This will make the learning process much more difficult and time consuming as it needs to learn by exploring each and every one of them. Earlier works has shown the reinforcement learning is applied in a conventional flight control development that still uses a complete dynamic model.

There are also studies that started to use reinforcement learning for navigation in soaring flight and for coordinating multiple UAV. Recent developments has seen studies in trying to tackle the curse of dimensionality problem head on in order to apply reinforcement learning to control the flight of an air vehicle.

One particular study is the development of deep learning sparked a study in Deep Deterministic Policy Gradient in 2016 [2]. This method combines the deep Q network (DQN) with Deterministic Policy Gradient (DPG). Presumably, this method provides a solution for continuous and high dimensional space.

DQN has the advantage of solving problems with high-dimensional state space, but it cannot handle continuous action spaces. By combining it with DPG, it can solve problems with continuous action space as well. This method has been applied for various simple and classic problems such as cart-pole swing-up and legged locomotion[2]. It has also been applied for more complex tasks such as controlling a biped robot[3][4], an autonomous car [5] and a quadrotor UAV [6],

Although recent studies on developing model-free flight control system for quadrotors and fixed wing aircraft exist, the difference between aerial vehicles characteristics are evident as shown in the work listed above. For fixed-wing 6-degree-of-freedom vehicles, more variables are clearly required to be observed. In addition, control surfaces characteristics vary between vehicles and affect the operation of the aerial vehicles in different ways.

Research studies using DDPG method, cited previously, have shown advantages for model-free control design but also challenges when handling multiple variables and its effect on convergence behaviour. Hence a particular aspect of the work in this thesis is to propose a learning strategy, within the DDPG remit, assisting the RL to obtain sufficient knowledge about operational scenario in lesser time.

This learning strategy is designed to pace the reinforcement learning agent to step-by-step acquiring one skill after another to finally be able to control an aircraft in flight. The skill here is to utilize the control surface of the air vehicle which comprises of elevator, aileron and rudder.

This thesis will focus in developing a learning strategy that can be used as a a baseline to train reinforcement learning to develop a flight control for a general fixed wing aircraft.

1.2 Research Objectives

The main objective of this research is develop a learning strategy to train the DDPG learning agent to control the flight performance of an air vehicle. The specific objectives are as followed:

- Determining the most suitable reward function that can represent the best performance of an air vehicle as the action with the best value of reward.
- Developing a learning strategy for a single action variable, which is the elevator. This learning strategy is developed for a fixed operating condition and a variating operating condition.
- Developing a learning strategy for dual action variable that can accommodate the coupling nature of lateral mode and directional mode of a fixed wing aircraft. The variables are aileron and rudder.
- Developing a learning strategy for reinforcement learning to be able to control a full 6-degree-of freedom fixed wing aircraft with three action variable which are elevator, aileron and rudder.

1.3 Thesis Overview and Contributions

This thesis consists of eight chapters. **Chapter 2** consists of theory regarding reinforcement learning in general, and reviews previous studies in reinforcement learning for continuous and high-dimensional spaces. **Chapter 3** will focus on describing the Deep Deterministic Policy Gradient (DDPG) and the learning strategy that is being developed.

Chapter 4 focuses in determining the most suitable reward function for the DDPG. The flight control for the longitudinal mode of a missile is designed by implementing DDPG with various reward function.

Chapters 5-7 consists of several study cases that utilize and assess the learning strategies of DDPG agent to control the flight performance for certain conditions.

In these 3 chapters, the learning strategy is developed from a single action variable to a three action variables, Figure 1-4.

Chapter 5 study case will be developing learning strategy for DDPG agent to control the longitudinal mode of an aircraft with one available action (elevator). The aircraft is a Jetstream 3102. Here the DDPG agent is expected to develop a flight control system in a fixed operating condition and in variating operating condition.

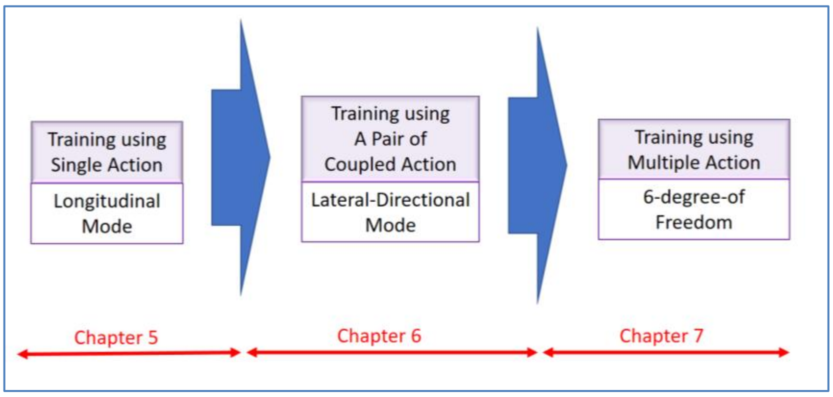

Training using Single Action
Longitudinal Mode
Training using A Pair of Coupled Action
Lateral-Directional Mode
Training using Multiple Action
6-degree-of Freedom
Chapter 5
Chapter 6
Chapter 7

Figure 1-4 Phases In Exploring DDPG for Flight Control

Chapter 6 will delve in developing a learning strategy for DDPG agent in Variating Operation Condition (VOC). For this case study, the air vehicle is a missile. The aim is to control its longitudinal mode with one available action (elevator). The VOC consists of variating airspeed and/or altitude.

Chapter 7 focuses on developing a learning strategy for the DDPG agent to handle the coupling of the lateral mode and the directional mode of a conventional fixed wing aircraft. The aircraft in this study case is a UAS (Unmanned Aerial System) and it will have two action variables at its disposal (aileron and rudder).

Chapter 8 contains the learning strategy for the DDPG agent that will enable it to control the dynamic of a full 6-degree-of-freedom aircraft. It has three action variables at its disposal (elevator, aileron and rudder). Similar to Chapter 7, the study case is a UAS.

Chapter 9 consists of the overall conclusion of this research regarding the learning strategy and extracting the final policy network. It also consists of possible future works that can be explored further.

The contribution of this research is to develop a learning strategy that will help to shape an unsupervised machine learning process in developing a flight control for an air vehicle. As are many other methods, DDPG has its own limitations, specifically regarding to multiple action. Without certain help in the learning process, it might not produce a suitable and usable final policy network.

2 LITERATURE REVIEW

2.1 Introduction

Reinforcement learning (RL) is a branch of machine learning that learns to map situations to actions unsupervised [1]. According to Dr. Danko Nikolic from Max-Planck Institute[44], the definition of machine learning is the science of getting computers to act without being *explicitly* programmed but instead letting them learn a few tricks on their own.

Another definition [7] states that RL is an algorithmic method for solving problems in which actions (decisions) are applied to a system over an extended period of time, in order to achieve a desired goal. Overall, RL mimics the way an animal learn and adjust to its environment without any outside interference in *how* to learn it.

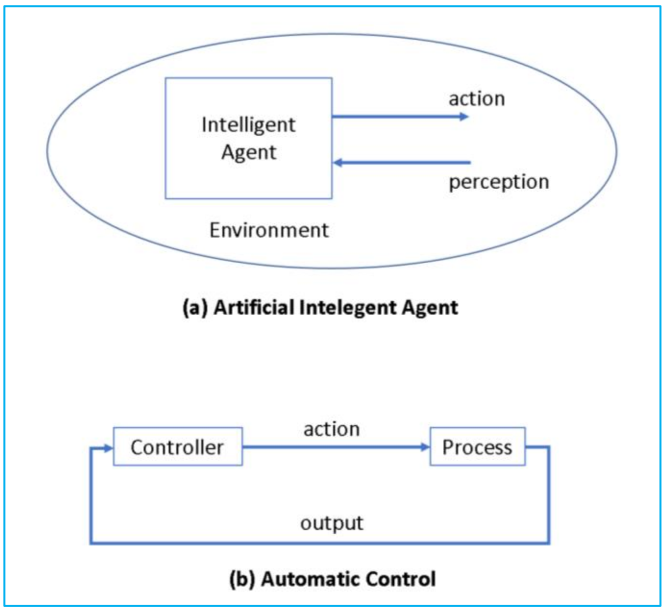


Figure 2-1 Application in two different domain [7]

RL comprises of two elements, a learner and an environment. A learner is the decision maker that decides which action to apply to a situation (state), monitor how those actions changes the state of the environment and learns from it for further decision making. Defined specifically for each cases, the RL has a list of potential states (called state space) and a list of actions available at its disposal (called action spaces). The nature of these spaces (discrete, continuous, deterministic, stochastic, etc) will determine the method used to solve it.

The designation for these elements may vary depending on the field that it is used for. An example, in the field of artificial intelligence, the learner is called an RL agent. Its task is to learn how to behave optimally towards its environment by interacting with it and monitoring the changes that happen because of it.

Another example is in automatic control. The learner is called the controller and the environment is called a system. A controller receives output measurement from a process and applies action to this process in order to make its behaviour satisfy a certain requirement. A general description of the slight difference in definition can be seen in **Figure 2-1**.

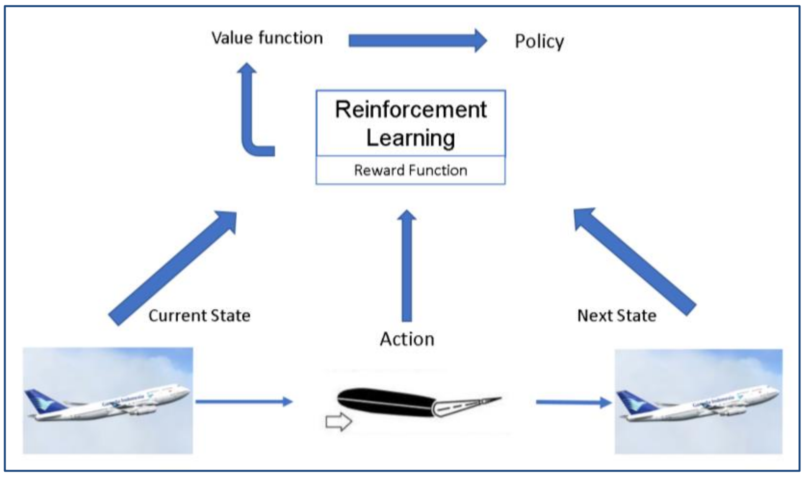


Figure 2-2 Interaction between sub elements

Aside from these 2 elements, there are also 4 sub elements of a RL system [1]. They are:

- a policy (Q), this sub element determines how the agent behaves towards different states at any given time. The policy maps the states to specific and suitable actions.
- a reward function(r) is the sub element that gives numerical reward to the state or state-action pair that occurs. For a given state, the agent makes a move (an action) that result in a change in state. This reward shows how far the resulting state is from the final goal.

- a value function (V/Q) gives value to a string of state (V-function) or state-action pairs (Q-function) by judging how close is the final state to the goal. In other words, it compiles a string of rewards and compares the final state with the final goal.
- model of the environment (*optional*)

The interactions between these sub elements can be described in **Figure 2-2**. In this example, the RL controlled the attitude of an aircraft. Based on [1], there are three class of methods to solve a RL problem, which are:

- Dynamic Programming
  DP methods mathematically computes optimal policies, but require a perfect model of the systems behaviour and the environment as an MDP (Markov Decision Process). This method has a downside in that requires a big computational expense[1]. Therefore, this method is often limited to discrete space. It is not suitable to handle a system with large finite state sets or high-dimensional state-space [8].
- Monte Carlo
  MC methods do not require a priori knowledge of the environment's dynamics and are conceptually simple. It learns through samples that represents interaction between action and any changes in the state. However, the value estimates and policies changed only upon the completion of an episode. This method has a known problem of maintaining sufficient '*exploration'*.
- Temporal Difference
  TD methods is a combination of both dynamic programming and monte carlo. It can learn directly from only experience without a model and can update estimates based in part on other estimates without waiting for an outcome

However according to [7], dynamic programming is different from reinforcement learning because of its requirement of a model. Dynamic programming requires an exact model while reinforcement learning does not (model-free). Despite the slight difference of definition, the objective in reinforcement learning is to gain the optimal policy in order for the agent to control a system in the desired behaviour.

2.2 Early Works Utilizing Reinforcement Learning

Since 2000, researchers has explored the use reinforcement learning on controlling various types of air vehicle, such as airship [9][10][11], quadcopter [12][13], unmanned helicopter [14] and unmanned fixed wing [15]. Reinforcement learning develops flight trajectory for navigating soaring flight [16] [17] and even applied indirectly in attitude control[18].

A few examples of indirect application in attitude control includes determining the optimal value in a classical flight control method using LQR [14] and H_∞ [19]. Another example is producing the dynamic model of an air vehicle based on flight data compiled when a human pilot flies the vehicle. Such cases were researched for a quadrotor [20], and an unmanned helicopter [21][22].

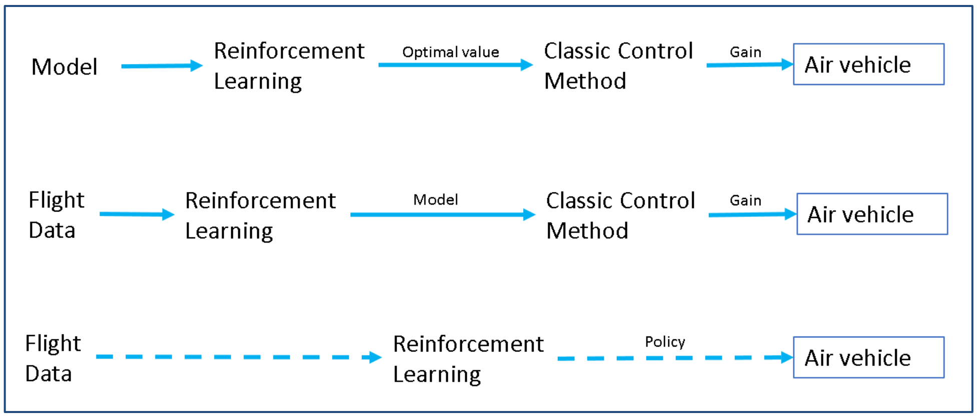


Figure 2-3 Research Gap

However not many researches have been done to use reinforcement learning entirely to develop a flight control[23][24][25]. This means using a reinforcement-learning agent as an artificial intelligence that controls the air vehicle. Research by [16] used a dynamic model and dynamic programming to control an air vehicle. Other research didn’t use a model and therefore, used either Monte Carlo method or Temporal Difference method[26][27].

Developing a policy to control an air vehicle always presents challenges due to the nature of the state-action spaces. It is continuous and high dimensional. This made the number of values very large when the agent has to explore and put value to every possible state-action pair. This is known as the ‘curse of dimensionality’.

2.3 Reinforcement Learning in Large and Continuous Space

It takes a very long time for the RL agent to find the most suitable value function for every one of the possible state-action pairs. Each state-action pairs needs to be visited often to determine its value. To tackle the ‘curse of dimensionality’, earlier works started with using discretising [28] and Learn from Demonstration (LfD).

Discretising method is a method that discretize both the state space and the action space and is usually applied in combination with the Monte Carlo method. This method, however, can only be applied for a small state space and action space[28]. There is always a possibility that a state value might be overlook and yet it may has significant influence on the system.

In LfD, the range of operation is confined and set by a previous flight demonstration by a pilot. The RL agent is trained to copy the manoeuvre set by a human pilot. This method negates the advantage of reinforcement learning, which is to find the optimal flight control. By exploring various possibilities, exploiting the best actions that has been discovered and maintaining a certain amount of exploring every now and then, the learning agent is expected to produce an optimal and robust flight control.

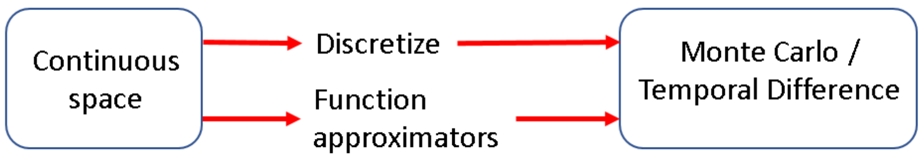


Figure 2-4 Earlier work in continuous space

Another method had started to emerge was by using a more compact-size Q-function (policy). This is achieved by utilising state-dependent basis functions (BFs) and discretising action space[7]. The state space is grouped into different BFs ($\phi_1, \phi_2, \ldots, \phi_N$). The Q-function is compacted by storing the values for each BF-action pair. **Figure 2-5** illustrates the difference in Q-function.

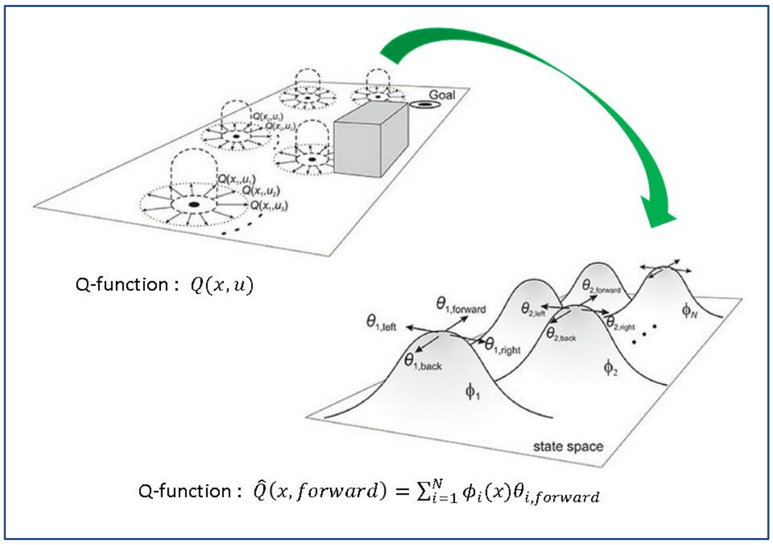


Figure 2-5 Q-function comparison [7]

Instead of storing Q-values of state and action pairs, it stores parameters θ with pairs of BF and action. This is called function approximators. Applying function approximator depends on:

- The algorithm used for the class of reinforcement learning method. They are value iteration, policy iteration and policy search **Figure 2-6**.
- Type of approximators, parametric or nonparametric

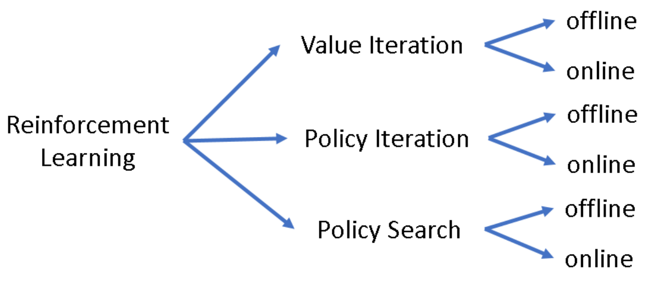


Figure 2-6 Taxonomy of reinforcement learning algorithms

As described in chapter 1, an RL agent has two main task, which are exploring its environment and learning from those experiences. In **Figure 2-6**, there is a mention of off-line and on-line methods[7]. The difference between the two is whether the behaviour policy and the target policy are separate policies or not[29][30][31].

The behaviour policy is the policy that incites the random actions for the RL agent to learn. The target policy or estimation policy is the policy that evaluates the actions are taken by the behaviour policy. The result of the evaluation will be used to update the behaviour policy periodically.

An offline method does not separate the behaviour policy and the target policy. Therefore, the policy updates very quickly after each run. This method sometimes also called off-policy. In the other hand, an on-line method does separate these two policies, sometimes also called on-policy[1].

Researchers has explored many variations of method to approximate the Q-function of a state. One of the examples is by using Gaussian process[11] [32]. This function approximator was applied for a Monte Carlo (MC) class method. As the MC method can only be applied for a series of episodic tasks, the samples it has were divided into several episodic tasks and its value estimation was changed/updated after each episode is finished[1]. Another example of works that uses function approximator to estimate the value function for a Monte Carlo method is [33]

However, this method still poses a problem as it only tackles the problem of continuous state space. The action space itself, still needs to be discretized for it to work. Another alternative that seems promising is a Temporal Difference (TD) method.

A popular online temporal difference method is SARSA. The name SARSA is derived from the elements of data tuple that is used by the algorithm. They are state, action, reward, (next) state, (next) action. SARSA is also most commonly used for policy iteration. The value of Q-function in this method is updated as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \qquad \textbf{(2-1)}$$

Another popular TD method but is an offline one is Q-learning. One example of the use of Q-learning is [22]. This method is most commonly used for value iteration. The value of Q is updated with the new information according to this formula:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \qquad \textbf{(2-2)}$$

The term between square brackets in equations **(2-1**) and (**2-2**) is the temporal difference. It is the difference in estimates Q-value between the updated one with the current one. α is the learning rate that is set between 0 and 1. Setting α at 0 means that the values of Q are never updated, therefore it learned nothing. The higher the setting of α meant that the learning process can occur faster.

γ is the discount factor, also set between 0 and 1. If the value of γ is nearer to 1 then the value of future rewards are much higher than the immediate rewards. In other words, the learner aims for long-term rewards.

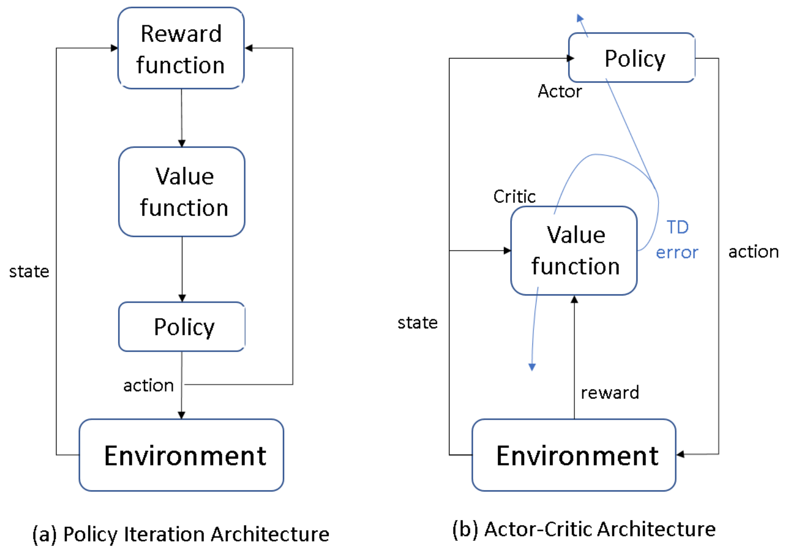


Figure 2-7 Architecture comparison [1]

An advantage in using online methods is a higher probability (guarantee) that the policy will converge[29] albeit the exploration has to be annealed over time to achieve it. Offline methods however, sometimes finds themselves diverging for certain problems. However, when it does successfully converge, it happens faster than an online method.

Another popular TD method is actor-critic method[34]. Here the memory structures for the two policies are separated. One memory structure is allocated for the policy that determines the action to choose and another is allocated for the estimated value function that evaluates those actions.

Because of the separation between actor and critic function, this method require minimal computation in order to choose an action. This is favourable for a continuous action space. However, determining the actor policy should be done with care as complete random actions may result in the RL agent not visiting the important part of the state space[35].

2.4 Reinforcement Learning in Stochastic State Space

Another problem in using RL for flight control is the stochastic nature of the flight data. To generate the policy for decision-making, the Markov Decision Process (MDP) is used as a mathematical framework to model the decision-making uncertainty [30].

However, in stochastic MDP, the next state is not deterministically given by the current state and action[7]. This represents a problem for the model-free reinforcement learning.

Previous works solved this problem by converting the stochastic problem into a deterministic one. The method is called Pegasus method [36]. The same researchers combined Pegasus method with MC method to develop a dynamic model of an unmanned helicopter [21][22]. Then the flight control is developed with conventional method based on this dynamic model. In this method, the RL agent learned from samples (flight data) previously acquired by a pilot and developed a model of the helicopter.

The drawback in this is that the model is based on a limited area of operation done by the pilot. It did not explore the state-action outside of it, so it missed out on finding other state-action pair that may have greater value function. Also, the helicopter was not controlled by the reinforcement learning controller.

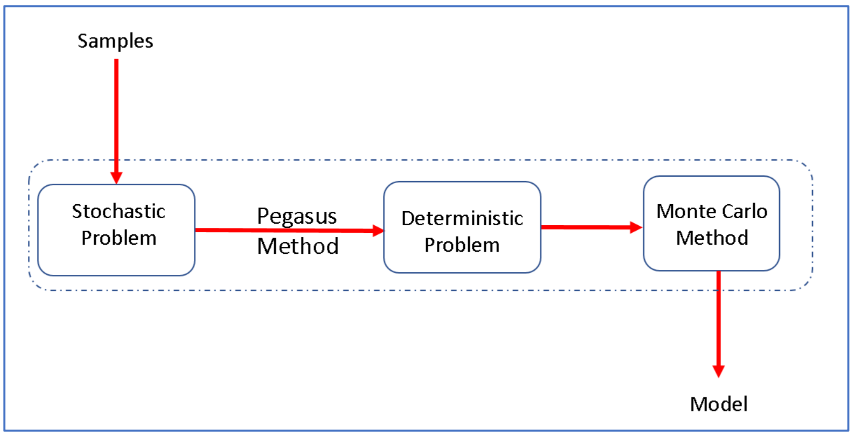


Figure 2-8 The work flow of [22][21]

However, recently develop method (2014) called Deep Deterministic Policy Gradient (DDPG) [2] has shown to be able to solve a continuous, high-dimensional and stochastic problem. This has been shown to solve several simulated physics tasks such as cart-pole problem and mountain car.

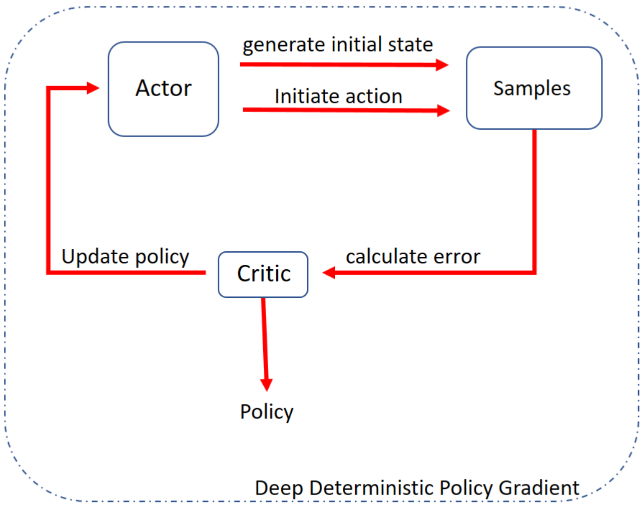


Figure 2-9 The work flow of DDPG

This method combined Deterministic Policy Gradient (DPG) [31] with Deep Q Network (DQN) [37]. The DPG method combines policy gradient and actor-critic to develop deterministic policy based on samples from stochastic policy. While the DQN can solve the continuous and high-dimensional observation spaces. By combining the two, a policy network for a continuous and high-dimensional state (observation) and action spaces can be developed.

This work focuses on developing a flight control of a full 6-degree-of-freedom air vehicle using DDPG. Specifically, it will focus on developing a learning strategy for the RL agent to develop an optimal and robust flight control.

Several works had use DDPG for controlling bicycle [38], bipedal walking[4] [39], Quadrotor [40][41], and autonomous land vehicle [5]. Yet, these works did not have as many state (observation) spaces and action spaces as an air vehicle.

Despite of the advantageous of DDPG, controlling a full 6-degree-of-freedom air vehicle is not an easy feat [42]. The RL agent has to learn how to use the elevator to control the longitudinal mode of the air vehicle. Then it also has to learn to control both aileron and rudder simultaneously to control the lateral-directional mode.

Aside from those fixed operating procedures, there is also the varying operating condition. This would require the RL agent to learn to control the air vehicle when there exist changes in its altitude or airspeed. These entire training goals requires a learning strategy in order to develop the appropriate policy quickly. Without a learning strategy, the RL agent can take a very long time to develop as it sorts through every possible state and action.

2.5 Conclusion

As with any other methods, there is disadvantages along with the advantages in using DDPG. Below is the list of both advantages and disadvantages of DDPG method.

Table 2-1 Advantages and Disadvantages of DDPG

No.	Advantages	Disadvantages
1.	DDPG can be applied for problems with large and continuous state and action spaces.	Difficulty to achieve convergence in performance if there are more than one action variable
2.	The behaviour policy and the target policy are separated. Exploring the action spaces will not immediately affect the behaviour policy.	A problem in eliminating chattering in the action performance. An adjustment in the reward function only limits the range of the action but not the chattering.
3.	Optimisation in the learning process is enhanced by learning using a replay buffer.	

This research proposes a learning strategy that guarantee the best policy will represent the optimal performance of the air vehicle within a reasonable development time. The next chapter will describe the methodology of deep deterministic policy gradient and a general description of how to determine the reward function and the learning strategy.

3 THEORY/METHODOLOGY

3.1 Introduction

The focus of this research is to generate a reinforcement learning policy that can adequately control an air vehicle without the need to have its dynamic model. In other words, utilizing reinforcement learning to generate a model-free flight control. To achieve this, the reinforcement-learning (RL) agent needs to learn the flight characteristic of the air vehicle by interacting with it. This interaction is represented as samples of flight data, demonstrating how the attitude (state) of the air vehicle changes due to changes in control surface (action) given by the agent.

Figure 3-1 and Figure 3-2 below shows how an RL agent generally learn a policy by using samples. The RL agent gives numerical value (reward) to each state-action pair in regards to the desired state. Then the RL agent iterates the value function(v_t) from this reward until it can reach an optimal policy(Q_t).

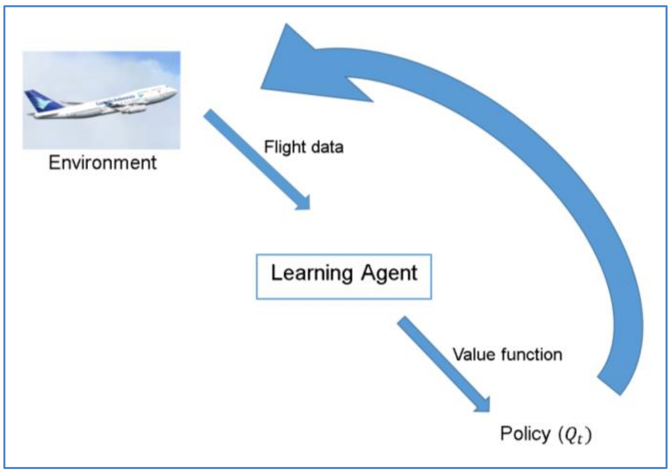


Figure 3-1 A Model-Free Reinforcement Learning Flow

The flight data provides the attitude of the air vehicle that is being observed or state(s_t, s_{t+1}). It also provides the action(a_t) that caused the change in those states. A reward value is given to these sets of state and action (s_t, a_t, s_{t+1}) and this will determine the value of the state-action pairing(s_t, a_t). The RL agent then compiles these value functions into a policy.

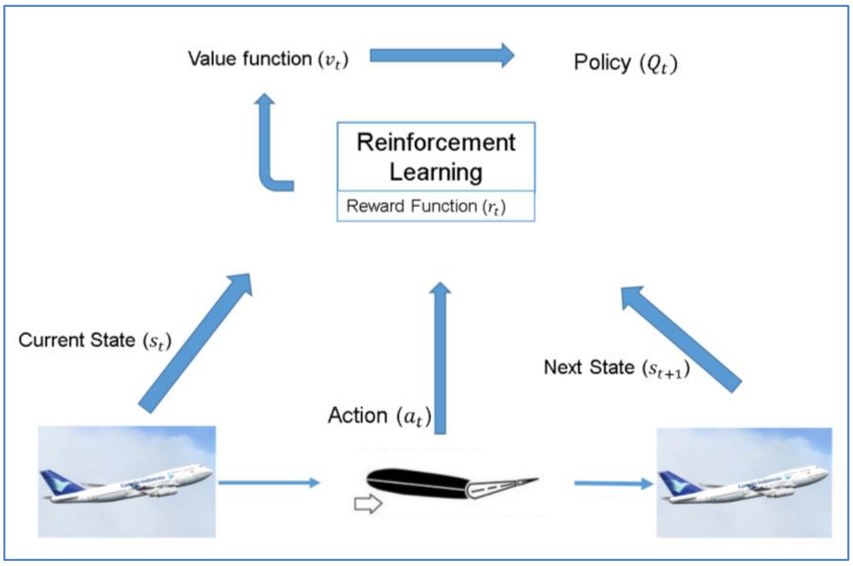


Figure 3-2 Determining the Policy

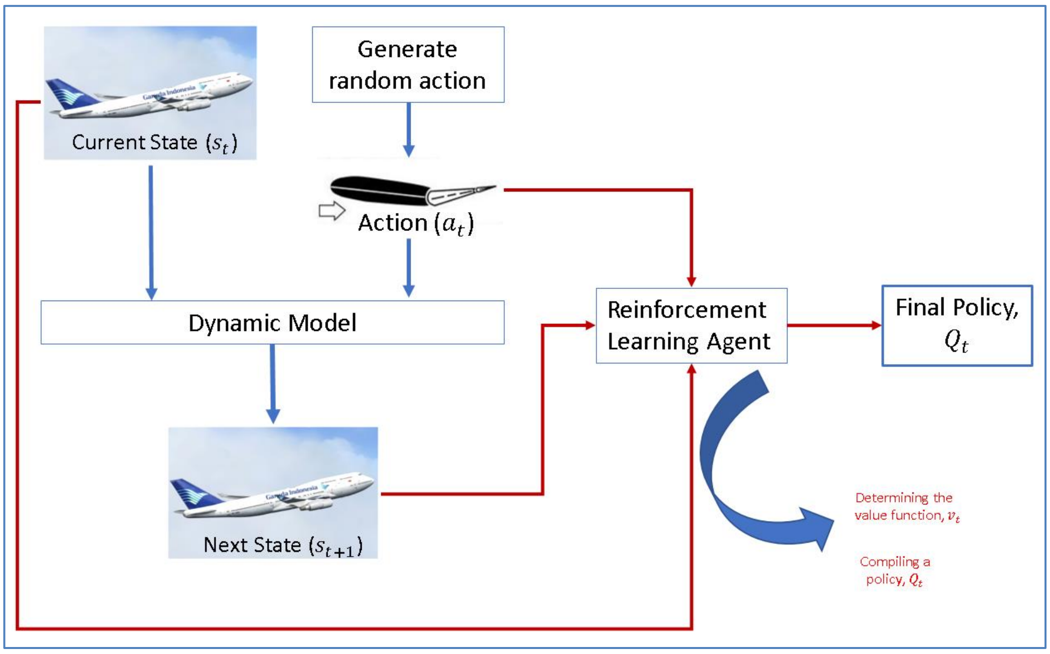


Figure 3-3 Generating Samples for RL Agent Learning Process

In the absence of flight data, the samples can be generated by simulation using a dynamic model of the air vehicle. This dynamic model exist outside of the learning process. First, a random action(a_t) is generated and applied to an initial state(s_t). Then the dynamic model produced the appropriate resulting state(s_{t+1}). Once these three sets of data are available, then the RL agent can generate the optimal policy as described in Figure 3-2. Figure 3-3 demonstrates clearly that the RL agent still learns without the use of a dynamic model.

3.2 Reinforcement Learning

In reinforcement learning, the RL agent is interacting with the environment in discrete timesteps. Here, the environment is the air vehicle. At each timestep t, the RL agent receives an observation x_t, takes an action a_t and receives a scalar reward r_t for the state-action pair. In this environment, it is assumed that:

1. the actions are real-valued $a_t \in \mathbb{R}^N$
2. the environment is fully observable, therefore the initial state is the observation, $s_t = x_t$.

An RL agent's behaviour is defined by a policy,π, which maps states to a probability distribution over the actions $\pi: S \to P(A)$. The environment, E, may also be stochastic. We model it as a Markov decision process with a state space S, action space $A = \mathbb{R}^N$, an intial state distribution $p(s_1)$, transition dynamics $p(s_{t+1}|s_t, a_t)$, and reward function $r(s_t, a_t)$.

The return from a state is defined as the sum of discounted future reward $R_t = \sum_{i=t}^{T} \gamma^{(i-t)} r(s_i, a_i)$ with a discounting factor $\gamma \in [0,1]$. This depends on the actions chosen, therefore also depends on the policy π, which may be stochastic. Mathematically, the goal in reinforcement learning is to learn a policy which maximizes the expected return from the start distribution $J = \mathbb{E}_{r_i, s_i \sim E, a_i \sim \pi}[R_1]$. The discounted state visitation distribution for a policy π is denoted as ρ^{π}.

The action-value function describes the expected return after following policy π by taking an action a_t in state s_t :

$$\mathrm{Q}^{\pi}(s_t, a_t) = \mathbb{E}_{r_{i \geq t}, s_{i>t} \sim E, a_{i>t} \sim \pi}[R_t | s_t, a_t] \tag{3-1}$$

Here the reinforcement learning uses the Bellman equation:

$$\mathrm{Q}^{\pi}(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}\left[r(s_t, a_t) + \gamma \mathbb{E}_{a_{t+1} \sim \pi}[\mathrm{Q}^{\pi}(s_{t+1}, a_{t+1})]\right] \tag{3-2}$$

3.3 Deterministic Policy Gradient

In 2014, [31] proposed DPG to develop a deterministic policy from stochastic samples. DPG stands for Deterministic Policy Gradient. It is a method that combines the use of policy gradient and actor-critic algorithm, which are widely used for reinforcement learning with continuous action spaces, by using policy gradient algorithm to drive the stochastic behaviour policy to a deterministic target policy.

For a deterministic target policy, the Bellman equation can described as a function $\mu: S \leftarrow A$:

$$\mathrm{Q}^{\mu}(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}[r(s_t, a_t) + \gamma \mathrm{Q}^{\mu}(s_{t+1}, \mu(s_{t+1}))] \tag{3-3}$$

The expectation depends only on the environment. This means that it is possible to learn Q^{μ} off-policy, using transitions which are generated from a different stochastic behaviour policy β.

Q-learning is the most commonly used off-policy algorithm and it uses the greedy policy $\mu(s) = arg \max_{a} Q(s, a)$. However, Q-learning cannot be applied directly to continuous action spaces because finding its greedy policy requires an optimization of a_t at every timestep. This optimization is too slow to be practical with large, unconstrained function approximators and nontrivial action spaces. Therefore, the off-policy algorithm uses actor-critic approach based on DPG algorithm.

In this method, While the critic criticize the action based on the result.

The DPG algorithm applies a parameterized actor function $\mu(s|\theta^{\mu})$ which specifies the current policy by deterministically mapping states to a specific action. This means that the actor selects actions to apply on various initial states or observations based on its current policy(μ) or stochastic behaviour policy.

Then the critic makes an estimate of the value function and updates the policy(Q) or deterministic target policy. The critic $Q(s, a)$ is learned using the Bellman equation as in Q-learning.

The actor is updated by following the applying the chain rule to the expected return from the start distribution J with respect to the actor parameters:

$$\nabla_{\theta^{\mu}} J \approx \mathbb{E}_{s_t \sim \rho^{\beta}}\left[\nabla_{\theta^{\mu}} \mathrm{Q}(s, a|\theta^{Q})|_{s=s_t, a=\mu(s_t|\theta^{\mu})}\right] \tag{3-4}$$

$$= \mathbb{E}_{s_t \sim \rho^{\beta}}\left[\nabla_{a} \mathrm{Q}(s, a|\theta^{Q})|_{s=s_t, a=\mu(s_t)} \nabla_{\theta_{\mu}} \mu(s_t|\theta^{\mu})|_{s=s_t}\right]$$

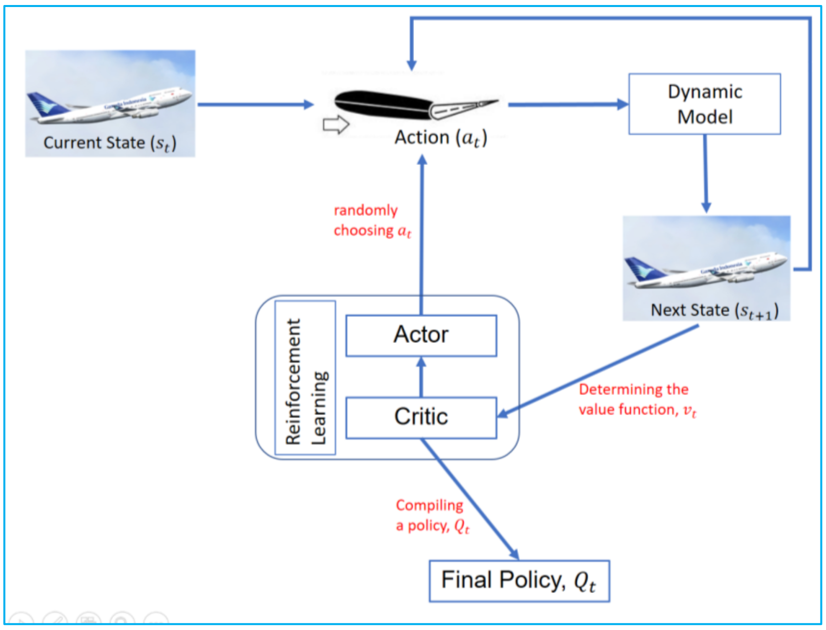


Figure 3-4 Actor-Critic Method

3.4 Deep Deterministic Policy Gradient

Deep Deterministic Policy Gradient is first introduced in 2015. This method combined Deterministic Policy Gradient (DPG) with Deep Q network (DQN). In 2013, [37] showed how an RL agent produce a policy to control a system with a high-dimensional observation states. This RL agent uses deep learning.

DDPG basically uses neural network function approximators (DQN) to learn in large state and action space online. One challenge when using neural networks for reinforcement learning is that most optimization algorithms assume that the samples are independently and identically distributed. Obviously, when the samples are generated from exploring sequentially in an environment this assumption no longer holds. Additionally, to make efficient use of the hardware optimizations, it is essential to learn in mini-batches, rather than online.

As in DQN, a replay buffer is used to address these issues. The replay buffer is a finite sized cache $\mathcal{R}$. Transitions were sampled from the environment according to the exploration policy and the tuple (s_t, a_t, r_t, a_{t+1}) stored in the replay buffer. When the replay buffer was full, the oldest samples were discarded. At each timestep, the actor and critic are updated by sampling a minibatch uniformly from the buffer. Because DDPG is an off-policy algorithm, the replay buffer can be large, allowing the algorithm to benefit from learning across a set of uncorrelated transitions.

Directly implementing Q learning with neural networks proved to be unstable in many environments. Since the network $Q(s, a|\theta^Q)$ being updated is also used in calculating the target value, the Q update is prone to divergence. The solution is to create a copy of the actor-critic networks $Q'(s, a|\theta^{Q'})$ and $\mu'(s|\theta^{\mu'})$ respectively, that are used for calculating the target values. The weights of these target networks are then updated by having them slowly track the learned network: $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$ with $\tau \ll 1$. This means that the target values are constrained to change slowly, greatly improving the stability of learning. This simple change moves the relatively unstable problem of learning the action-value function closer to the case of supervised learning, a problem for which robust solutions exist. Having both a target μ' and Q' is required to have stable targets y_i in order to consistently train the critic without divergence.

$$y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'}\right) \tag{3-5}$$

The critic is then updated by minimizing the loss:

$$L = \frac{1}{N}\sum_i \left(y_i - Q(s_i, a_i|\theta^Q)\right)^2 \tag{3-6}$$

This may slow learning, since the target network delays the propagation of value estimations. However, in practice we found this was greatly outweighed by the stability of learning.

Algorithm 1[2]

```
Randomly initialize critic network $Q(s,a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$
Initialize target network $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer $R$
for episode = 1, M do
    Initialize a random process $\mathcal{N}$ for action exploration
    Receive initial observation state $s_1$
    for t = 1, T do
        Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
        Execute action $a_t$ and observe reward $r_t$ and observe new state $s_{t+1}$
        Store transition $(s_t, a_t, r_t, s_{t+1})$ in $R$
        Sample a random minibatch of N transitions $(s_i, a_i, r_i, s_{i+1})$ from $R$
        Set $y_i = \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1}|\theta^{\mu'}\right)|\theta^{Q'}\right)$
        Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i\left(y_i - Q(s_i, a_i|\theta^Q)\right)^2$
        Update the actor policy using the sampled policy gradient:
            $\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s,a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$

        Update the target networks:    $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
                                       $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$

    end for
end for
```

To generalise the parameters across environments with different scales of state values, these features are scaled by adapting a recent technique from deep learning called batch normalization. This technique normalizes each dimension across the samples in the minibatch to have unit mean and variance.

A major challenge of learning in continuous action spaces is exploration. An advantage of DDPG, as is any other off-policy algorithms, is that the exploration problem can be learned independently. An exploration policy μ' is constructed by adding noise sampled from a noise process N to actor policy.

$$\mu'(s_t) = \mu\left(s_t|\theta_t^\mu\right) + N \qquad \textbf{(3-7)}$$

N is chosen to be an Ornstein-Uhlenbeck process.

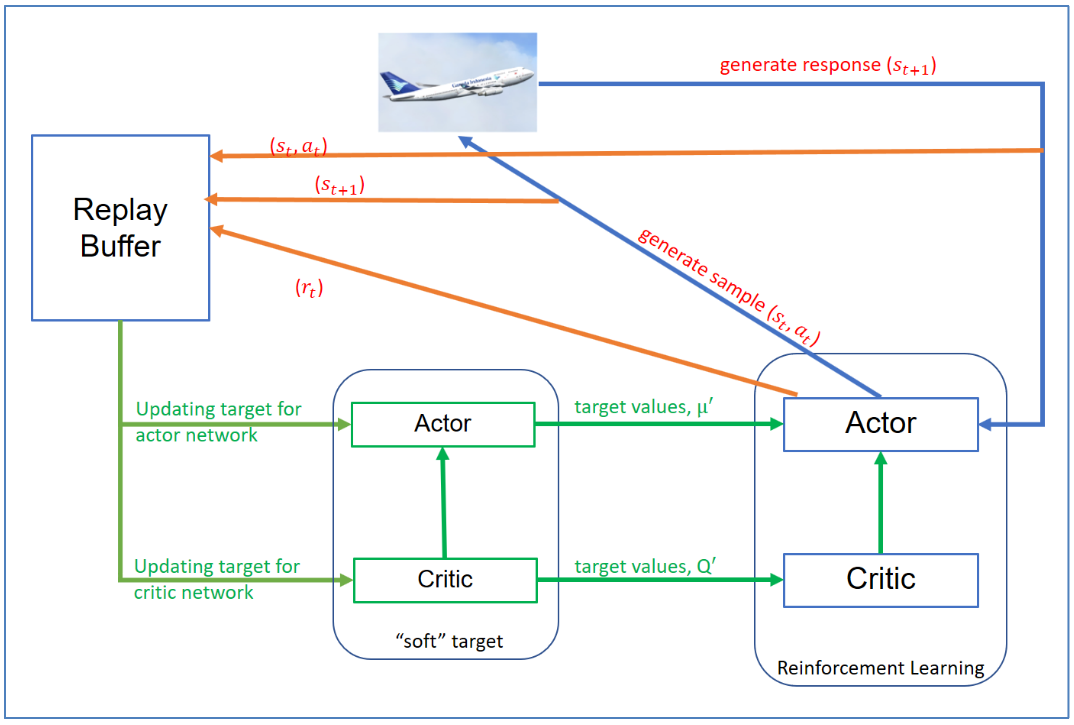


Figure 3-5 DDPG Method

3.5 Formulating The Reward Function

The general process of reinforcement learning is that the agent learns through a series of episodes. These episodes are called training episodes. At the end of the training, the resulting policy is assumed to be the optimal policy and can be applied in an air vehicle.

Two aspects need to be addressed to determine whether a policy is optimal and can be applied in an air vehicle. One is that the return (total rewards) of the training episodes converges as the number of episodes increases. This shows that the RL agent has found an optimal policy that the RL agent will visit often. However, this doesn't mean that the performance of the air vehicle is as desired.

This leads to the second aspect, which is making sure that the best performance, according to the RL agent, coincides with the best performance of the air vehicle itself. With only the desired final state as a goal, the agent can find many ways to achieve it. More than one flight performance can be considered suitable to achieve the final state from the initial state. However, the RL agent will consider the fastest way to achieve the goal is the best performance. Not the optimal one.

Therefore, this work proposed to define an optimal trajectory for the RL agent to follow. This trajectory should be guarantee that the flight performance that is desired. This trajectory started at the same value as the initial state, while the changes in trajectory is defined as follows:

$$\dot{\eta}_{ref} = -2\xi_{ref}\omega_{n_{ref}}\eta_{ref} - \omega_{n_{ref}}^{2}(\eta_{ref} - \eta_t) \qquad \textbf{(3-8)}$$

$\dot{\eta}_{ref}$ is defined as the flight attitude that intends to be controlled by the control surface. In this work, $\dot{\eta}_{ref}$can be the pitch angle ($\dot{\theta}_{ref}$), the roll angle ($\dot{\phi}_{ref}$) and the yaw angle ($\dot{\psi}_{ref}$). ξ_{ref} is a damping ratio of 1 and $\omega_{n_{ref}}$ is the natural frequency of 2π.

In RL, the reward function gives a numerical value to a state compared to the final desired state. In order to compare the performance of each episode, the possible reward values has to have the same limit that defines the best value. Therefore, the reward function is defined as a negative value.

$$R = -f(S_t) \qquad \textbf{(3-9)}$$

More detail on this work is described in chapter 4.

3.6 Learning Strategy

In order to develop a reinforcement learning agent that can control an air vehicle in 6-degree-of-freedom, this research is divided into three separate stages based on the number of action variable involved.

1) For single action variable, the study case will involve training a reinforcement learning agent to control the longitudinal mode of an air vehicle. It focuses on controlling the pitch angle of the air vehicle.
2) For dual action variable, the study case will involve training a reinforcement learning agent to control the lateral-directional mode of an

air vehicle. It is trained to control the roll angle, yaw angle and both roll-yaw angle together.

3) The study case for training a reinforcement learning agent to control a 6-degree-of-freedom will involve three action variables which are elevator, aileron and rudder.

Designing the training steps for the first stages will consist of 2 main goals. They are:

1) Learning how to use its control surface. This translate to learning how much impact the changes in a control surface has on its overall state. the RL agent is trained to utilize the elevator in both directions to reach its goal to change the pitch angle.
2) Learning how to achieve certain attitude in different conditions (such as different airspeed and altitude).

For the second goal of this stage, several simulations are run with different set of variables for state or observation. These simulations will also have different layers in the network.

For the second stage, there is a slightly different training strategy as it needs to:

1) Learning how to use 2 control surfaces. One (aileron) has slightly more influence than the other (rudder).
2) Learning coordinate the movement of two different control surfaces. This is due to the nature of the lateral and directional mode of an air vehicle are usually coupled. But setting the RL agent to explore and find the perfect combination on its own could take a long time as it presents a large number of action possibilities.

Therefore, the RL agent is trained to first control the lateral mode and the directional mode by its own. Then they are followed by a training series that coordinated both control surface. This will ensure that the RL agent will find the optimal policy in a relatively faster time.

Training the reinforcement agent for a complete 6-degree-of-freedom air vehicle (stage 3) will focus on:

1) Learning how to use 3 control surfaces (elevator, aileron, and rudder).
2) Learning to coordinate the movement of three different control surfaces with different degree of influence for different mode (longitudinal, lateral, directional).
3) Extracting the resulting policy (network)

In longitudinal mode, the influence of aileron and rudder are very small compared to the elevator. But in lateral-directional mode, the degree of influence is reversed. Elevator movement has a small impact when the agent is trying to control its roll angle but it has a large impact when it is trying to control the pitch angle. The reinforcement learning agent has to developed its knowledge when moving in and out of a certain flight mode.

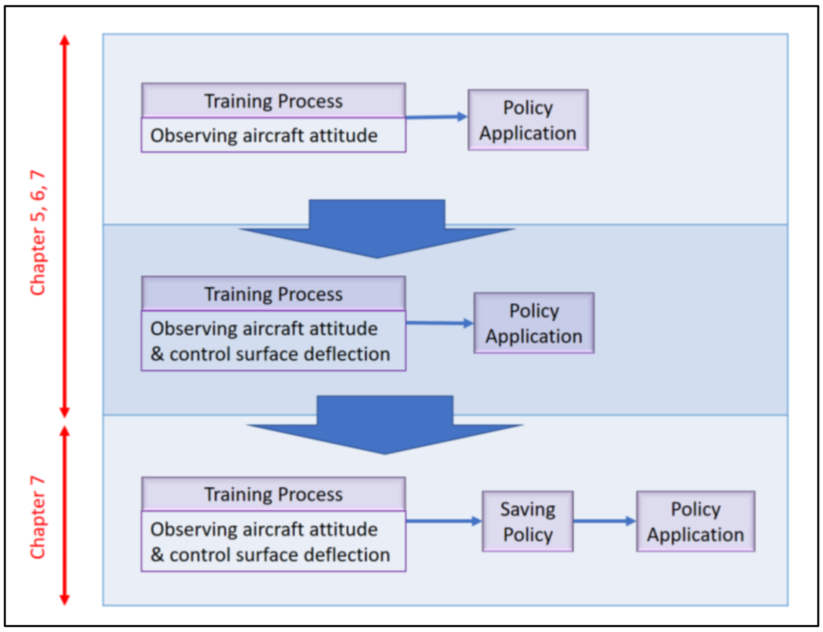

Chapter 5, 6, 7
Chapter 7
Training Process
Observing aircraft attitude
Policy Application
Training Process
Observing aircraft attitude & control surface deflection
Policy Application
Training Process
Observing aircraft attitude & control surface deflection
Saving Policy
Policy Application

Figure 3-6 Policy Application After Training Process

Extracting the policy (network) is also very important. Because unlike developing an AI to control a robot, an air vehicle can't go back up after it has crashed during the first few episodes where it is trying to gain stability in the system. In order to use this flight control, the reinforcement learning trains in a simulation or trains using a recorded flight data. Then the result can be used or uploaded to an on-board system of an air vehicle.

3.7 Conclusion

This chapter describes the way DDPG method works and how the programme is adjusted to develop a DDPG agent that can control the flight of an air vehicle. It is assumed to be a straight forward affair in adopting DDPG in flight control.

However, as the investigation progressed changes were made in the training procedure and even in the way to define a policy is suitable or not. The following chapters will explain these discoveries.

4 DETERMINING THE OBJECTIVE OF THE RL AGENT

4.1 Introduction

A major issue in using reinforcement learning in developing a flight control is “how to formulate the reward function to best represent the way to control the aircraft?”. As mentioned earlier in sub chapter 3.5, the reward function gives value to a pair of state-action. The goal of the RL agent is to find the pair with the best value. In the physical term, it would be to find the best action for each and every state.

The first step is to determine is the state to be observed by the RL agent. In the aircraft system, the state represents the attitude of the aircraft. Therefore, the determined flight variables to be observed (state) will be the ones that the RL agent will look out for and use it for guidance to achieve its goal.

The second step is to find a common goal for each training episode. Each training needs to have the same objective to achieve so that the result of each episode can be compared to one another. For each and every initial state, the appropriate action needs to be found, so that the final goal can be achieved from whatever state it initially started.

4.2 Determining the State

For this chapter, the case study is to develop a flight control for the longitudinal mode of a missile. The objective is to control the pitch angle (θ). So, pitch angle should be one of the states to be observed, as it is the attitude variable, the RL agent has to control.

However, just using the pitch angle as the state is not enough. In order for the RL agent to control the pitch angle, it uses the elevator deflection (δE). Yet, the elevator deflection doesn’t directly control the pitch angle. So there will be a gap in the RL agent’s knowledge in how to achieve a pitch angle with a certain value.

$$a = [\delta E] \tag{4-1}$$

Due to this reason, the pitch rate (q) is added to the state. The pitch rate will directly show the RL agent how each elevator deflection influenced the changes in q and therefore, changes the pitch angle from its initial value. So at the start of this investigation, the state to be observed is proposed as:

$$s = [q \quad \theta] \tag{4-2}$$

4.3 Model for Data Generation

The model of missile that is used for the simulation is a nonlinear model from[43]. The longitudinal mode of the missile model can be seen as follows. **Table 4-1** consists of the configuration of the missile and the aerodynamic data that correlates to that configuration.

$$\dot{\alpha} = \frac{QS}{m}\frac{Cz_{\alpha}}{V}\alpha + q + \frac{QS}{m}\frac{Cz_{\delta E}}{V}\delta E \tag{4-3}$$

$$\dot{q} = \frac{QSd}{I_{yy}}(X_{cp} - X_{cg})\frac{Cz_{\alpha}}{d}\alpha + \frac{QSd}{I_{yy}}\frac{d}{2V}Cm_{q}q + \frac{QSd}{I_{yy}}Cm_{\delta E}\delta E \tag{4-4}$$

$$\dot{\theta} = q \tag{4-5}$$

Table 4-1 Parameters of Missile

Mass : 200 kg	Diameter (d): 0.3 m	$X_{cp} = X_{cg} + d$	V : 300 m/s
Iyy : 450 $kg.m^2$	X_{cg} : 2.5 m	ρ : 1.21 kg/m	
Cz_{α} : -6.0	Cm_{α} : -500.0	$Cz_{\delta E}$: -0.1340	$Cm_{\delta E}$: -26.180

4.4 Reward Function I

In the first method, the state is defined as (4-2). The reward function in this method is defined as the square error between the desired pitch angle (θ_d) and the current pitch angle (θ). The objective is for the RL agent to minimize this and achieve the desired pitch angle (θ_d).

$$r = -(\theta - \theta_d)^2 \tag{4-6}$$

The desired pitch angle is a constant value throughout the entire timestep. The error is squared so that it will eliminate the influence of positive or negative value of the error and focuses solely on the difference itself.

For this case study, the number of layers for the network are respectively 300 and 500. The parameters for the noise factor in the actor policy is listed in **Table 4-2**

Table 4-2 Ornstein-Uhlenbeck Parameter for Simulation 04.01

$\theta_{OU} = 0.3$	$\sigma_{OU} = 0.075$

Figure 4-2 showed that the RL agent failed to achieve the desired pitch angle value even though the total reward per episode during the training process clearly showed convergence, therefore indicating that the RL agent has learned the system Figure 4-1.

However, the problem with this simulation is that by using the pitch angle (θ) as one of the observed states, it means that the training set would have to train the RL agent through all the initial pitch angle value and all the possible pitch angle desired. This is ineffective. A solution for this problem is proposed in the next sub chapter.

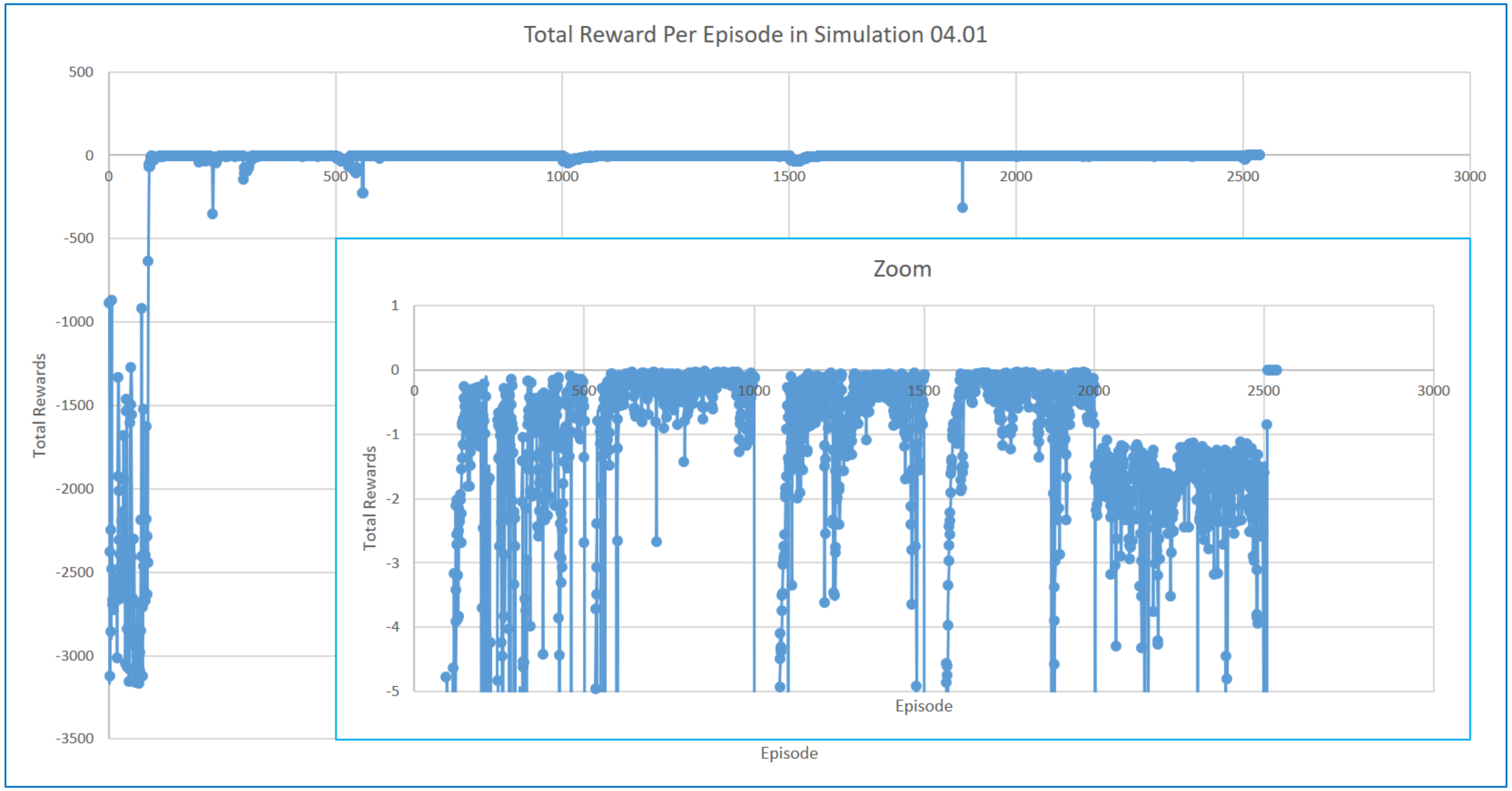


Figure 4-1 The Total Reward Per Training Episode in Simulation 04.01

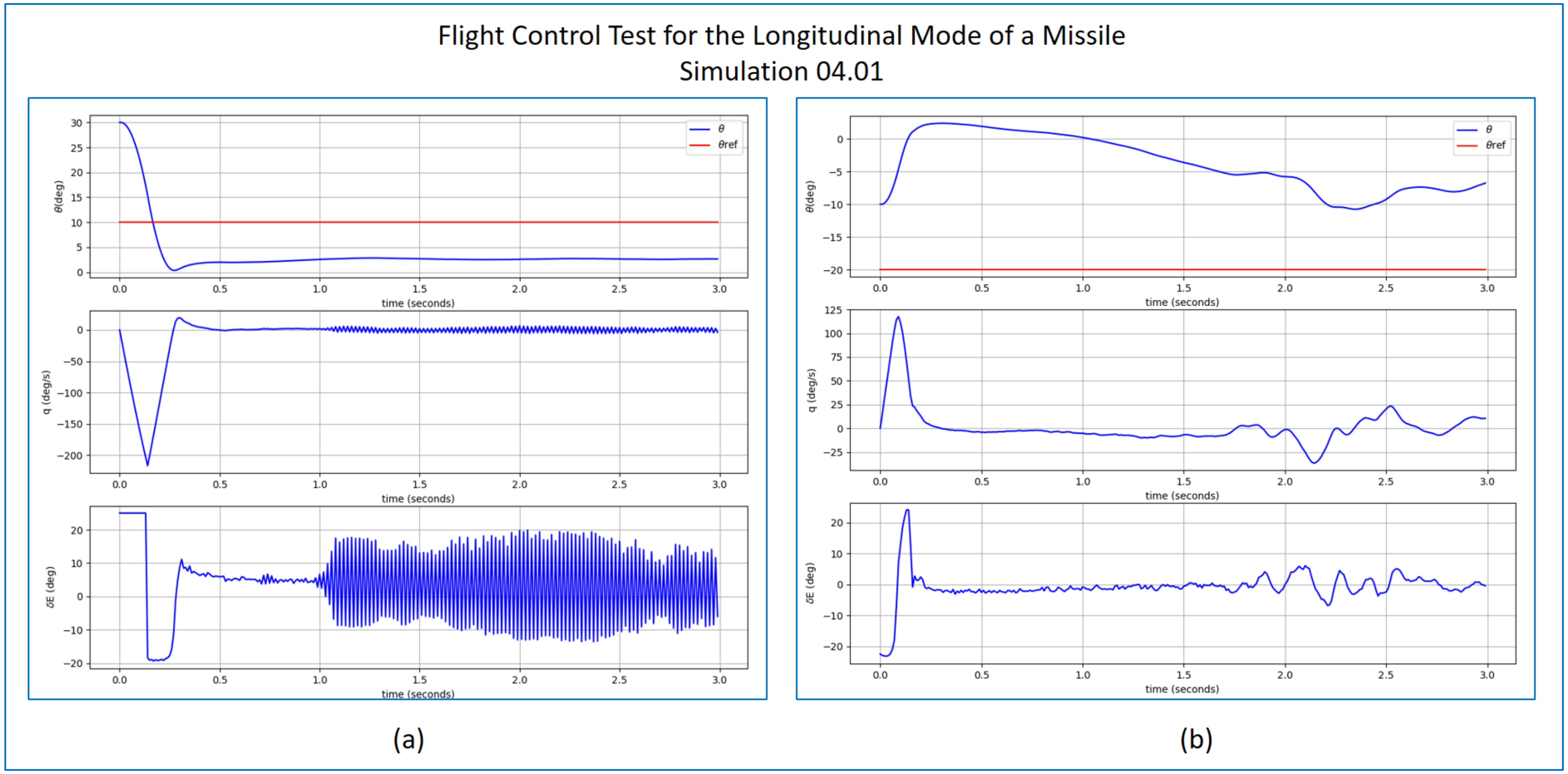


Figure 4-2 The Result of Test in Simulation 04.01

4.5 Reward Function II

As described in the previous sub chapter, it is ineffective to train the RL agent through all the values of the initial pitch angle and towards all the possible desired pitch angle. To shorten this training process, a change is proposed in the state definition. Instead of the pitch angle (θ), it should be the error of the pitch angle itself (θ_{err}).

$$s = [q \quad \theta_{Err}] \tag{4-7}$$

With the same reward function as in (4-6), simulation 04.02 is done with the same number of network layers and the same noise parameter.

The second method seems to produce a better result than the first method. But it does produce a question whether or not the best performance has been reached. As seen in the result, the bigger value of total reward does show a performance with the quickest response to achieve the desired pitch angle. It doesn't show the best performance. It only shows the goal of the pitch angle. In this method, there is always a possibility that a better performance can be produce after the best one.

Also, a question in this method is whether the manoeuvre chosen by the RL agent is possible with the limitations of the aircraft's control surface deflection. In a commercial flight, a pitch rate bigger than $28^0/s$ can exceed the structural capability of the aircraft.

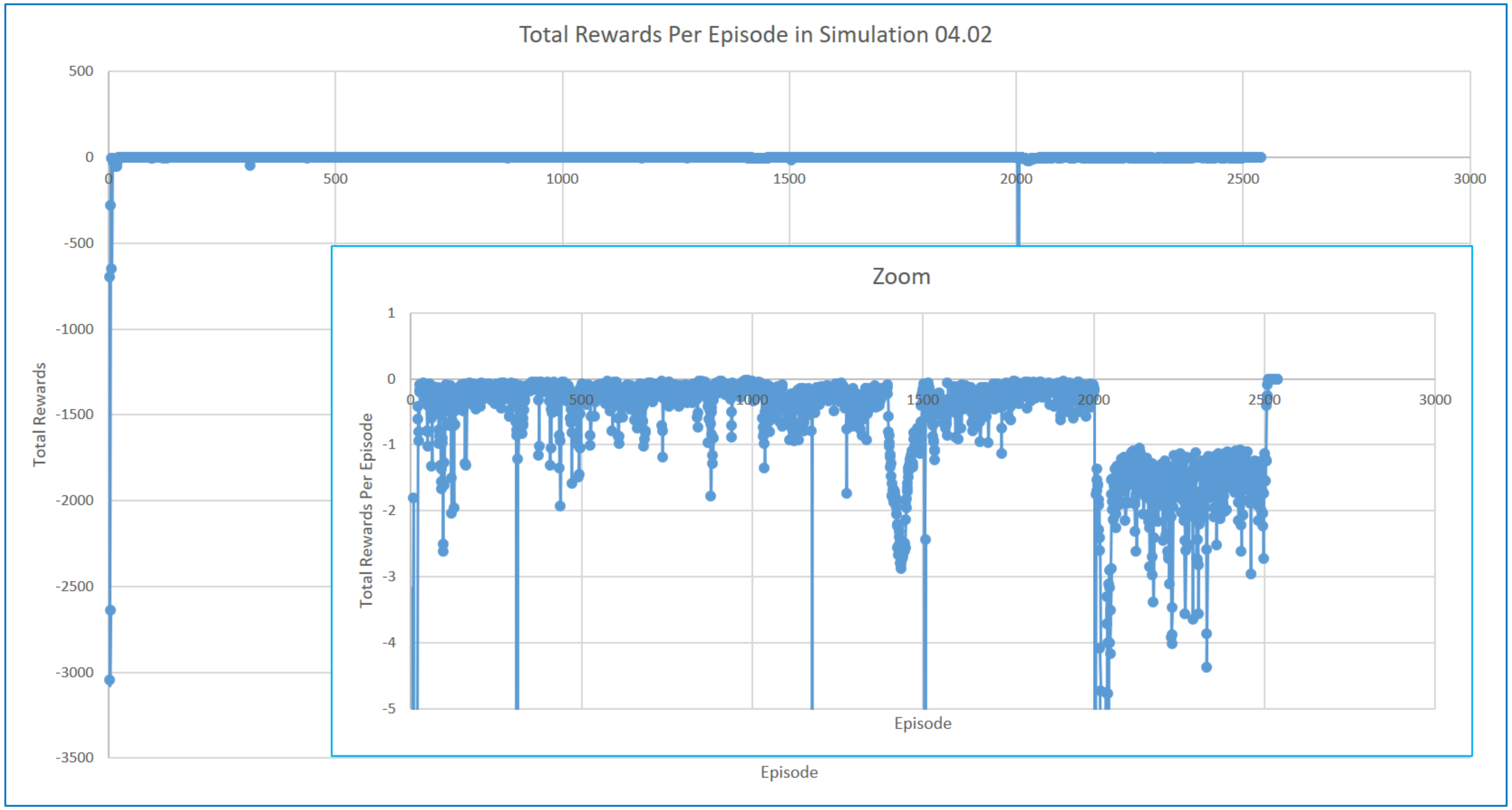


Figure 4-3 The Total Reward Per Training Episode in Simulation 04.02

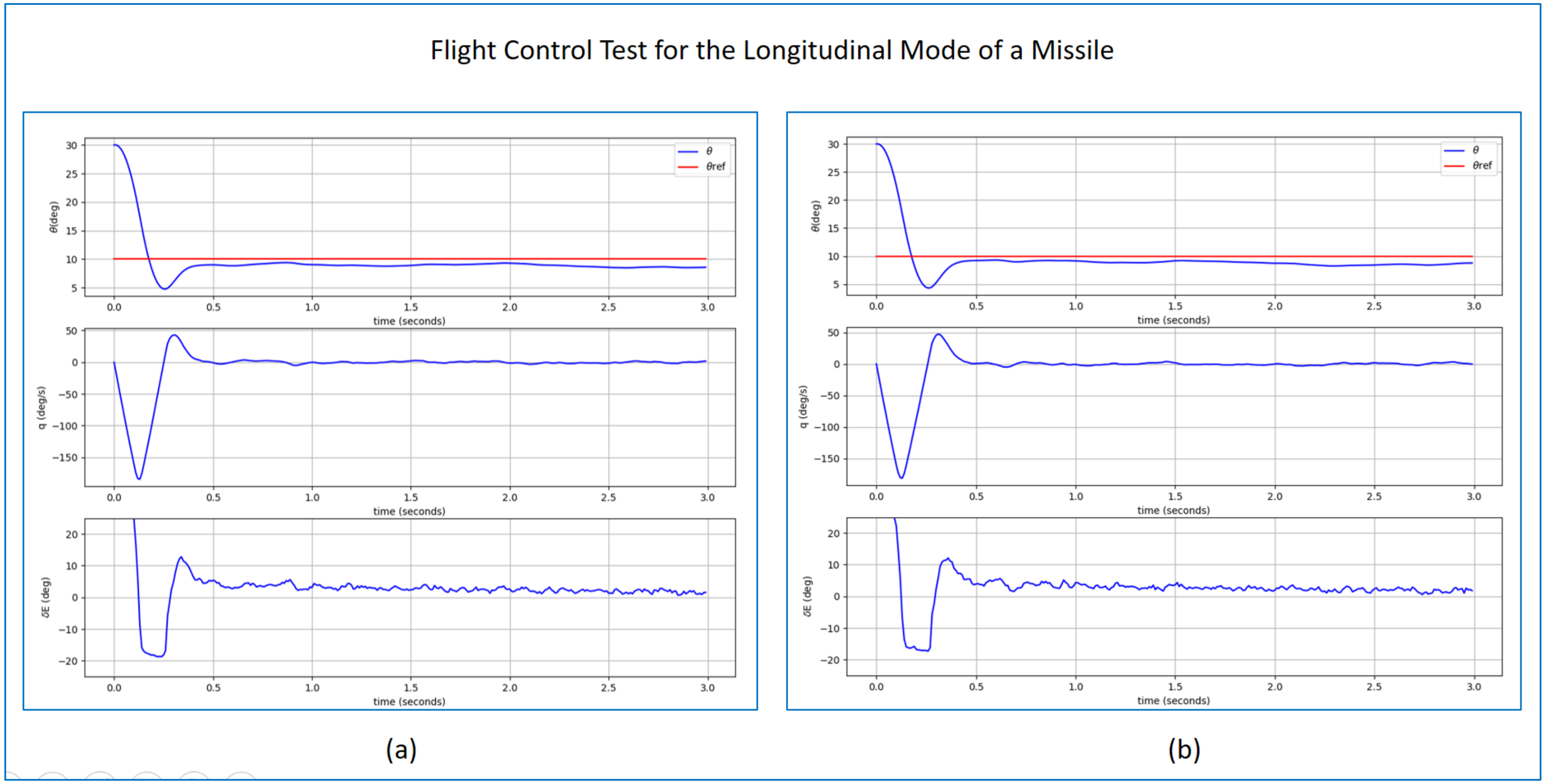


Figure 4-4 The Result of Test in Simulation 04.02

4.6 Reward Function III

Based on the previous sub chapter, there is a problem regarding the performance of the RL agent. There is no guarantee that the best performance produced is the best performance overall. Nor is there any sort of measurement of how far away is the resulting performance compared to the best performance.

A solution to this particular question is to create an optimal trajectory for the RL agent to follow. This defines the best performance as this optimal trajectory. It will provide the RL agent with a benchmark for it to follow and/or to measure its performance against. In other words, this provides a guarantee of what the best performance looks like.

So, if y_{ref} is the optimal trajectory of the changes in pitch angle, then y is the current changes of pitch angle that the RL agent has executed. The reward function is defined as follows.

$$r = -\left(y - y_{ref}\right)^2 \tag{4-8}$$

The learning agent is learning to produce a performance result as close as possible with the reference performance. The reward value of the best performance can be very close to zero, as depicted the second performance in Figure 4-5.

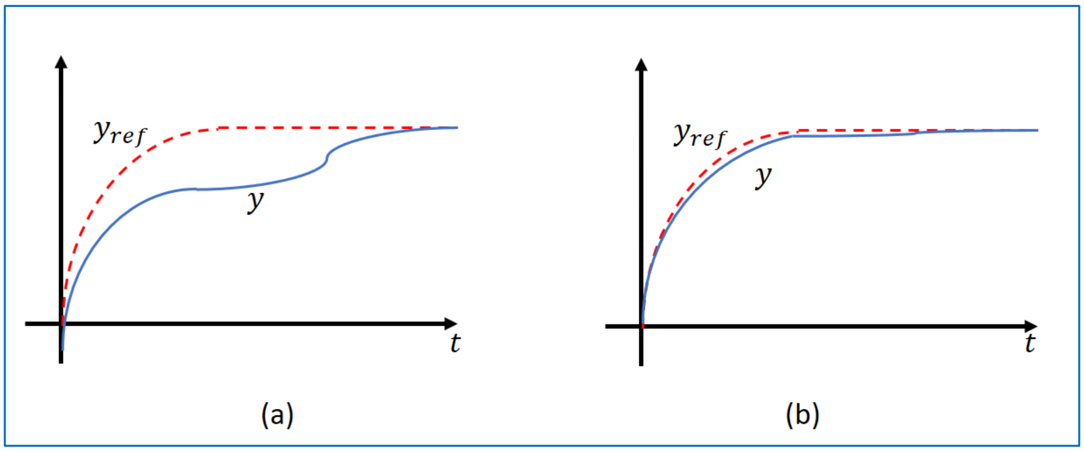


Figure 4-5 Two Potential Performance with a Reference Trajectory

This trajectory started at the same value as the initial state, while the changes in trajectory is defined as follows:

$$\dot{\eta}_{ref} = -2\xi_{ref}\omega_{n_{ref}}\dot{\eta}_{ref} - \omega_{n_{ref}}^{2}(\eta_0 - \eta_t) \tag{4-9}$$

$\dot{\eta}_{ref}$ is defined as the flight attitude that intends to be controlled by the control surface. In this work, $\dot{\eta}_{ref}$ can be the pitch angle ($\dot{\theta}_{ref}$), the roll angle (ϕ_{ref}) and the yaw angle ($\dot{\psi}_{ref}$). ξ_{ref} is a damping ratio of 1 and ($\omega_{n_{ref}}$) is the natural frequency of 2π.

Figure 4-7 showed that the RL agent manage to follow the reference trajectory that even though there is still a steady state error. The performance with the best value in total reward in this method truly represents the desired performance demanded from the aircraft.

However, there is also a disadvantage in only identifying the magnitude of the error but not when it occurs. The same value of total reward can have two significantly different performance results, as shown in Figure 4-8.

The first performance in Figure 4-8 is convergent while the second performance is divergent. The first performance is the performance result that is desired. Therefore, this reward function doesn't suitably represent the relationship between the reward value and the performance.

This problem can be easily tackled by the adding a time component $W(t)$. By multiplying the squared error with $W(t)$, the total reward value would differ between the first performance in Figure 4-8 and the second. This time component will guarantee that divergent performance will result in a large negative reward value.

$$r = -(\theta - \theta_d)^2 W(t) \tag{4-10}$$

$$W(t) = 0.01nStep \tag{4-11}$$

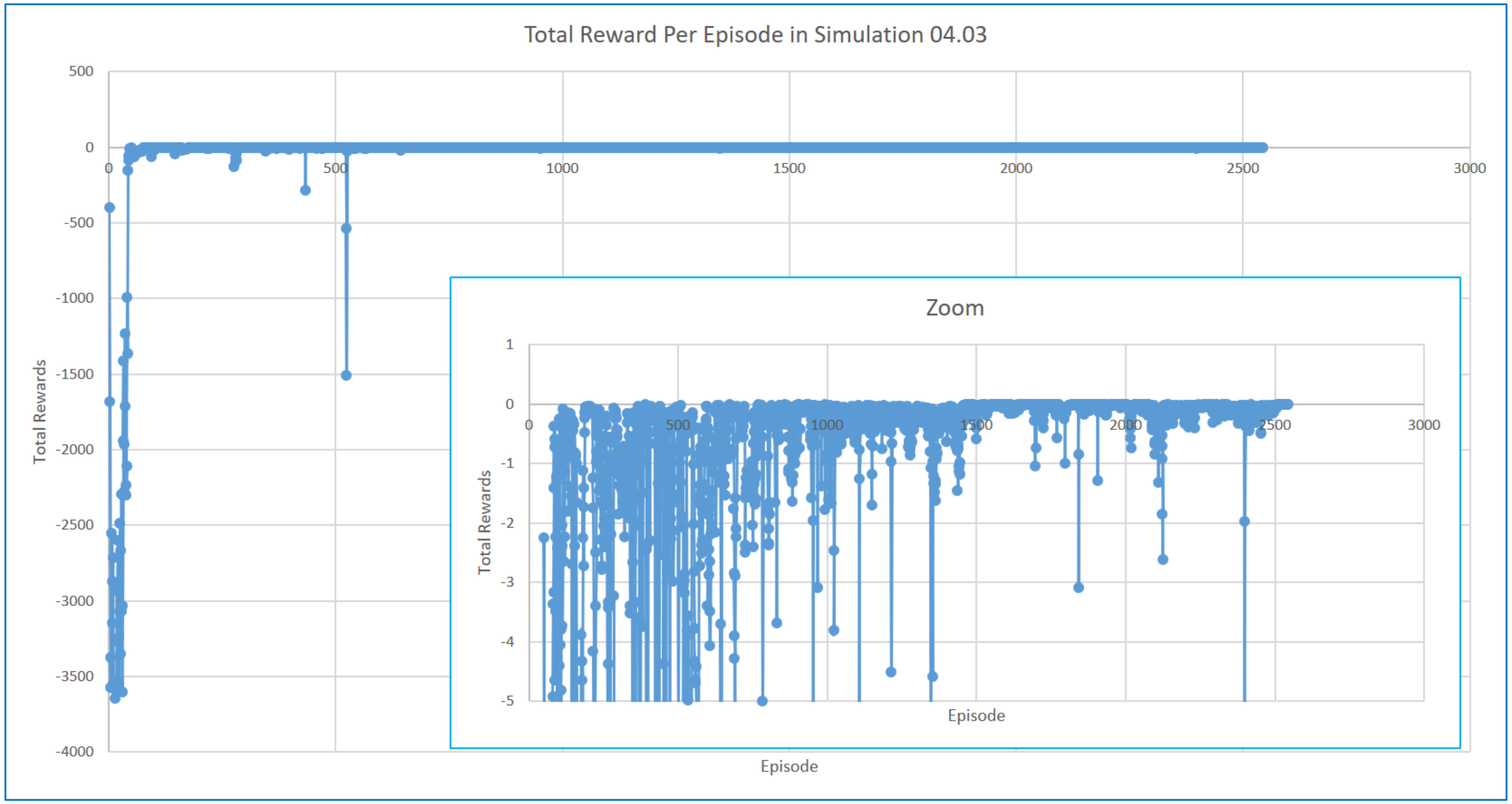


Figure 4-6 The Total Reward Per Training Episode in Simulation 04.03

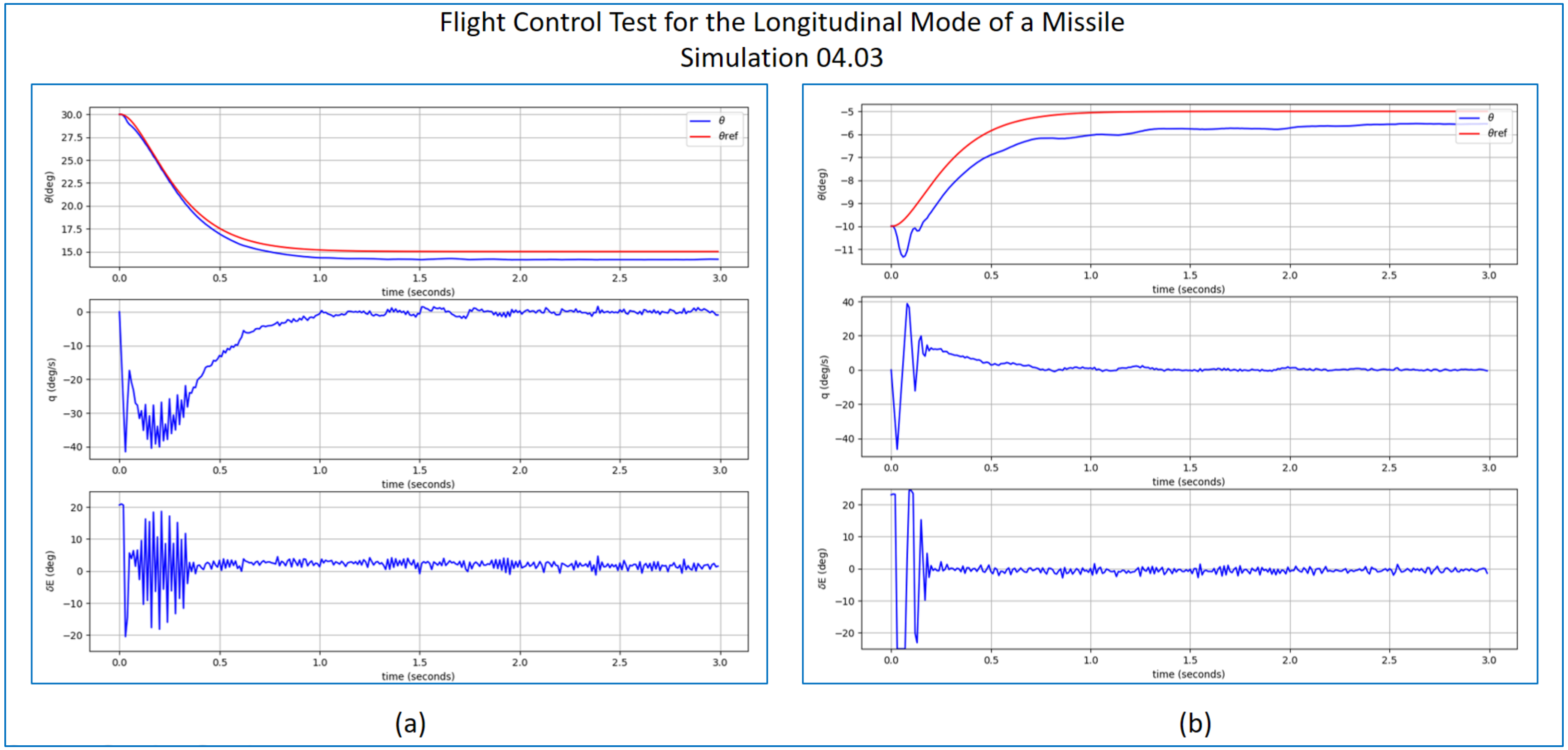


Figure 4-7 The Result of Test in Simulation 04.03

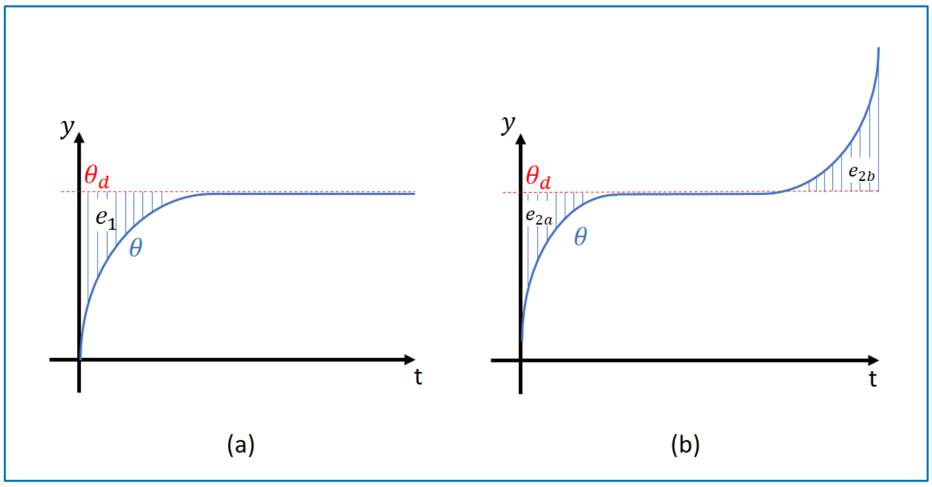


Figure 4-8 Two Potential Performance with the Same Reward Value

In these simulations, each second is divided into 10 nStep. By using equation (4-11) as the time component, it will guarantee that the reward value will negatively increase as it occurs further along the nStep. A constant value of 0.01 is multiplied to the time component to normalize the reward value.

4.7 Conclusion

Based on the result of Simulation 04.01, 04.02 and 04.03, in order for the best value of total reward represents the desired performance of the aircraft, a combination of state definition and reward function is required as follows.

$$s = [q \quad \theta_{Err}]$$

$$r = -\left(y - y_{ref}\right)^2$$

To guarantee which performance is the desired performance, an optimal performance is predetermined $\left(y_{ref}\right)$ for the RL agent to follow. To ensure that the steady state error of the performance is minimized, adding a time component as in equation (4-10).

5 DEVELOPING FLIGHT CONTROL IN LONGITUDINAL MODE USING DDPG WITH SINGLE ACTION

This chapter focuses on how to apply DDPG method to develop a flight control system. This is pursued by developing a training strategy for the RL agent to learn how to control the longitudinal mode of an air vehicle. In this case study, the air vehicle is a Jetstream J-3102 aircraft in a fixed operating condition.

This work will focus on how to judge the RL agent's ability to follow the performance goal at the end of the training session, instead of comparing the result with other conventional method. Examples of conventional method in flight control system development are model based PID [44][45] and backstepping [46] or model-free iPID [47] and NIB-MFC [48]. Comparing these results might be done in future works.

5.1 Introduction

One of the main issues in DDPG is determining the learning strategy that can cover a certain range of flight envelope and doing it at a definite amount of reasonable time. The problem with DDPG, if the range of possible states are too large then the RL agent learning process will diverge and not be able to produce a suitable policy. This learning strategy will be described in the next sub chapter.

Another issue in DDPG is determining the reward function that can best represent the flight performance. The best value of reward function has to be equal, and only equal, to the desired flight performance.

Based on the result of the previous chapter, the reward function that can best represent the desired flight performance is by following a predetermined path of flight. The desired performance in attitude change is defined as the changes discussed in sub chapter 4.5. The change of attitude for the predetermined path of pitch angle is defined as followed:

$$\dot{\theta}_{ref} = -2\xi_{ref}\omega_{n_{ref}}\dot{\theta}_{ref} - \omega_{n_{ref}}^{2}(\theta_0 - \theta_t) \quad \textbf{(5-1)}$$

θ_0 is defined as the initial state of pitch angle and θ_t is defined the final goal of the pitch angle. ξ_{ref} is a damping ratio of 1 and ($\omega_{n_{ref}}$) is the natural frequency of 2π. **This will guarantee that the best performance is the predetermined flight attitude change.**

The reward function is defined as the difference between the current state of pitch angle with the current desired pitch angle. To ensure that the learning agent minimized the difference as timestep t increases, the reward function is also multiplied by the timestep component.

$$\theta_{err} = \left(\theta - \theta_{ref}\right)^2 * 0.01 * nStep \quad \textbf{(5-2)}$$

5.2 Generating Samples

For this case study, the simulation is run for a Jetstream J-3102 aircraft. The goal in this simulation is to control the pitch angle attitude of the aircraft. A desired change in attitude (pitch angle) is predetermined from subchapter 5.1. This path/trajectory showed the best way for the RL agent to change its initial pitch angle attitude towards the desired pitch angle attitude.

The RL agent is trained through several episodes (300 - 1000 episodes) to learn how to control the aircraft. The indicator for its success will be the stabilizing of the total reward per episode (return) in which the value is negative and closest to zero.

Another indicator is the stabilization of the Q_{max} value. This value shows the changes done to the policy after certain episodes. The stabilizing of its value showed that the changes becomes less and less. When the changes become almost zero, then the policy is considered fully developed.

As stated in Chapter 0, the reinforcement-learning agent learned to determine the best policy for a system by using experience, compiling correlations between the state-action pairings and its rewards. In flight control, this experience is defined as samples of flight data that shows the correlation between the initial state of the air vehicle (s_t), actions that was taken (a_t), the state of the air vehicle after an action has been taken (s_{t+1}), and the reward of the action as it is correlated to the final state (r_t).

As described in subchapter 3.4, the algorithm for exploration are treated separately from the learning algorithm. The samples are generated by DDPG itself. It is generated by the exploration policy μ'. To ensure randomness, the actor policy is added by noise $\mathcal{N}$ from Ornstein-Uhlenbeck process.

In this simulation the DDPG is connected to a model that can generate the changes in state (s_{t+1}) when a random action (a_t) is applied to a given initial state (s_t). These data (s_t, a_t, s_{t+1}) are then given a reward value (r).

For this case study, the state consists of two variables that are defined as the difference between pitch angle and its desired pitch angle, known as θ_{err}, and pitch rate, q.

In DDPG method, the samples are generated by the exploration policy. This policy is achieved by adding noise to the actor policy. The policy initiates an initial state of the air vehicle and a random action (s_t, a_t). Then using the model that is outside of the DDPG method, the resulting state (s_{t+1}) from applying the action (a_t) to the initial state (s_t) is produced. These three variable fill out ¾ of the tuple needed to determine the value function in DDPG learning agent.

The remaining ¼ of the tuple is the reward value (r_t) that the action policy has given based on the previous ¾ of the tuple (s_t, a_t, s_{t+1}). Based on the complete tuple (s_t, a_t, s_{t+1}, r_t)., the RL agent updates its policy.

The action in this case study is defined as the elevator deflection (δ_E). The state that is being observed are the pitch rate (q) and the difference (θ_{Err}) between pitch angle (θ) and the desired pitch angle (θ_d), $s = [q \;\; \theta_{Err}]$.

Err_θ is chosen with the idea that it can be a benchmark for the RL agent to pursue the smallest value of it no matter the value of the desired pitch angle (θ_d).

The model of Jetstream (J-3102) in longitudinal mode are derived from [49]. The linear equation of motion for the longitudinal mode of Jetstream (J-3102) are as follows:

$$\dot{\alpha} = \frac{QS}{m}\frac{Cl_\alpha}{V}\alpha + q + \frac{QS}{m}\frac{Cl_{\delta E}}{V}\delta E \tag{5-3}$$

$$\dot{q} = \frac{QSc}{I_{yy}}Cm_\alpha\alpha + \frac{QSc}{I_{yy}}\frac{c}{2V}Cm_q q + \frac{QSc}{I_{yy}}Cm_{\delta E}\delta E \tag{5-4}$$

$$\dot{\theta} = q \tag{5-5}$$

The parameters below consist of the flight configuration of Jetstream (J-3102) and the aerodynamic configuration that correlates to the flight configuration.

Table 5-1 Parameters of Jetstream (J-3102) Aircraft

Mass = 6421 kg	S= 25.08 m^2	V = 91.57 m/s	X_{cg} = 0.28*c
Iyy = 35765 $kg.m^2$	c = 1.72 m	$\rho =$ 1.0193 kg/m	Cm_α = -0.993
Cl_α = -5.534	$Cl_{\delta E}$ = -0.3367	Cm_q = -15.96	$Cm_{\delta E}$ = -1.396

The dynamic pressure equation is:

$$Q = \frac{1}{2}\rho V^2 \tag{5-6}$$

5.3 Learning Strategy

The learning strategy for flight control in longitudinal mode in fixed operating condition focuses on exploring the capability of the action available to the RL agent. This means the control surface available on the air vehicle to change its attitude. For Jetstream J-3102, it is the elevator deflection.

In this case study, the RL agent is trained to familiarize itself with the possible movement of the elevator which is upward and downward and limited to a certain value of deflection. It is also trained to learn how each movement of the elevator affects the attitude of the whole aircraft. Just like a young bird that learn to flap its wings to its fullest or a baby that's learning how much strength they can put on their legs, the RL agent needs to learn how to use the elevator to control its pitch angle attitude.

The RL agent is trained from two different initial conditions, which are negative pitch and positive pitch. Then it is trained to achieve a pitch angle with a certain value. This value is determined early on in the simulation. The RL agent is trained not only to achieve the final pitch angle, it is also trained to do it by following the desired predetermined trajectory/path.

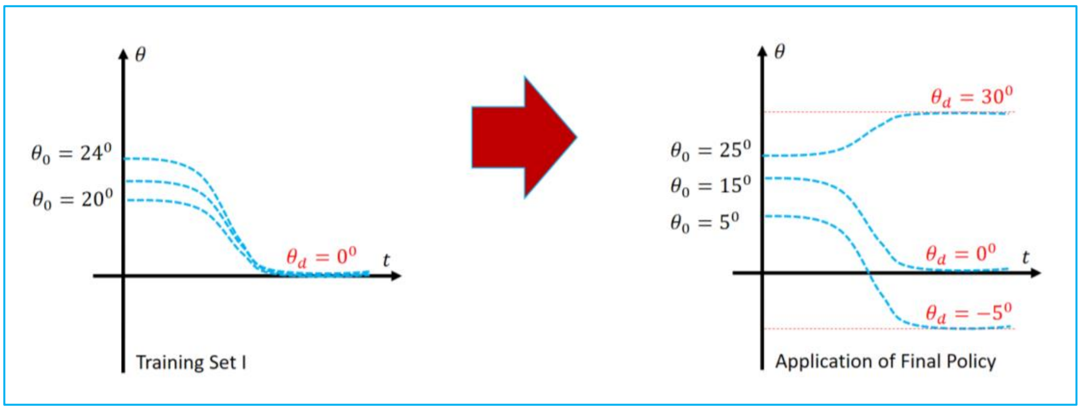


Figure 5-1 Diagram of Learning Process (Training Set I)

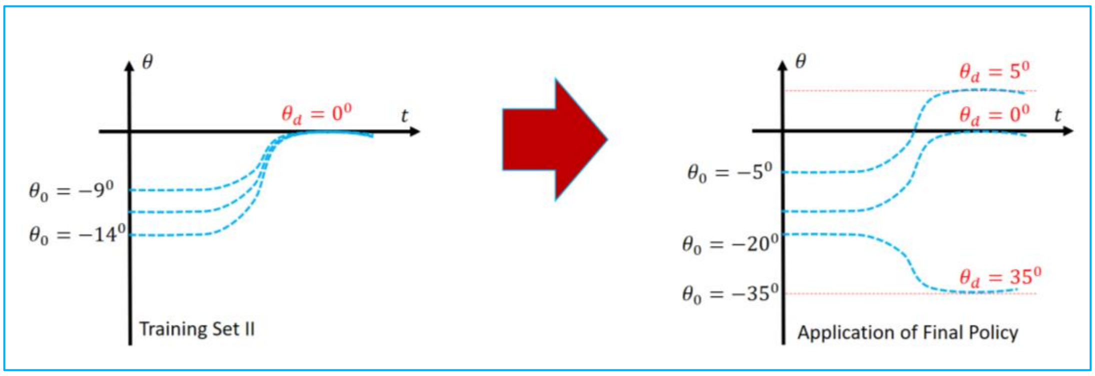


Figure 5-2 Diagram of Learning Process (Training Set II)

The learning strategy consists of 2 set of training set. Training set I focuses on exploring the downward elevator deflection and its impact on the pitch angle. The initial pitch angle (θ_0) is randomly chosen between 20^0 and 24^0. This range is chosen at 5^0. A range larger than 6^0 will result in the learning process to diverge.

Table 5-2 A Learning Strategy for Longitudinal Mode

No	Training Set (Fixed Operating Condition)	
	Initial State (θ_0)	Command (θ_d)
01	$random\ 20^0 - 24^0$	0^0
02	$random\ 20^0 - 24^0$	30^0
03	$random\ (-14^0) - (9^0)$	0^0
04	$random\ (-14^0) - (9^0)$	-20^0

Because one of the state defined is θ_{err}, then the RL agent is assumed to learn how to control the pitch angle based on how much difference there is from the desired final pitch angle. Therefore, even though the learning strategy only covers a certain portion of the possible pitch angle, the resulting policy will be able to adapt to it, Figure 5-1.

Table 5-3 Test Set for Application of Policy

Test Episode	Test Set (Fixed Operating Condition)	
	Initial State (θ_0)	Command (θ_d)
01-04	$random\ (-30^0) - (30^0)$	$random\ (-12^0) - (12^0)$

Training set II focuses on exploring the upward elevator deflection and its impact on the pitch angle. The initial pitch angle (θ_0) is randomly chosen between -9^0 and -14^0, Figure 5-2. As with the first set of training, the RL agent will learn based on θ_{err}.

For each training set, the RL agent is given 300 episodes to converge its learning process. At the end of the learning process, the RL agent is then tested to control the pitch angle from a random initial pitch angle to a random desired pitch angle. At this test, the RL agent doesn't stop learning. The policy is still updated, even if the changes are close to zero.

5.4 Pitch Angle Control

In this simulation, the number of layers for the network are respectively 350 and 350. This simulation will be focused on aircraft's response in the first 3 seconds. Each seconds will be divided into 300 timestep.

For the exploration policy, the Ornstein-Uhlenbeck parameter are:

Table 5-4 Ornstein-Uhlenbeck Parameter for Simulation 05.01 & 05.02

$\theta_{OU} = 0.4$	$\sigma_{OU} = 0.06$

In this sub chapter, two simulations will be executed with two different reward function. The reward function for simulation 05.01 is:

$$R_1 = -\sum_{j}^{nStep} {\theta_{Err}}^2 \tag{5-7}$$

The reward function for simulation 05.02 is:

$$R_2 = -\sum_{j}^{nStep} \left({\theta_{Err}}^2\right) * 0.01 * nStep \tag{5-8}$$

The theory is that the result of simulation 05.02 should give better result than simulation 05.01. The $nStep$ component should provide a guarantee that the performance will definitely follow the desired trajectory/path as time increases.

The result of Simulation 05.01 can be seen in Figure 5-3,Figure 5-4, Figure 5-5. and Figure 5-6. While the result of Simulation 05.02 can be seen in Figure 5-7,Figure 5-8,Figure 5-9, and Figure 5-10.

Both Figure 5-3 and Figure 5-7 show a value of total reward (return) per episode that is converging as the time increases. This means that for both simulations, the DDPG learning agent can find stability in the system and it can control the aircraft. A closer look at the converging value of both simulations can be seen in Figure 5-4 and Figure 5-8.

The Q_{max} graphs in Figure 5-6 and Figure 5-10 also showed that it stabilizes its value in the range of 1 and (-1). This indicate that the policy created is stable enough that it merits less changes to the policy after every episode.

However, the performance from the test episodes from both simulations do give different result. Figure 5-5 shows that the RL agent manage to control the pitch angle in 1 out of 4 test episodes. Test Episode 01 shows the best result of the pitch angle changes follow the desired trajectory/path. In test episode 02, the value of pitch rate is too high as the RL agent seems to go beyond the elevator deflection limit. This suggests that this episode might be one of the exploration episodes of the RL agent.

Test episodes 3 and 4 shows that the RL agent manage to change the pitch angle by 'generally' follow the trajectory/path that is desired. However, the steady state error of the pitch angle is more than 10. It is assumed that this problem can be solved in Simulation 05.02, where the reward function includes the $nStep$ component.

Figure 5-9 shows the result of Simulation 05.02. It shows that 4 out of 4 test episodes, the RL agent manage to control the pitch angle to 'generally' follow the desired trajectory/path. The steady state error of all 4 test episodes are less than 10%. However, in test episode 2, the pitch rate is too big (more than 0.5 rad/s or $\approx 28^0/s$). This is unacceptable. This problem is assumed can be solved by adding a component of pitch rate in the reward function.

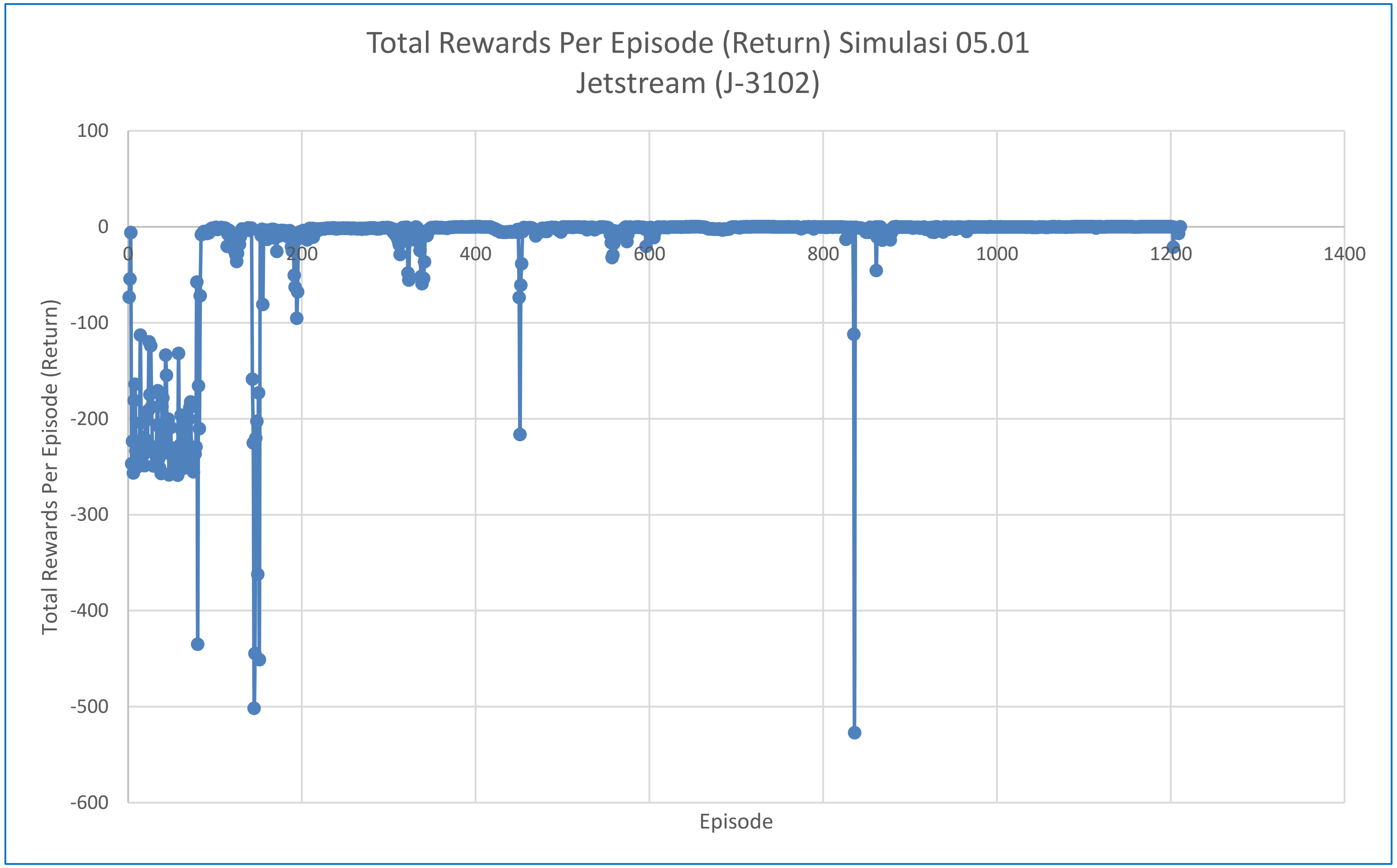


Figure 5-3 The Total Reward Per Training Episode in Simulation 05.01

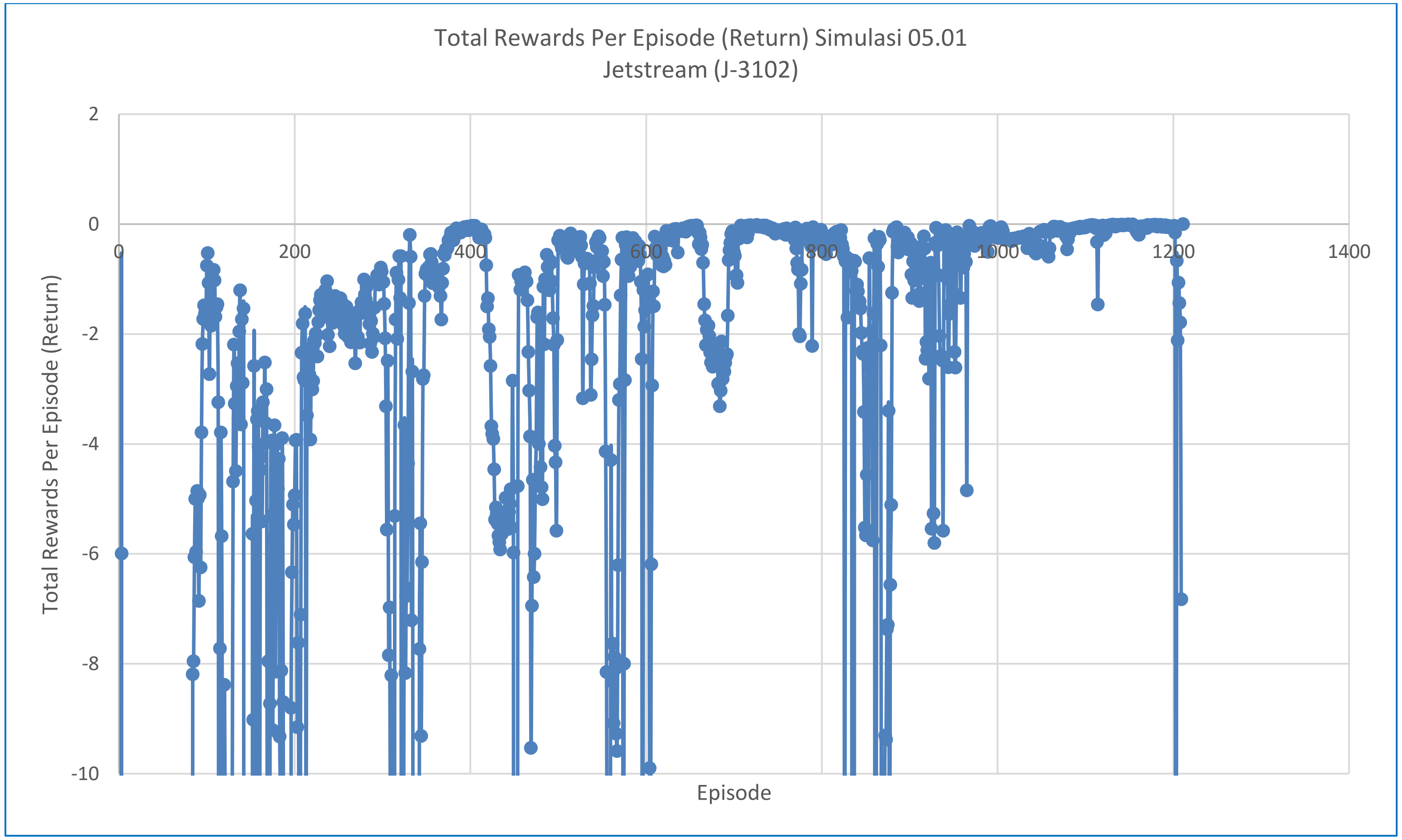


Figure 5-4 The Total Reward Per Training Episode in Simulation 05.01-- Zoom

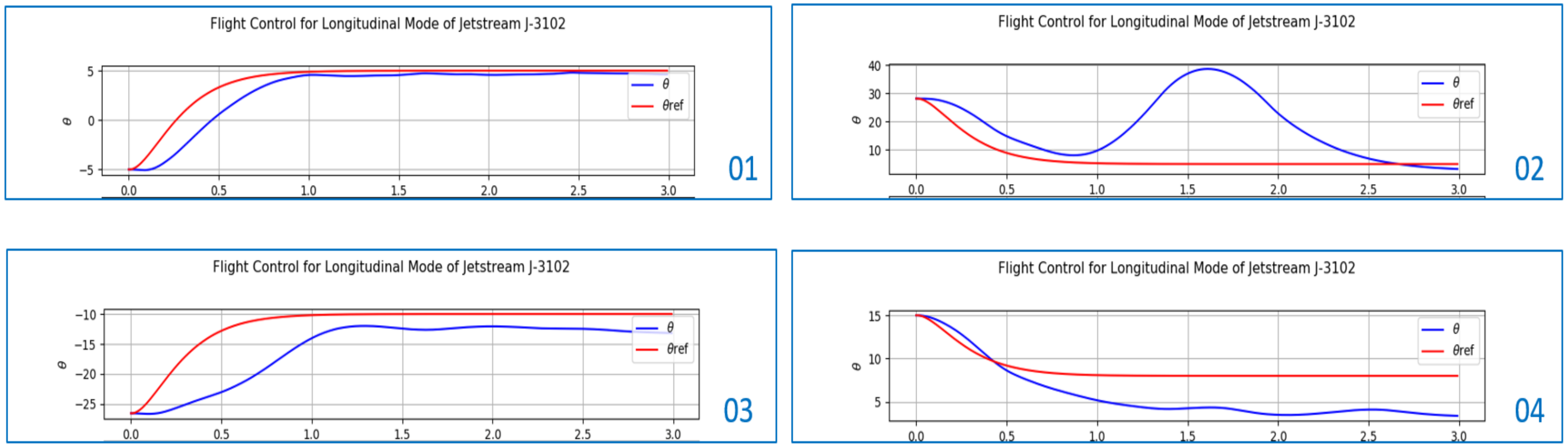


Figure 5-5 Test Result in Simulation 05.01

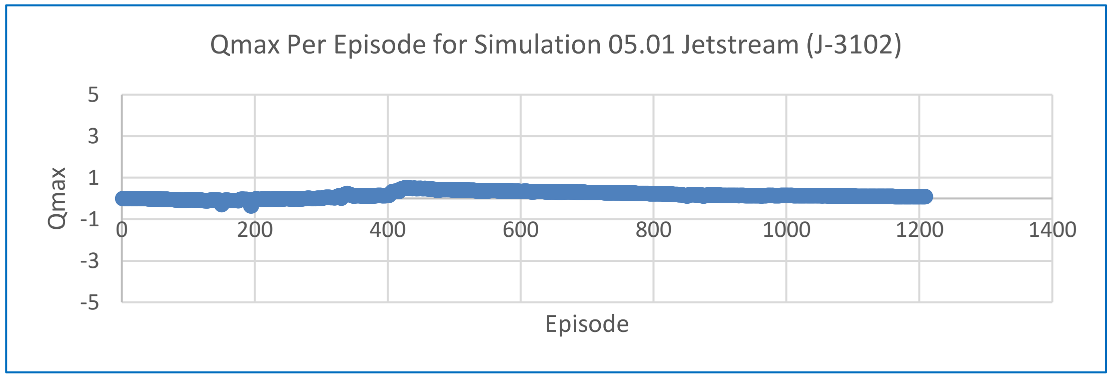


Figure 5-6 Qmax Changes in Simulation 05.01

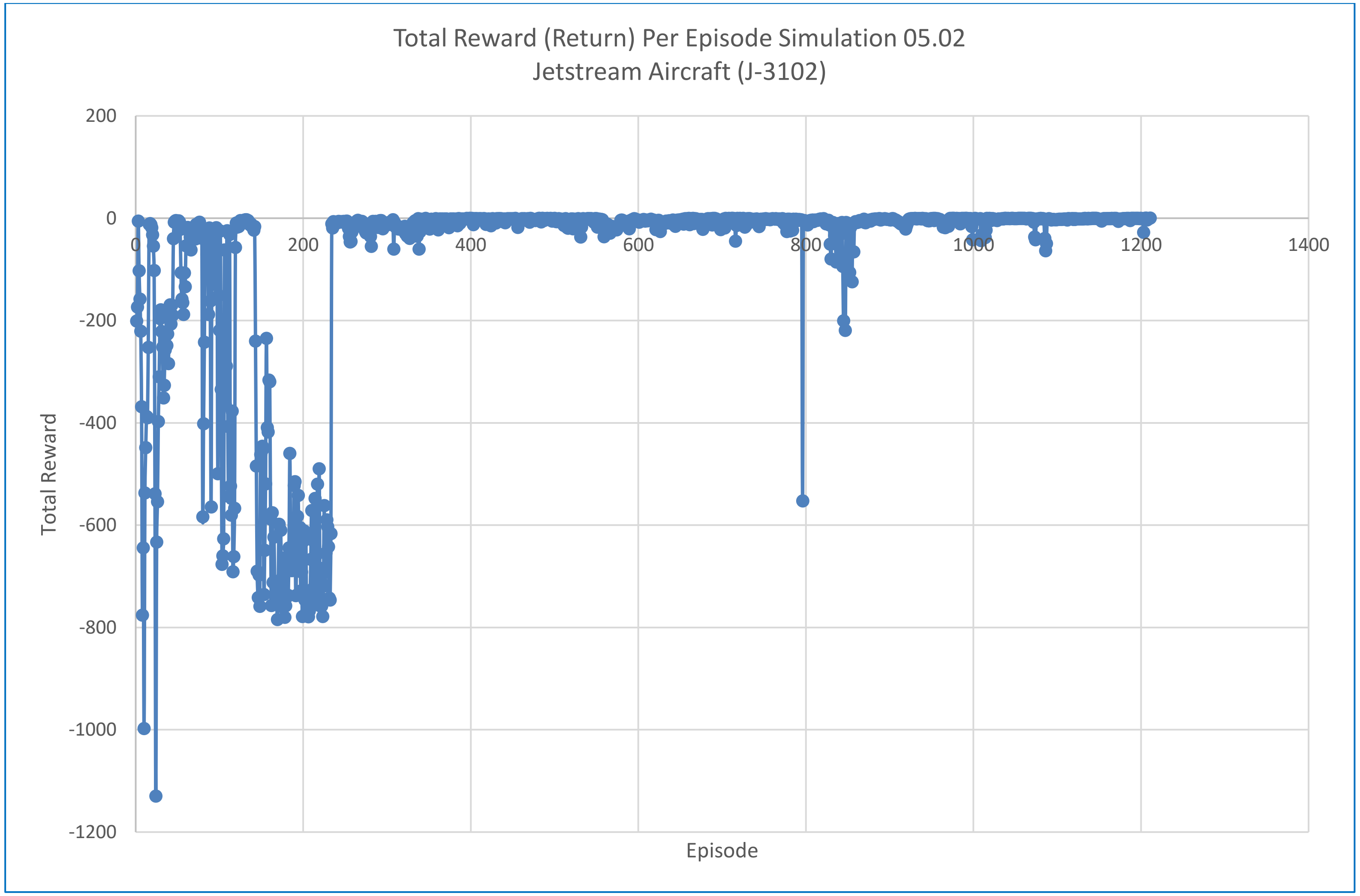


Figure 5-7 The Total Reward Per Training Episode in Simulation 05.02

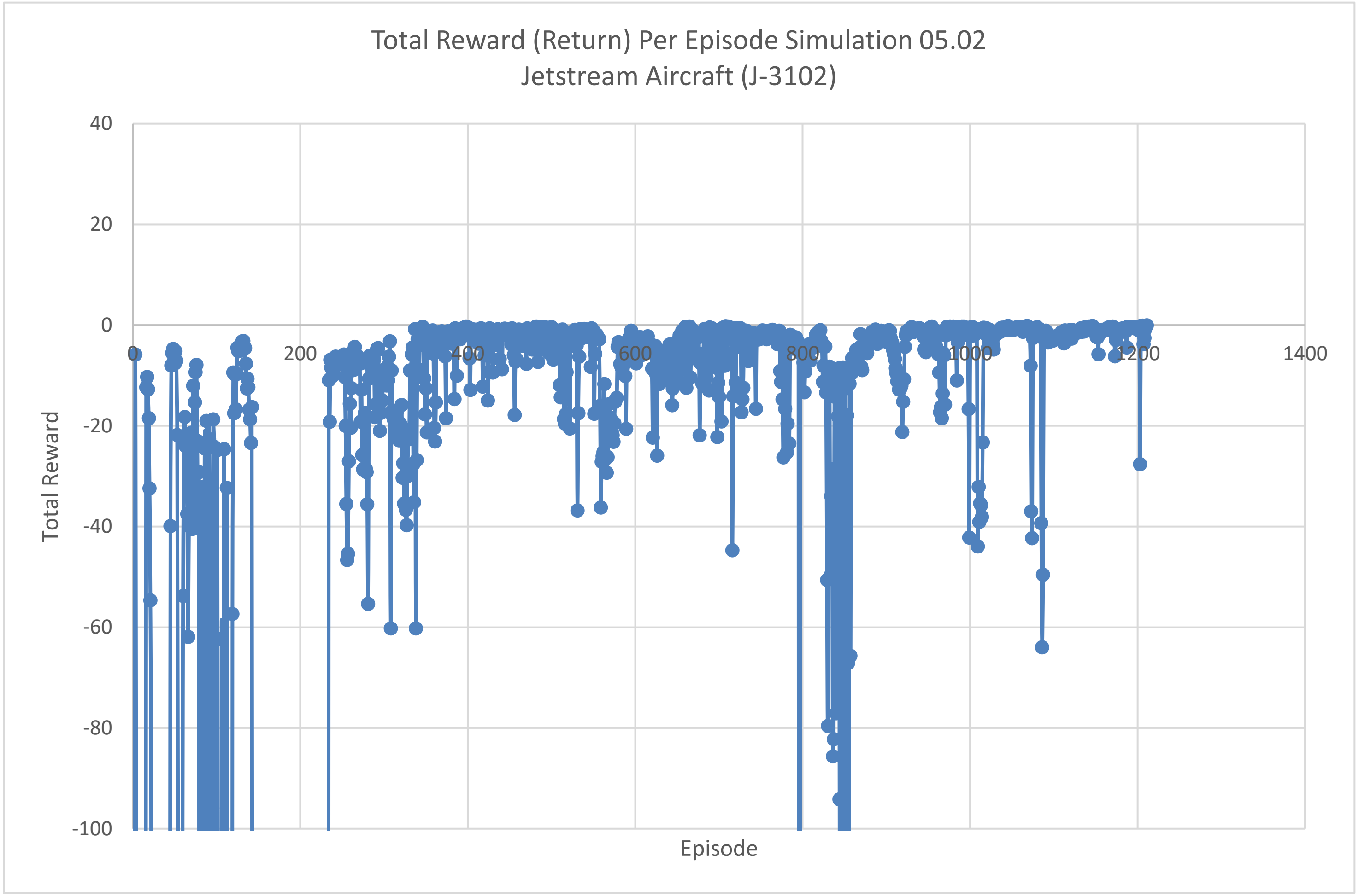


Figure 5-8 The Total Reward Per Training Episode in Simulation 05.02-- Zoom

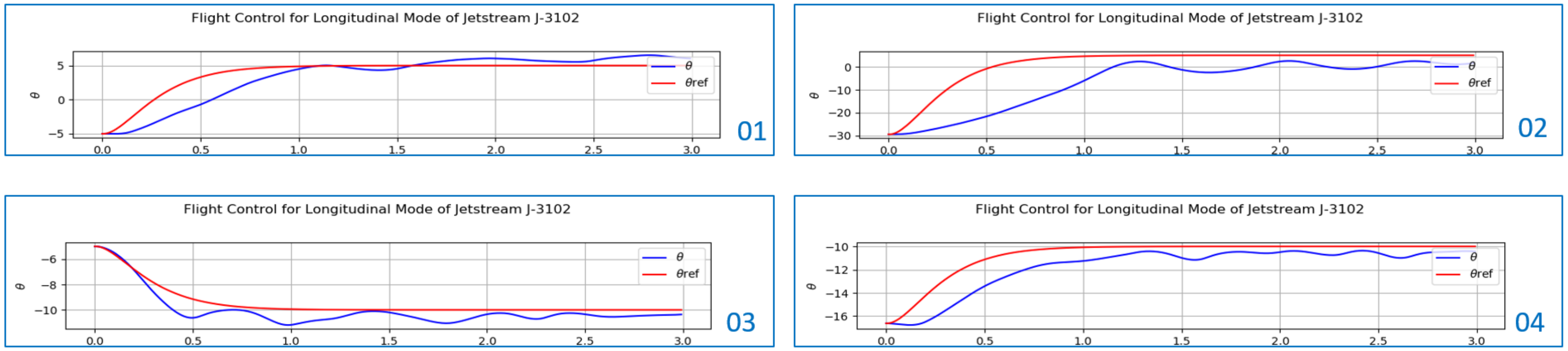


Figure 5-9 Test Result in Simulation 05.02

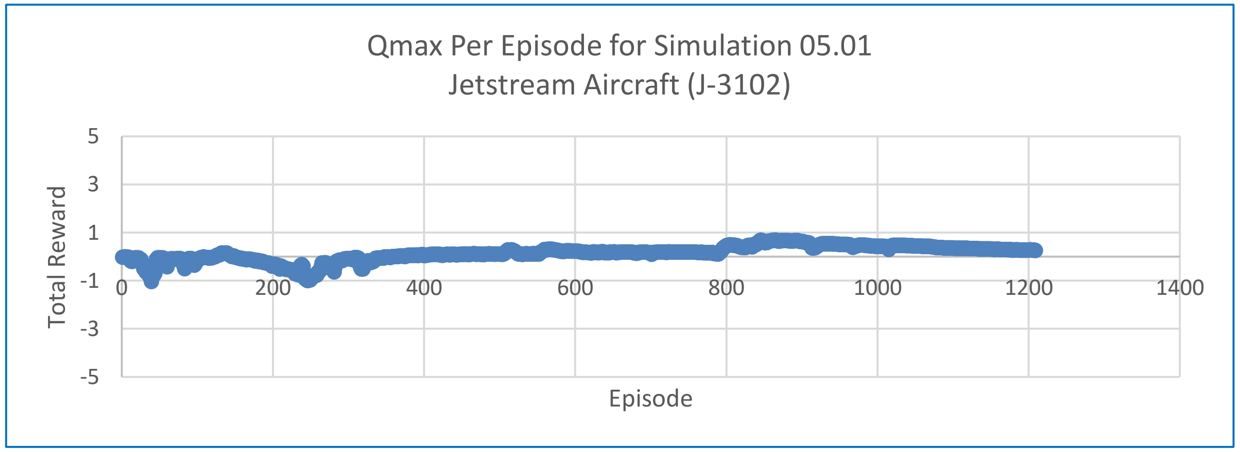


Figure 5-10 Qmax Changes in Simulation 05.02

5.5 Validation

In this sub chapter, the final policy network from the training process in Simulation 2 is extracted and applied as the policy network for another programme. This programme uses the same external model to demonstrate how the RL agent responded to different initial state.

This phase is designed to validate the resulting policy network and prove that indeed the policy can provide the knowledge for the RL agent to act appropriately for different state. The noise parameter in the actor policy is assumed zero, therefore forcing the RL agent exploit policy already given to produce a response instead of exploring for a new and possibly positive response.

Figure 5-11 shows how the RL agent uses an initial policy network that has been through training process. It shows how the pitch angle is changed throughout the time by following the desired path. At the end of 3 seconds, the pitch angles are all stable on the desired value.

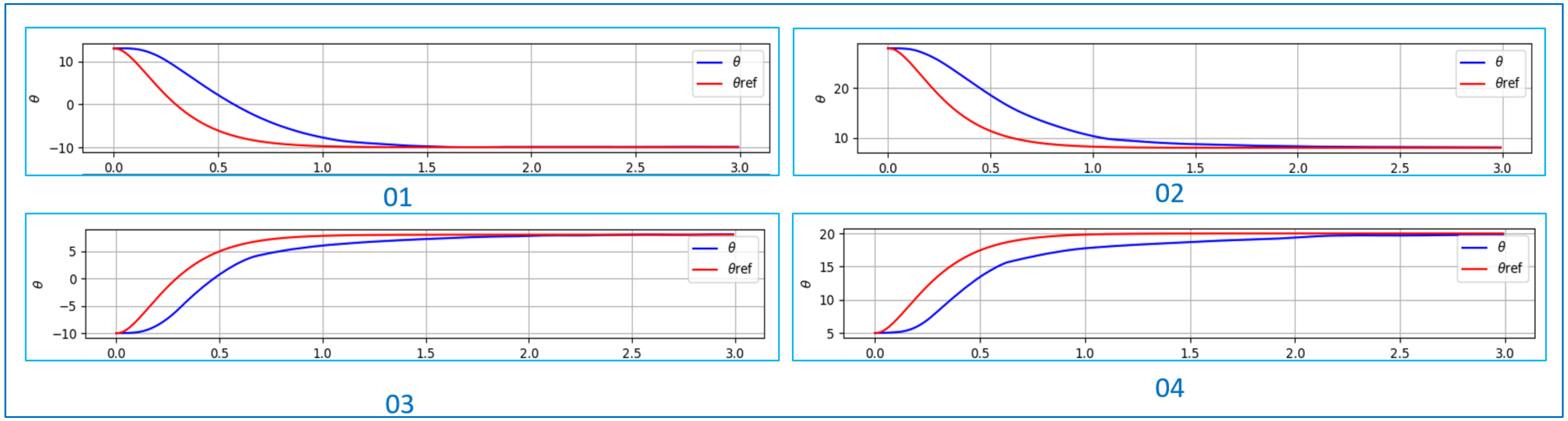


Figure 5-11 Test Result in Simulation 05.03

5.6 Conclusion

The investigation through several simulations that focuses on controlling the pitch angle and the pitch rate shows that:

- the DDPG method can be used to develop a control system for the longitudinal mode of an aircraft. In this case study, the aircraft is Jetstream J-3102.
- adjusting the components in the reward function can apply a limitation to the pitch angle and pitch rate response of the aircraft
- the final policy network is considered suitable when the value of Q_{max} during the test episodes are within the range of [-1,1].
- the policy network post training process can be extracted and used in another programme and expected to give an appropriate response to almost every state

6 DEVELOPING FLIGHT CONTROL IN LONGITUDINAL MODE USING DDPG IN VARIATING OPERATION CONDITION

This chapter focuses on developing a training strategy for the RL agent to learn how to control the longitudinal mode of an air vehicle in variating operating condition. The operating condition that are considered to be varied are velocity, and altitude. In this case study, the air vehicle is a missile and the model is derived from [43].

6.1 Introduction

One of the main issue in DDPG is determining the variables to be observed, known as *state*. From Chapter 5, it is found that the state definition in a fixed operating condition is $s = [\theta_{err}\ \ q]$. However, for a variating operating condition it is unknown whether the same state definition is enough to control the pitch angle.

Therefore, in this chapter, a variety of state definition is simulated to train the DDPG agent for:

- Velocity variation
- Altitude variation

In training for velocity variation, two definition of state is used. One is the state definition already done in Chapter 5, where $s = [\theta_{err}\ \ q]$. Another state definition is where the variable of velocity added, $s = [\theta_{err}\ \ q\ \ V]$. For altitude variation, one simulation is done with the state definition is $s = [\theta_{err}\ \ q]$.

The missile model used to generate the samples is described in the following sub chapter. The specific training strategy for each varying operation condition will be explained in each sub chapter that follows after.

6.2 Generating Sample

For this case study, the simulation is run for a Missile. The goal in this simulation is to maintain control of the pitch angle attitude of the missile in variating operation condition. The variating operating condition is achieved by variating the velocity and/or altitude.

Similar to Chapter 5, the desired change in attitude (pitch angle) is predetermined according to Equation (5-1). The desired performance will be for the RL agent to follow this path/trajectory.

However, the indicator for success in the learning process will be the stabilizing of the total reward per episode (return) on a value close to zero and the stabilization of the Q_{max} value.

The model of the missile in longitudinal mode are derived from [43]. The linear equation of motion of the longitudinal mode for the missile are the same as the equation for the Jetstream aircraft in Chapter 5. However, for clarity, it is written again below.

$$\dot{\alpha} = \frac{QS}{m}\frac{Cz_{\alpha}}{V}\alpha + q + \frac{QS}{m}\frac{Cz_{\delta E}}{V}\delta E \tag{5-3}$$

$$\dot{q} = \frac{QSd}{I_{yy}}(X_{cp} - X_{cg})\frac{Cz_{\alpha}}{d}\alpha + \frac{QSd}{I_{yy}}\frac{d}{2V}Cm_{q}q + \frac{QSd}{I_{yy}}Cm_{\delta E}\delta E \tag{5-4}$$

$$\dot{\theta} = q \tag{5-5}$$

The parameters below consist of the flight configuration of the missile and the aerodynamic configuration that correlates to the flight configuration.

Table 6-1 Parameters of The Missile

Mass : 200 kg	Diameter (d): 0.3 m	V : 300 m/s	$X_{cp} = X_{cg} + d$
Iyy : 450 $kg.m^2$	X_{cg} : 2.5 m	ρ : 1.21 kg/m	
Cz_{α} : -6.0	Cm_{α} : -500.0	$Cz_{\delta E}$: -0.1340	$Cm_{\delta E}$: -26.180

6.3 Varying Velocity

In this case study, two simulations are executed to explore the impact of adding a condition that varies during the operation of the missile. Those two state definitions are:

- $s = [q \;\; \theta_{err}]$
- $s = [q \;\; \theta_{err} \;\; V]$

It is assumed that the latter state definition will require more number of layers for the input network than that of the first state definition.

6.3.1 Training Strategy

The learning strategy for flight control in longitudinal mode in variating operating condition focuses on two things. One, that it explores the capability of the action available to the RL agent. This means the control surface available on the air vehicle to change its attitude, which is the elevator deflection. Two, it explores how much the impact of its capability changes due to a variation of velocity.

Table 6-2 Training Set for FOC

No	Training Set (Fixed Operating Condition)	
	Initial State (θ_0)	Command (θ_d)
01	$random\ 20^0 - 24^0$	0^0
02	$random\ (-14^0) - (9^0)$	0^0
03	$random\ 20^0 - 24^0$	$\theta_{d1} = 15^0, \theta_{d2} = 0^0$
04	$random\ (-14^0) - (9^0)$	$\theta_{d1} = 0^0, \theta_{d2} = 15^0$
05	$random\ 20^0 - 24^0$	$\theta_{d1} = 15^0, \theta_{d2} = 5^0, \theta_{d3} = 30^0$

The first focus is achieved by putting the RL agent through 5 sets of training in fixed operating condition (FOC), **Table 6-2**. The second focus is achieved by adding 3 sets of training in variating operating condition (VOC),**Table 6-3**. The RL agent go through all of these training sets before a set of episodes is added at the end to test the policy.

Table 6-3 Training Set for VOC (Velocity Variation)

No	Training Set	
	Initial State (θ_0)	Command (θ_d)
	24^0	0^0
06	$random\ V = 10\text{-}60\ \mathrm{m/s}$	
07	$random\ V = 260\text{-}310\ \mathrm{m/s}$	
08	$random\ V = 360\text{-}410\ \mathrm{m/s}$	

In VOC training, the RL agent is pushed to control the pitch angle in a range of varying velocity. For each set of VOC training, the range of velocity is 50 m/s. A range larger than that will not result in the agent converging its learning process.

For both FOC training and VOC training, each set of training is given a range of 500 episodes. This means that the FFPG agent is given 500 episodes to learn and produce a converging policy for each training set.

6.3.2 Velocity Variation ($\boldsymbol{s = [q\ \ \theta_{err}]}$)

In Simulation 06.01, the number of layers for the network are respectively 300 and 500. As are the simulations in previous chapters, this simulation will focus on the missile's response in the first 3 seconds, specifically on the performance of the pitch angle. Each seconds will be divided into 300 timestep.

For the exploration policy, the Ornstein-Uhlenbeck parameter are:

Table 6-4 Ornstein-Uhlenbeck Parameter for Simulation 06.01

$\theta_{OU} = 0.3$	$\sigma_{OU} = 0.075$

The reward function for simulation 06.01 is:

$$R_1 = \sum_{j}^{nStep} {\theta_{Err}}^2 * 0.01 * nStep \tag{6-1}$$

Because the test episodes were executed following the training episodes, the RL agent is still updating its policy. Therefore, previous to this simulation, it is assumed that the result should be that the RL agent will be able to still control the pitch angle of the missile through different velocity.

Figure 6-1 shows that the value of the total reward (return) are converging as the number of episodes grow, through the fixed operating condition training (episodes 1- 2500) and the variating operating condition training (episodes 2501-4000).

There are some spikes of big negative value during the variating operating condition. These episodes happened as the RL agent enters a new training set with a different velocity. Figure 6-5 supports this claim as it also shows spikes in the changes of policy when those spikes in the total reward value occurs. It also shows stabilization towards the end of the simulation as its value stayed between 1 and (-1).

Figure 6-2 shows the result of the test episodes at the end of the training episodes. The policy is tested through different velocity condition and different change in pitch angle. The result shows that the RL agent manage to follow the desired trajectory/path of pitch angle change. Though the steady state error differs.

Figure 6-3 and Figure 6-4 show the result of the test episode, performed at the end of the training process. They show the result of test episodes for different velocity. Test episodes (b) and (f) show how the DDPG agent control the pitch angle of the aircraft for velocities that are trained in the previous training process.

However, test episodes (a), (c), (d) and (e) are all test episodes where the velocities of the aircraft have different value than the ones being used for training. This shows that the DDPG agent can make a response that is an interpolation based on its training process.

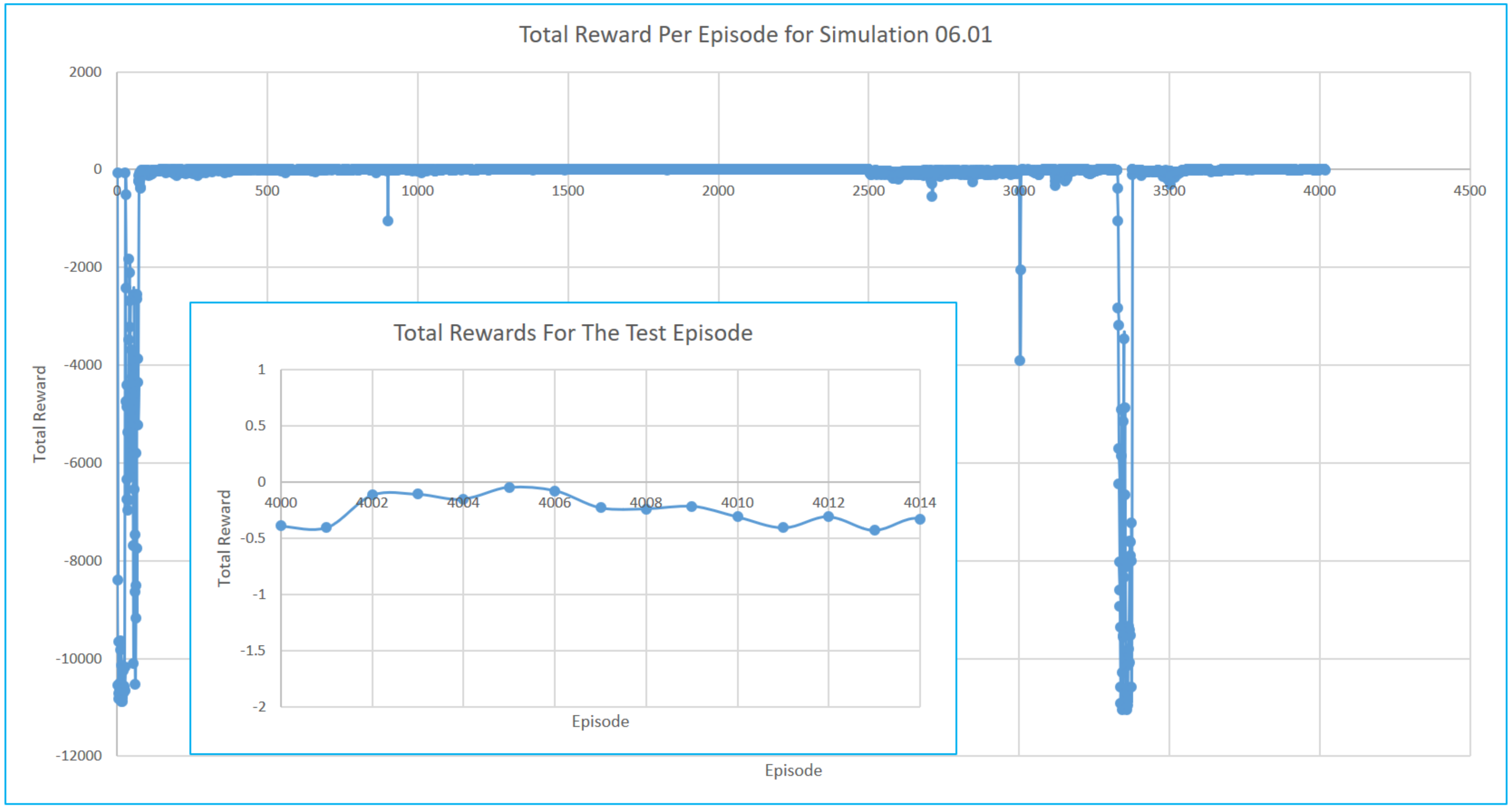


Figure 6-1 The Total Reward Per Training Episode in Simulation 06.01

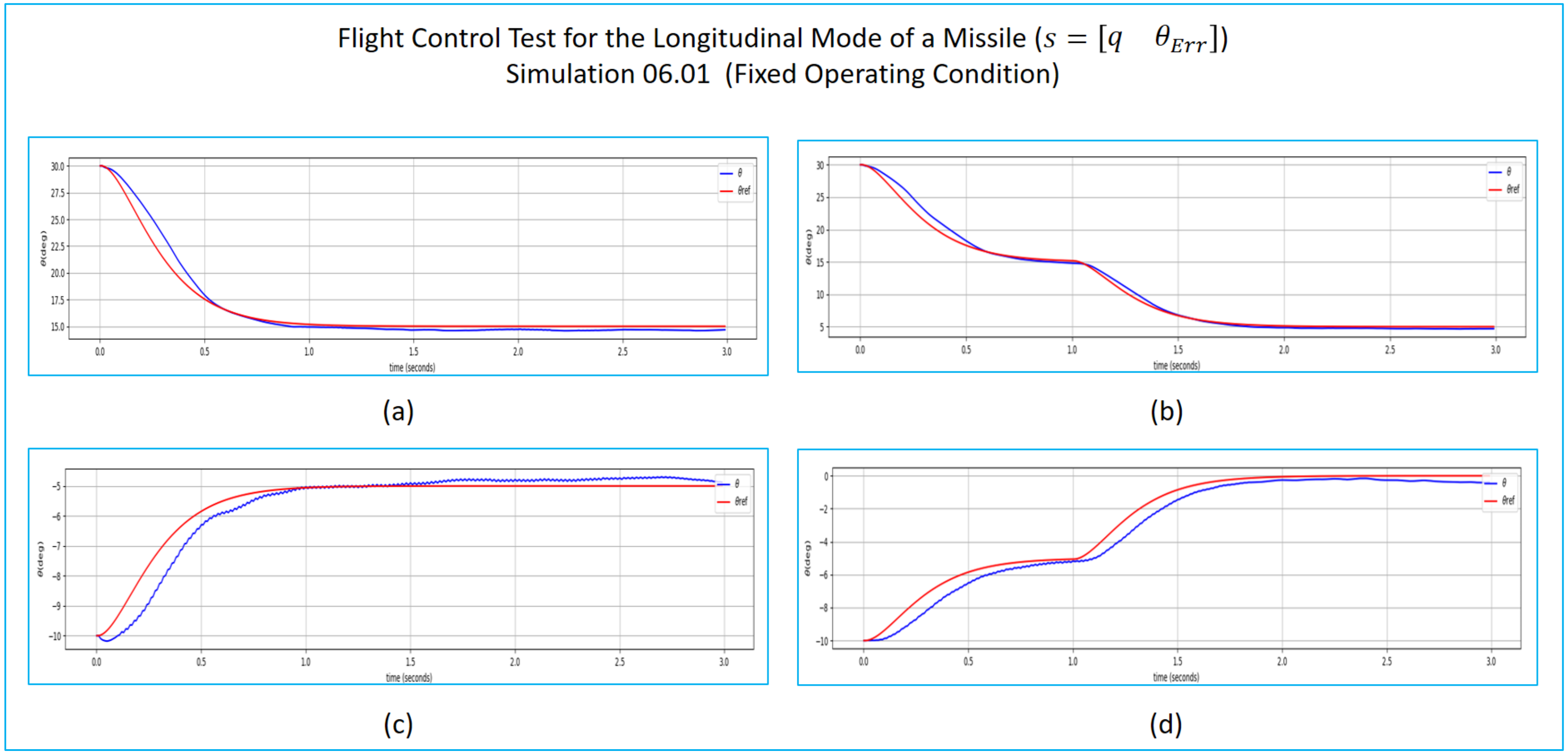


Figure 6-2 FOC Test Episode Following Training in Simulation 06.01

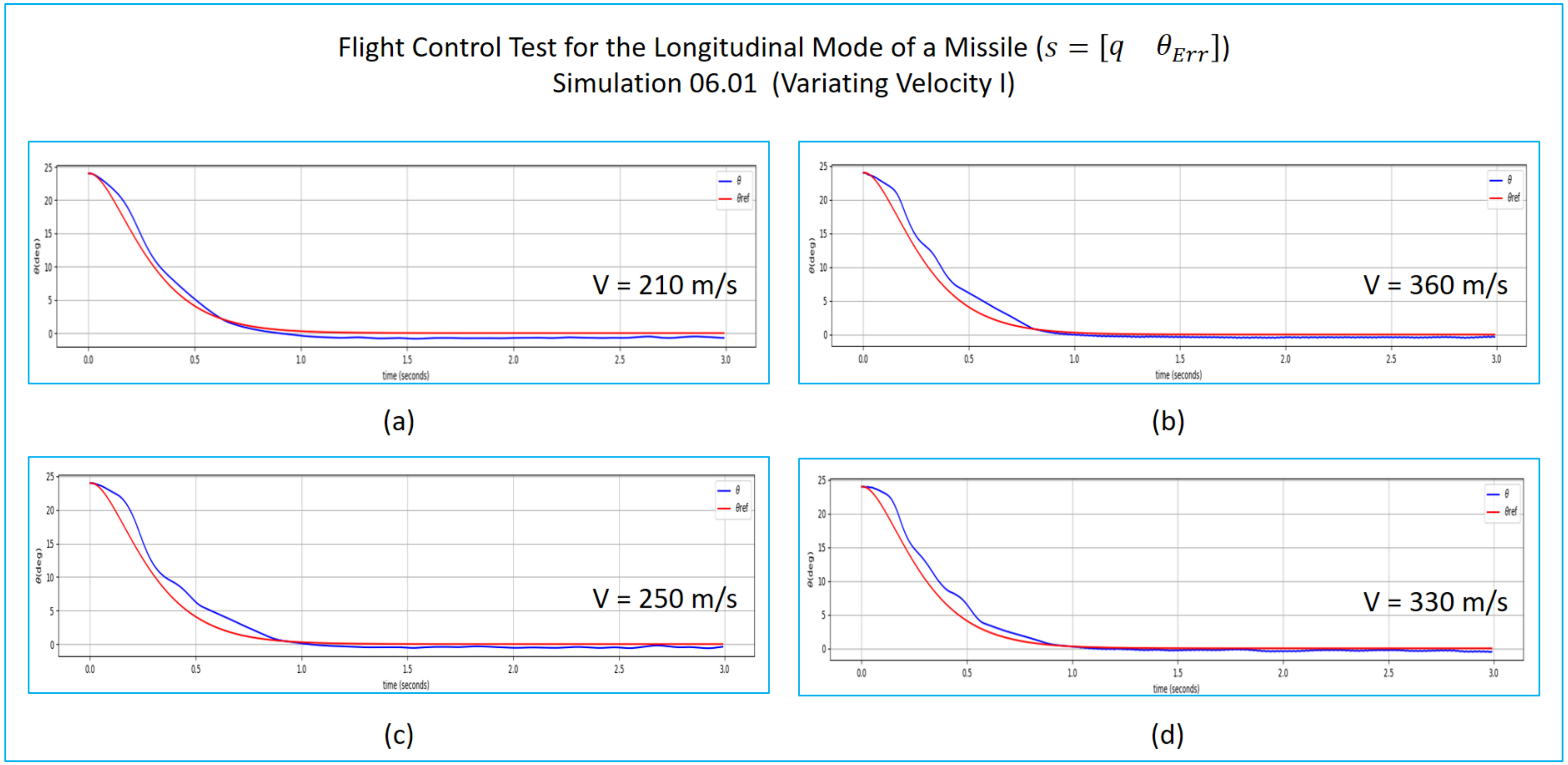
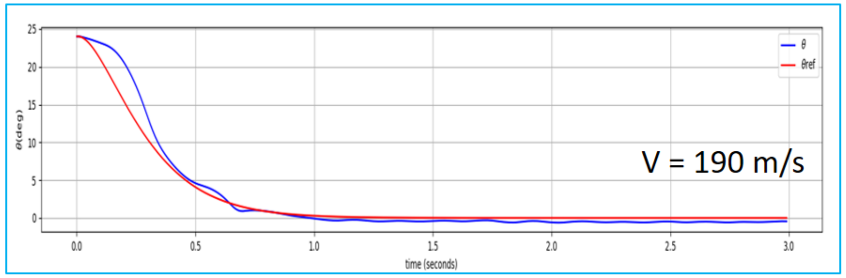
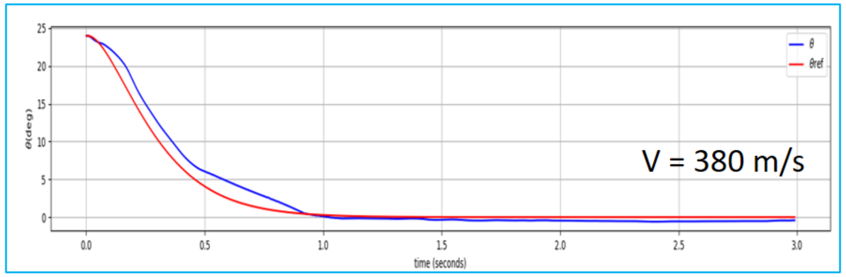


Figure 6-3 VOC Test Episodes 1-4 Following Training in Simulation 06.01

Flight Control Test for the Longitudinal Mode of a Missile ($s = [q \quad \theta_{Err}]$)
Simulation 06.01 (Variating Velocity II)



(a)



(b)

Figure 6-4 VOC Test Episodes 5-6 Following Training in Simulation 06.01

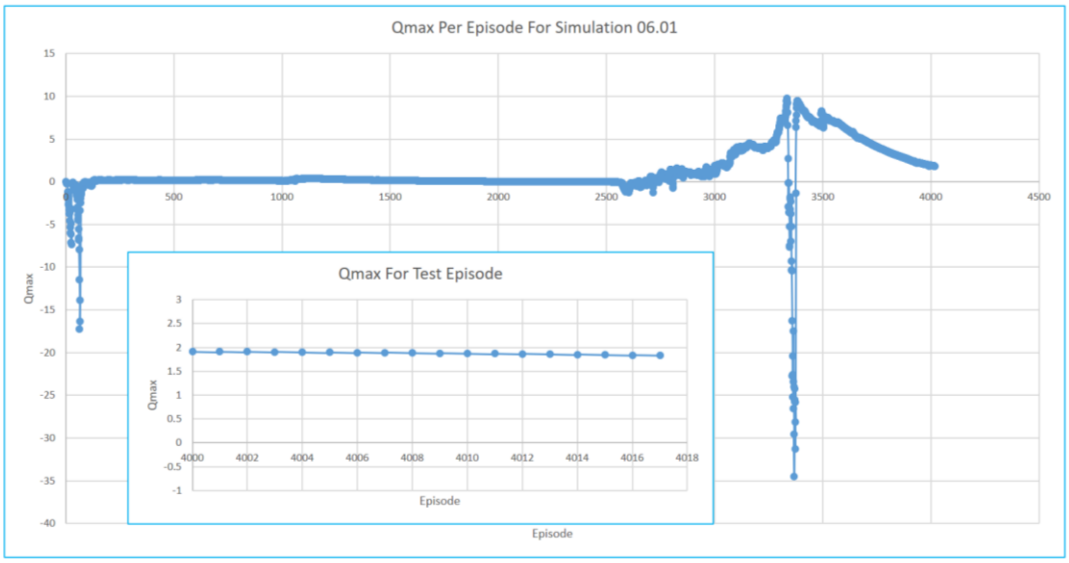


Figure 6-5 Qmax Changes in Simulation 06.01

Figure 6-5 shows the DDPG agent stabilizing to FOC training process followed by its value spiked during the VOC training. However, the agent manages to find stability again and it shows in its values during the test episodes (4000-4016).

6.3.3 Velocity Variation ($s = [q \ \ Err\theta \ \ V]$)

In Simulation 06.02, several hyper-parameters are kept similar to Simulation 06.01, such as the timestep (nStep) for each second and the reward function.

However, the number of layers for the network are different because it needs to accommodate for the added number of state. In simulation 06.02, the number of state is 3 which are pitch rate, the error in pitch angle and velocity ($s = [q \ \ \theta_{err} \ \ V]$). Therefore, the number of network layers are respectively 400 and 600.

Another difference is the number of episodes given to the DDPG agent to learn a converging policy for each training set. Because the number of state is added, then there are one more variable to be observed by the DDPG agent. This included variations of velocity variable whilst tied with variations in pitch rate and pitch angle. This means more episodes required to master one VOC training set.

For the FOC training set, the DDPG agent is given 500 episodes to learn. This is the same as simulation 06.01. the reason behind this is that because for this training set, the velocity is locked in one value only (300 m/s). This effectively cuts down the variations possible in the velocity. Therefore, it is assumed that the number of episodes that the DDPG agent needs to master each training set is similar to that in simulation 06.01.

However, for the VOC training set, the DDPG agent is given 1000 episodes to learn a converging policy. As stated in previous explanations, with more variations due to the variating velocity, the number of episodes needed to master a specific training process is assumed to larger.

For the exploration policy, the Ornstein-Uhlenbeck parameter are exactly the same as in simulation 06.01:

Table 6-5 Ornstein-Uhlenbeck Parameter for Simulation 06.02

$\theta_{OU} = 0.3$	$\sigma_{OU} = 0.075$

Due to its complexity as a result of the added definition of state, therefore, also the added number of network layer, the reward function is simplified by eliminating the timestep (nStep) component. In simulation 06.02, the reward function for is:

$$R_1 = \sum_{j}^{nStep} {\theta_{Err}}^2 \tag{6-2}$$

Simulation 06.02 is run through the training process described in **Table 6-2** and **Table 6-3**. The result of this simulation can be observed in Figure 6-6, Figure 6-7, Figure 6-8 and Figure 6-9.

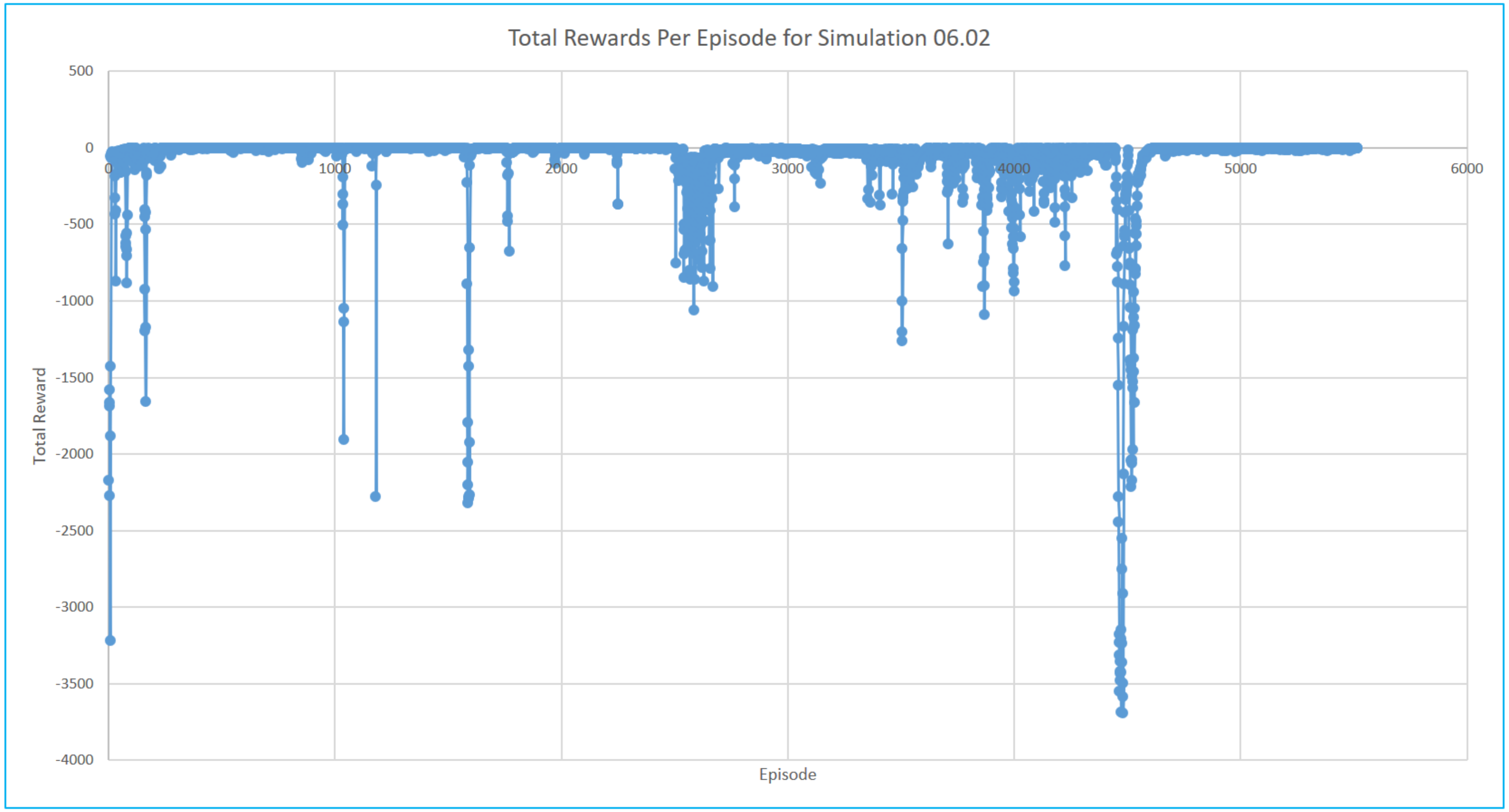


Figure 6-6 The Total Reward Per Training Episode in Simulation 06.02

In Figure 6-6, the total reward per episode shows convergence that are marked by spikes in value in certain areas. Focusing on the FOC training set, Figure 6-10 shows a comparison between simulation 06.01 and 06.02 with the same range of value. Here, it shows that the DDPG agent in both simulations converge its total rewards value. Despite that, Simulation 06.02 does have more spikes in value. This is assumed due to the difference in state definition. An exploration might have different result as the velocity is observed therefore a slight change in can affect the whole response.

The result of test episodes in Figure 6-7 shows that although the DDPG agent ‘generally’ followed the desired pitch angle change, the steady state error is significant. It is assumed that this is due to the fact that the reward function has no timestep (nStep) component. However, adding this component could jeopardize the DDPG agent learning process. It could lead to divergence in the total rewards per episode.

Specifically, for graph (d), it is assumed that this response coincides with the episode where the DDPG agent is exploring instead of exploiting. It is advised to separate the training program and the test programme to fully see the use of the policy resulted from the training process.

Figure 6-8 displays the performance of the air vehicle as it is controlled by the DDPG agent. The performance from the DDPG agent towards situations with different velocity, after its training, shows that it has manage to generally follow the desired path of pitch angle change. However, its steady state error suggests that the absence of timestep (nStep) component is quite influential.

From Figure 6-9, it can be seen that there is quiet a big spike in the value of Q_{max} somewhere between episodes 3500 and 4500. This is the episode range in which the DDPG agent trained for random velocity between 260 m/s and 310 m/s. This also corresponds with the large negative value of total reward in Figure 6-6.

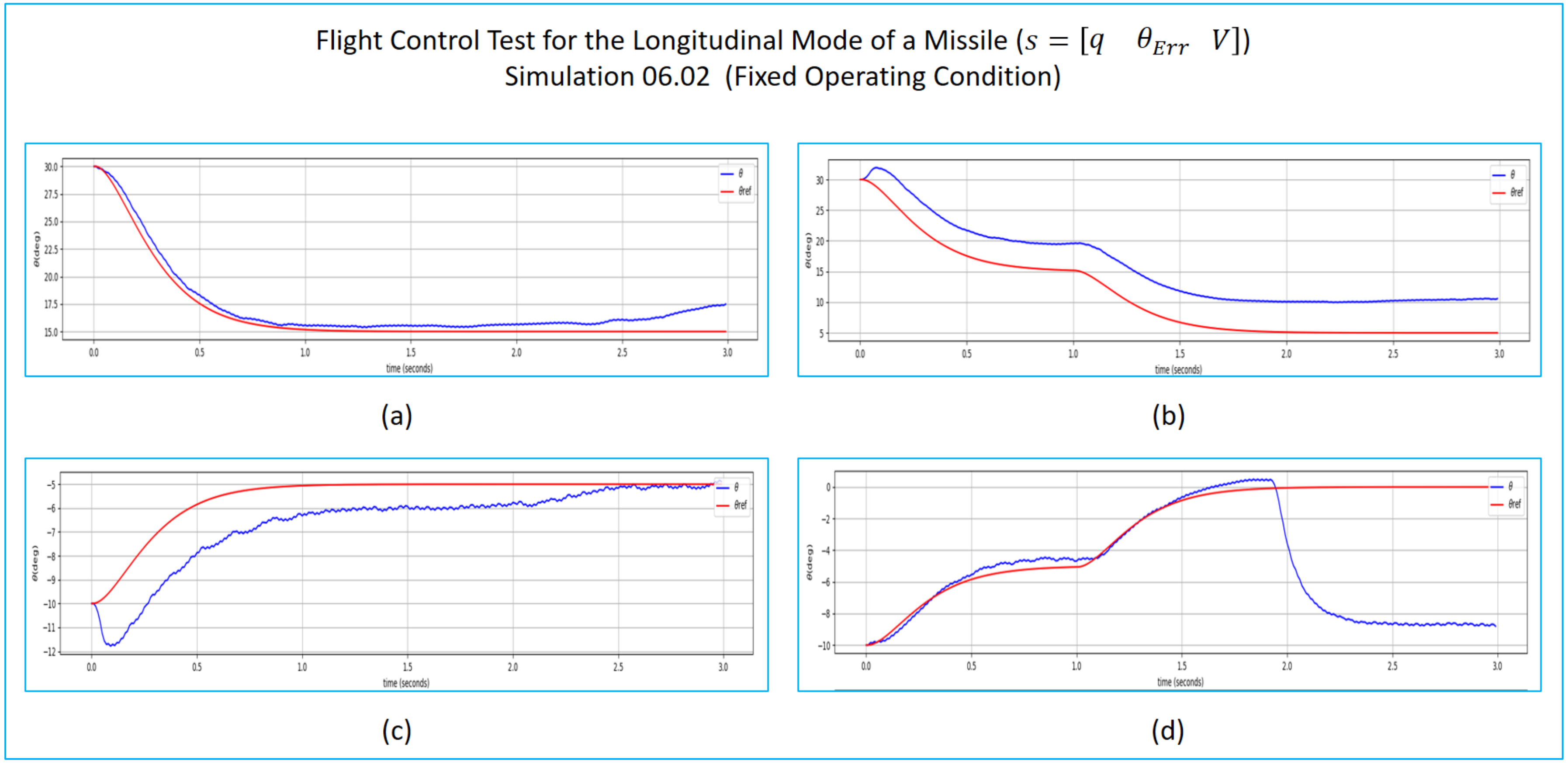


Figure 6-7 FOC Test Episode Following Training in Simulation 06.02

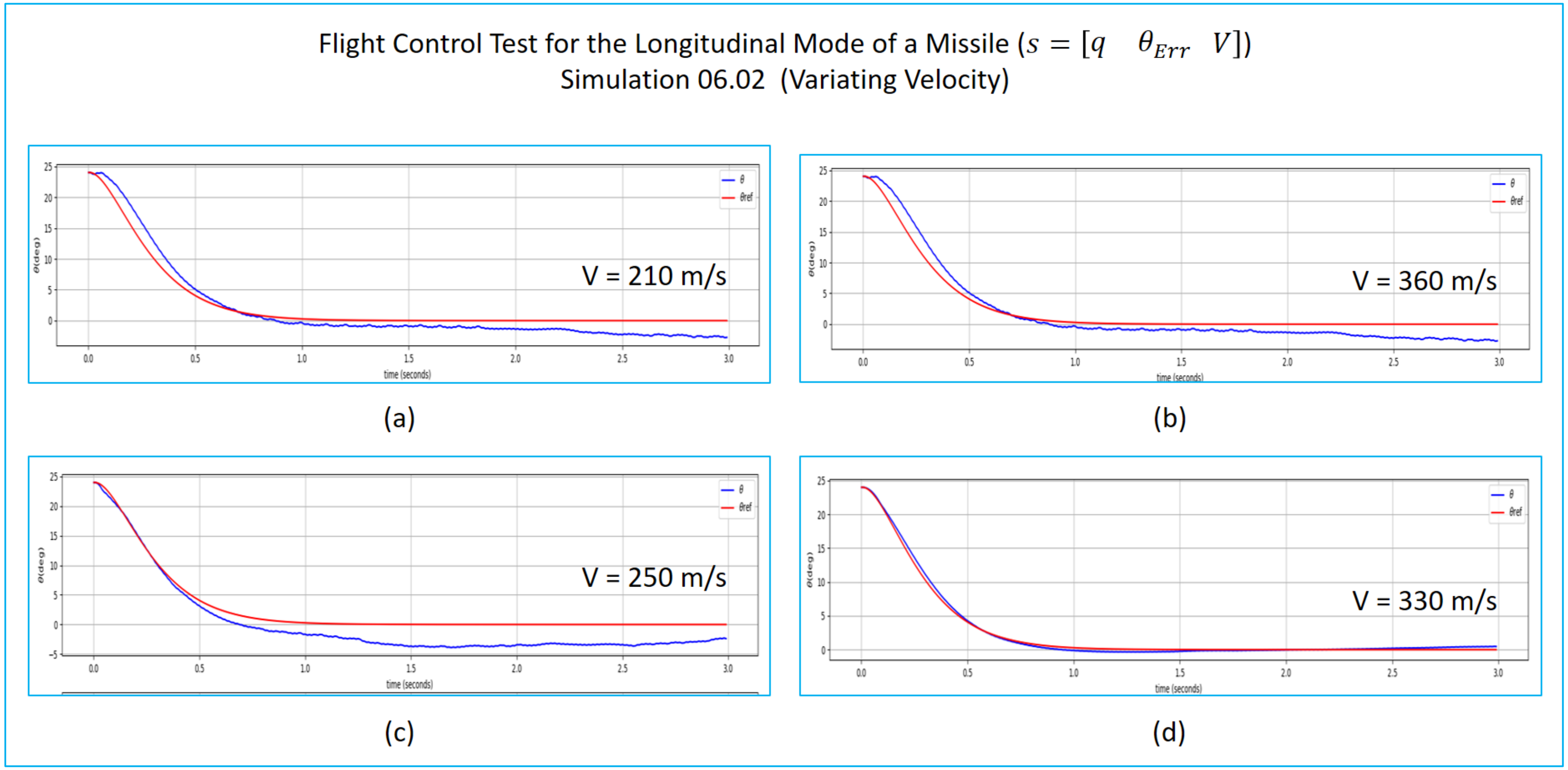


Figure 6-8 VOC Test Episode Following Training in Simulation 06.02

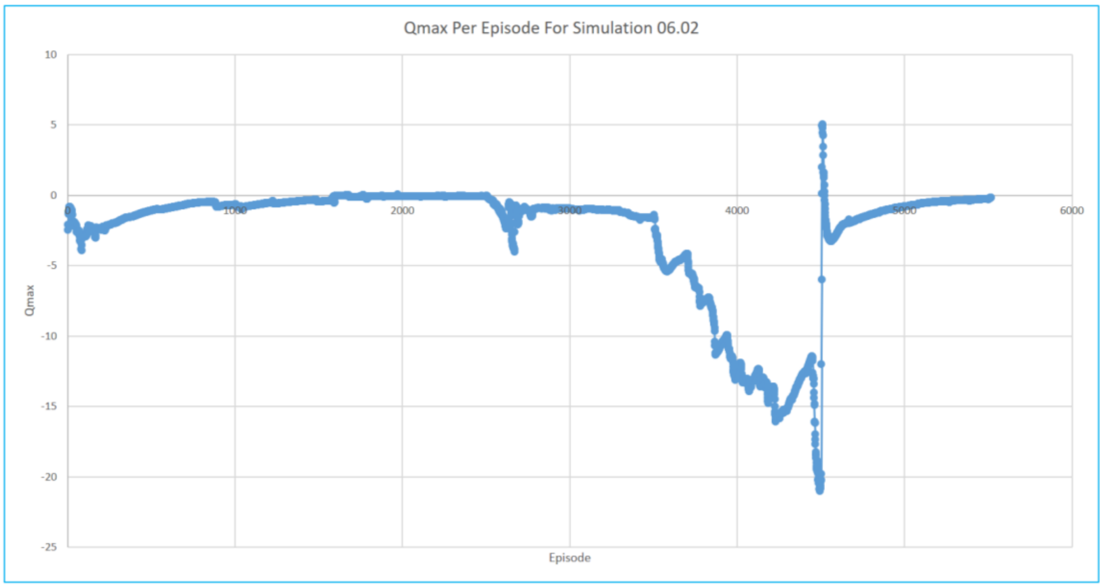


Figure 6-9 Qmax Changes in Simulation 06.02

This spike in value through this one training set can be incited by the existence of an already formed policy for velocity 300 m/s. The selected area of this policy is quiet extensive as it is trained for FOC in this velocity alone. Therefore, DDPG agent has to adjust and update quiet significantly the policy surrounding the policy for 300 m/s velocity. Hence, the changes in the value of Q_{max} is quiet extensive.

A further look into VOC training set for random velocity between 10- 60 m/s and between 360 m/s – 410 m/s shows a spike in value early on its training episodes (episode 2500s and 4500s) before settling and stabilizing towards the end of the training set. In these area, the policy value is zero as it is not yet being filled and given value. Changes in this area are compared to the initial value, which is zero.

Based on simulations 06.01 and 06.02, the result of the training for simulation 06.01 is more preferable. Adding the number of state will impact not only the number of layers but also the number of episodes for training and the formulation of reward function. For this case study, it is recommended to not change the state definition for VOC training.

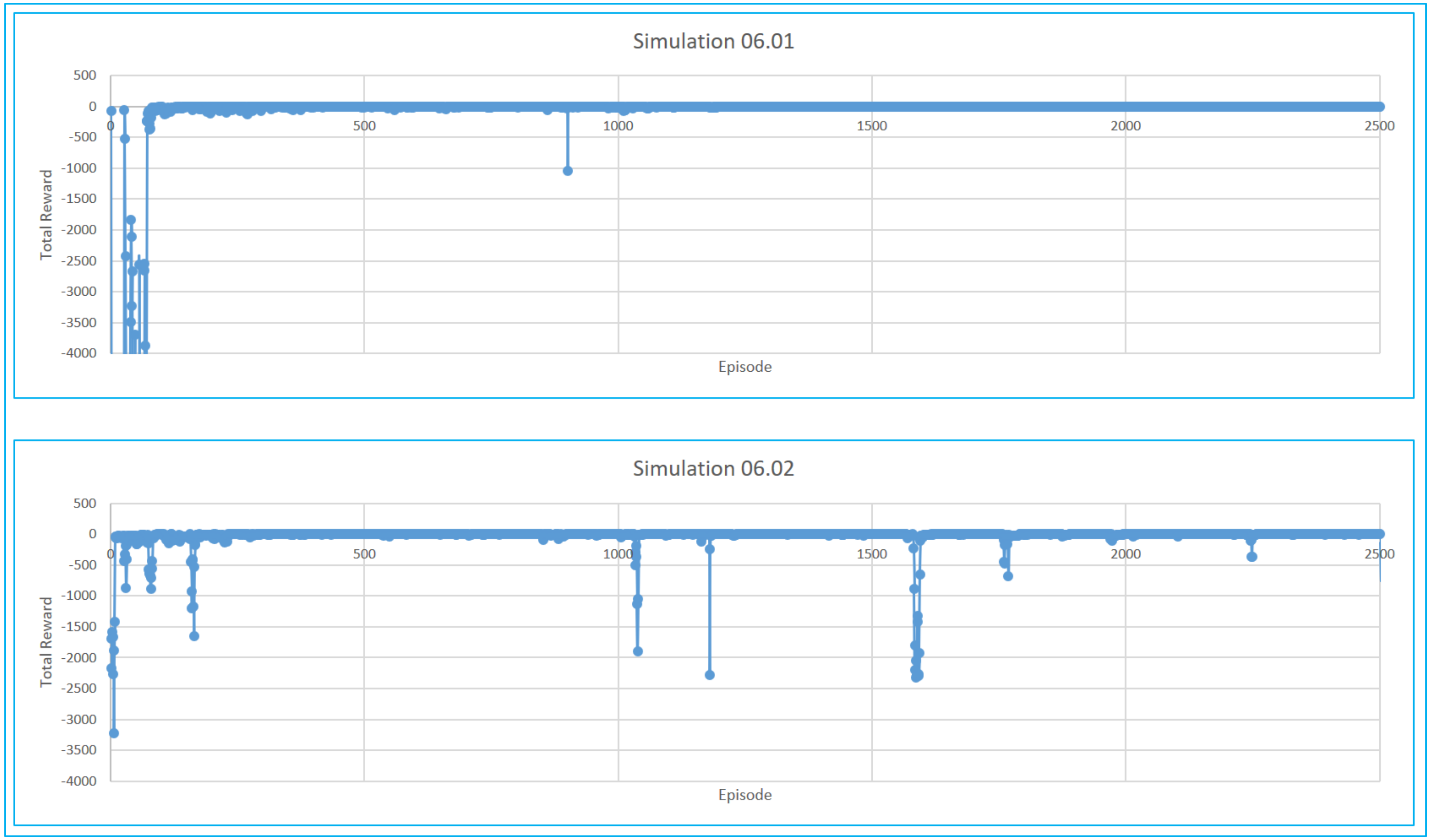


Figure 6-10 Comparison of FOC Training Set of Simulation 06.01 & 06.02

6.4 Varying Altitude

Based on the conclusion in the previous sub chapter, the simulation performed for this case study has the state definition: $s = [\theta_{err} \;\; q]$. Here the DDPG agent is trained to control the pitch angle of an air vehicle in fixed operating condition (FOC) and variating operation condition (VOC). In this case, the variation is the altitude of the air vehicle.

6.4.1 Training Strategy

As with sub chapter 6.3.1, the learning strategy for this case study focuses on two things. One, exploring the capability of the action available to the RL agent and two, exploring how much the impact of its capability changes due to a variation of altitude.

The number of layers for the network are respectively 300 and 500. As are the simulations in previous sub chapters, this simulation will focus on the missile's response in the first 3 seconds, specifically on the performance of the pitch angle. Each seconds will be divided into 300 timestep.

For the exploration policy, the Ornstein-Uhlenbeck parameter are:

Table 6-6 Ornstein-Uhlenbeck Parameter for Simulation 06.03

$\theta_{OU} = 0.3$	$\sigma_{OU} = 0.075$

The reward function for simulation 06.03 contains the timestep ($nStep$) component. It is expected that the existence of this component will guarantee the convergence of the flight performance carried out by the DDPG agent is:

$$R_1 = \sum_{j}^{nStep} {\theta_{Err}}^2 * 0.01 * nStep \tag{6-3}$$

Because the test episodes were executed following the training episodes, the RL agent is still updating its policy. Therefore, previous to this simulation, it is assumed that the result should be that the RL agent will be able to still control the pitch angle of the missile through different altitude.

The first focus is achieved by putting the RL agent through 5 sets of training in fixed operating condition (FOC), **Table 6-2**. The second focus is achieved by adding 6 sets of training in variating operating condition (VOC),**Table 6-7**. The RL agent go through all of these training sets before a set of episodes is added at the end to test the policy.

Table 6-7 Training Set for VOC (Variating Altitude)

No	Initial State (θ_0)	Command (θ_d)
	24^0	0^0
06	$random\ \ \rho = 1.225 - 1.2133\ kg/m^3$ (0 m – 200 m)	
07	$random\ \ \rho = 1.0846 - 1.0633\ kg/m^3$ (1250 m – 1450 m)	
08	$random\ \ \rho = 1.0065 - 0{,}9864\ kg/m^3$ (2000 m – 2200 m)	
09	$random\ \ \rho = 0.8191 - 0.8020\ kg/m^3$ (4000 m – 4200 m)	
10	$random\ \ \rho = 0.7364 - 0.7203\ kg/m^3$ (5000 m – 5200 m)	
11	$random\ \ \rho = 0.5252 - 0.5130\ kg/m^3$ (8000 m – 8200 m)	

For this VOC training, the RL agent is pushed to control the pitch angle in a various samples of varying altitude. These samples have an altitude range of 200 meters. A range larger than that will not result in the agent converging its learning process. However, to cover the wide range of altitude, the training samples are taken from different altitude in a range between sea level and 8500 meters. Based on previous chapters and sub chapter, the DDPG agent is expected to be able to interpolate for altitudes not used in training.

For each training set, the DDPG agent is given 500 episodes to converge its learning process.

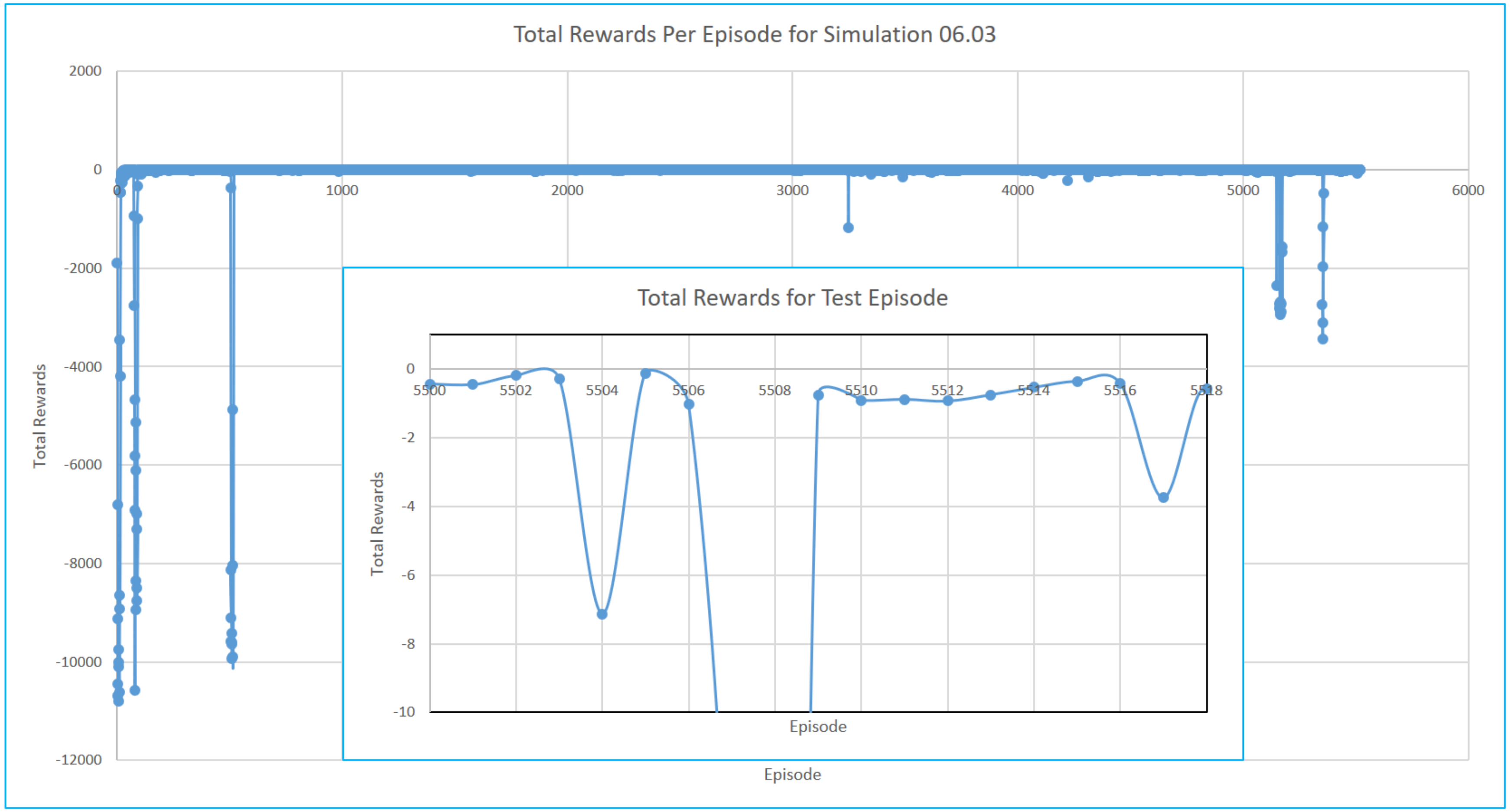


Figure 6-11 The Total Reward Per Training Episode in Simulation 06.03

For this VOC training, the RL agent is pushed to control the pitch angle in a various samples of varying altitude. These samples have an altitude range of 200 meters. A range larger than that will not result in the agent converging its learning process. The agent is given 500 episodes to converge its learning process in each range.

Figure 6-11 shows that the DDPG agent has manage to learn and produce a suitable policy after VOC and FOC training. The test episodes' numbers 5501 through 5516, except for 5507 and 5508, shows that the DDPG agent can utilize its final policy to control the flight performance of an air vehicle. This can be seen in Figure 6-12,Figure 6-13 and Figure 6-14. Test episode 5507 and 5508 are test episodes where the DDPG agent has to control the air vehicle through three different changes of pitch angle s, which it hasn't trained for. Therefore, the result is not good.

The performance of the DDPG agent in controlling the air vehicle in Figure 6-12, shows undesirable result in (c) form episode 5504. This is suspected due to the DDPG agent exploring instead of exploiting its policy. Even the steady state error in (d) shows a significant value.

Performance result in VOC shows desirable responses for various altitude condition. The timestep component in the reward function almost guarantee the convergence of the flight performance with minimal steady state error in VOC test episodes.

Figure 6-15 shows the value of Q_{max} which exceed the range [-1,1] in the last training set, which is a training set for altitude condition of 8000 – 8200 meters. This can be an indicator that the DDPG agent is unable to learn effectively to control the air vehicle. This altitude might be beyond the flight envelope of the air vehicle.

This is indicated by the value of Q_{max} during the test episodes (the insert graph in Figure 6-15). It shows that the values are within the acceptable range of [-1,1]. A further testing should be done using the policy to test the altitude condition above 8000 meter.

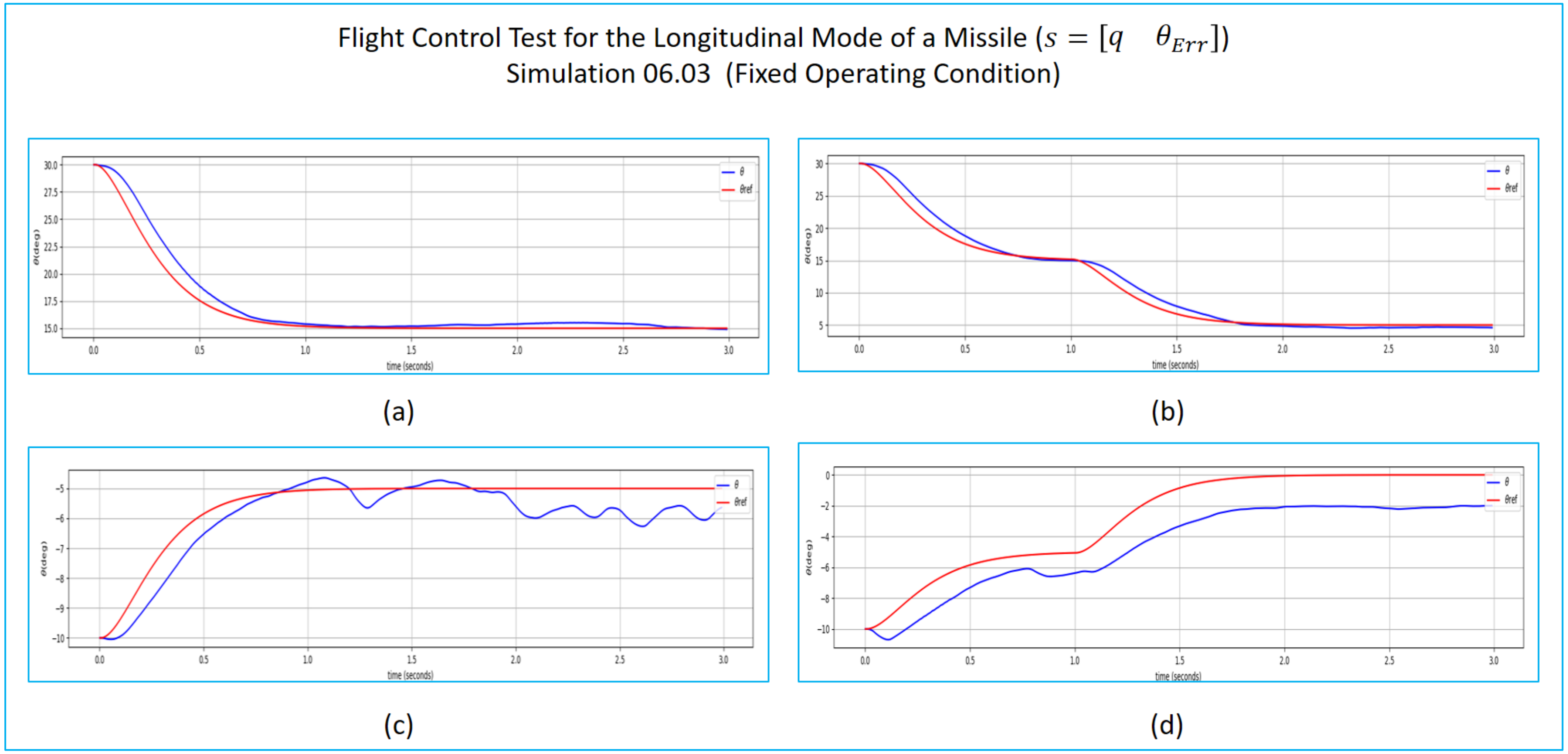


Figure 6-12 FOC Test Episode Following Training in Simulation 06.03

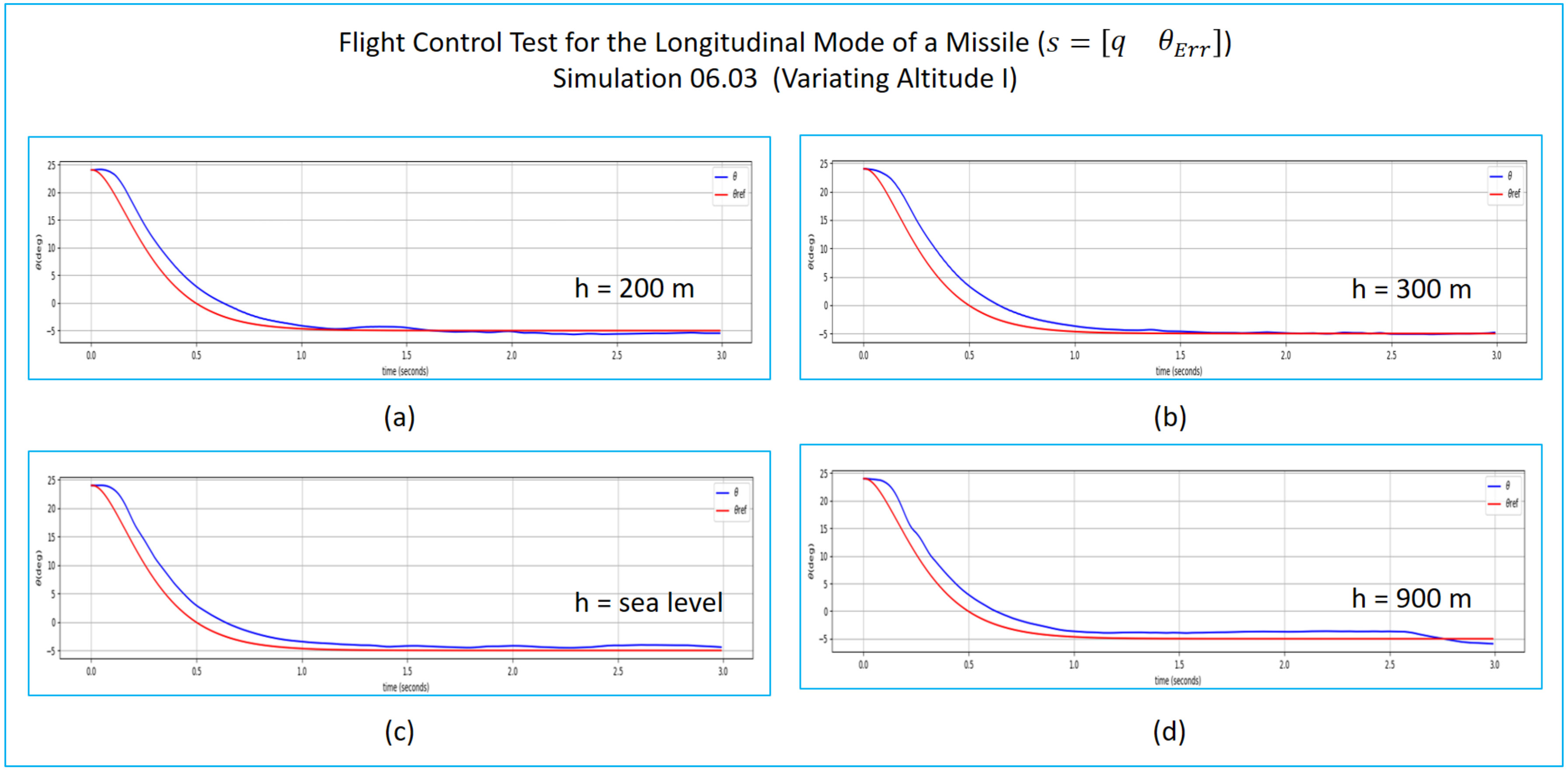


Figure 6-13 VOC Test Episode Part 1 Following Training in Simulation 06.03

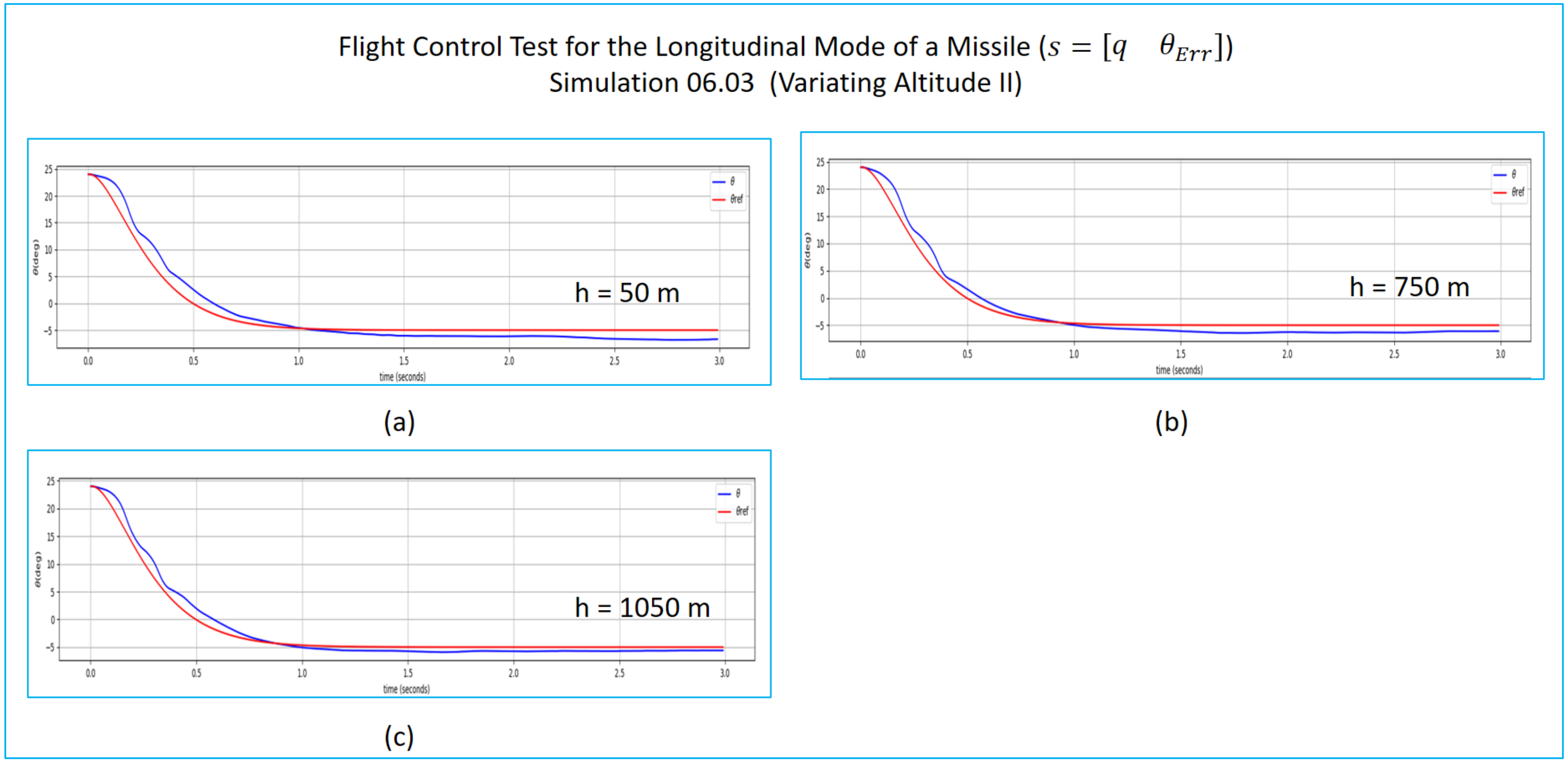


Figure 6-14 VOC Test Episode Part 2 Following Training in Simulation 06.03

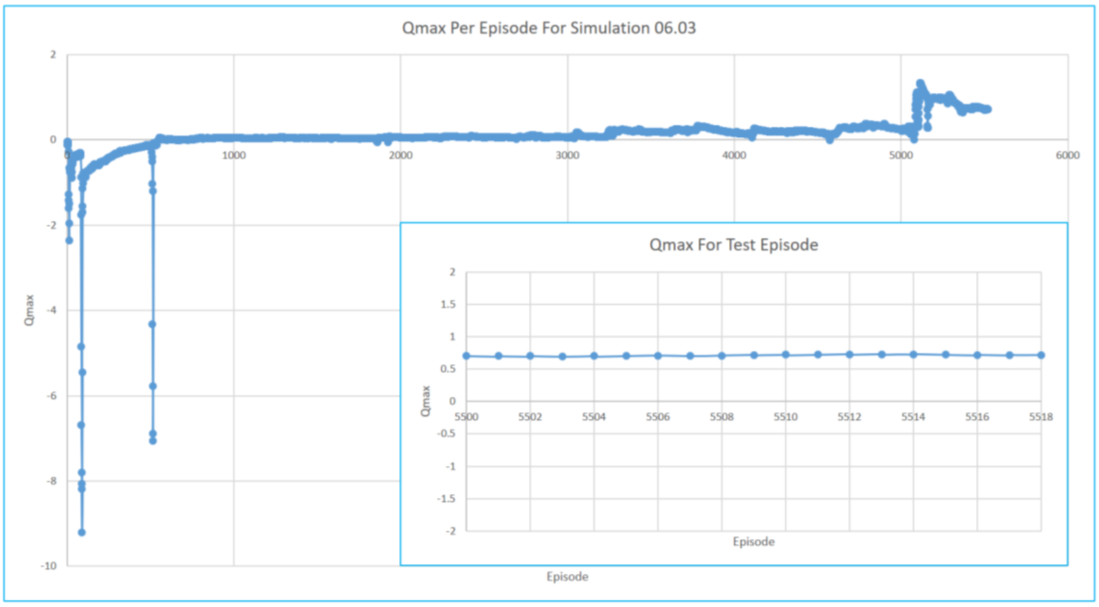


Figure 6-15 Qmax Changes in Simulation 06.03

6.5 Conclusion

Through the investigation in chapter 6, where the DDPG agent is trained to control the air vehicle in several variating operating condition, several conclusions are showed as follows.

By comparing the result of simulation 06.01 and 06.02 where both DDPG agent are trained for variations in velocity, it is concluded that:

- Adding another state to observe for this purpose is not recommended. As this is adding a complication to the ultimate goal of learning towards convergence. The DDPG agent itself can manage to add velocity to its observations implicitly with better result.
- It is recommended also to use different altitude for VOC training purposes and FOC training. Changing or updating an already established area of policy network can cost time towards convergence.

Further investigation through simulation 06.03 shows that:

- The DDPG agent can adapt to variations of altitude and velocity without having its state definition adjusted.
- A further investigation should be performed to confirm that the spike of Q_{max} value and total rewards during the last training set is due to the limitation of the air vehicle itself.

A general recommendation is that the program to train the DDPG agent for FOC and VOC is separated. The final policy from the FOC training programme can be extracted and used as the initial policy in VOC training programme. This can cut down the run time of the programme and minimize having to do the whole programme all over again when a mistake occurs somewhere in the training or test programme.

7 DEVELOPING FLIGHT CONTROL IN LATERAL-DIRECTIONAL MODE USING DDPG WITH DUAL ACTION

This chapter focuses on developing a training strategy for the RL agent to learn how to control the lateral-directional mode of an air vehicle. This is a problem because for this purpose, the DDPG method needs to use two action variables. They are aileron and rudder. In this case study, the operating condition is fixed. The air vehicle for this case study is an unmanned aerial vehicle.

7.1 Introduction

Another issue in DDPG is the use of more than one action variable. The use of more than one action variable increases the chance of the RL agent to not converge on its learning process.

Previous works that applies DDPG for a control problem has used it for one action only, such as [50].The author of this work only use the steering wheel as the action to control an autonomous land vehicle using DDPG. For a simulation using two actions, the authors use ACER.

In [51], a Model-driven Deep Deterministic Policy Gradient (MDDPG) is utilized to develop a control policy for a system with 6 dimension of action. This method basically made the search for optimal policy easier by not starting its knowledge of the system from zero. The goal was to minimize the amount of training needed and avoiding actions that may result in undesirable performance.

[4] uses only DDPG to control a bi-pedal walking robot using 4-dimension action. But it is unclear exactly how the researchers did this.

In this chapter, it is proposed that just by customizing the learning strategy, the DDPG method can develop a policy that can control the lateral-directional mode of an aircraft. For this case study, a UAS (Unmanned Aerial System) model from [52] is used.

7.2 Model

The model of Unmanned Aerial System (UAS) that is used for the simulation is derived from [52]. The nonlinear equation of motion for the lateral-directional mode of UAS are as follows:

$$\dot{v} = -ur + wp + g\sin\phi\cos\theta + \frac{QSC_Y}{m} \tag{7-1}$$

$$\dot{p} = \frac{I_{xz}}{I_{xx}}(\dot{r} + pq) - \frac{I_{zz} - I_{yy}}{I_{xx}}qr + \frac{QSbC_l}{I_{xx}} \tag{7-2}$$

$$\dot{r} = -\frac{I_{xz}}{I_{zz}}(\dot{p} - qr) - \frac{I_{yy} - I_{xx}}{I_{zz}}pq + \frac{QSbC_n}{I_{zz}} \tag{7-3}$$

$$\dot{\phi} = p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \tag{7-4}$$

$$\dot{\psi} = q\sin\phi\sec\theta + r\cos\phi\sec\theta \tag{7-5}$$

The side force coefficient (C_Y), the rolling moment coefficient (C_l) and the yawing moment coefficient (C_n) are determined by the following equations. **Table 7-2** consists of the parameters of the UAS and the aerodynamic data that correlates to its configuration.

$$C_Y = C_{Y,\beta}\beta + \left(C_{Y,p}p + C_{Y,r}r\right)\left(\frac{b}{2V}\right) + C_{Y,\delta R}\delta R \tag{7-6}$$

$$C_l = C_{l,\beta}\beta + \left(C_{l,p}p + C_{l,r}r\right)\left(\frac{b}{2V}\right) + C_{l,\delta R}\delta R + C_{l,\delta A}\delta A + \left(x_{cg} - x_{cg,ref}\right)C_Y\left(\frac{c}{b}\right)\sin\alpha \tag{7-7}$$

$$C_n = C_{n,\beta}\beta + \left(C_{n,p}p + C_{n,r}r\right)\left(\frac{b}{2V}\right) + C_{n,\delta R}\delta R + C_{n,\delta A}\delta A + \left(x_{cg} - x_{cg,ref}\right)C_Y\left(\frac{c}{b}\right)\cos\alpha \tag{7-8}$$

The maximum and minimum deflection of the control surface of the UAS is described in **Table 7-1**.

Table 7-1 Control Surface Limitation

Aileron deflection limit	$\pm 21^0$
Rudder deflection limit	$\pm 10^0$

Table 7-2 Parameters of UAS for Lateral-Directional Mode

m = 12.5 kg	c = 0.2 m	V = 20 m/s	$\rho =$ 1.21 kg/m
b = 3 m	S = 0.6 m^2		
I_{xx} = 1.446 $kg.m^2$	I_{yy} = 1.181 $kg.m^2$	I_{zz} = 2.269 $kg.m^2$	I_{xz} = 0.1 $kg.m^2$
X_{cg} = 0.564 m	$X_{cg,ref}$ = 0.512 m	$C_{l,\delta A}$ = 0.2549	$C_{n,\delta A}$ = 0.0
$C_{Y,\beta}$ = -0.6328	$C_{Y,p}$ = -0.0520	$C_{Y,r}$ = 0.2609	$C_{Y,\delta R}$ = 0.3236
$C_{l,\beta}$ = -0.1195	$C_{l,p}$ = -0.5796	$C_{l,r}$ = 0.1898	$C_{l,\delta R}$ = 0.0439
$C_{n,\beta}$ = 0.1151	$C_{n,p}$ = -0.0730	$C_{n,r}$ = -0.0901	$C_{n,\delta R}$ = -0.1041

7.3 Learning Strategy

In this case study, the RL agent observed the error between the aircraft's roll and yaw angle with the desired roll and yaw angle, (ϕ_{Err} and ψ_{Err}). However, the agent also needs to observe the roll rate (p) and yaw rate (r). It needs to see how to achieve its goal by controlling the angle rate. The state,s_t, is defined as follows.

$$s_t = [p \quad \phi_{Err} \quad r \quad \psi_{Err}] \qquad \textbf{(7-9)}$$

Based on Chapter 4, the reward function that can best represent the desired flight performance is by following a predetermined path of flight. The error in roll angle and yaw angle are defined as follows.

$$\phi_{Err} = \phi - \phi_{ref} \tag{7-10}$$

$$\psi_{Err} = \psi - \psi_{ref} \tag{7-11}$$

The desired changes in roll angle and yaw angle can be seen in the following equations.

$$\dot{\phi}_{ref} = -2\xi_{ref}\omega_{n_{ref}}\dot{\phi}_{ref} - \omega_{n_{ref}}^{2}(\phi_0 - \phi_t) \tag{7-12}$$

$$\dot{\psi}_{ref} = -2\xi_{ref}\omega_{n_{ref}}\dot{\psi}_{ref} - \omega_{n_{ref}}^{2}(\psi_0 - \psi_t) \tag{7-13}$$

ϕ_0 is defined as the initial state of roll angle and ϕ_t is defined as the final goal of the roll angle. ψ_0 is defined as the initial state of yaw angle and ψ_t is defined as the final goal of the yaw angle. ξ_{ref} is a damping ratio of 1 and ($\omega_{n_{ref}}$) is the natural frequency of 2π.

The reward function defined in the following equation.

$$R = -\sum_{j}^{nStep}\left\{-\left(\phi_{Err}{}^{2}\right) - \left(\psi_{Err}{}^{2}\right)\right\} * 0.01 * nStep \tag{7-14}$$

The reward function is multiplied by a timestep component to ensure that the steady state error in roll and yaw angle are the smallest towards the end.

Aside from the definition of the reward function, there are two things that are important in this case study. They are:

- the definition of the action space
- the learning strategy

The action, a_t, is defined as a vector that consists of aileron deflection and rudder deflection. So, the action is considered a single action but with a larger number of possibilities as it contains possible variations of two variables.

$$a_t = [(\delta A \quad \delta R)] \tag{7-15}$$

In determining the learning strategy, there is a problem with letting the RL agent loose to find its own way to balance the use of aileron and rudder. There are a large number of action possibilities that can be taken. This is added to the large number of possibilities of the state of the aircraft itself. The RL agent has to learn to control the roll angle and the yaw angle of the aircraft.

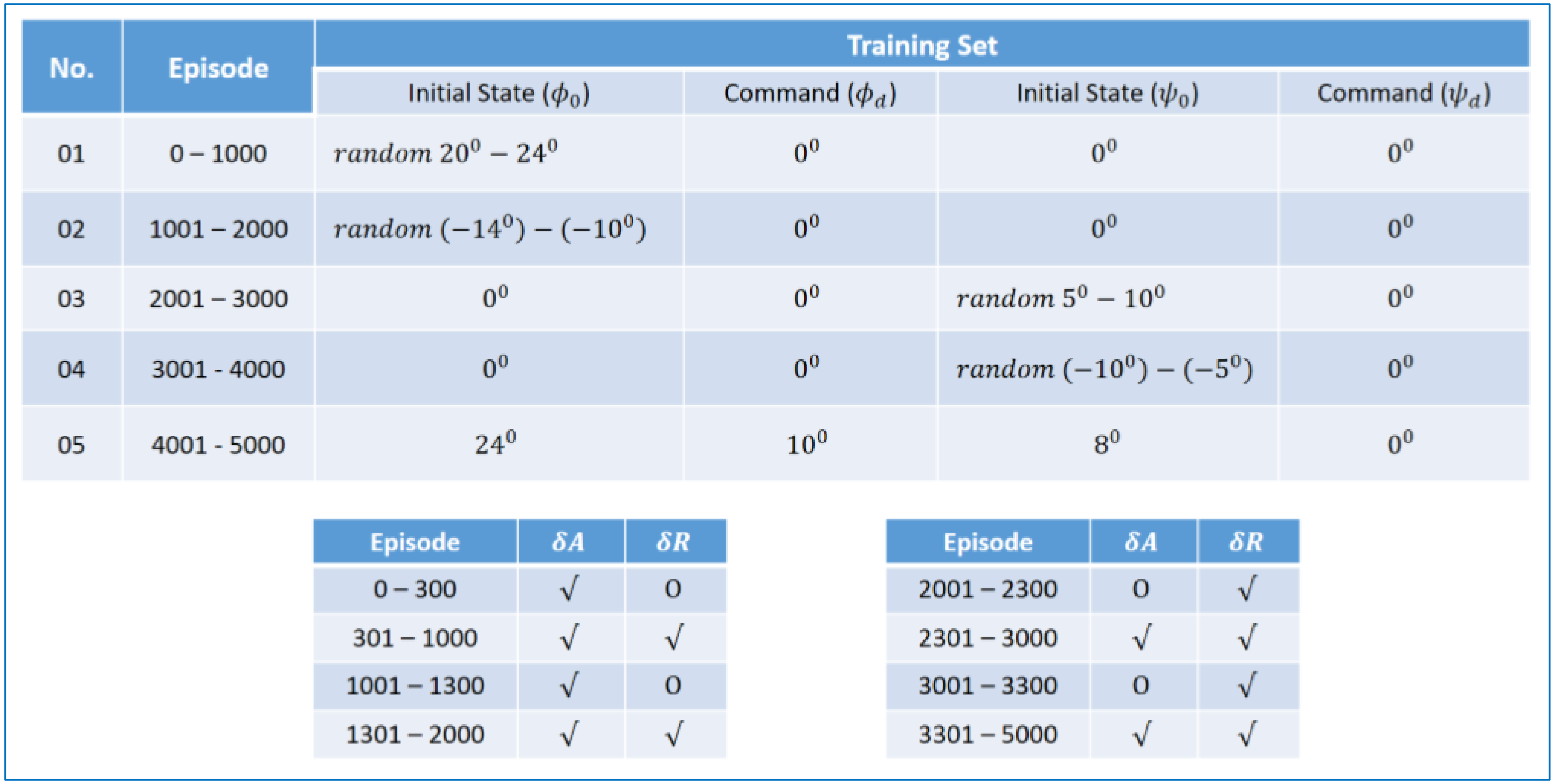
In this, an approach is made similar to that of a child learning how to use a pair of spoon and fork to eat. In a child's development, they first learn to use only the spoon to eat. Once they mastered the spoon then the fork is introduced. The child began to learn to coordinate with both spoon and fork to eat.

In learning how to control the lateral-directional mode, the RL agent is first trained how to control the roll angle. This training set is divided into two parts. In the first part, the rudder deflection is considered is considered zero. The RL agent has to learn to control the roll angle only with aileron. This will push the RL agent to a total reward value that converges to a best performance with only the use of the aileron.

Then in the second part of the training set, the rudder is added and the RL agent started to coordinate the use of both aileron and rudder to control the roll angle. This should be able to produce a converging learning process easily because the RL agent has already found the suitable policy for zero rudder deflection. How this training set works is visually explained in Figure 7-1.

The training set to learn how to control the roll angle is followed by another training set that trains the RL agent to learn how to control the yaw angle (Figure 7-2). As is the training for controlling the roll angle, the RL agent has to learn to control the yaw angle by using rudder deflection only. Then it is followed by learning to control it using both aileron and rudder deflection.

Table 7-3 The Training Sequence for Lateral-Directional Mode



No.	Episode	Training Set			
		Initial State (ϕ_0)	Command (ϕ_d)	Initial State (ψ_0)	Command (ψ_d)
01	0 – 1000	$random\ 20^0 - 24^0$	0^0	0^0	0^0
02	1001 – 2000	$random\ (-14^0) - (-10^0)$	0^0	0^0	0^0
03	2001 – 3000	0^0	0^0	$random\ 5^0 - 10^0$	0^0
04	3001 - 4000	0^0	0^0	$random\ (-10^0) - (-5^0)$	0^0
05	4001 - 5000	24^0	10^0	8^0	0^0

Episode	δA	δR
0 – 300	√	O
301 – 1000	√	√
1001 – 1300	√	O
1301 – 2000	√	√

Episode	δA	δR
2001 – 2300	O	√
2301 – 3000	√	√
3001 – 3300	O	√
3301 – 5000	√	√

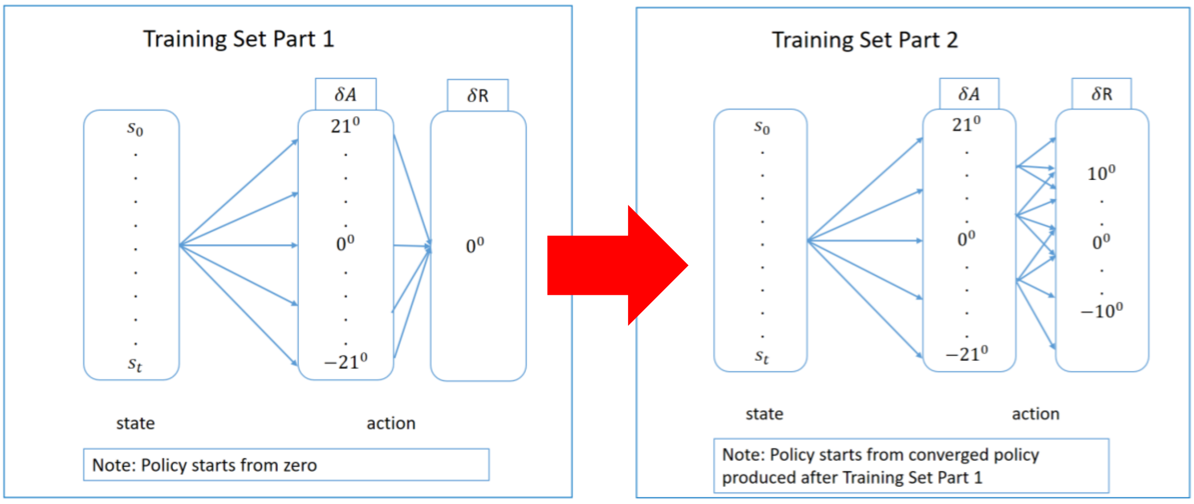


Figure 7-1 The Training Set To Control The Roll Angle

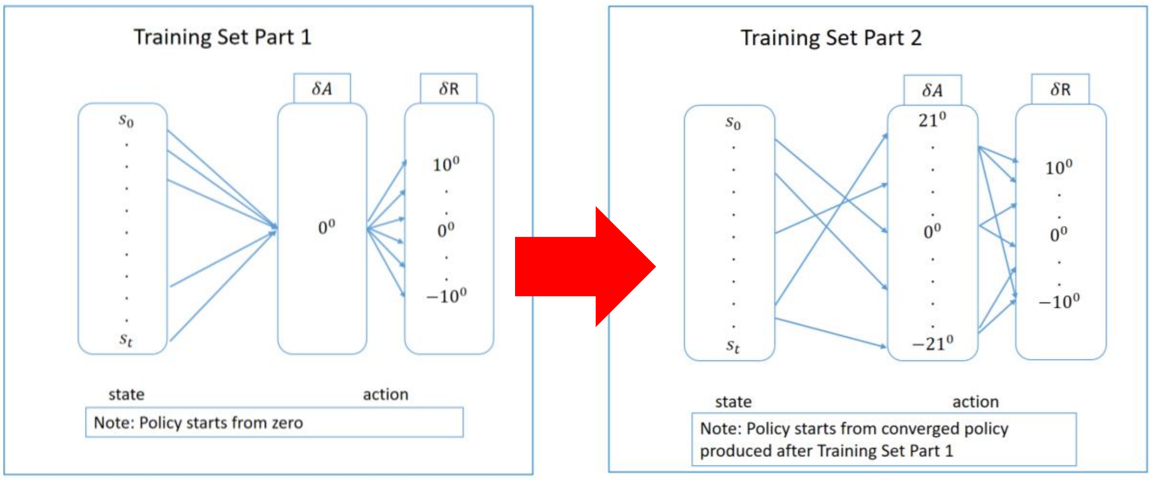


Figure 7-2 The Training Set To Control The Yaw Angle

Table 7-3 Shows the training sequence for this simulation and which part uses only one action that varies and which part has to coordinate with two actions. Each training set provides an overall of 1000 episodes for the RL agent to master its training. 2/3 of those episodes are used for learning to coordinate two actions. It Is assumed that controlling a roll or a yaw angle by coordinating two different actions takes longer to learn.

7.4 Result

In this chapter, two simulations are executed. The first simulation consists of the training process of the RL agent and the testing episode following the end of the training process. The training process applied the learning strategy to train the RL agent to learn to control the lateral-directional mode of the aircraft. The test episode following it will be to test the resulting policy.

However, there is a chance that during the testing episode, the RL agent might not respond in the best possible action. This is due to the fact that the RL agent often times explores a new or untried action for certain situation. That is why the second simulation needs to be performed.

The second simulation will use the final policy from the first simulation and applied it to another programme with the same aircraft model. However, the noise that is added to the action policy is assumed non-existent. The Ornstein-Uhlenbeck parameter are assumed zero. Therefore, the RL agent won't be exploring anymore and just exploiting the initial policy.

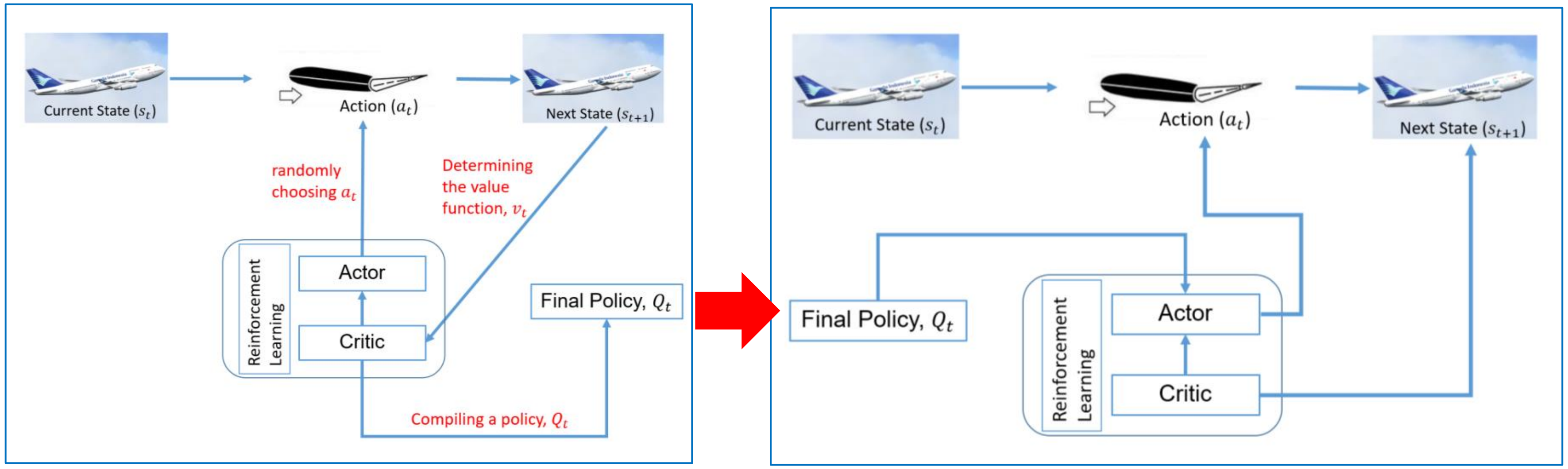
For the second simulation, the number of layers for the network must be the same between the first simulation and the second. This is because the final policy saved is in the form of a network. In this investigation the number of layers for the network are respectively 500 and 600.

For the exploration policy in the first simulation, the Ornstein-Uhlenbeck parameter are:

Table 7-4 Ornstein-Uhlenbeck Parameter for Simulation 07.01

$\theta_{OU} = 0.2$	$\sigma_{OU} = 0.05$

Figure 7-3 shows how the final policy from simulation 07.01 is extracted and applied as the initial policy in simulation 07.02. In simulation 07.02, there is no more training process for the RL agent. The RL agent simply responded to the initial condition (state) and initiate an action based on the initial policy that is no longer empty.



(i) Simulation 07.01

(ii) Simulation 07.02

Figure 7-3 The Schematic For Simulation 07.01 and 07.02

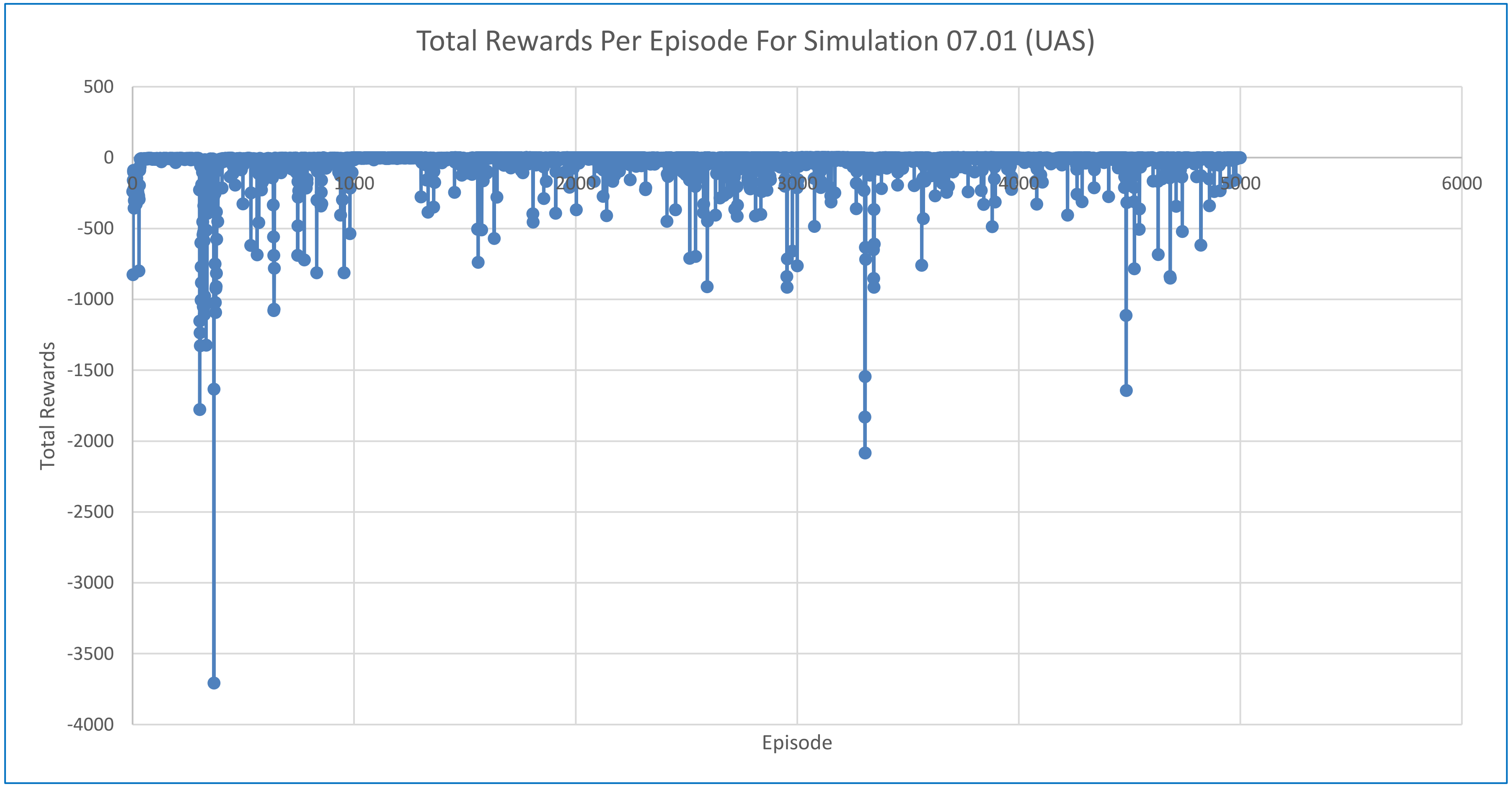


Figure 7-4 The Total Reward Per Training Episode in Simulation 07.01

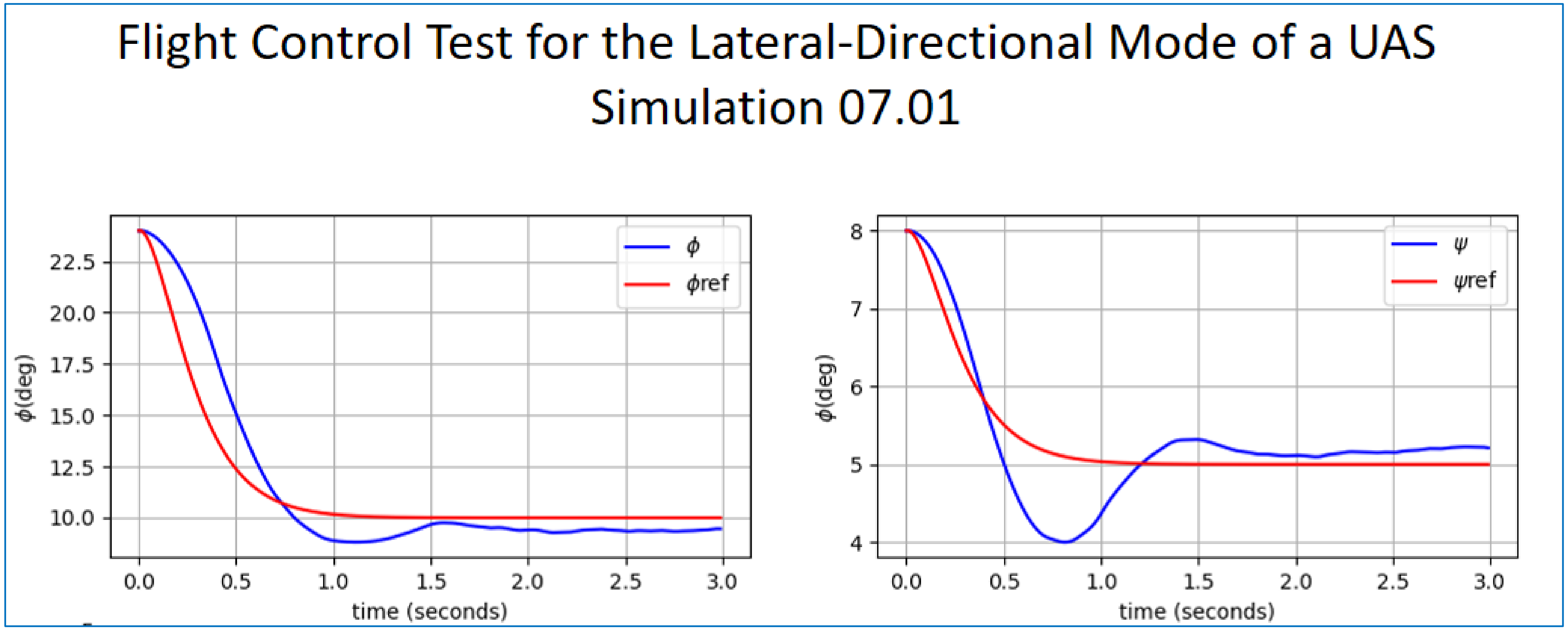


Figure 7-5 Test Episode Following Training in Simulation 07.01

The result of simulation 07.01 can be seen in Figure 7-4, Figure 7-5 and Figure 7-6. The total reward value in Figure 7-4 shows that the RL agent has manage to converge even though there are instances where the exploring episode has deviated so far from the desired response. This can be attributed to the small parameters of the noise parameters that led the RL agent to explore extremely throughout the learning process instead of clustered together throughout the earlier episodes.

For the test episode in simulation 07.01, the only changes made is the value of desired yaw angle (ψ_d). Figure 7-5 shows that the RL agent seems to be following the desired path of change in yaw angle. However, the response initiated by the RL agent doesn't overlap perfectly with the desired path. It has given a steady state error of about $\pm 10\%$. There is a possibility that this value doesn't represent the value of the RL agent exploiting its already developed policy. There is a possibility that this is a value where the RL agent are actually exploring other possibility for this initial state.

Despite the result of the episode following the training episodes, the value of Q_{max} in Figure 7-6, shows that it is trying to find its stability inside the value range of ± 0.2. This shows that the final policy resulted from simulation 07.01 is suitable enough to be applied for the purpose of controlling the system.

Based on this investigation into applying DDPG for flight control, if the value of Q_{max} towards the end of the training process is maintained at the range of ± 1.0, then the resulting policy is suitable enough to control the system.

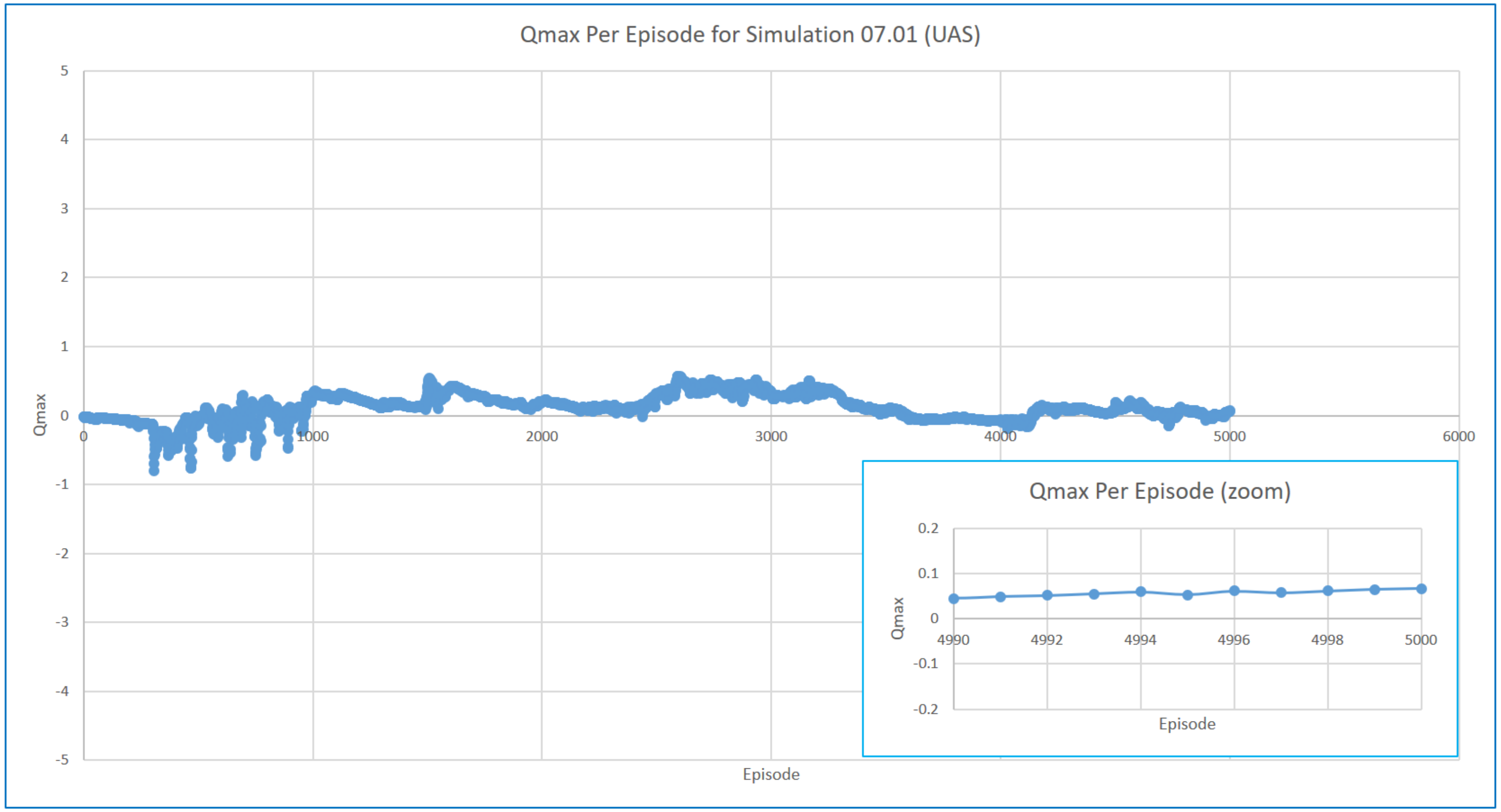


Figure 7-6 Qmax Changes in Simulation 07.01

The result of Simulation 07.02 can be seen in Figure 7-7, Figure 7-8 and Figure 7-9. Figure 7-7 shows that the initial policy used in the simulation is no longer empty (zero). This is shown by the value of total reward that are already close to zero. The response that this reward value represents can be seen in Figure 7-8.

Episodes 1-3 tries to test the policy by attempting to change just one angle, either the roll angle or the yaw angle. However, the result shows a less than stellar performance. This is due the coupling nature of lateral-directional mode in an aircraft. Any movement of the aileron will inflict the roll angle and the yaw angle. Therefore, it shows in Figure 7-8 graph (a) and (b). There’s a slight change in yaw angle, but the RL agent manage to control it and stabilize the yaw angle albeit with a steady state error. Vice versa for the rudder. The same principal applies to changes in yaw angle only, as seen in graph (c).

The response for the changes in roll angle in graph (b) and yaw angle in graph (c) might be more exactly like the desired path of change if the final policy is allowed to train some more with both control surface not limited except for the capability of the aircraft.

Figure 7-8 in graph (d) shows how the RL agent response to a change in both the roll angle and the yaw angle. It shows that the RL agent manage to generally follow the desired path. The Q_{max} in Figure 7-9 shows that during the whole test in simulation 07.02, the value doesn’t change much.

It takes roughly 48 hours in a common desktop to train an RL agent to learn to control the lateral-directional of an aircraft.

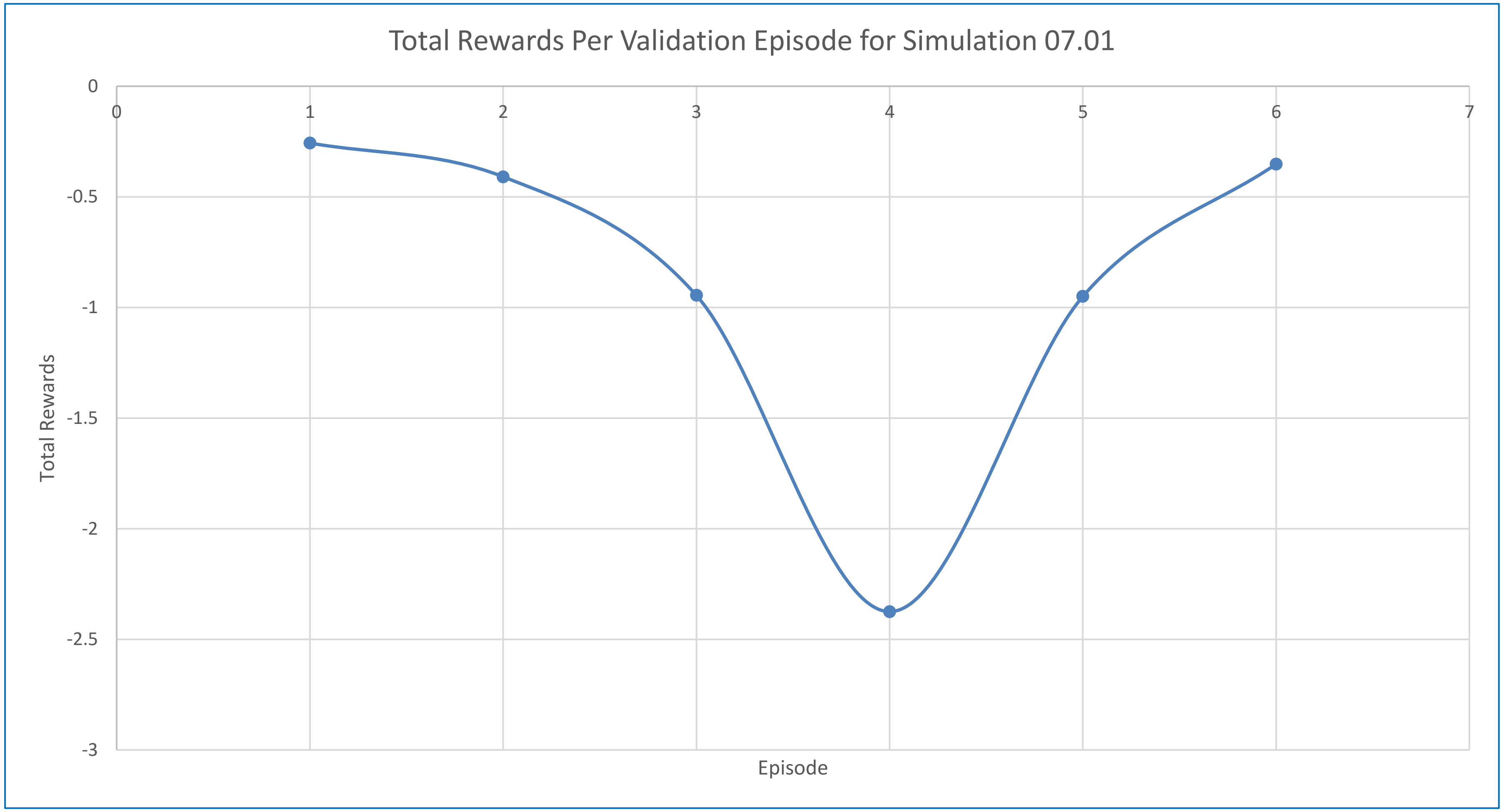


Figure 7-7 The Total Reward Per Episode for Simulation 07.02

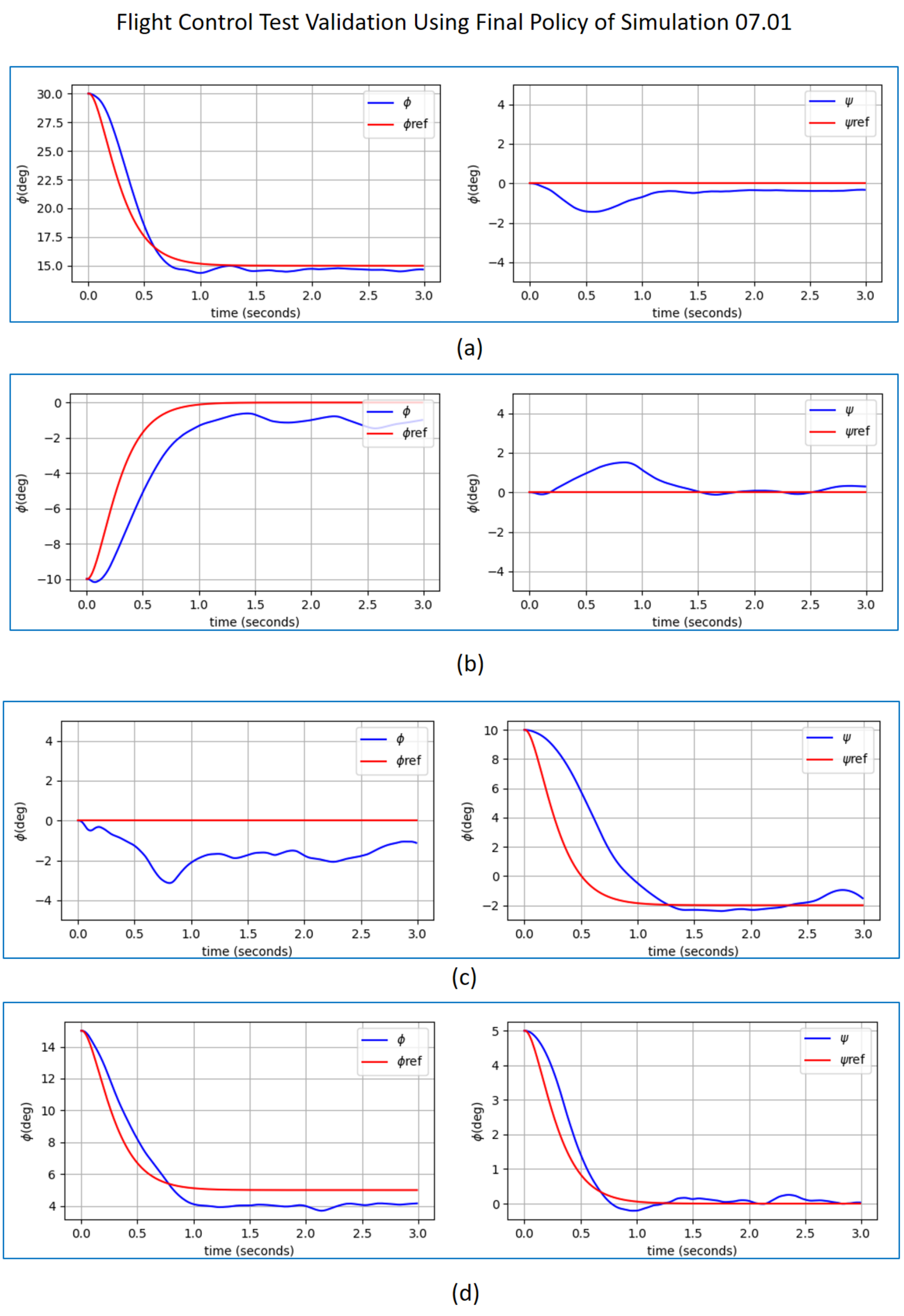


Figure 7-8 Result of Simulation 07.02

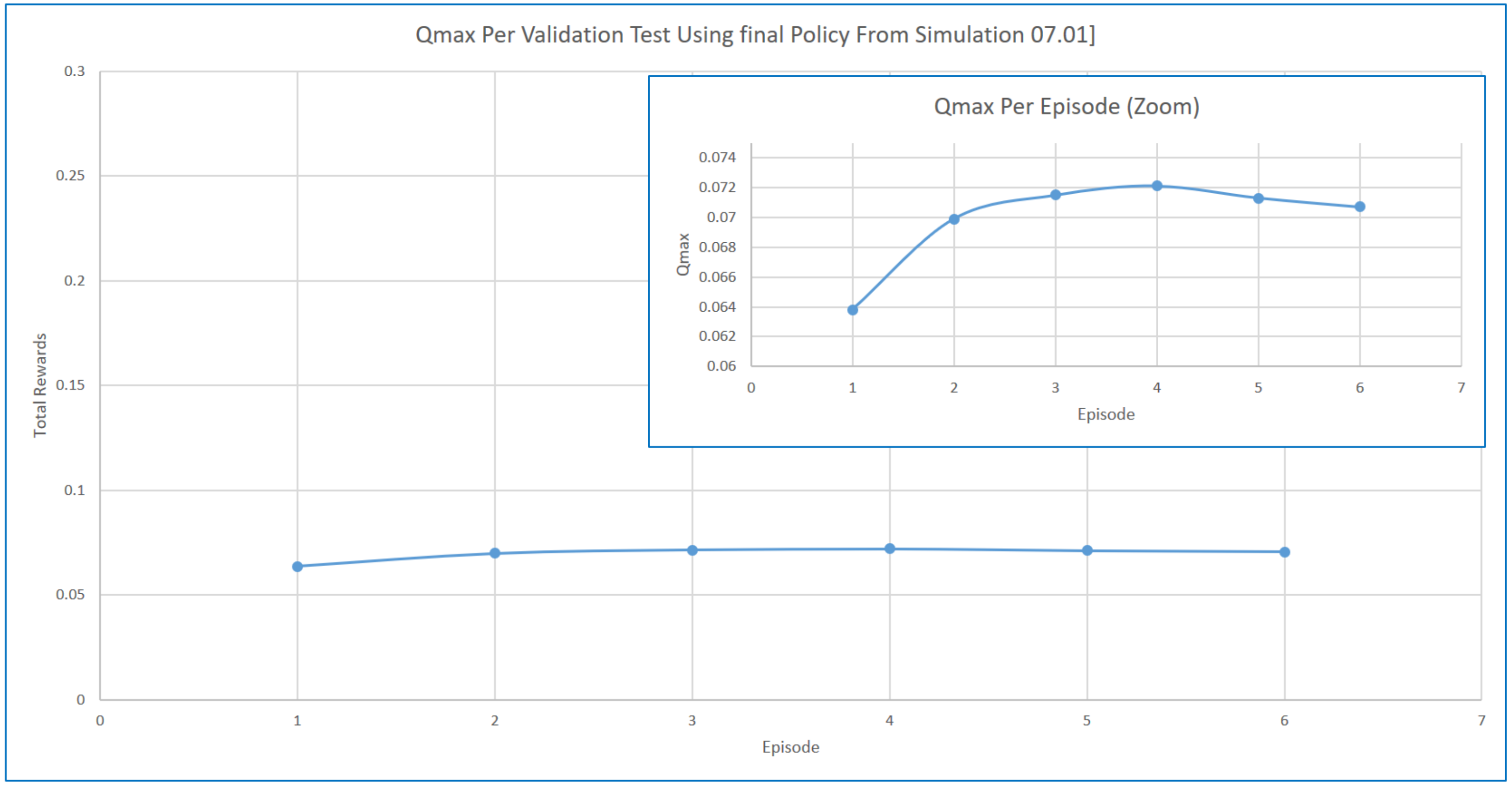


Figure 7-9 Qmax Changes in Validation of Simulation 07.02

7.5 Conclusion

Based on the results of simulations 07.01 and 07.02, it is proven that the by following the learning strategy described in sub chapter 7.3, the RL agent can learn how to utilise two control surface with coupling nature to control the lateral-directional of the aircraft

Also, by seeing the result of simulation 07.02, it is also confirming that the final policy network can be extracted and used as the initial policy in another program with the same aircraft model. This is important as in real situations the RL agent can be trained off board an aircraft and the final policy can be put on board once it is deemed suitable.

8 DEVELOPING FLIGHT CONTROL FOR 6-DEGREE-OF-FREEDOM USING DDPG WITH THREE ACTION

8.1 Introduction

This chapter focuses on developing a training strategy for the RL agent to learn how to control the full 6 degree of freedom of an air vehicle. This is a problem because in this case study, there are three action variables, which made the number of possible action combination increased exponentially. The action variables are the elevator, aileron and rudder.

The learning strategy in Chapter 7 works by first limiting the range of possible action by locking one action equals to zero. Then when the RL agent has mastered the use of one action, the locked action is unlocked and given the chance to explore.

Applying this strategy to this case study resulted in the RL agent failing to learn to control system. This is shown in the result of the total rewards per episode that doesn't converge. It failed to learn to control the longitudinal mode of the air vehicle once it masters the lateral-directional mode. Vice versa, if it learns to control the longitudinal mode of the air vehicle, then it will fail to learn to control the lateral-directional mode.

So, for this case study, a new learning strategy has to be develop to unofficially guide the RL agent through its learning process. In this case study, the operating condition is fixed. The air vehicle for this case study is an unmanned aerial vehicle.

8.2 Model

The model of UAS that is used for the simulation is a nonlinear model from[52]. The full 6-degree-of-freedom of the UAS model can be seen in the following 9 equations.

$$\dot{u} = vr - wq - g\sin\theta + \frac{QSC_x}{m} \tag{8-1}$$

$$\dot{v} = -ur + wp + g\,sin\,\phi\,cos\,\theta + \frac{QSC_Y}{m} \tag{8-2}$$

$$\dot{w} = -vp + uq + g\cos\phi\cos\theta + \frac{QSC_Z}{m} \tag{8-3}$$

$$\dot{p} = \frac{I_{xz}}{I_{xx}}(\dot{r} + pq) - \frac{I_{zz} - I_{yy}}{I_{xx}}qr + \frac{QSbC_l}{I_{xx}} \tag{8-4}$$

$$\dot{q} = -\frac{I_{xz}}{I_{yy}}(p^2 - r^2) - \frac{I_{xx} - I_{zz}}{I_{yy}}pr + \frac{QScC_m}{I_{yy}} \tag{8-5}$$

$$\dot{r} = -\frac{I_{xz}}{I_{zz}}(\dot{p} - qr) - \frac{I_{yy} - I_{xx}}{I_{zz}}pq + \frac{QSbC_n}{I_{zz}} \tag{8-6}$$

$$\dot{\phi} = p + q\,sin\,\phi\,tan\,\theta + r\,cos\,\phi\,tan\,\theta \tag{8-7}$$

$$\dot{\theta} = q\cos\phi + r\sin\phi \tag{8-8}$$

$$\dot{\psi} = q\,sin\,\phi\,sec\,\theta + r\,cos\,\phi\,sec\,\theta \tag{8-9}$$

The aerodynamic force coefficients (C_X, C_Y, C_Z) and the aerodynamic moment coefficients (C_l, C_m, C_n) can be seen in the equations below.

$$C_X = -C_D\cos\alpha + C_L\sin\alpha \tag{8-10}$$

$$C_Y = C_{Y,\beta}\beta + \left(C_{Y,p}p + C_{Y,r}r\right)\left(\frac{b}{2V}\right) + C_{Y,\delta R}\delta R \tag{8-11}$$

$$C_Z = -C_D\sin\alpha - C_L\cos\alpha \tag{8-12}$$

$$\begin{aligned} C_l = C_{l,\beta}\beta + \left(C_{l,p}p + C_{l,r}r\right)\left(\frac{b}{2V}\right) + C_{l,\delta R}\delta R + C_{l,\delta A}\delta A \\ + \left(x_{cg} - x_{cg,ref}\right)C_Y\left(\frac{c}{b}\right)sin\,\alpha \end{aligned} \tag{8-13}$$

$$C_m = C_{m,u}u + C_{m,\alpha}\alpha + \left(C_{m,\dot{\alpha}}\dot{\alpha} + C_{m,q}q\right)\left(\frac{c}{2V}\right) + C_{m,\delta E}\delta E + \left(x_{cg} - x_{cg,ref}\right)(C_L\, cos\,\alpha + C_D\, sin\,\alpha) \tag{8-14}$$

$$C_n = C_{n,\beta}\beta + \left(C_{n,p}p + C_{n,r}r\right)\left(\frac{b}{2V}\right) + C_{n,\delta R}\delta R + C_{n,\delta A}\delta A + \left(x_{cg} - x_{cg,ref}\right)C_Y\left(\frac{c}{b}\right) cos\,\alpha \tag{8-15}$$

The drag coefficient (C_D) and the lift coefficient (C_L) equations are as follows.

$$C_D = C_{D,\alpha}\alpha \tag{8-16}$$

$$C_L = C_{L,u}u + C_{L,\alpha}\alpha + C_{L,q}q\left(\frac{b}{2V}\right) + C_{L,\delta E}\delta E \tag{8-17}$$

Table 8-1 Parameters of UAS for 6-degree-of-freedom

m = 12.5 kg	c = 0.2 m	V = 20 m/s	ρ = 1.21 kg/m
b = 3 m	S = 0.6 m^2		
I_{xx} = 1.446 $kg.m^2$	I_{yy} = 1.181 $kg.m^2$	I_{zz} = 2.269 $kg.m^2$	I_{xz} = 0.1 $kg.m^2$
X_{cg} = 0.564 m	$X_{cg,ref}$ = 0.512 m		
$C_{L,\alpha}$ = 5.5138	$C_{L,q}$ = 7.4673	$C_{L,u}$ = 0.0024	$C_{L,\delta E}$ = 0.2649
$C_{m,q}$ = -22.5924	$C_{m,\delta E}$ = -1.1893	$C_{m,u}$ = 0.0003	$C_{m,\dot{\alpha}}$ = -4.1034
$C_{m,\alpha} = -1.6510$	$C_{D,a} = 0.2188$	$C_{l,\delta A}$ = 0.2549	$C_{n,\delta A}$ = 0.0
$C_{l,\beta}$ = -0.1195	$C_{l,p}$ = -0.5796	$C_{l,r}$ = 0.1898	$C_{l,\delta R}$ = 0.0439
$C_{n,\beta}$ = 0.1151	$C_{n,p}$ = -0.0730	$C_{n,r}$ = -0.0901	$C_{n,\delta R}$ = -0.1041

8.3 Learning Strategy

In this case study, the RL agent has to be able to learn to control the roll ϕ, pitch θ and yaw angle ψ using three control surfaces which are elevator, aileron and rudder. Therefore, the definition of state and action is as followed.

The state being observed are the error between the aircraft's roll, pitch and yaw angle with the desired roll, pitch and yaw angle, (ϕ_{Err} , θ_{Err} and ψ_{Err}). In addition, the agent also needs to observe the roll rate (p), pitch rate (q) and yaw rate (r). It needs to see how to achieve its goal by controlling the angle rate. The state,s_t, is defined as follows.

$$s_t = [p \quad \phi_{Err} \quad r \quad \psi_{Err} \quad q \quad \theta_{Err}\,] \tag{8-18}$$

Based on Chapter 4, the reward function that can best represent the desired flight performance is by following a predetermined path of flight. The error in roll angle and yaw angle are the same as defined in Chapter 7.

$$\phi_{Err} = \phi - \phi_{ref}$$

$$\psi_{Err} = \psi - \psi_{ref}$$

The error in pitch angle are the same as defined in Chapter 5.

$$\theta_{Err} = \theta - \theta_{ref}$$

The equation desired changes in roll angle and yaw angle are the same as the one in Chapter 7. The equation for the desired changes in pitch angle is the same as in Chapter 5. Those equations are written again below.

$$\dot{\phi}_{ref} = -2\xi_{ref}\omega_{n_{ref}}\dot{\phi}_{ref} - \omega_{n_{ref}}^2(\phi_0 - \phi_t)$$

$$\dot{\psi}_{ref} = -2\xi_{ref}\omega_{n_{ref}}\dot{\psi}_{ref} - \omega_{n_{ref}}^2(\psi_0 - \psi_t)$$

$$\dot{\theta}_{ref} = -2\xi_{ref}\omega_{n_{ref}}\dot{\theta}_{ref} - \omega_{n_{ref}}^2(\vartheta_0 - \theta_t)$$

ϕ_0 is defined as the initial state of roll angle and ϕ_t is defined as the final goal of the roll angle. ψ_0 is defined as the initial state of yaw angle and ψ_t is defined as the final goal of the yaw angle. θ_0 is defined as the initial state of pitch angle and θ_t is defined as the final goal of the pitch angle. ξ_{ref} is a damping ratio of 1 and ($\omega_{n_{ref}}$) is the natural frequency of 2π.

The reward function defined in the following equation.

$$R = -\sum_{j}^{nStep}\{-(\phi_{Err}{}^2) - (\psi_{Err}{}^2) - (\theta_{Err}{}^2)\} * 0.01 * nStep \tag{8-19}$$

Again, the reward function is multiplied by a timestep component to ensure that the steady state error in roll, pitch and yaw angle are the smallest towards the end.

The action, a_t, is defined as a vector that consists of aileron deflection, elevator deflection and rudder deflection. So, the action is considered a single action but with a larger number of possibilities as it contains possible variations of three variables.

$$a_t = [(\delta A \quad \delta R \quad \delta E)] \tag{8-20}$$

The limitation of the control surface deflection is determined as follow.

Table 8-2 Control Surface Limitation

Aileron deflection limit	$\pm 21^0$
Rudder deflection limit	$\pm 10^0$
Elevator deflection limit	$\pm 25^0$

In determining the learning strategy, another problem has occurred while moving on from the learning strategy of controlling the lateral-directional mode. The same concept cannot be directly applied to the d DoF system. This is because the amount of possible combination of actions has greatly increased.

Another aspect is that even though in conventional flight control design, the longitudinal mode is usually handled separately to the lateral-directional mode, there is however an influence that crosses between the two. Therefore, when learning to control one mode, all three actions must engage/active from the start of learning.

This case study proposes an approach by applying the way a child learns to stand/walk. When they first learn to stand, they needed their hands to lean on other surfaces to have balance. Then as they started to master how to stand, they started to walk with both hands still stretched for balance. Much further and they would start to keep their hands by their side. However, the hands will always function to maintain balance, because if a person loses one of the hands, it will impact how they try to stand.

In learning how to control the aircraft in a 6 DoF system, the learning strategy is divided into three training set. Each training set will train the RL agent to learn to control its one mode, either it is longitudinal/lateral/directional, while still maintaining the stability of the other modes.

In each of the training set, no control surface is left with zero deflection. However, certain limitation is put on one or two of the control surfaces, depending on the flight mode its currently trying to control. This is done so the RL agent will have a less number of possible combination of action and therefore incited a faster convergence.

This is also done to avoid the RL agent from having to re-learn and repopulate its policy network. This will happen, for example, when the RL agent has to learn to control the roll angle whilst the elevator and rudder are locked and unable to operate. When they are unlocked, then the policy would have to be adjusted again as the pairing of state-action suddenly grew and the RL agent has to re-learn to control the roll angle with all control surfaces operating.

However, if the RL agent already learning from the beginning how to control the roll angle with all control surfaces operating, albeit with some range limitation, then the RL agent don't have to work twice and the range limitation will help converge the learning process faster. **Table 8-3** and **Table 8-4** shows how the training is executed.

Table 8-3 Training Set A To Learn Control of Roll Angle

No	Episode	Training Set				Control Surface		
		Initial State (ϕ_0)	Command (ϕ_d)	Initial State (θ_0)	Command (θ_d)	δA	δR	δE
01	0 – 300	$random$ $20^0 - 24^0$	0^0	0^0	0^0	√	χ_R	χ_E
02	301-600					√	√	χ_E
03	601-900					√	√	χ_E
04	901-1200	$random$ $(-14^0) - (-10^0)$	0^0	0^0	0^0	√	χ_R	χ_E
05	1201-1500					√	√	χ_E
06	1501-1800					√	√	χ_E

Further explained, in training set A, the RL agent learns how to control the roll angle. In the first 300 training episode, the agent is given a full range of the aileron deflection. This means in the range between -21^0 to $+21^0$. However, the range of rudder deflection and elevator deflection is limited.

The rudder deflection is limited in a range of $\chi_R = \pm 3^0$. The elevator deflection is limited to a range of $\chi_E = \pm 8^0$. These two control surfaces are important because it is needed to control any movement in yaw and pitch angel due to movement in roll angle. However, it is limited so that the RL agent can converge swiftly.

The purpose of giving the RL agent the availability to utilize a small part of the other two control surfaces is not to explore in them. That will be done in the other part of the training. The rudder deflection is explored during the second 300 episodes and the elevator deflection is explored during the third 300 episodes.

In training set B, the RL agent learns how to control the pitch angle. In the first 300 training episode (episodes 1801-2100), the agent is given a full range of the elevator deflection. This means in the range between -25^0 to $+25^0$. However, the range of rudder deflection and aileron deflection is limited. The rudder deflection is limited in a range of $\chi_R = \pm 3^0$. The aileron deflection is limited to a range of $\chi_A = \pm 5^0$.

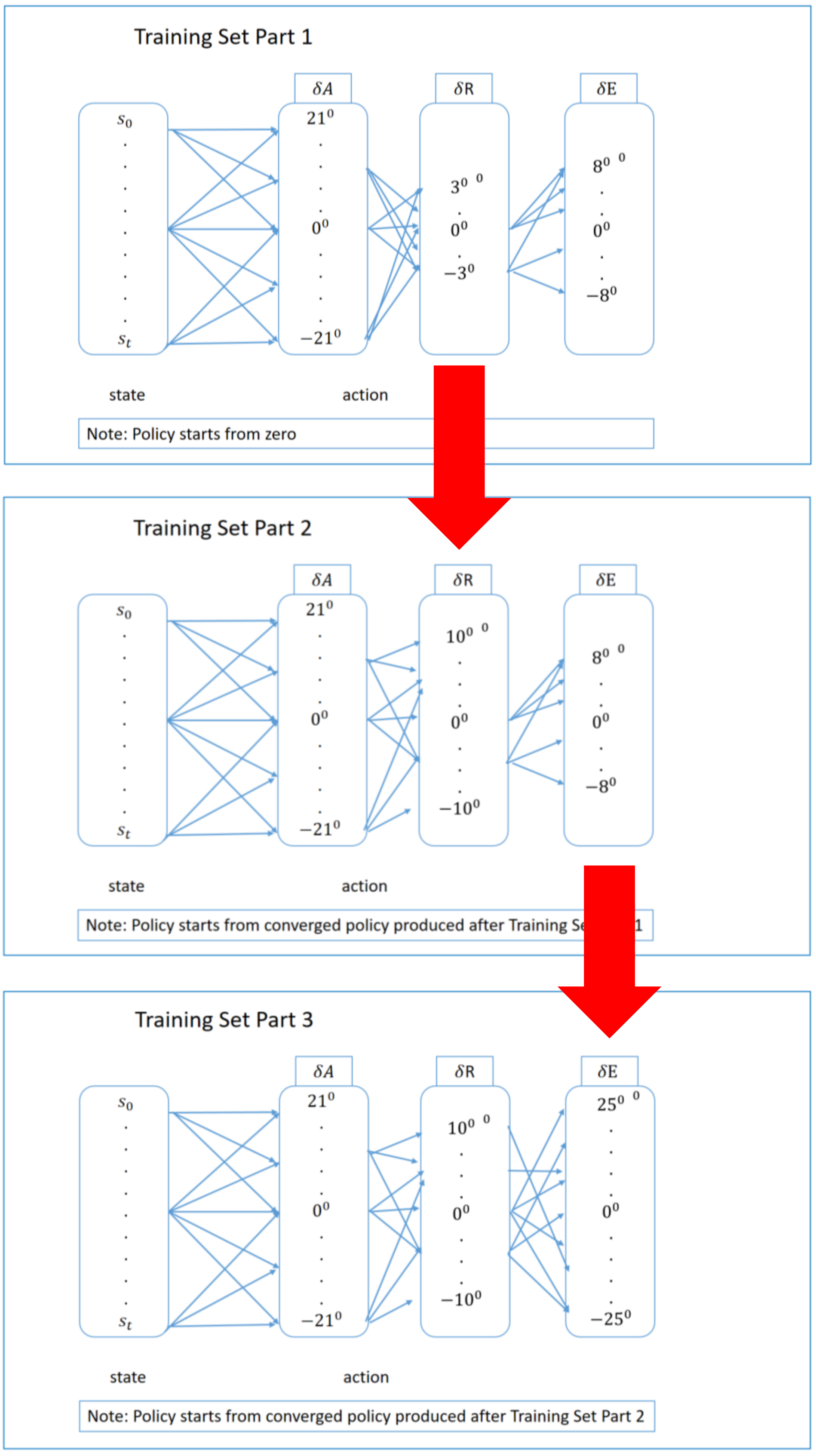


Figure 8-1 The Training Set To Control The Roll Angle

Table 8-4 Training Set B To Learn Control of Pitch Angle

No	Episode	Training Set				Control Surface		
		Initial State (ϕ_0)	Command (ϕ_d)	Initial State (θ_0)	Command (θ_d)	δA	δR	δE
07	1801-2100	0^0	0^0	$random$ $20^0 - 24^0$	0^0	χ_A	χ_R	√
08	2101-2400					√	χ_R	√
09	2401-2700					√	√	√
10	2701-3000	0^0	0^0	$random$ $(-14^0) - (-10^0)$	0^0	χ_A	χ_R	√
11	3001-3300					√	χ_R	√
12	3301-3600					√	√	√

Similar with training set A, in training set C, the RL agent learns how to control the yaw angle. In the first 300 training episode (episodes 3601-3900), the agent is given a full range of the rudder deflection. This means in the range between -8^0 to $+8^0$. However, the range of elevator deflection and aileron deflection is limited. The elevator deflection is limited in a range of $\chi_E = \pm 8^0$. The aileron deflection is limited to a range of $\chi_A = \pm 5^0$.

In this case study, the number of layers for the network are respectively 1400 and 1500. All the simulations will focus on the aircraft's response in the first 3 seconds and each second will be divided into 300 timestep (nStep). For the exploration policy, the Ornstein-Uhlenbeck parameter are:

Table 8-5 Ornstein-Uhlenbeck Parameter for Simulation 08.01

$\theta_{OU} = 0.3$	$\sigma_{OU} = 0.05$

Running the whole programme with three control surface requires a lot of computer memory and processor capability. To lessen the load and to made trouble shooting more easily, the training programme is divided into 4 programmes. These programmes consists of:

- Training set A, to learn to control the lateral mode
- Training set B, to learn to control the longitudinal mode
- Training set C, to learn to control the directional mode

- Training set D, to learn to control by coordinating between all 3 control surfaces.

8.4 Result

In the simulation to train the RL agent to control the 6 degree-of-freedom of an aircraft, the training session is divided into 4 separate programs. This is because it takes a lot of memory power and processor to run the training all at once. So, it is decided to separate the training program. Each final policy network from the previous program becomes the initial policy for the next training program.

The first three program trains the RL agent to focus on controlling the roll angle (training set A), pitch angle (training set B) and the yaw angle (training set C). In the fourth program, the RL agent is trained to utilize the whole range of the 3 control surfaces. Figure 8-2 describes how the policy network from a training programme is extracted and used as the initial policy network in another training programme.

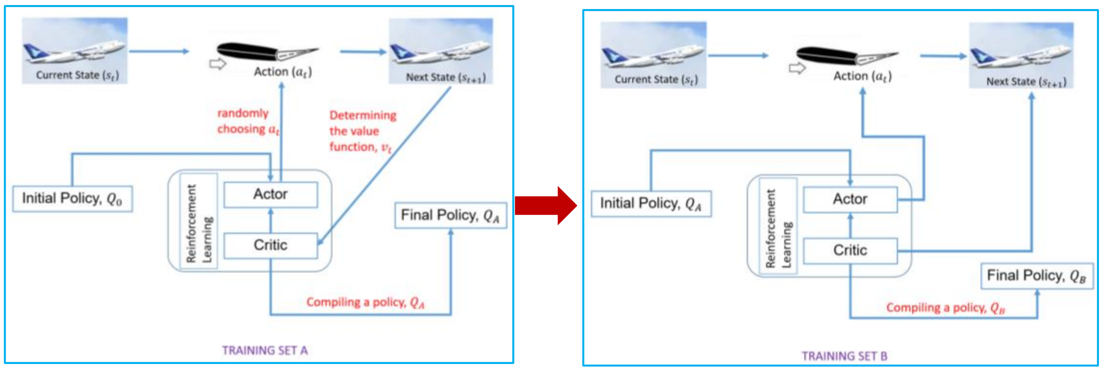


Figure 8-2 Using Policy From Another Programme As The Initial Policy

The following figures shows the result for each training program and the resulting response of the RL agent towards the initial state.

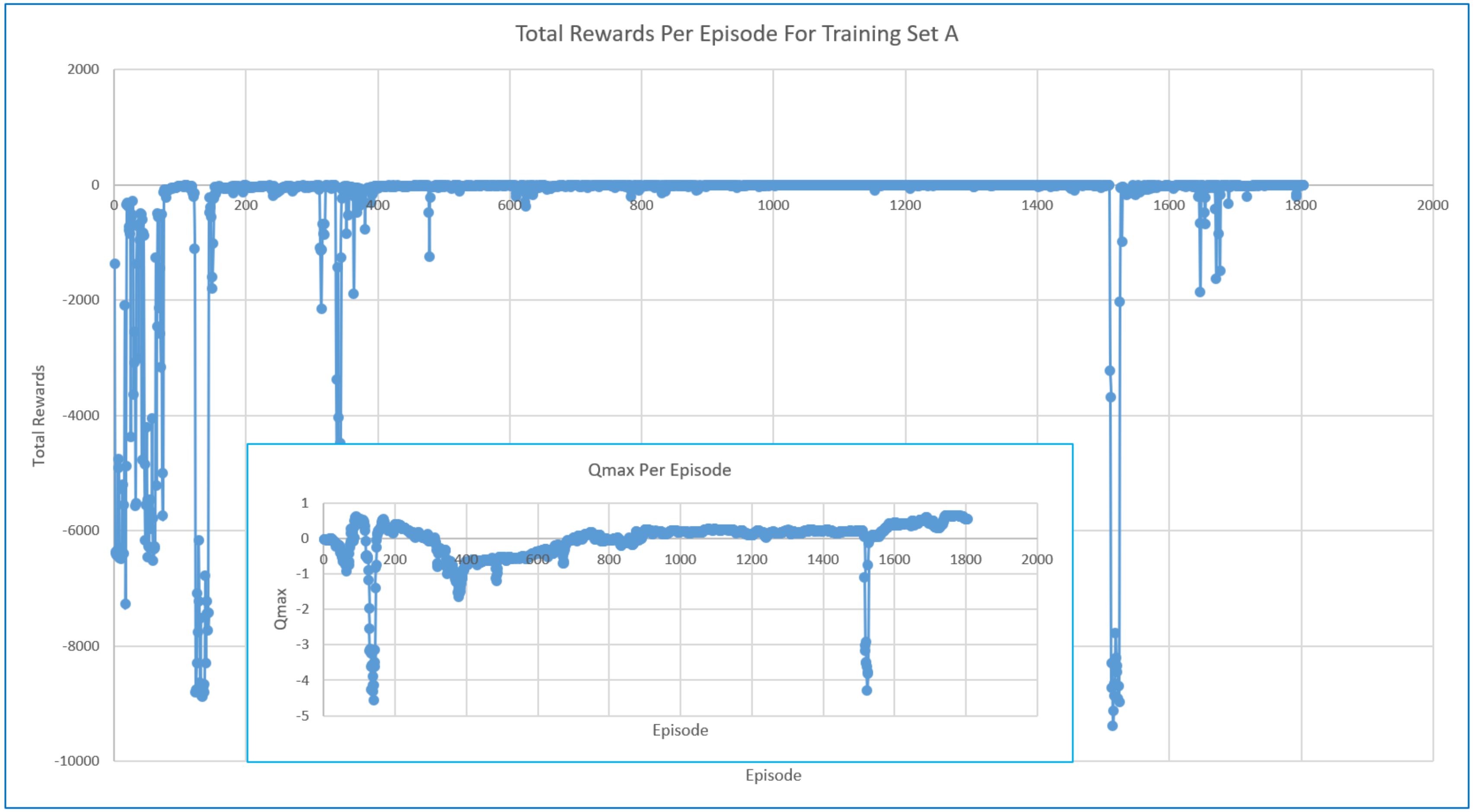


Figure 8-3 The Total Reward and Qmax Per Episode in Training Set A

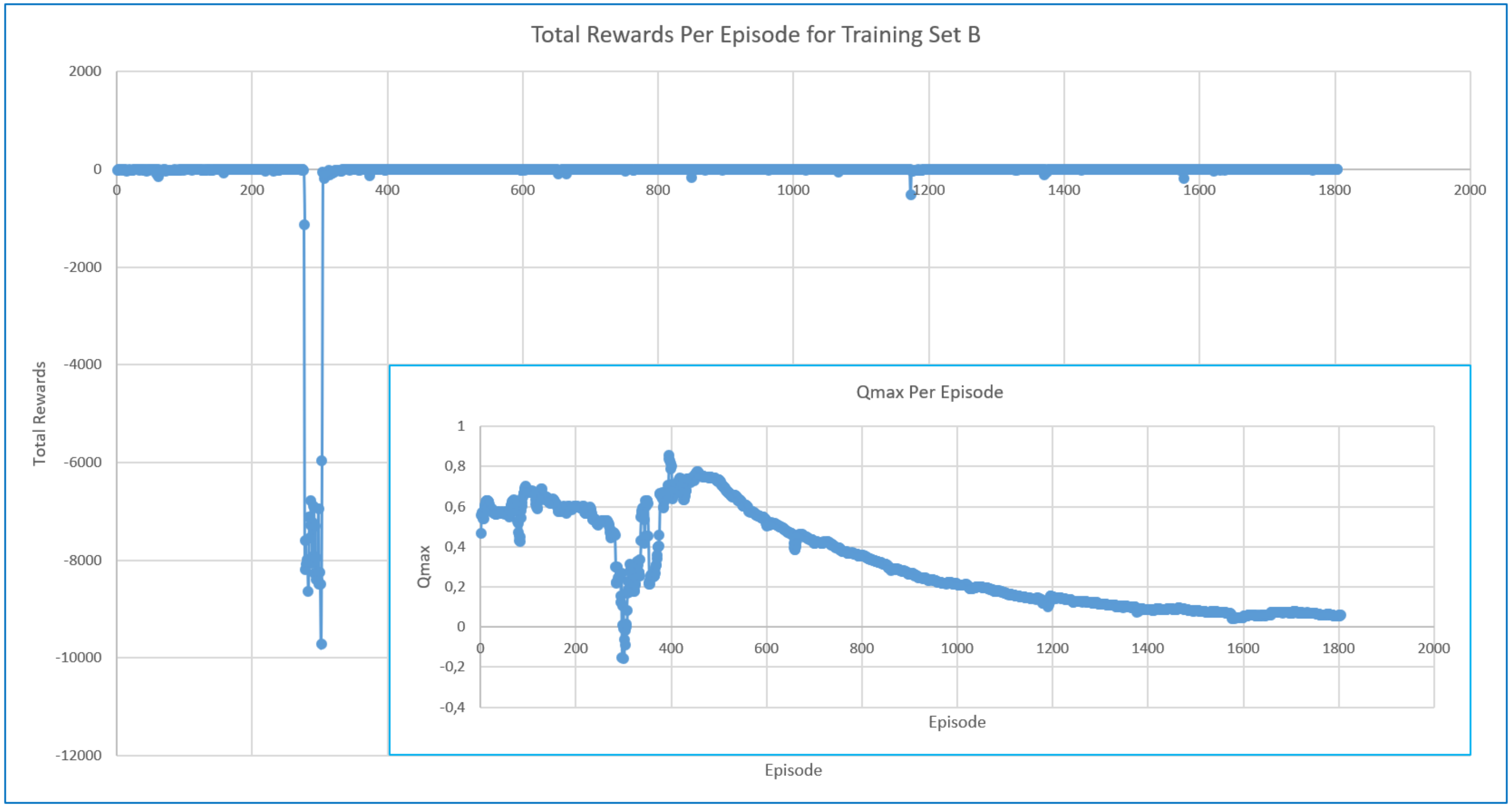


Figure 8-4 The Total Reward and Qmax Per Episode in Training Set B

Figure 8-3 and Figure 8-4 shows that the RL agent's learning process through training set A and B have gone successful. This is shown with the converging of the total reward value for each episode in the training set. Training set A was designed to train to control the roll angle and training set B was designed to train to control the pitch angle. Judging from the value of Q_{max} on training set A, it shows the value in the last 200 episodes are between -1 and 1, which previous simulations show that the final policy is suitable.

The result performance of training set A in Figure 8-5 shows that although the roll angle manages to follow the desired trajectory, the response itself is oscillating. This could be attributed to the fact that it is still exploring the possible actions.

In Figure 8-6, the RL agent has manage to control the pitch angle response to generally follow the desired path of change. The response does give a large steady state error. This result is considered acceptable at the moment as it is not the final policy that will be used.

Figure 8-9 shows that although the total rewards value shows convergence, the value of Q_{max} is barely 1. This shows in the performance result in Figure 8-7, where the angles of pitch roll and yaw all diverge even though the yaw angle generally follow the desired path first before diverging.

Now, the resulting policy from training set C is still used for the initial policy in training set D. This is with an assumption that the RL agent will continue to learn to control the whole system in training set D. It is expected that the response of the RL agent to control the roll angle, yaw angle and the pitch angle can be improved during training set D. In training set D, the RL agent is trained to control the roll, yaw and pitch angle using the full range of all 3 control surfaces.

Figure 8-10 shows that the total reward value of the training episodes steadily close to zero. The value of Q_{max} also shows that its value slowly and steadily comes down to zero. This should suggest that the final policy network is ready to be used for 6 degree-of-freedom control.

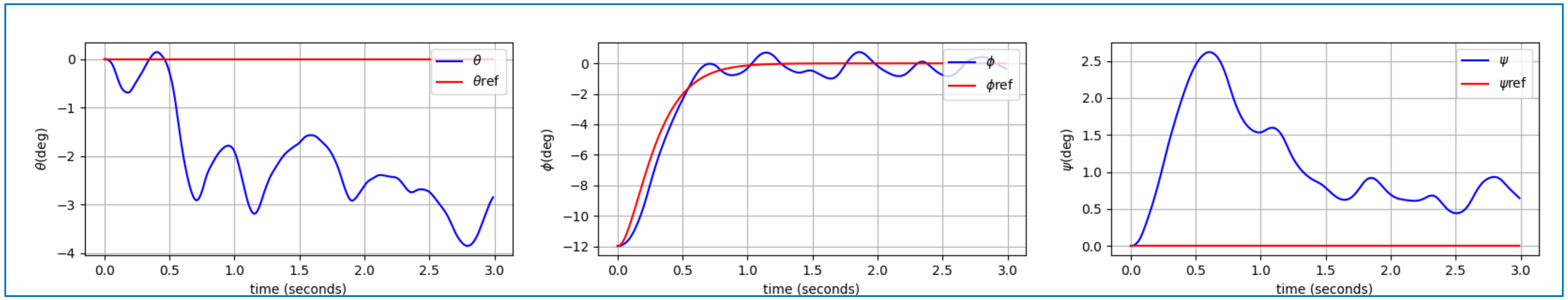


Figure 8-5 Final Episode of Training Set A

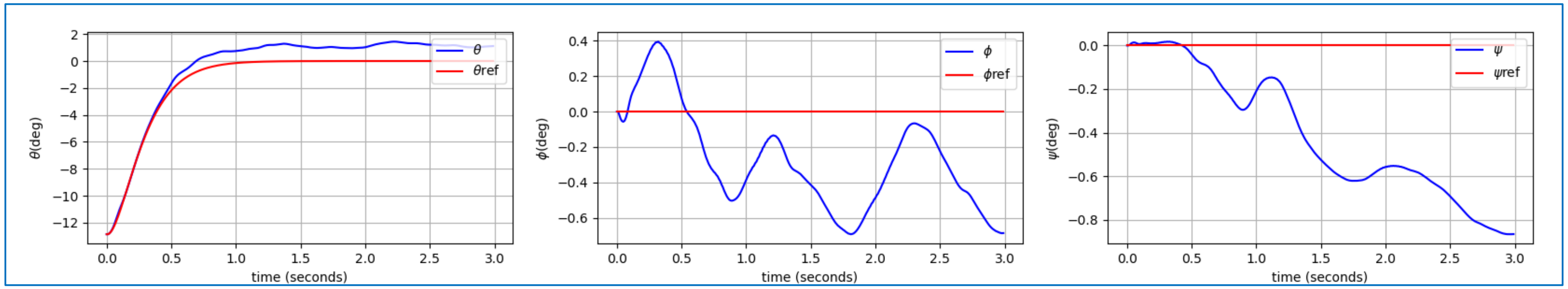


Figure 8-6 Final Episode of Training Set B

In Figure 8-8 shows a sampling episode towards the end of training set D. It shows that although the RL agent manage to control the pitch, roll and yaw angle to ‘generally’ follow the desired trajectory, it is not exactly following it.

Figure 8-11, Figure 8-12, and Figure 8-13 shows the result of the testing phase. This is the phase where the extracted final policy network from training set D is applied in a new program. This second program has no noise component in its actor policy.

There are 5 episodes for testing the final policy network. The first three tested the policy for single attitude control. The last tested for multiple attitude control. Figure 8-11 shows that the total reward value of the test episodes starts to decline from episode 4. In episode 5, the value plummets even more. Yet, the Q_{max} value shows a spike of value change for episode 5, even though it is relatively small.

Figure 8-12 shows how the RL agent controlled the roll angle (a), pitch angle (b) and yaw angle (c) separately, while also maintain the other respective angles. Figures (a) and (b) shows that the RL agent manage to achieve the desired roll angle and pitch angle while still maintaining the other angles to not diverge. However, in figure (c), the RL agent has a hard time following the desired yaw angle trajectory. Despite the fact that it still manages to control the other angles to not diverge.

Figure 8-13 shows the RL agent trying to control multiple attitude angles of the air vehicle. In both figures, where the RL agent tries to change the roll-yaw angle (d) and where the RL agent tries to change the roll-pitch-yaw angles (e), the yaw angle is not giving the best performance. This might be due to the policy resulted from training set C that is less than stellar. A solution is proposed to do the training set C again with adjusted hyper-parameter, in order to make the policy result for yaw control more suitable.

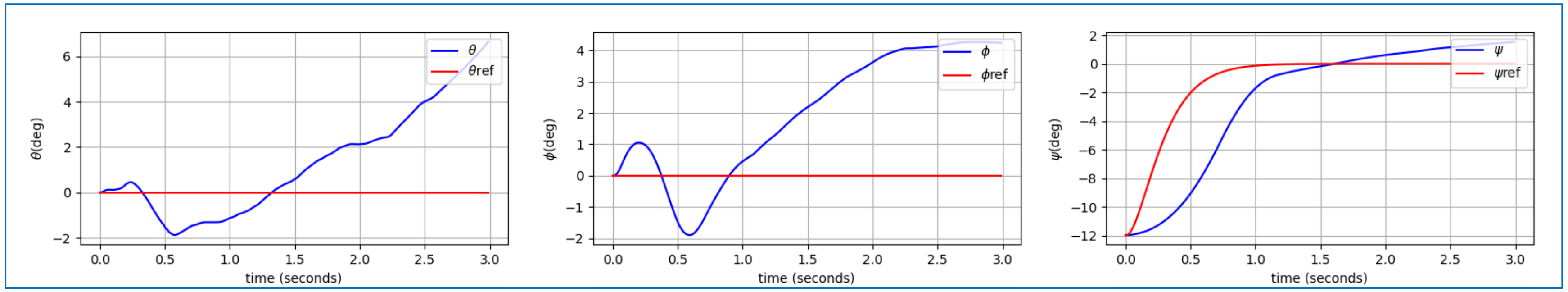


Figure 8-7 Final Episode of Training Set C

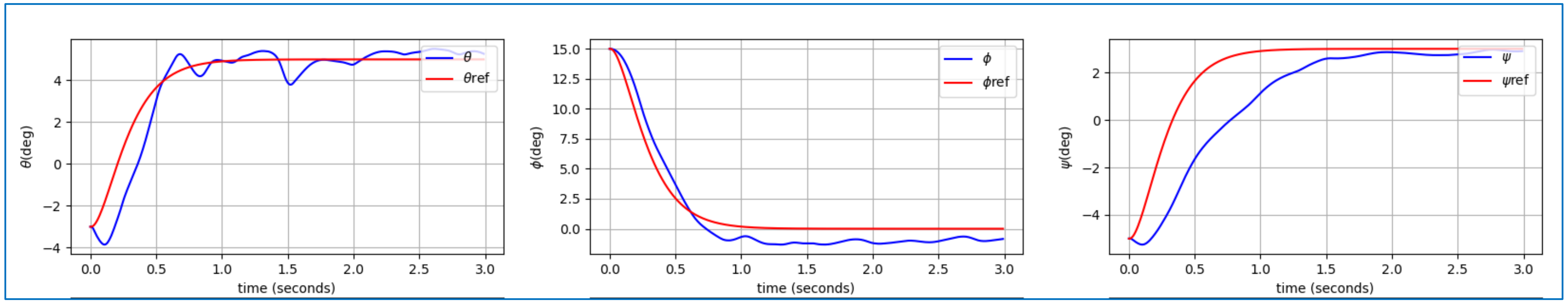


Figure 8-8 Final Episode of Training Set D

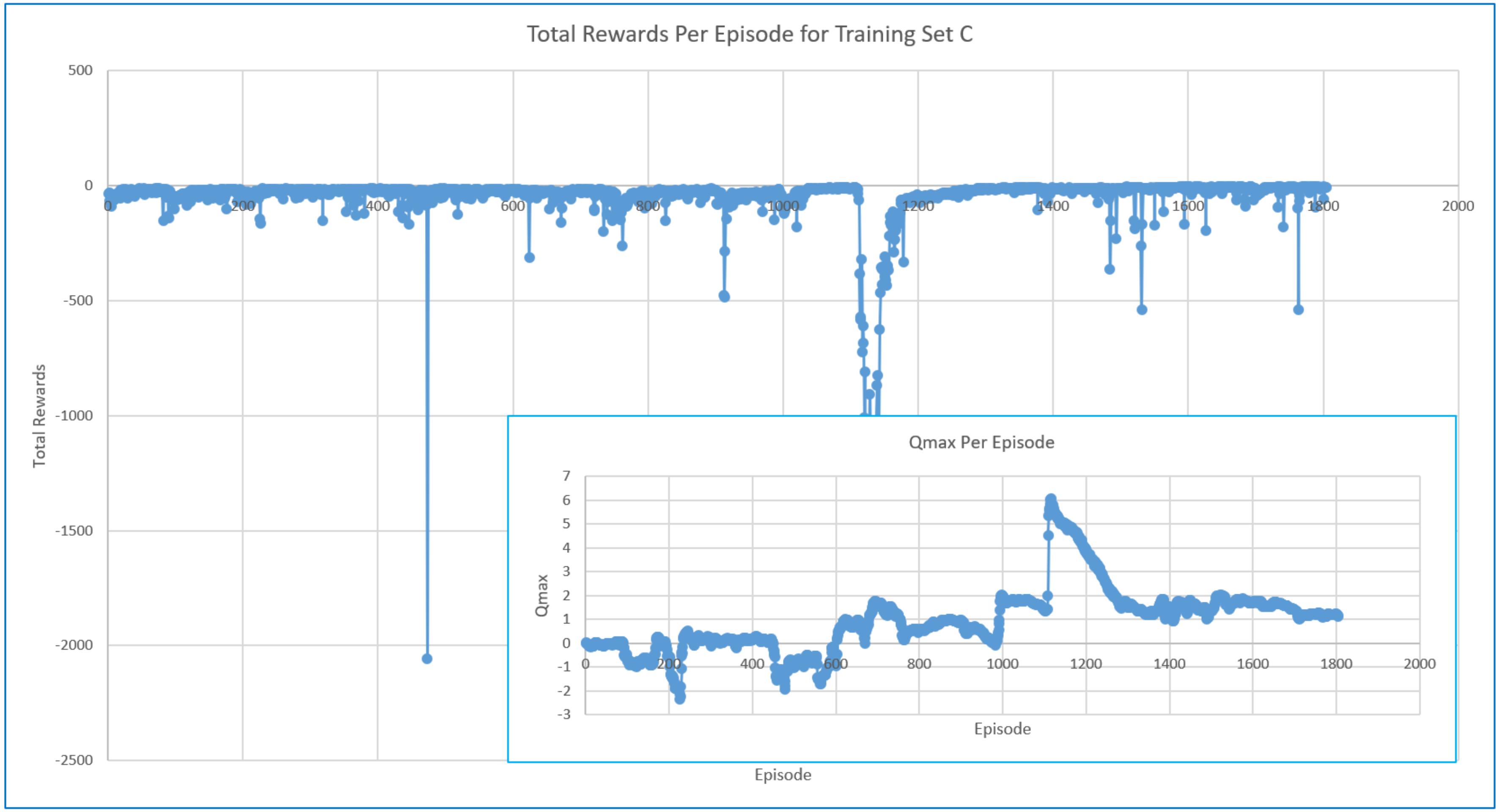


Figure 8-9 The Total Reward and Qmax Per Episode in Training Set C

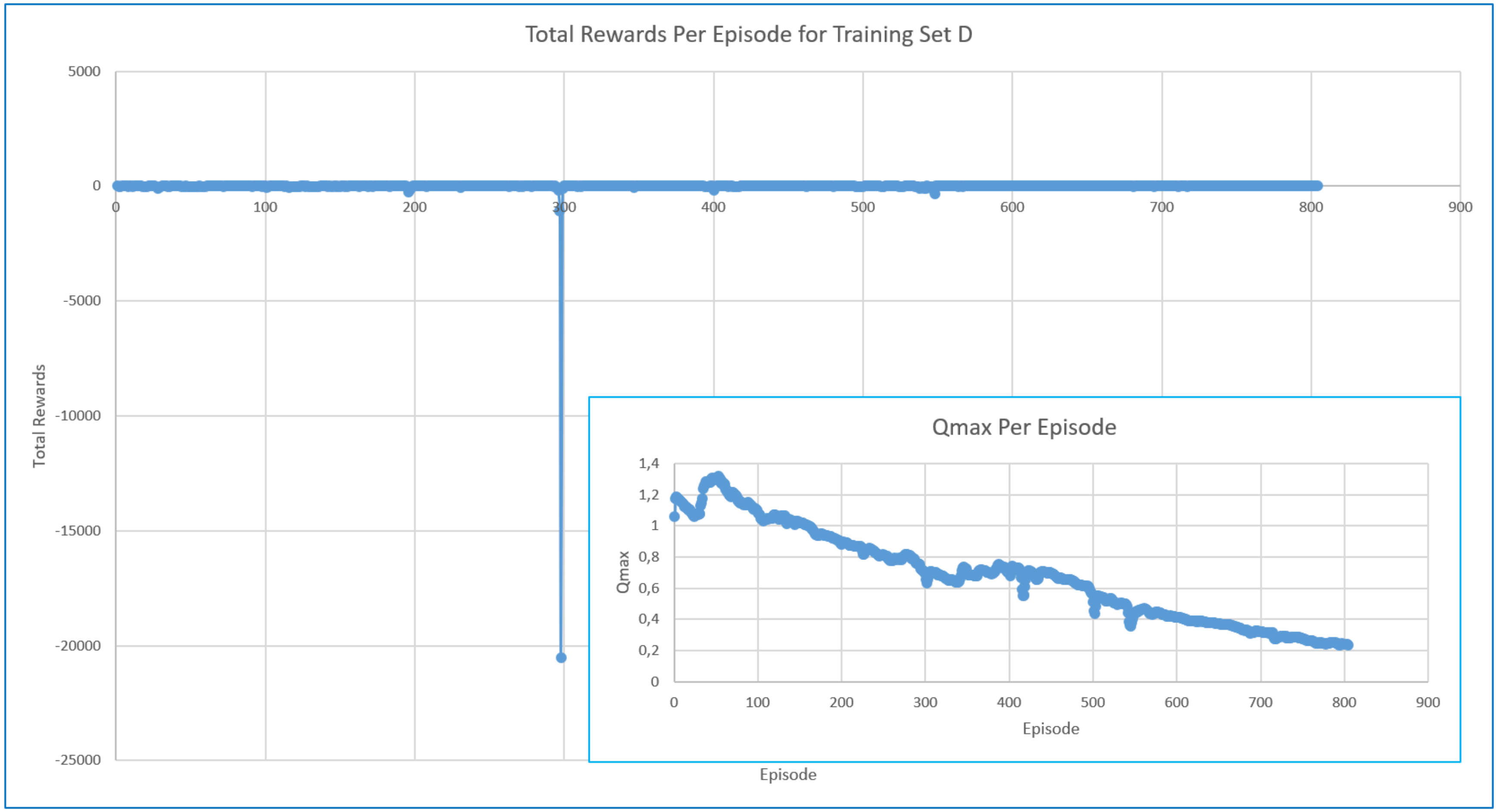


Figure 8-10 The Total Reward and Qmax Per Episode in Training Set D

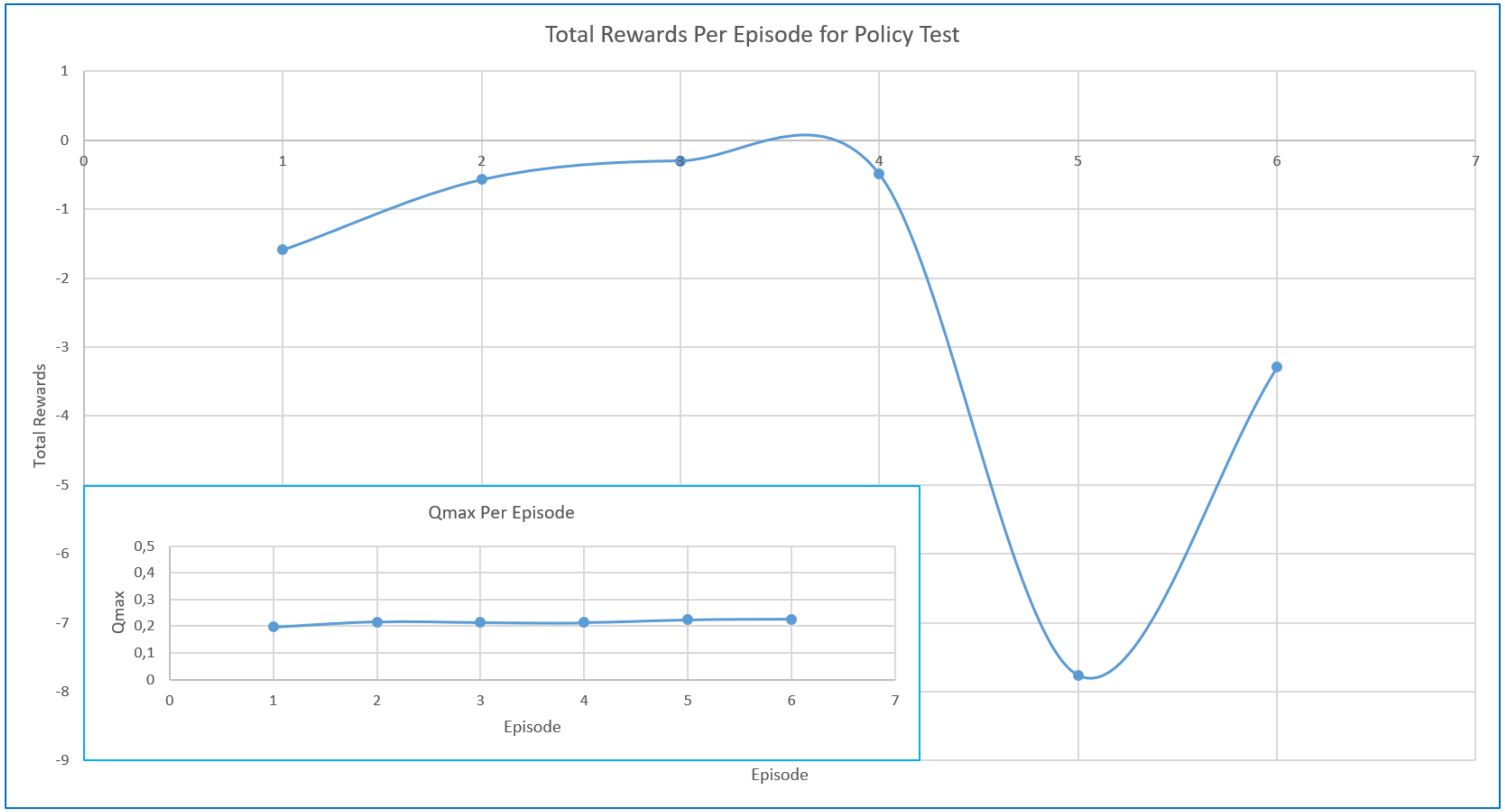
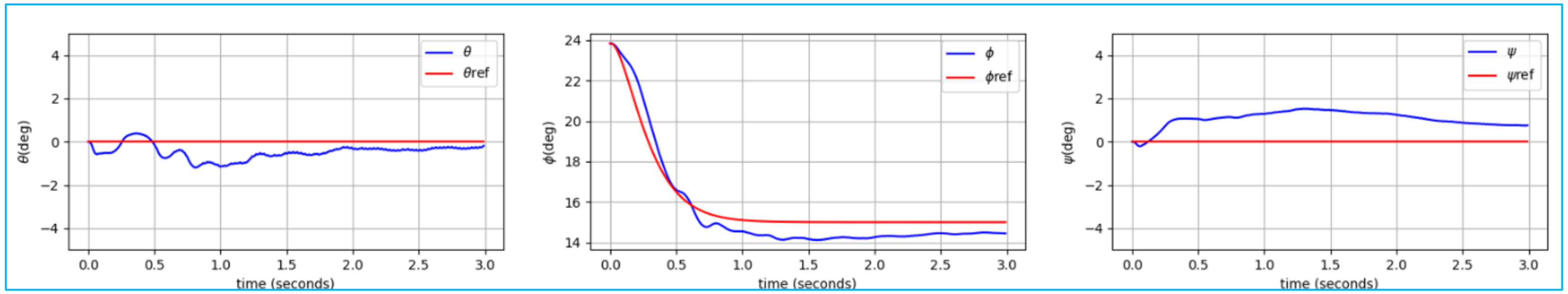
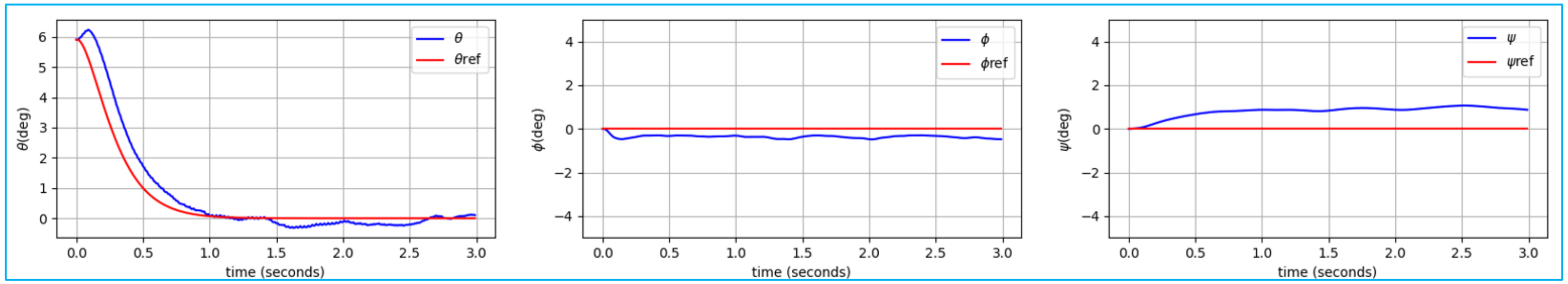


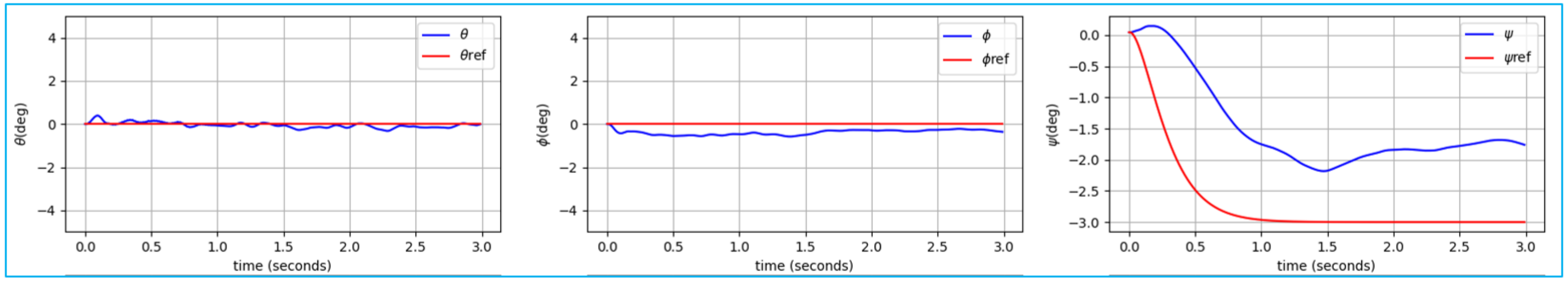
Figure 8-11 The Total Reward and Qmax Per Episode in Policy Test



(a)

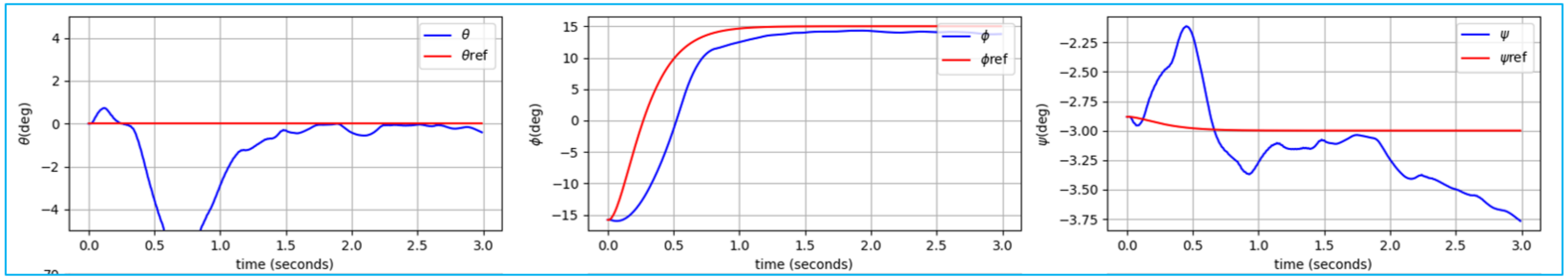


(b)

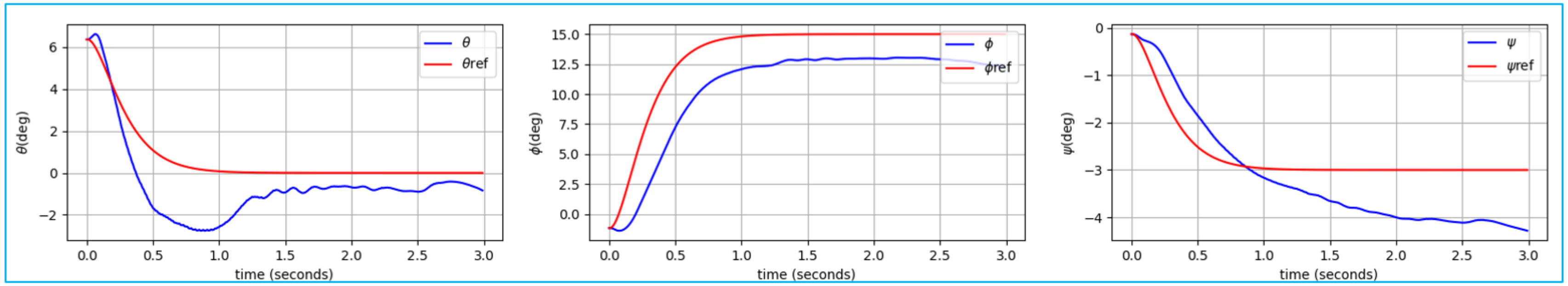


(c)

Figure 8-12 Test Episode For Single Attitude Control with The Final Policy Network



(d)



(e)

Figure 8-13 Test Episode For Multiple Attitude Control with The Final Policy Network

8.5 Conclusion

Based on the simulation done in this chapter, it shows that the RL agent can develop a policy network to control an air vehicle in 6 degree-of-freedom using all three control surfaces. By using all the control surface at each training process and also limited the range of the secondary control surface, the RL agent can converge it learning process to control an air vehicle with multiple control surfaces. Therefore, DDPG method can be used to obtain a policy network to control an air vehicle.

it shows that the policy network can be extracted and use as the initial policy network in another programme or another training programme. This advantage can be utilized to optimize the learning process. By dissecting the learning process into several training set, trouble in the learning process can be identified quickly and can be dealt with without having to redo the whole learning process. This saves time and effort.

9 CONCLUSION

This chapter consists of the research conclusion and the future works that can be seen on the horizon following this thesis.

9.1 Conclusion

Based on this research, it is possible to shape the learning process of the reinforcement learning agent using the deep deterministic policy gradient method. For this purpose, there are two aspects that is essential in shaping the learning process of a DDPG agent. They are:

- Guaranteeing the response in controlling the system, in this case an air vehicle.
- Developing the learning process of the agent by designing the training strategy in order to achieve the desired knowledge and skill

Based on the investigations in chapters 4 and 5, the way to guarantee the response of the RL agent in controlling a system is by:

- Determining the state definition, which consists of variables to observe and control. These variables would determine the number of network layers needed for the learning process. Excessive variable can lead to a growing number of layers and the time and computer memory to executed on.
- Defining the desired path to follow in order to control a variable/variables of the system. Here the variable that is being controlled is the pitch angle (longitudinal mode). It is easier for the system to learn to follow a desired path instead of finding its own path without knowing the limitations of the air vehicle itself.
- Determining the reward function that is representative so that the most desired performance equals the highest reward.

Learning from simulations in chapters 6, 7 and 8, it is clear that the learning strategy for controlling an air vehicle are comprised to these:

- For variating operating condition, the state definition shouldn’t be added or changed. The DDPG agent itself will implicitly consider it in its policy.
- To handle dual action with coupling nature, the strategy is to first learn the use of its most dominant action. The secondary action is confined to zero. Once the total reward is stabilized then the secondary action is released and variated alongside the dominant one.
- To handle multiple actions that have some level of influence in one another, the training strategy is to learn the effects of one action at the time. But re-learning the value of a pair of state-action1 with action2 cost time and may also cost convergence. Therefore, during training for one action, the other actions are not locked in zero value. Instead, they are given a small range to move, so that throughout the training process, the RL agent is accustomed to work with multiple actions.

Also, a conclusion of this research is that the final policy network post-training process can be extracted and used in another program. This is similar to removing a pilot from training in a simulator to a real air vehicle. The policy is considered suitable when it is tested and gives a value of Q_{max} between [-1,1].

The work of this thesis shows that shaping the learning process of a RL is essential in developing a 6-degree-of-freedom flight control system. This is an important stepping stone for incorporating RL in the flight control system development.

9.2 Future Works

For future works, this work should be developed further to also control the action profile given to system, as it currently still shows damaging chattering. A method can also be developed to determine the number of layers for the neural network so that it can eliminate the trial-and-error phase of determining the appropriate number of layers.

Another possible future work is to investigate and explore the possibility of applying a final policy network from a certain air vehicle to a different air vehicle within the same flight envelope. If this is possible, it can shed a lot of time to develop a control system for a newly develop air vehicle.

Further advanced work will be of investigating the use of DDPG method to develop fault tolerant flight control. An adjustment needs to be made to the DDPG method so as to allow a small room to update its policy following a fault occurrence and yet not so big a room that it would explore its policy during flight.

REFERENCES

1. Sutton RRS., Barto AGA., Book a B. Reinforcement Learning : An Introduction. 1998; Available at: DOI:10.1109/TNN.1998.712192

2. Lillicrap TP., Hunt JJ., Pritzel A., Heess N., Erez T., Tassa Y., et al. Continuous control with deep reinforcement learning. 2015; Available at: DOI:10.1561/2200000006

3. B CL., B AGL., B MJN. Implementation of Deep Deterministic Policy Gradients for Controlling Dynamic. Springer International Publishing; 2018. 276–287 p. Available at: DOI:10.1007/978-3-319-95972-6

4. Kumar A., Paul N., Omkar SN. Bipedal Walking Robot using Deep Deterministic Policy Gradient. 2018; Available at: DOI:arXiv:1807.05924v2

5. Kardell S., Kuosku M. Autonomous vehicle control via deep reinforcement learning Master's thesis in Systems, Control and Mechatronics. 2017;

6. Koch W., Mancuso R., West R., Bestavros A. Reinforcement Learning for UAV Attitude Control. : 1–13.

7. Busoniu L., Babuska R., De Schutter B., Ernst D. Dynamic programming and reinforcement learning in Large and Continuos Space. Reinforcement Learning and Dynamic Programming using Function Approximators. Automation. CRC Press Taylor & Francis Group; 2011. pp. 1–270. Available at: DOI:10.1515/9781400821334.toc

8. Si J., Barto A., Powell W., Wunsch D. Handbook of Learning and Approximate Dynamic Programming. 2004.

9. Nie C., Zhu M., Zheng Z., Wu Z. Model-free control for stratospheric airship based on reinforcement learning. Chinese Control Conference, CCC. 2016; 2016-Augus: 10702–10707. Available at: DOI:10.1109/ChiCC.2016.7555054

10. Daskiran O., Huff B., Dogan A. Low Speed Airship Control using Reinforcement Learning and Expert Demonstrations. AIAA Atmospheric

Flight Mechanics Conference. 2017; (January): 1–35. Available at: DOI:10.2514/6.2017-0934

11. Rottmann A., Plagemann C., Hilgers P., Burgard W. Autonomous blimp control using model-free reinforcement learning in a continuous state and action space. IEEE International Conference on Intelligent Robots and Systems. 2007; : 1895–1900. Available at: DOI:10.1109/IROS.2007.4399531

12. Junell J., Kampen E Van., Visser C De., Chu Q. Reinforcement Learning Applied to a Quadrotor Guidance Law in Autonomous Flight. 2015; (January): 1–13.

13. Han J-H., Lee D-K., Lee J-S., Chung S-J. Teaching micro air vehicles how to fly as we teach babies how to walk. Journal of Intelligent Material Systems and Structures. 2013; 24(8): 936–944. Available at: DOI:10.1177/1045389X13478270

14. Lee D., Choi M., Bang H. Model-free LQ Control control for unmanned helicopters using reinforcement learning. The 5th International Conference on Automation, Robotics and Applications. 2011; (6): 19–22. Available at: DOI:10.1109/ICARA.2011.6144849

15. Zhu Y., Mottaghi R., Kolve E., Lim JJ., Gupta A., Fei-Fei L., et al. Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning. 1609.05143V1. 2016; Available at: http://arxiv.org/abs/1609.05143

16. Wharington J. Autonomous Control of Soaring Aircraft by Reinforcement Learning. Doctoral Thesis. 1998; (November): 22–53.

17. Woodbury T., Dunn C., Valasek J. Autonomous Soaring Using Reinforcement Learning for Trajectory Generation. 52nd Aerospace Sciences Meeting. 2014; (January): 13–17. Available at: DOI:doi:10.2514/6.2014-0990

18. Lee DJLDJ., Bang HBH. Reinforcement learning based neuro-control

systems for an unmanned helicopter. Control Automation and Systems (ICCAS), 2010 International Conference on. 2010; (2): 2537–2540.

19. Kim JH., Lewis FL. Model-free H??? control design for unknown linear discrete-time systems via Q-learning with LMI. Automatica. 2010; 46(8): 1320–1326. Available at: DOI:10.1016/j.automatica.2010.05.002

20. Hoffmann G., Jang JS., Tomlin CJ. Multi-Agent X4-Flyer Testbed Control Design: Integral Sliding Mode vs. Reinforcement Learning. International Conference on Intelligent Robots and Systems. 2005; : 468–473.

21. Ng AY., Kim HJ., Jordan MI., Sastry S. Autonomous helicopter flight via Reinforcement Learning. 2003;

22. Ng AY., Coates A., Diel M., Ganapathi V., Schulte J., Tse B., et al. Autonomous inverted helicopter flight via reinforcement earning. Springer Tracts in Advanced Robotics. 2006; 21: 363–372. Available at: DOI:10.1007/11552246_35

23. Koch W. Flight Controller Synthesis Via Deep Reinforcement Learning. Boston University; 2019. Available at: http://arxiv.org/abs/1909.06493

24. Morrison S., Fisher A., Zambetta F. Towards Intelligent Aircraft Through Deep Reinforcement Learning. 10th International Micro-Air Vehicles Conference. Melbourne; 2018. pp. 1–8. Available at: http://www.imavs.org/papers/2018/IMAV_2018_paper_52.pdf

25. Kroezen D. Online Reinforcement Learning for Flight Control. TU Delft. TU Delft; 2019.

26. Zhou Y., Kampen E Van., Chu QP. Incremental Approximate Dynamic Programming for Nonlinear Flight Control Design. Proceedings of the 3rd CEAS EuroGNC. Toulouse; 2015. pp. 1–18.

27. Bhatnagar S., Panigrahi JR. Actor-critic algorithms for hierarchical Markov decision processes. Automatica. 2006; 42(4): 637–644. Available at: DOI:10.1016/j.automatica.2005.12.010

28. Al-Tamimi A., Lewis FL., Abu-Khalaf M. Model-free Q-learning designs for linear discrete-time zero-sum games with application to H-infinity control. Automatica. 2007; 43(3): 473–481. Available at: DOI:10.1016/j.automatica.2006.09.019

29. Seijen H Van., Hasselt H Van., Whiteson S., Wiering M. A Theoretical and Empirical Analysis of Expected Sarsa.

30. Feldbrugge RL. Using Reinforcement Learning to Make Optimal Use of Available Power and Improving Overall Speed of a Solar-Powered Boat. University of Groningen; 2010.

31. Silver D., Lever G., Heess N., Degris T., Wierstra D., Riedmiller M. Deterministic Policy Gradient Algorithms. Proc. of the 31st International Conference on Machine Learning. 2014; : 387–395. Available at: http://jmlr.org/proceedings/papers/v32/silver14.html

32. Ko J., Klein DJ., Fox D., Haehnel D. Gaussian Processes and Reinforcement Learning for Identification and Control of an Autonomous Blimp. 2007; (April): 10–14.

33. Faust A., Palunko I., Cruz P., Fierro R., Tapia L. Automated aerial suspended cargo delivery through reinforcement learning. Artificial Intelligence. 2017; 247: 381–398. Available at: DOI:10.1016/j.artint.2014.11.009

34. Konda VR., Tsitsiklis JN. Actor-Critic Algorithms. Proceedings of Advances in Neural Information Processing Systems. 2000; 12: 1008–1014.

35. Crites RH., Barto AG. An Actor/Critic Algorithm That is Equivalent to Q-learning. Proceedings of the 7th International Conference on Neural Information Processing Systems. 1994; (1983): 401–408. Available at: http://dl.acm.org/citation.cfm?id=2998687.2998737

36. Ng AY., Jordan M. PEGASUS: A Policy Search Method for Large MDPs and POMDPs. Uncertainty in Artificial Intelligence Proceedings. 2000. pp. 406–415.

37. Mnih V., Kavukcuoglu K., Silver D., Graves A., Antonoglou I., Wierstra D., et al. Playing Atari with Deep Reinforcement Learning. 2013; : 1–9. Available at: DOI:10.1038/nature14236

38. Tuyen LP., Chung TC. Controlling bicycle using deep deterministic policy gradient algorithm. 2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence, URAI 2017. 2017; : 413–417. Available at: DOI:10.1109/URAI.2017.7992765

39. Decayeux T. Autonomous Docking of a Quadrotor on a Moving Platform. 2016.

40. Bansal S., Akametalu AK., Jiang FJ., Laine F., Tomlin CJ. Learning Quadrotor Dynamics Using Neural Network for Flight Control. 55th IEEE Conference on Decision and Control. 2016; (0931843). Available at: DOI:10.1109/CDC.2016.7798978

41. Hwangbo J., Sa I., Siegwart R., Hutter M. Control of a Quadrotor with Reinforcement Learning. 2017; 2(4): 2096–2103. Available at: DOI:10.1109/LRA.2017.2720851

42. Xu D., Hui Z., Liu Y., Chen G. Morphing control of a new bionic morphing UAV with deep reinforcement learning. Aerospace Science and Technology. Elsevier Masson SAS; 2019; 92: 232–243. Available at: DOI:10.1016/j.ast.2019.05.058

43. Chang-Hun L. Lecture Notes 0n Nonlinear Controls for Aerospace System.

44. Sanchez E., Becerra H., Velez CM. Combining fuzzy and PID control for an unmanned helicopter. NAFIPS 2005 - 2005 Annual Meeting of the North American Fuzzy Information Processing Society. 2005; : 235–240. Available at: DOI:10.1109/NAFIPS.2005.1548540

45. Salih AL., Moghavvemi M., Mohamed HAF., Gaeid KS. Flight PID controller design for a UAV quadrotor. Scientific Research and Essays. 2010; 5(23): 3660–3667. Available at: http://www.academicjournals.org/SRE%5Cnhttp://www.researchgate.net/

publication/230633819_Flight_PID_Controller_Design_for_a_UAV_Quadrotor/file/d912f511361f422fdd.pdf

46. Bouabdallah S., Siegwart R. Backstepping and Sliding-mode Techniques Applied to an Indoor Micro Quadrotor. Proceedings - IEEE International Conference on Robotics and Automation. 2005; 2005(April): 2247–2252. Available at: DOI:10.1109/ROBOT.2005.1570447

47. Chand AN., Kawanishi M., Narikiyo T. Non-Linear Model-free Control of Flapping Wing Flying Robot using iPID. IEEE International Conference on Robotics and Automation. 2016; : 2930–2937. Available at: DOI:10.1109/ICRA.2016.7487458

48. Younes Y Al., Drak A., Noura H., Rabhi A., Hajjaji A El. Robust Model-Free Control Applied to a Quadrotor UAV. Journal of Intelligent and Robotic Systems: Theory and Applications. 2016; : 1–16. Available at: DOI:10.1007/s10846-016-0351-2

49. Yusuf S., Lone M., Cooke A., Lawson N. Regressor time-shifting to identify longitudinal stability and control derivatives of the Jetstream 3102. Aerospace Science and Technology. Elsevier Masson SAS; 2017; 69: 218–225. Available at: DOI:10.1016/j.ast.2017.06.003

50. Kardell S., Kuosku M. Autonomous vehicle control via deep reinforcement learning. 2017; : 73.

51. Xu J., Hou Z., Wang W., Xu B., Zhang K., Chen K. Feedback Deep Deterministic Policy Gradient with Fuzzy Reward for Robotic Multiple Peg-in-hole Assembly Tasks. IEEE Transactions on Industrial Informatics. IEEE; 2018; PP(c): 1. Available at: DOI:10.1109/TII.2018.2868859

52. Hong J-H. Lecture Notes on Model Building for UAS.

Website:

53. https://emerj.com/ai-glossary-terms/what-is-machine-learning/ visited on 13 March 2019