A New Control Technology for the Development of an Air-to-Air Refueling System

By

Jonathan Kaban

A thesis submitted to Lakehead University in partial fulfilment of the requirements for the Degree of Master of Science in Mechanical Engineering

Lakehead University
Department of Mechanical Engineering
2023

Abstract

Air to Air Refueling (AAR) was first performed over 100 years ago and until now it has almost exclusively been used in military applications. This is due to the prohibitive cost of maintaining a tanker fleet to enable refueling operations as well as the amount of training required by both tanker and receiver pilots to mitigate the risk involved with operating aircraft in close proximity. There are two methods of performing AAR operations: probe-drogue and flying boom. This work investigates the feasibility of converting a civilian tanker into a probe-drogue tanker for use in civilian applications. The aircraft chosen for this work is FuelBoss AT-802 as a fuel hauler. The first objective of this thesis is to model the AT-802 and explore its potential role as an AAR tanker.

The second objective of this thesis is to address the issue of risk in AAR by modeling a hose-drogue and proposing a new control technology to stabilize a drogue in flight. As no drogue system is available for experimental testing, a flexible smart structure lab workstation will be used to investigate control strategies for vibration suppression under variable system dynamics. A Deep Deterministic Policy Gradient (DDPG) algorithm is proposed in conjuncture with Domain Randomization for reinforcement training of the controller. The effectiveness of the proposed control technique and learning algorithm is verified by experimental tests, with comparison to other related control methods such as the built-in PD controller and an intelligent NF controller. Dynamic conditions of the flexible structure are simulated by placing magnetic mass blocks at different positions on the beam. Experimental results show that the proposed DDPG controller outperforms other related control methods in terms of settling time, overshoot, and mean error, without sacrificing robustness and stability. It can learn a decision-making policy in environments with large action spaces such as in vibration suppression and has potential to used for hose-drogue system control.

Acknowledgement

My deepest appreciation goes to my thesis supervisor, Dr. Wilson Wang, for his mentorship and guidance throughout this journey. His expertise, dedication, and belief in my abilities were the driving forces behind the success of this thesis. I would also like to extend my sincere thanks to the members of the thesis review committee, Dr. Qiang Wei, Dr. Hao Bai, and Dr. Ali Tarokh.

I would like to acknowledge the support provided by Alan Cheeseman, Dave Gaudino, and Dan Murray from Wilderness North and FuelBoss. Their belief in the significance of this project contributed greatly to this work.

I want to express my gratitude to Meagen Lepage for her patience, understanding, and unwavering belief in me. Finally, and most importantly to my parents, Lynda and Michael Kaban, your sacrifices and constant support have allowed me to reach the places I have today and will tomorrow.

Table of Contents

List of Figures

List of Tables

List of Symbols

Symbol	Definition
π	Reinforcement Learning Decision Making Policy
S^{π}	Policy State Space
s	State space variables
A^{π}	Policy Action Space
a, u	Action taken
V^{π}	Policy Value Function
γ	Discount Factor for future rewards
r	Environment Reward
F_W	Tanker Frame of Reference
w_1, w_2, w_3	Unit vectors of tanker frame
ω_W, α_W	Angular velocity and acceleration of hose-link angles relative to the tanker frame, respectively
$-l_K$	Length of hose link
$p_K{}'$	Position of lumped mass K relative to the tanker frame
$p_K, \dot{p}_K, \ddot{p}_K$	Position, velocity, and acceleration of lumped mass K relative to the previous link, respectively
θ_{K1}, θ_{K2}	Angular position of lumped mass K about the pitch and yaw axis respectively
$\dot{\theta}_K, \ddot{\theta}_K$	First and second derivatives of hose link angle for link of lumped mass K, respectively
v_K, a_K	Velocity and Acceleration of lumped mass K, respectively
g	Gravitational Constant
Q_K	External Force Vector acting on hose-drogue

T_K	Link tension in the link for lumped mass K
n_{Ki}	Unit Vector of position of lumped mass K
$[\Gamma]$, $\{q\}$	Matrix form of link tension equation constants
$\{T\}$	Link Tension vector
μ_K	Reciprocal of the mass of node k
D_K	Aero force on k
D_d	Drogue Drag
m_K	Mass of any link K
m_{drogue}	Mass of the drogue
R	Reward Function
Q^*	Optimum Action-Value Function
u	Actor Network
Q	Critic Network
ψ^u, ψ^Q	Parameters (Weights) of the actor and critic networks respectively
u'	Target Actor Network
Q'	Target Critic Network
$\mathbb{E}$	Expected value of a finite number of random outcomes
J	Starting distribution of reinforcement learning NN
$\nabla_{\psi^u}, \nabla_{\psi^Q}$	Gradients of the weights for the actor and critic NNs respectively
∇_a	Gradient of actor outputs
L	Loss function
τ	Target network learning rate
$\varkappa_t$	Experience replay sample at time step, t

$\hat{m}_t, \hat{v}_t$	First and second moment vectors of the loss gradient at time step t, t
$\hat{m}_t', \hat{v}_t'$	Corrected first and second moment vectors of the loss gradient at time step t, t
ϱ	ADAM Optimization algorithm learning rate
β_1, β_2	ADAM Optimization Correction hyperparameters, chosen through trial and error
ϕ^u, ϕ^Q	Actor and critic learning rates, respectively
$\mathbb{B}$	Experience replay buffer
N_b	Number of samples of previous experiences used for each training step
$\mathcal{N}$	Action Noise Component
μ, σ	Action noise hyperparameters, mean and standard deviation, respectively
b	Replay buffer size
N_E	Number of training episodes
ε_k	State observation noise scaling constant, $k = 1,2, \dots, n$ where n is the number of inputs into the actor network
l_b	Smart structure flexible beam length
l_r	Smart structure rigid bar length
m_m	Mass of motor and gear reducer mounted atop flexible beam and connected to the rigid bar
m_r	Mass of the rigid bar
x_b	Deflection of the flexible beam
θ	Angular position of rigid bar
$\dot{s}_t$	Rate of change of state space variables
$\dot{x}_b$	Velocity of deflection of the flexible beam
$\dot{\theta}$	Angular velocity of the rigid bar

$T_{X,Y}^{T}$	Translation matrix between any two points (X and Y) on the smart structure
$T_{X,Y}^{R}$	Rotation matrix between any two points (X and Y) on the smart structure
$T_{0,3}$	Transformation matrix from the base of the smart structure to the free end of the rigid bar
x^r, y^r, z^r	Three dimensional cartesian coordinates of the free end of the rigid bar
$\dot{x}^r, \dot{y}^r, \dot{z}^r$	Rate of change of cartesian coordinates of the free end of the rigid bar
La	Lagrangian from the Euler-Lagrange Method
K	Total kinetic energy of system
V	Total potential energy of system
K_s	Spring constant of the flexible beam, determined experimentally
ω_n	Natural frequency of the flexible beam
J_r	Mass moment of inertial of the rigid bar
Q_1, Q_2	Generalized external forces for Lagrangian
τ_m	Torque produced by motor
η_g	Gearbox efficiency
K_g	Gearbox ratio
η_m	Motor efficiency
K_t	Motor torque constant
V_m	Motor applied voltage
R_m	Motor armature resistance
K_m	Back-emf constant
$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$	State space matrices
$k_{x_b}^{p}, k_{\theta}^{p}$	Proportional gains for x_b and θ, respectively

$k_{x_b}^d, k_\theta^d$	Derivative gains for $\dot{x}_b$ and $\dot{\theta}$, respectively
$\boldsymbol{u}$	General control output
$g(s)$	General membership function for NF controller
N_s	Number of state space variables
a_j, c_j	Non-linear parameters of NF MFs
p_{h_n}	Linear parameters of NF MFs
W_h	Firing strength of fuzzy rule h
O_j^i	O_j^i is the firing strength of the node O, at position j, in layer i
$U(x)$	Rectified Linear Units (ReLU) function
N_i	Number of nodes in a given layer i
r	Reward for a given step
$H(O_1^4)$	hyperbolic tangent function output for the output node O_1^4 of the actor
ζ	State-space model step size
E	Current episode during training

Chapter 1 - Introduction and Literature Review

1.1 Problem Statement

Canada as a country is in a unique position in terms of air power and air force. With a large amount of territory that is uninhabited and most of the population living along the southern section of the country, defence and air coverage of northern areas may be difficult when deploying from major air bases. For these reasons Canada can benefit greatly from air to air refueling (AAR) of its own air force and other civilian aircraft such as rescue helicopters. However, currently the Royal Canadian Air Force relies on tankers from the U.S. Air Force for refueling needs [1, 2]. Consequently, a tanker that suits the unique AAR needs of Canada would and will be greatly important in the future. This work will focus on the AT-802-FuelBoss tanker which, is an aircraft that is outfitted to deliver fuel to remote communities in Canada and in the world.

Figure 1.1 shows a FuelBoss, which is a modified AT-802. It is manufactured by Air Tractor in Onley, Texas and modified by FuelBoss to carry fuel, becoming the AT-802-FuelBoss Boasting a deliverable load of 8000 lbs from a single engine aircraft, the FuelBoss can carry 4000 litres of fuel while remaining versatile and being able to respond swiftly [3]. This aircraft was developed off of the AT-802-FireBoss, firefighting aircraft, that can carry a large capacity for its size, while being able to take off in a short distance and respond much more quickly than a larger tanker [4]. This made the AT-802 well suited to serve as an aerial refueling tanker.

Figure 1.1: AT-802 FuelBoss [3]

This work focuses on the development of AAR system to refuel helicopters using AT-802-FuelBoss as a tanker.

1.2 Overview on Aerial Refueling

Aerial refueling has been performed for almost 100 years, to refuel a receiver aircraft from a tanker aircraft while flying. The first test was performed between two bi-planes on June 27th, 1923, with 75 gallons of fuel delivered [5]. Since then, many innovations have been made to improve the capabilities of air forces in extending range and operational area of missions, as well as increasing the flexibility and versatility of strategy in aerial operations [6]. Figure 1.2 shows an early refueling test between two biplanes using the looped hose method.

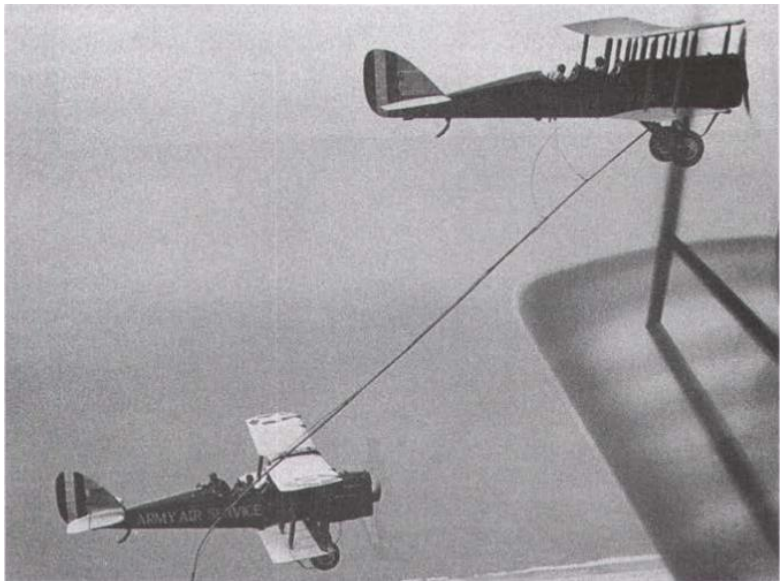


Figure 1.2: Refueling test: de Havilland DH-4B hanging hose to be grabbed by second DH-B4 using looped hose method [1]

There are two common types of AAR; the “flying boom” and the probe-and-drogue methods. The flying boom is mainly used by large tankers to deliver fuel to large receivers with a high fuel flow rate [7]. The flying boom method relies on an operator within the refueling tanker to manipulate a boom into a receptacle on the receiver and deliver fuel, shown in Figure 1.3. The operator ‘flies’ the boom into a refueling port on the receiver aircraft.

Figure 1.3: KC-10 refueling a B-52 using the 'flying boom' method [8]

Probe and drogue is typically applied for refueling small receiver aircraft such as fighters and helicopters [8], [9]. Probe and drogue refueling is performed by the tanker aircraft crew extending a hose with a drogue at the end. As illustrated in Figure 1.4, the drogue creates a drag as its shape is similar to a parachute. The drag force generates tension in the hose and ensures the drogue is aerodynamically stable enough to not be negatively effected by turbulence [10]. The drogue acts as a moving target, for the receiving aircrafts pilot to ‘fly into’ such that the probe on the receiver couples within the center of the drogue [8]. The drag on the drogue is enough to allow the receiver pilot to overcome the connection force of the drogues coupling. The FuelBoss is a small tanker and will mainly be refueling small planes and helicopters and therefore this work focuses on probe-and-drogue refueling technology.

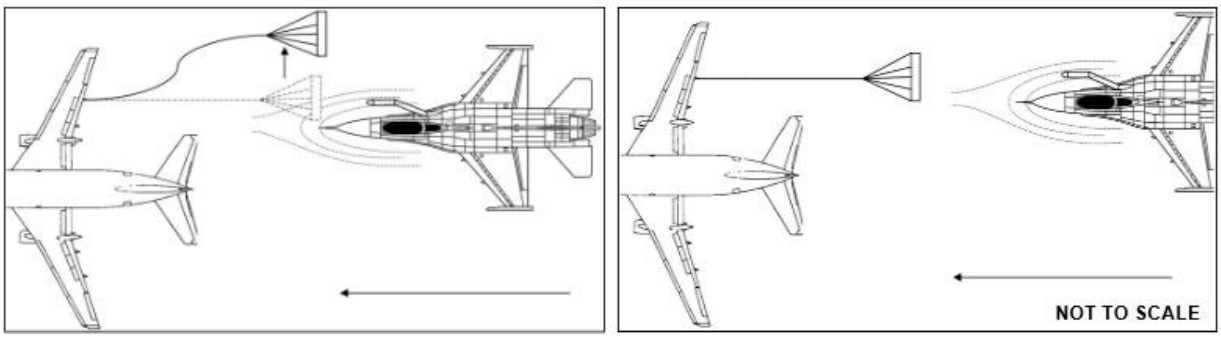
Figure 1.4: KC-130 refueling multiple F-18s using the probe and drogue method [5]

1.3 Challenges in Aerial Refueling

AAR can benefit both military and commercial sectors. When a hose and drogue are extended behind the tanker, they experience aerodynamic disturbances that are affected by a number of factors such as air speed and atmospheric conditions. The hose and drogue possess little to no ability to respond to these external disturbances, which leads to many challenges when docking with the receiver aircraft [11]. For example, the hose and drogue can be disturbed by the same factors that may disturb a conventional aircraft such as constant wind, wind gusts, and turbulence. The drag acting on the drogue may be able to dampen these effects in light conditions, but larger effects can lead to instability. Disturbances like the tanker's wake and the bow-wave effect are unique to aerial refueling. As the tanker flies through the air, it disturbs the air behind it; this can be seen in Figure 1.5, which is taken from a study simulating the effects of the tanker's wake in helicopter aerial refueling. The topology in the figure shows aerodynamic disturbances from the fuselage, propellers, wingtip, flaps, and refueling pod, all of which effect the flight of the drogue behind the tanker. [12]

Figure 1.5: U.S. Coast Guard photo of C-130J with wake turbulence visible in clouds [12]

In addition, the bow-wave effect can perturb the hose-drogue as the receiver flies close enough to the drogue in order to couple. The bow-wave effect is caused by the aerodynamic disturbance in-front of a fighter as it flies. This disturbance pushes away the drogue as the receiver approaches and creates a difficult challenge for refueling pilots. [13-16]. An example of this effect on a drogue's flight can be seen in Figure 1.6.



(a) (b)

Figure 1.6: Illustration of the bow-wave effect in AAR [16]: (a) drogue is displaced by approaching receiver due to the bow-wave effect. (b) receiver reverses approach and drogue returns to natural position.

Another challenge in AAR is referred to as the hose whipping phenomenon, which occurs when slack is created in the refueling hose as the receiver aircraft attempts to couple with the drogue [17]. Unfortunately, there is very limited research in literature on the dynamics of the hose and drogue during refuelling, as well as on the study of the response and control methods when hose whipping is occurring. [18, 19]. Advanced research in drogue stabilization and docking control could be very beneficial as it could reduce requirement from pilots and allow for a less expensive transition to the commercial flight space.

1.4 Literature Review for Air to Air Refueling

1.4.1 Modeling and Dynamic Analysis of Hose-Drogue Systems

In existing literature several aspects of probe and drogue refuelling have been studied This section will investigate the research and development done in this area. The first dynamic model of the hose and drogue part of the refueling system in was made by Eichler in 1978. It studied the effects of variable hose length, drogue drag, and drogue weight using a flight test of a 50 foot hose; some valuable findings and suggestions were made [20]. More modeling studies of

AAR operations were made in the last two decades. For example, J. Yan et al. proposed a finite element approach to hose modeling in 2004 with the intention of studying control and sensor requirements for a future autonomous refueling system [21]. This approach simplified the refueling hose as a single cantilever beam and found that increasing the damping effects on the hose could, lead to faster settling of the drogue. Zhu et al. suggested a finite element method to model the hose-drogue system with elastodynamic principals [22]. It was found that opting to use a curved beam rather than a straight cantilever, resonance in the hose was not caused by disturbance at the tow point, but instead by the tanker wake vortex. The hose-drogue will orbit the wake vortex, as the hose was lengthened this disturbance became less severe [19]. Ro and Kamman modeled the hose and drogue as a series of rigid links, attached by ball and socket joints [23]. This method accounted for tanker wake, steady wind, and atmospheric turbulence. Ro, et al. conducted research to study the dynamics of the hose-drogue during coupling with the receiver [24]. It accounted for the bow-wave effect generated by the approaching receiver and explored the effects of slack being created in the hose during coupling. This research also showed that the drogue is displaced more by an approach from the side than one from straight on. It was also observed that hose-whipping occurred during coupling when slack was formed ion the hose and no reel take-up control was applied. Paniagua et al. presented a new model of the hose and drogue, including often neglected effects such as the hoses internal bending influence, downwash angle induced by the tanker, and the phase lag between the hose oscillation and the aerodynamic forces [25]; this could be the most comprehensive dynamic model of a hose-drogue to date.

1.4.2 Control and Stabilization of Hose-Drogue Models and Systems

Ro et al. proposed a concept of drogue stabilization using PID (proportional integral derivative) control in [26]. Its objective was to address erratic drogue behaviour caused by the fore-body or bow-wave effect of an approaching receiver aircraft. A linearized model of the hose-drogue was used in the PID control to actively stabilize the drogues flight. A post-contact tension control system for the hose is also applied in reducing the hose whipping phenomenon. Kuk et al. took their previous work [23, 24, 26] into the physical domain by building a one-third sized prototype of a drogue with control surfaces for use in wind tunnel testing. A 4-DOF, single link model was used in these tests. A linearized state space model of the drogue with the control

surfaces was applied in the feedback control. Dynamic testing results showed that an actively stabilized drogue could reduce drogue motion by up to 90% [27–29]. Figure 1.7 shows some photos of the prototype and its control surface arrangement.

(a) (b)

Figure 1.7: Prototype drogue from [29]: (a) side view, (b) front view

Liu et al. presented a modeling approach based on a boundary control method [30]. It extended Hamilton's principle to model the flexible hose as a distributed parameter system represented with partial differential equations. This model can accommodate varying speed, hose length, and input constraints, while backstepping control was used to reduce the hoses vibration. Simulation tests showed that this method could suppress hose vibration if control parameters could be selected properly. Yuan et al. investigated the controllability of a drogue in a hose-drogue refueling system in [31] and found that both the lateral and vertical displacement of the drogue can be reduced significantly when using control surfaces and a conventional PID control scheme. Fogeda et al. studied the dynamic response of the drogue under stabilization with control surfaces [32]; several types of excitations were applied at the pod and to the environment, and tested under different flight speeds. Test results showed that the controller could provide expected performance, especially at higher speeds.

There are also several researchers aiming to solve the hose-whipping phenomenon using reel take up systems driven by electric motors. For example, Cheng et al. developed a modified tensator system with a permanent magnet synchronous motor (PMSM) that can limit the rate of

take up acceleration and reduce vibration and remove the hose-whipping in testing [33]. Su and Wang suggested a non-singular fast terminal sliding mode control method in [34] using a PMSM as the reel-take up, rather than relying on the tensator. Satisfactory results were obtained from flight simulation data.

1.5 Intelligent Control in Hose-Drogue Refueling Systems

1.5.1 Overview on Reinforcement Learning

Traditional industrial controllers such as PD (proportional derivative), PI (proportional integral) or PID are mainly used to manipulate linear systems or some non-linear systems via proper linearizing strategies. They are relatively simple and easy to implement but they may not provide satisfactory performance for non-linear systems, especially under time-varying operating conditions [35]. A hose-drogue is a nonlinear system and subject to variable operating conditions in flight. It is difficult for these classical controllers to provide satisfactory performance as it is difficult to tune the related gains to accommodate for time-varying flight conditions. For this reason, advanced soft computing techniques such as machine learning, neural networks, fuzzy logic and their integrated neuro-fuzzy (NF) techniques may be used in hose-drogue system control [36-38].

Machine learning is a process to train and optimize a reasoning system using training data. For example, a typical supervised backpropagation training routine of a neural network involves first a forward propagation of the network from the input layer to the output layer. Then in the backward pass, by comparing the theoretical outputs with the desired outputs, neural network parameters such as link weights and node biases are optimized by using an appropriate training algorithm to minimize the mapping errors [39]. Neural networks mimic biological brain reasoning through parallel-distributed processing neurons for decision-making [40]. The main advantage of neural networks is that they can learn and be trained in order to achieve a desired input-output mapping; however, they have the drawbacks of using a black box decision-making and that convergence may not be guaranteed [41].

Fuzzy logic extends classic logic only allows a conclusion that is true or false, to decision making in situations where there is imprecise or vague information. For example, a person deciding what temperature they feels may be based on many imprecise factors, such as room

temperature, humidity, amount of clothing worn, and heart rate. Despite being unable to accurately measure these factors a person can determine a general idea of what they are experiencing. Several functions can be used to model each of these factors and logical operators can be used to determine a value of how true a response may be. Fuzzy reasoning allows for a conclusion that is somewhere between true and false, which can provide a better representation of a similar decision made biologically [37, 38]. The main problem with fuzzy logic is that it cannot modify its functions in response to feedback in order to improve responses.

A more advanced approach is the use of NF technique, which is an integration of the merits of neural networks and fuzzy logic and could be more effective in reasoning. This allows for fuzzy reasoning to be applied to inputs, while hidden layers of the network can be trained to provide a desired output that correlates to a given fuzzy input [41]. In the forward pass, inputs are fed through fuzzy rules to determine the degree of membership for each fuzzy rule set, which is then passed through the networks nodes to give an output. Using an appropriate training algorithm, the NF system parameters can be optimized properly. A NF technique can provide the network to capture complex relationships between the input and output variables and make more accurate predictions than both neural network- and fuzzy logic-based systems.

Reinforcement learning is a more advanced training approach, which can expand on these general supervised training concepts and excel in environments where an optimal action policy is unknown and training data is difficult or impossible to generate. In general, a reinforcement learning agent takes actions in a complex environment and receives a reward based on the success of that action. For example, the agent generates a decision-making policy π, which can map states to a probability distribution of actions [42]. This policy is shown in Equation ((1.1), where S^{π} is the state space and A^{π} is the action space.

$$\pi\colon S^{\pi} \rightarrow P(A^{\pi}) \tag{1.1}$$

Different from the classical training processes, reinforcement learning uses a Markov Decision Process rather than training data. Consider a NF system as an example. At a given state and at time moment t, s_t, the actor can take any action a that is made at the current state, which will lead to a new state at $t+1$, or s_{t+1}. The probability of state s_t changing to state s_{t+1} is represented by $P(s_t, s_{t+1})$ and the reward received for this state change is $R(s_t, s_{t+1})$. A policy, π must be developed to incentivise desired performance. This policy π will map the state

space to the action space; it is chosen to maximize the sum of the rewards generated over a series of decisions. The function to be optimized is referred to as a value function V^{π}, as illustrated in Equation 1.2; it uses a discount factor γ (usually close to 1) to incentivise current decisions over future decisions [43, 44]. A policy that is made by maximizing Equation (1.2) is the optimal policy:

$$V^{\pi} = \sum_{i=1}^{\infty} [\gamma^{i-1} r_i \mid s, \pi] \tag{1.2}$$

In typical applications, Equation (1.2) takes the form in Equation (1.3), which is known as the sum of discounted future rewards:

$$R_t = \sum_{i=1}^{\infty} [\gamma^{i-t} r_{(s_i, a_i)}] \tag{1.3}$$

The goal of reinforcement learning is to maximize the expected return from the starting distribution represented in Equation (1.4) is the goal of reinforcement learning.

$$J = [R_1 | s_t, a_t] \tag{1.4}$$

1.5.2 Literature Review for Reinforcement Learning

Reinforcement learning has been used for system control in literature. For example, Mnih et al. from Google DeepMind proposed the Deep-Q Network reinforcement learning algorithm in order to learn a policy to play Atari games from raw pixel data [45, 46]. The deep-Q network was able to develop strongly performing policies by taking only the raw pixel data and score from various Atari games. Silver et. al used reinforcement learning to master traditionally complex games such as Go and Chess [47, 48]. Previously these games were thought to be some of the most challenging games for artificial intelligence to learn, due to an extremely large search space and complex evaluation of board position and moves. Silver et al. also suggested a Deterministic Policy Gradient algorithm for determining the policy gradient of an action-value function, which could provide efficient performance and outperformed general stochastic methods [49]. Lillicrap combined the Deep-Q Network and Deterministic Policy Gradient and created the Deep Deterministic Policy Gradient (DDPG) algorithm, for continuous control [42].

It is shown that the DDPG algorithm outperforms classical controllers such as PID in performing different motor control and tracking control [50–52]. Gheisarnejad and Khooban developed a PID controller based on the DDPG algorithm that acted as a supplementary controller to adapt to uncertainties and disturbances; test results showed that the proposed controller outperformed traditional PID control methods [53]. Fujimoto et al. applied a pair of critics to address function approximation errors in actor critic methods with smaller outputs [54]; test results showed that it could limit overestimation and improve the DDPG algorithms performance. Tobin et al. addressed the challenges of applying an agent to a real-world problem that was training on a simulated version of that environment with domain randomization [55]. This introduced random variation in images taken by a virtual camera in a simulation when training an agent to detect objects from the images. When the agent was applied to a real camera, the agent was able to detect real objects due more accurately to the increased uncertainty during training.

1.6 Objectives of this Work

The first objective of this work is to model the AT-802 and the hose-drogue system. A model of the aircraft will be developed and then a simulation of the hose-drogue in flight will be created. This analysis will aid in investigating the potential of the medium sized AAR tanker, AT-802, as a tanker, and providing a better understanding of the drogues flight-characteristics behind the AT-802. Results will also be used to verify the controllability of a hose-drogue system deployed by the aircraft.

The second objective is to propose a new control technique to stabilize a hose-drogue system in flight, using a reinforcement learning algorithm. As we have no hose and drogue prototype systems available for real testing, this research will be conducted using a smart structure workstation in our research lab. A new technique based on reinforcement learning will be proposed for flexible structure vibration control under varying dynamic conditions. Domain randomization is applied to the state-space model of the smart structure to train the DDPG controller and to increase control convergence. The effectiveness of the proposed reinforcement learning DDPG controller will be examined by systematic experimental tests and compared to other related controllers.

1.7 Thesis Structure

The following chapters of this work are organized by areas of focus for research related to aerial refueling:

Chapter 1 introduces the topics of focus of the thesis and includes a literature review. Research objectives and thesis structure is outlined.

Chapter 2 discusses the primary design and modeling of an aerial refueling from the Air Tractor 802 airframe. The A dynamic simulation of a hose-drogue system is performed and the AT-802 Airframe is modeled and simulated.

Chapter 3 lays out the proposed deep deterministic policy gradient algorithm and the related applied reinforcement learning concepts. The sim-to-real gap problem is discussed and the technique of domain randomization is proposed as a solution to this problem for this application.

Chapter 4 proposes the experimental setup used to approximate a hose and drogue as a flexible structure. The designed PD, NF, and DDPG controllers are introduced. System modeling of the smart structure is performed to create a simulation environment for the reinforcement learning agent to be trained in. Selected hyperparameters of the DDPG training algorithm are provided.

Chapter 5 examines the effectiveness of the proposed DDPG control technique experimentally using the flexible structure workstation. The tests are undertaken to simulate variable dynamic conditions. The performance of the proposed DDPG technique is compared with the related control techniques.

Chapter 6 summaries the findings from this work and draws conclusions from the analysis of test data. It also includes some future work for the improvement to the control technique, training algorithm and advanced development.

Chapter 2 – Modeling and Analysis

Hose-drogue system modeling is performed in this Chapter in order to determine the feasibility of operating the AAR using a small tanker like the AT-802. Some considerations include the length of hose, the packaging constraints and aircraft weight envelope. Currently, there are no good quality models of the AT-802 in literature. Consequently, it is valuable to create to a 3D model to support the related research and development. A 3D model of the AT-802 FuelBoss, a hose-drogue dynamic model and a weight envelope model are created.

2.1 AT-802 3D Model

The model of the AT-802 will be created in Autodesk Inventor based on drawings from AirTractor as well as measurements taken on-site. It will be a reference model and used in preliminary computational fluid dynamic (CFD) testing to represent the AT-802. Figures 2.1 and 2.2 show a comparison between a photo of a two-seat AT-802 tanker and a render of the model produced for this project. Some slight differences due to modeling complexity can be noticed, such as small differences in the cockpit and wing tip shape. These differences would not affect the accuracy of the modeling. The larger belly tank for carrying auxiliary fuel can be seen on the model in Figure 2.2, which can be seen more when comparing Figure 2.3 and Figure 2.4.

Figure 2.1:AT-802 photo – Front isometric view [56]

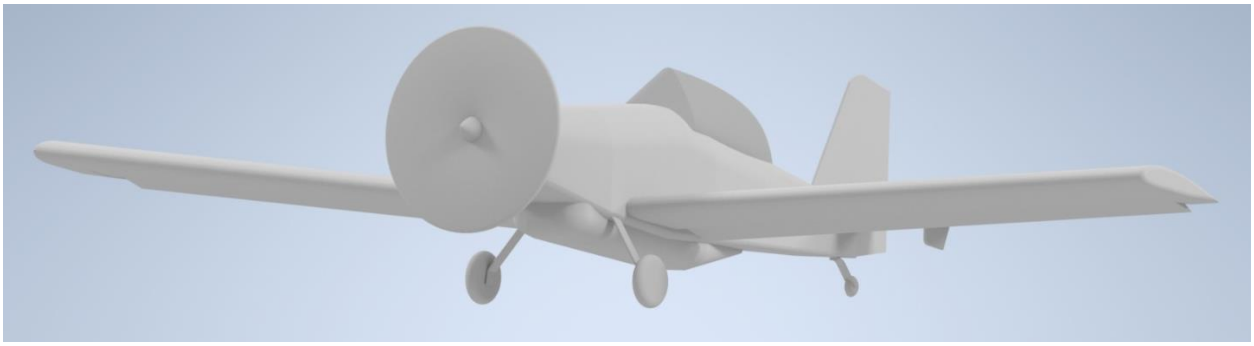


Figure 2.2: AT-802 model - Front isometric view

Figure 2.3: AT-802 photo - Rear underside view [57]

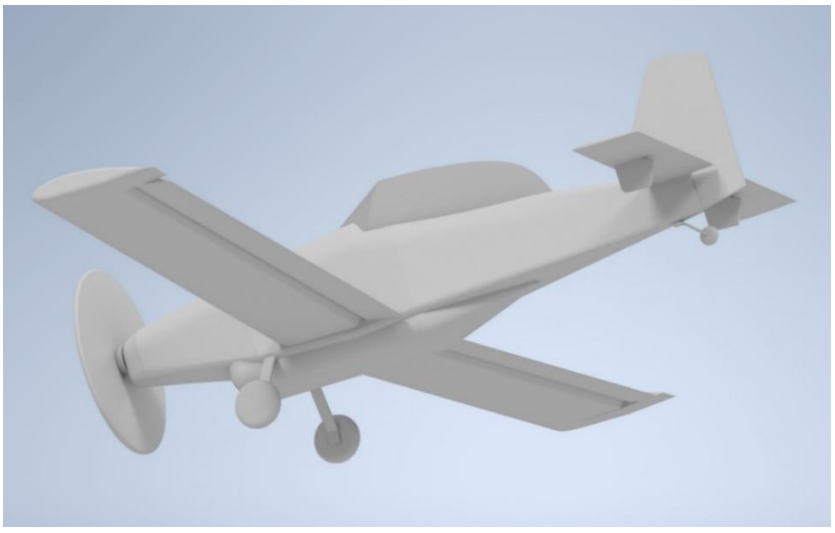


Figure 2.4: AT-802 model - Rear underside view

Pictured in Figure 2.5 is a belly tank that is installed on the AT-802 to add auxiliary fuel capacity. This version of the AT-802 is called the FuelBoss. The fuel load and off-load ports are covered by a long cowling, which can be seen on the underside of the model in Figure 2.4. This differs slightly from the comparison photo used in Figure 2.3, which is from a firefighting model with a water-dump system.

Figure 2.5:AT-802 FuelBoss belly tank

Figures 2.6-2.9 show the top, front, and side views of the 3D model next to the Air Tractor drawing for comparison. These examples can show the similarities between the 3D model and the aircraft in terms of body shape and aircraft profile. As illustrated in Figure 2.6 and Figure 2.7, the wings are longer on the model, this is due to the length of the wings on the AT-802 that the model was based on.

(a)

(b)

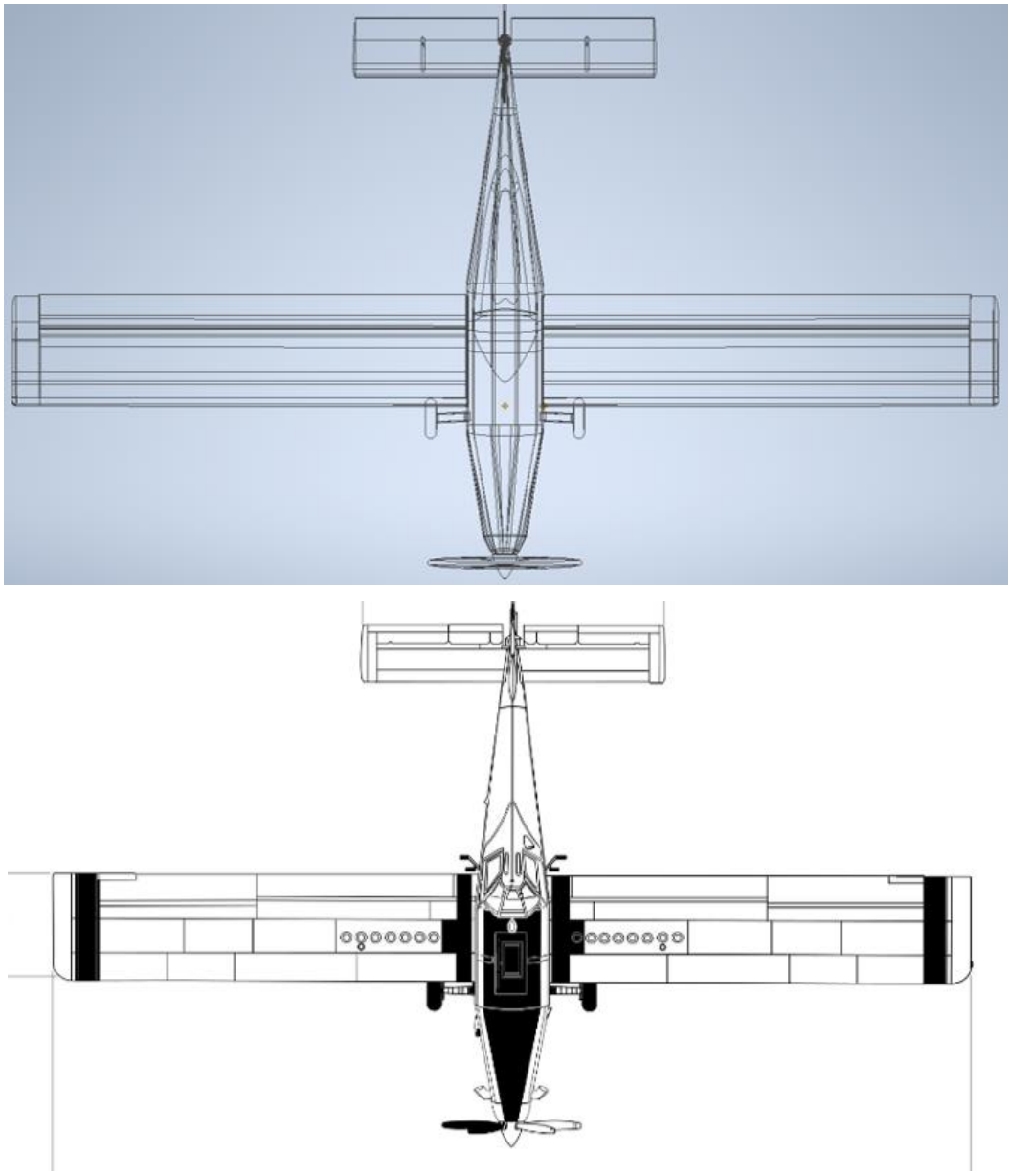
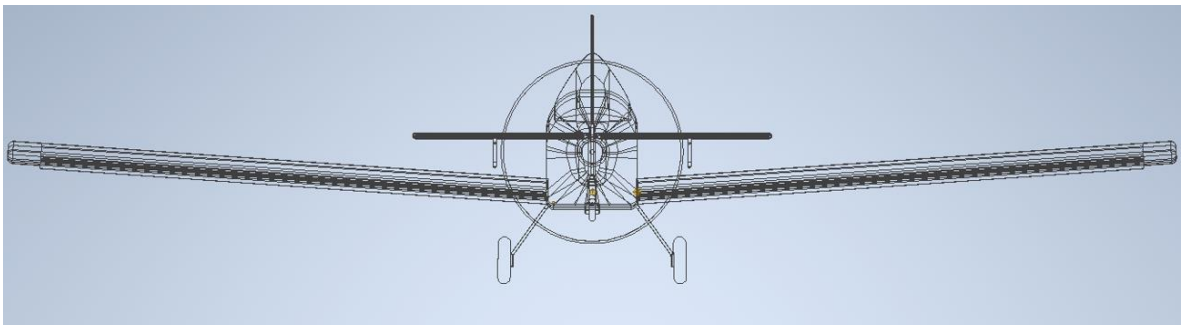


Figure 2.6: AT-802 model - Top view: (a) model, (b) drawing

(a)



(b)

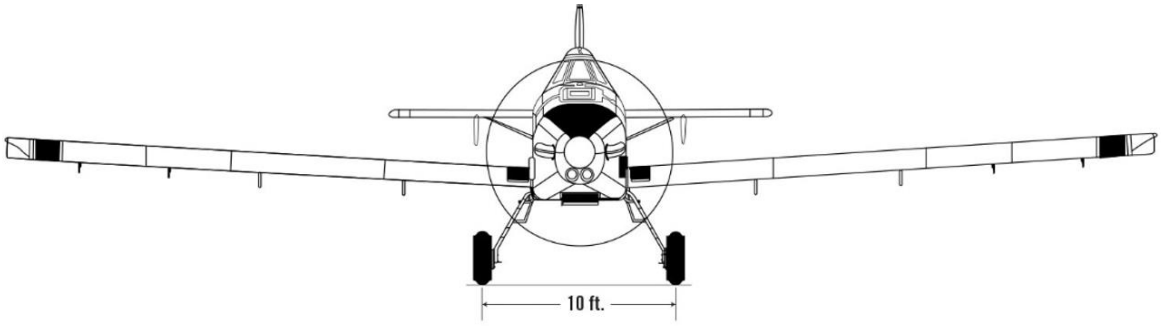
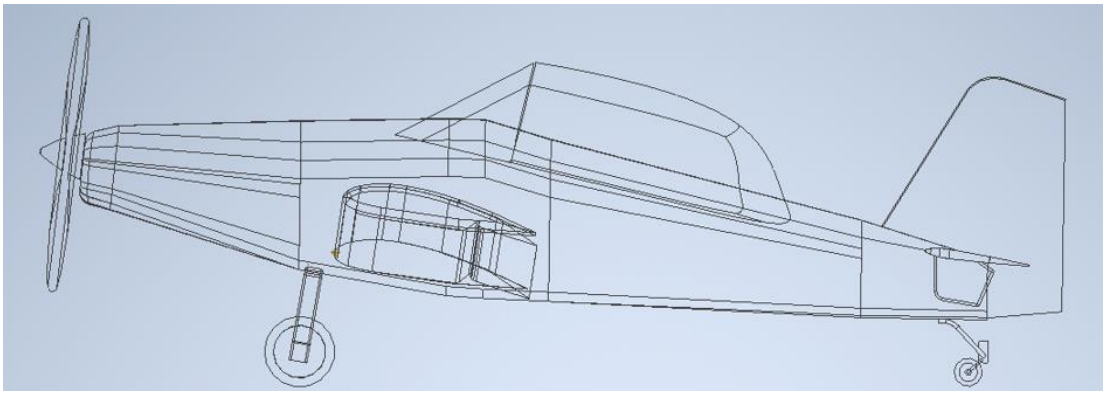


Figure 2.7: AT-802 model - Front view: (a) model, (b) drawing

(a)



(b)

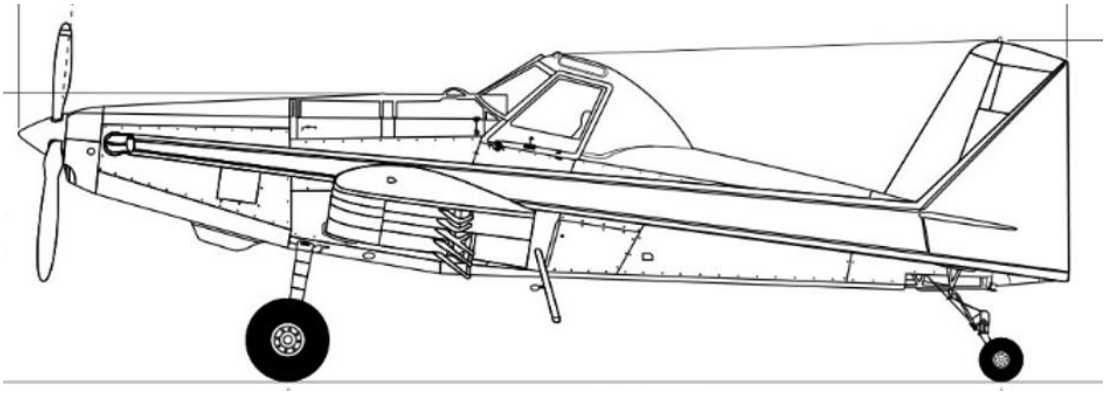


Figure 2.8: AT-802 model – Side view: (a) model, (b) drawing

As can be seen in Figure 2.6, Figure 2.7, Figure 2.8 proportions of the 3D model are very close to the real-life airframe. To gain a better understanding the wake behind the AT-802, the model will be simulated in computational fluid dynamics software. The wings in the model are modified to match a flap setting of 10 degrees, then the model is placed in a large volume of air moving at 110 knots horizontally. This simulates the plane moving through still air at the same speed, with correct settings for refueling. The solution converged after 286 iterations and results showing the velocity magnitude of the air around the aircraft from the side and rear are shown in Figures 2.9 and 2.10 respectively. Results are in the form of a heat map, with warmer colours representing a higher velocity magnitude.

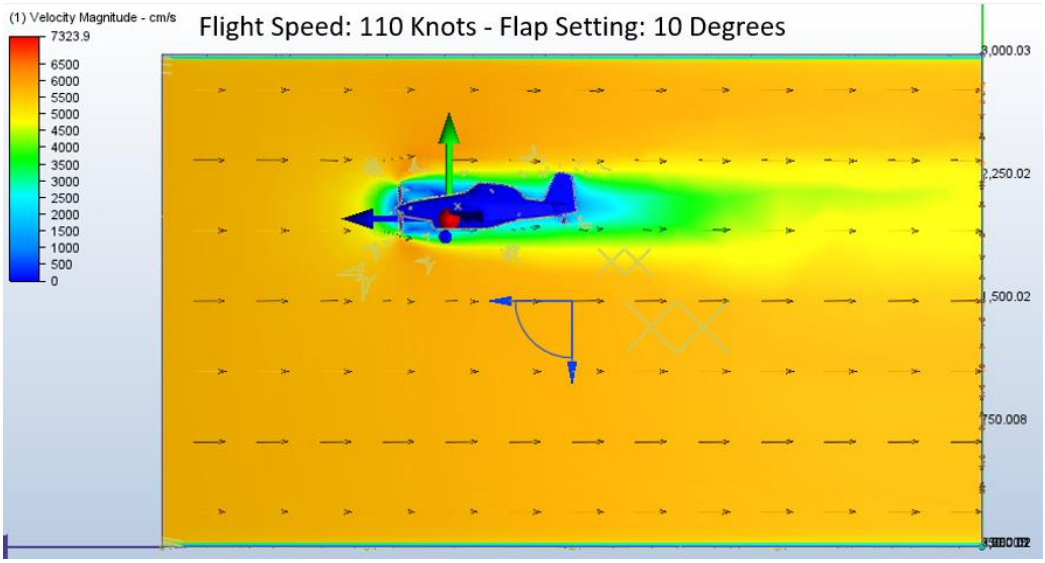


Figure 2.9: AT-802 side profile of CFD analysis (velocity magnitude)

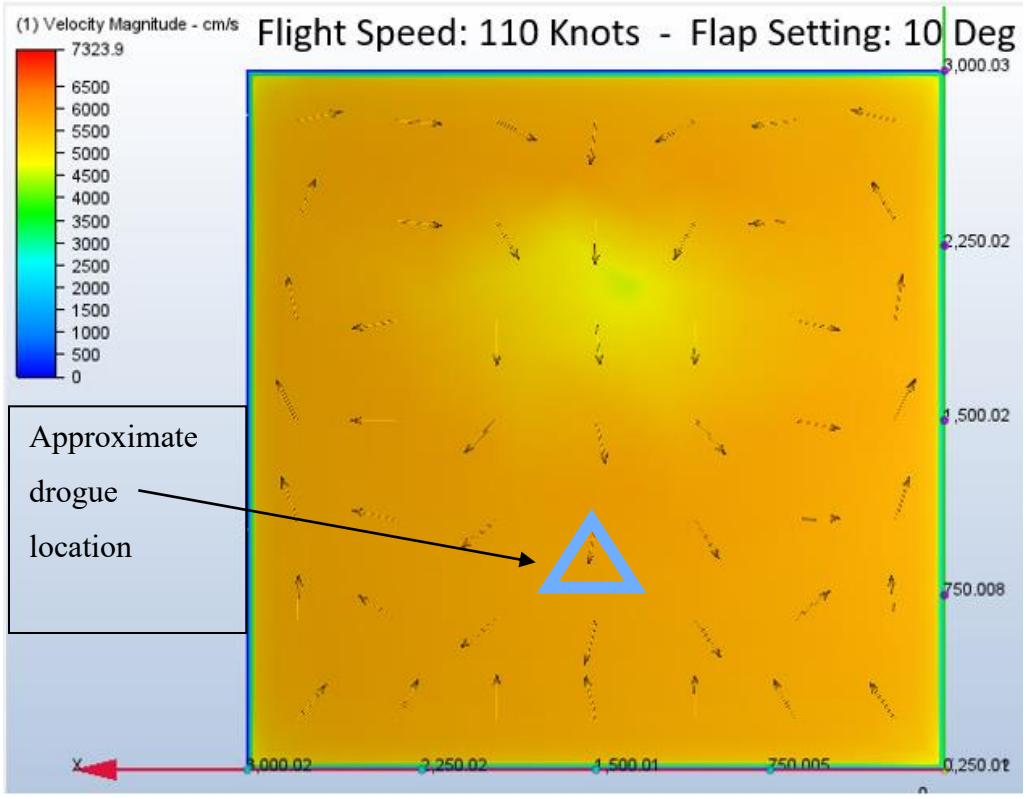


Figure 2.10: AT-802 rear profile of CFD analysis (velocity magnitude)

Results from the CFD test showed a homogenous wake behind the AT-802 with no low or high velocity areas. Orange coloured areas represent a speed around 105-115 knots which is the expected speed for AAR operations. The area with the most deviation from the expected air

speed is near the center and which is slowed down to around 90 knots. This is due to the wake from the aircraft and leads to an airspeed reduction of around 18%. This area is also upwards from the location the drogue would be expected to sit in during refueling operations, which is denoted by the blue triangle in Figure 2.10. Test results show that the wake behind the AT-802 is not very turbulent and that a drogue in flight behind the plane would be unlikely to behave unpredictably.

2.2 Center of Gravity (CoG) Considerations and Weight Envelope Modeling

Considering packaging of a refueling system into a FuelBoss, it is important to model the impact of weight and balance. The model will be created to analyze effects of loading on the position of the center of gravity and the weight envelope. Table 2.1 features sample loadings for both an acceptable and unacceptable take off condition. This table lists the major loads on the airframe, and the engine and body centers of gravity provided by the manufacturers. The station is listed beside the load for each component; the station is measured in inches between the center of the load and the datum of the airplane. There is no set location for a datum on the airframe, which is a reference point used by designers from which load location is measured. Typically, towards the front of the airplane from the datum is a negative measurement and towards the rear is a positive measurement. The CoG is calculated by the moment method about a reference. The loading differences between the presented acceptable and unacceptable cases are the position of the refueling pod and a ballast in the engine compartment.

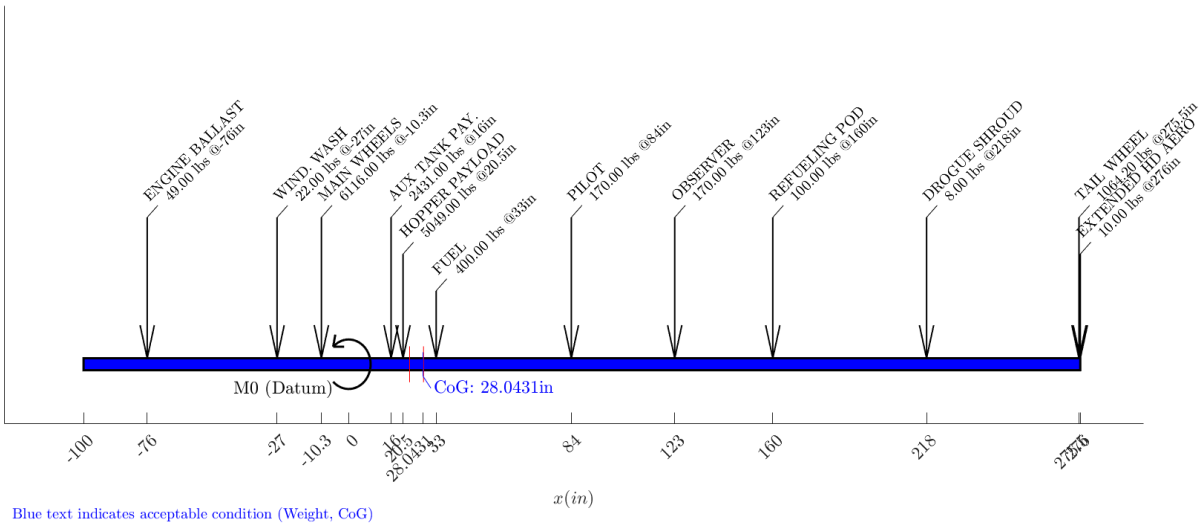
Table 2.1: Sample AT-802 refueling system loading

	Acceptable Sample Loading		**Unacceptable Sample Loading**	
	Load (lbs)	Station (in)	Load (lbs)	Station (in)
Main Wheels	6116	-10.3	6116	-10.3
Tail Wheel	1064.2	275.5	1064.2	275.5
Pilot	170	84	170	84
Observer	170	123	170	123
Fuel	400	33	400	33

Hopper Payload	5049	20.5	5049	20.5
Aux Tank Pay.	2431	16	2431	16
Wind. Wash	22	-27	22	-27
Refueling Pod	***100***	***160***	***100***	***220***
Drogue Shroud	8	218	8	218
Drogue	10	276	10	276
Engine Ballast	***49***	***-76***	***0***	***-76***
Total	15589.2		15540.2	
CoG (inches from datum)	28.0”		28.8”	

Figure 2.11 and Figure 2.12 provide a graphical representation of these weights using a weight and balance model. Figure 2.11 shows the acceptable loading case, in which a refueling pod is mounted underneath the second seat of the aircraft. This loading case requires an extra 49lb ballast in the engine compartment to improve the safety during take-off and landing, but the overall weight should not be over 16000lbs. From calculations, the CoG is located 28 inches behind the datum. In order to demonstrate an unacceptable loading case, the reel is moved 5 feet aft of where it was originally located. Figure 2.12 illustrates the effects on the aircraft after the ballast was removed. These changes have resulted in the variation of 28.8 inches in the CoG, which falls outside of the weight envelope threshold at a weight near 16000 lbs.

(a)



(b)

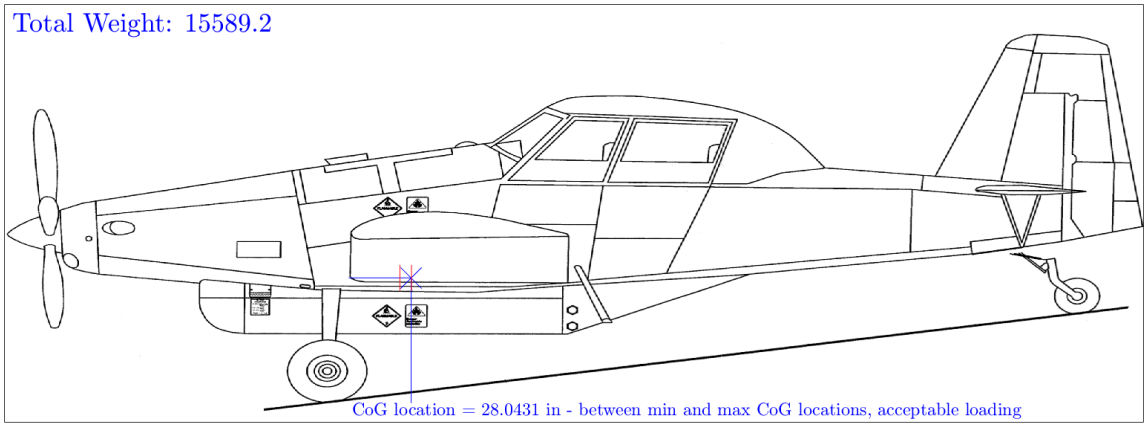


Figure 2.11: AT-802 weight envelope model - Acceptable loading condition: (a) simplified CoG diagram. (b) graphical representation of modified CoG position

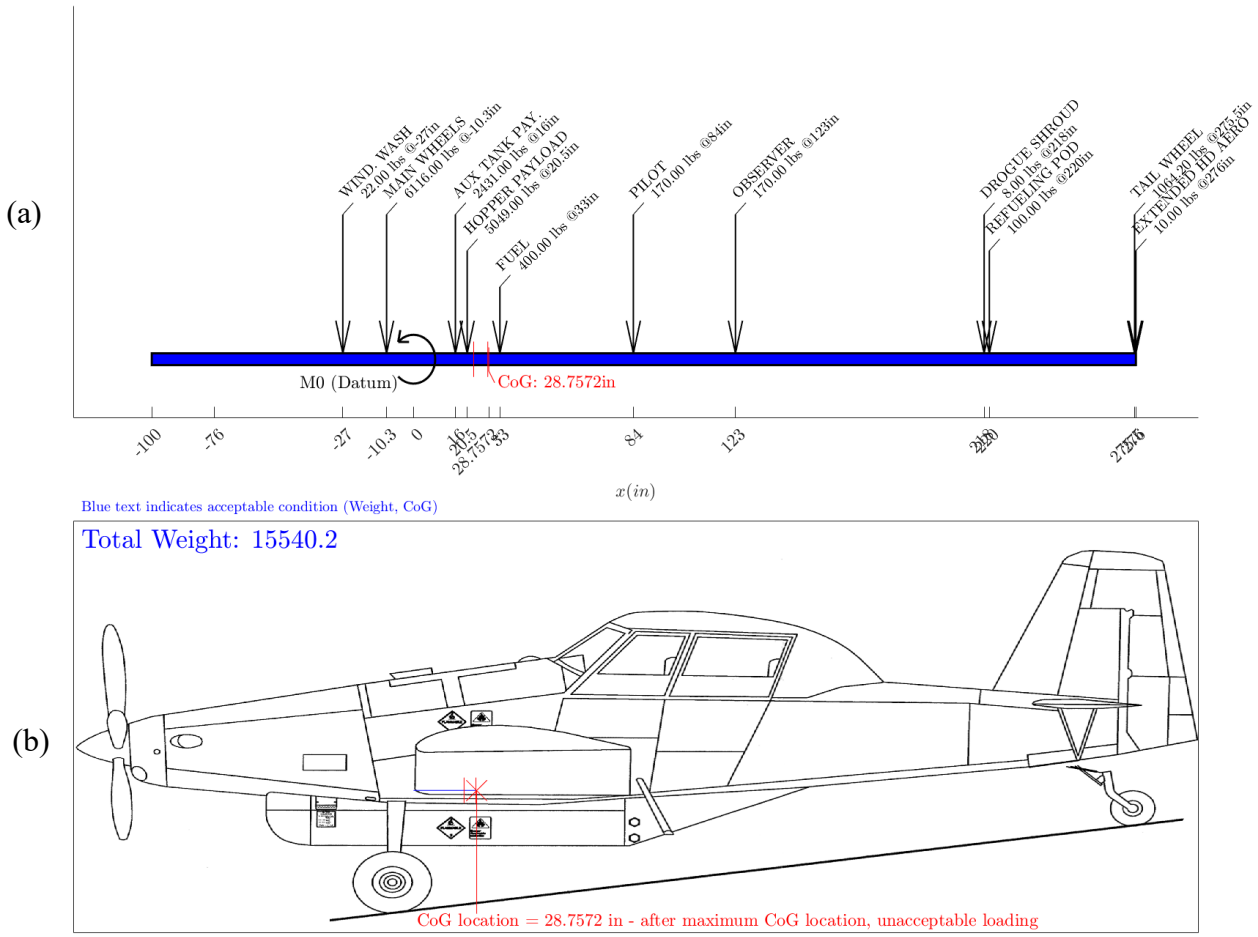


Figure 2.12: AT-802 weight envelope model - Unacceptable loading condition: (a) simplified CoG diagram. (b) graphical representation of modified CoG position

Figure 2.163 shows the center of gravities from Figures 2.11 and 2.12 on the weight envelope for safe AT-802 operation. It can be observed that the AT-802 loaded with a refueling system is very close to being outside of the weight envelope. Therefore, this issue must be taken carefully during system development to ensure the aircraft is within the allowed weight envelope when fully loaded for take-off. Suggestions for possible modifications include moving the belly tank forward and adding components of the refueling system to balance the weight, so as to balance the CoG.

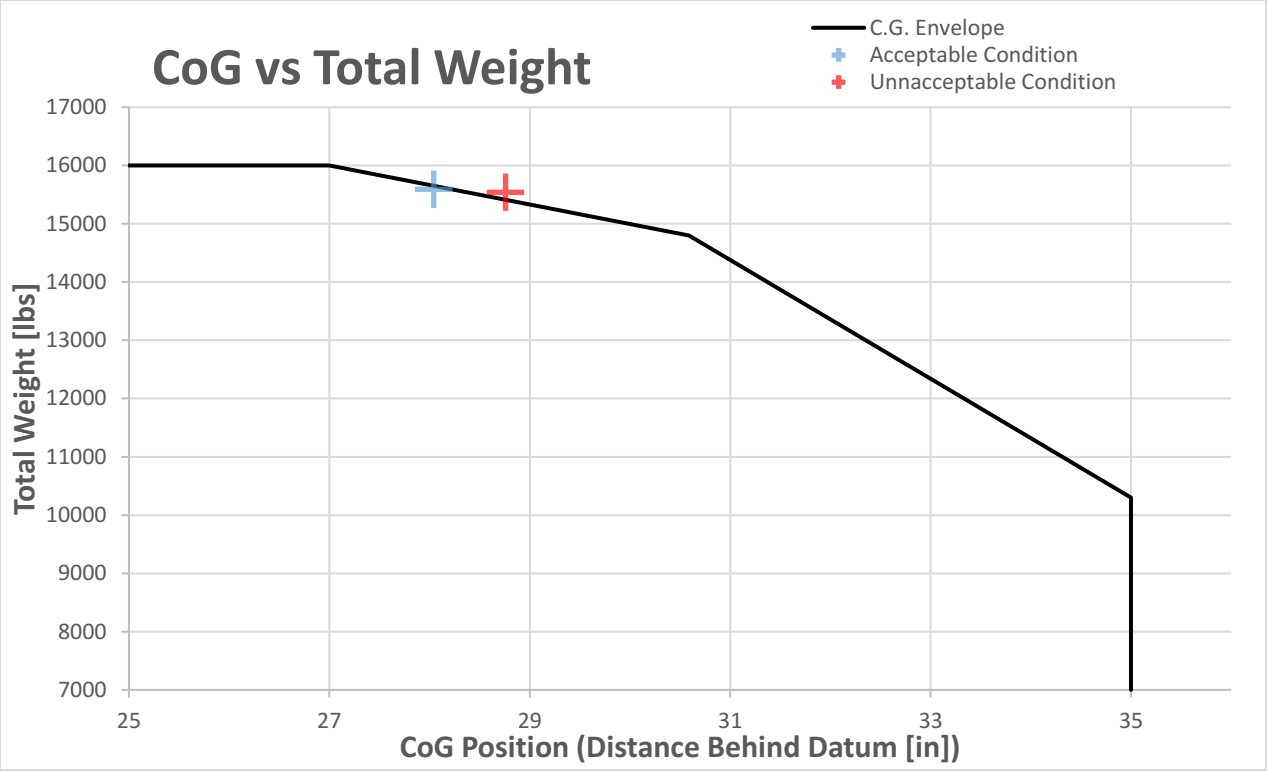


Figure 2.13: Weight envelope – plot of both acceptable and unacceptable conditions

2.3 Hose Drogue Dynamic Model

In order to gain a better understanding of the hose-drogue dynamics under aerodynamic flight conditions, a dynamic model will be developed in this section. This modeling will be an improved one from a simplified model in [23]. The simplified models include with the following assumptions: a static drag for the drogue and the constant variables in air density, drag coefficients, wind speed, wind direction, and initial conditions. In general modeling approaches, the hose is usually modeled as link units trailing from the tanker to the drogue attached at the other end of the hose. Hose links are connected by frictionless spherical joints that are modeled as lumped masses. The drogue is usually modeled as a lumped mass with a constant drag force acting on the free end of the hose. Figure 2.14 shows the reference frame used in this model F_W, as well as the hose-drogue treated as linked lumped masses.

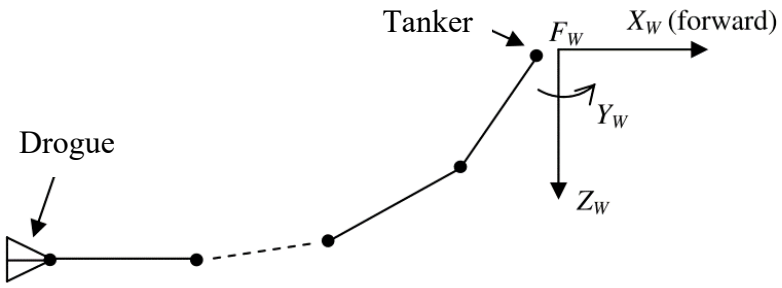


Figure 2.14: Side view of reference frame and lumped mass model [23]

The position of any lumped mass relative to the inertial frame can be characterized by Equation (2.1) and Equation (2.2) shows the position of lumped masses, relative to the previous linked mass.

$$p_K{}' = p_J{}' + p_K \tag{2.1}$$

$$p_K = -l_K(cos\theta_{K1}cos\theta_{K2}w_1 + sin\theta_{K2}w_2 - sin\theta_{K1}cos\theta_{K2}w_3) \tag{2.2}$$

where θ_{K1} and θ_{K2} represent the angles of link *K* in the vertical and horizontal directions, respectively, when viewing the drogue from behind. The unit vectors of the tanker frame F_W are given by w_1 , w_2, and w_3, which represent the X direction (roll axis of tanker), Y direction (pitch axis of tanker), and Z direction (yaw axis of tanker). Figure 2.15 displays the convention used to refer to any link, with link K being the currently observed link, link J being the link beside link K closer to the tanker, and link L being the link next to K that is closer to the drogue.

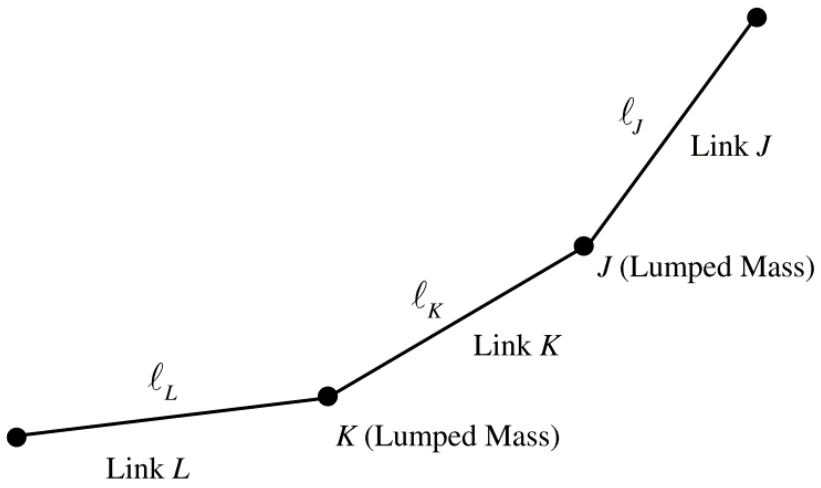


Figure 2.15: Link and lumped mass notation for any link K [23]

The derivatives of the link position vector, p_k, are demonstrated in Equations (2.3) and (2.4), whereas the velocity and acceleration of lumped masses are represented by Equations (2.5) and (2.6), respectively.

$$\dot{p}_K = \sum_i \left(p_{K,\theta_{Ki}} \dot{\theta}_{Ki}\right) + (\omega_W \times p_K) \tag{2.3}$$

$$\ddot{p}_K = \sum_i \left(p_{K,\theta_{Ki}} \ddot{\theta}_{Ki}\right) + \sum_i \left(p_{K,\theta_{Ki}} \dot{\theta}_{Ki}\right) + (\alpha_W \times p_K) + (\omega_W \times \dot{p}_K) \tag{2.4}$$

$$v_K = v_J + \dot{p}_K \tag{2.5}$$

$$a_K = a_J + \ddot{p}_K \tag{2.6}$$

The angular velocity and acceleration of any link relative to the tanker is given by ω_W and α_W, respectively. The length of each link is $-l_K$. If accelerations of all the lumped masses are obtained, the link orientation angles and their time derivatives, as well as the angular momentum of the frame F_W. Equation (2.7) is used to compute for the second derivatives of all link angles:

$$\ddot{\theta}_{Kj} = p_{K,\theta_{Kj}} * \frac{\left[a_K - a_J - \sum_i \left(\dot{p}_{K,\theta_{Ki}} \dot{\theta}_{Ki}\right) - (\alpha_W \times p_K) - (\omega_W \times \dot{p}_K)\right]}{\left(p_{K,\theta_{Kj}} \cdot p_{K,\theta_{Kj}}\right)}, \qquad j = 1,2 \tag{2.7}$$

Next the equations of motion can be represented by Equations (2.8) and (2.9), where $\vec{Q}_K$ is the external force vector acting on the *K*th lumped mass, $\vec{T}_K$ is the tension vector in the *K*th link, and n_{Ki} is the unit vector in the frame of the link *K*. Substituting Equation (2.8) into Equation (2.7) gives a set of tension equations for each link, as represented in Equation (2.10). This equation also can be represented in a matrix form as in Equation (2.11), where Equation (2.10) is used to fill in the matrices of [Γ] and $\{q\}$, with the exceptions being given by Equation (2.12)

$$a_K = \frac{\vec{Q}_K + \vec{T}_K + \vec{T}_L}{m_K} = \mu_K(\vec{Q}_K + \vec{T}_K + \vec{T}_L) \tag{2.8}$$

$$\frac{d\left((p_K \cdot p_K) = l_K^2\right)^2}{d^2t} \quad \Rightarrow \quad (a_K - a_J) \cdot n_{K1} = l_K \dot{n}_{K1}^2 \tag{2.9}$$

$$-\mu_J(n_{J1} \cdot n_{K1})T_J + (\mu_J + \mu_K)T_K - \mu(n_{L1} \cdot n_{K1})T_L = l_K \dot{n}_{K1}^2 + (u_J Q_J - u_K Q_K) \cdot n_{K1} \tag{2.10}$$

$$[\Gamma]\{T\} = \{q\} \tag{2.11}$$

$$\begin{gathered} \Gamma_{11} = \mu_K, \quad \Gamma_{12} = -\mu_K(n_{L1} \cdot n_{K1}), \\ \Gamma_{K(K-1)} = -\mu_J(n_{J1} \cdot n_{K1}), \quad \Gamma_{KK} = \mu_J + \mu_K, \\ q_K = l_K \dot{n}_{K1}^2 - \mu(\vec{Q}_K \cdot n_{K1}) + (a_0 \cdot n_{K1}) \end{gathered} \tag{2.12}$$

where μ_K is the reciprocal of the mass for the Kth link. The external force acting on any lumped mass K can be represented by Equation (2.13).

$$\vec{Q}_K = m_K g + \frac{1}{2}(\vec{D}_{K-1} + \vec{D}_K) \tag{2.13}$$

where D_K is the aerodynamic force acting on the Kth lumped mass and D_d is the drag of the drogue. The external force acting on the final lumped mass, or the drogue is given by Equation (2.14).

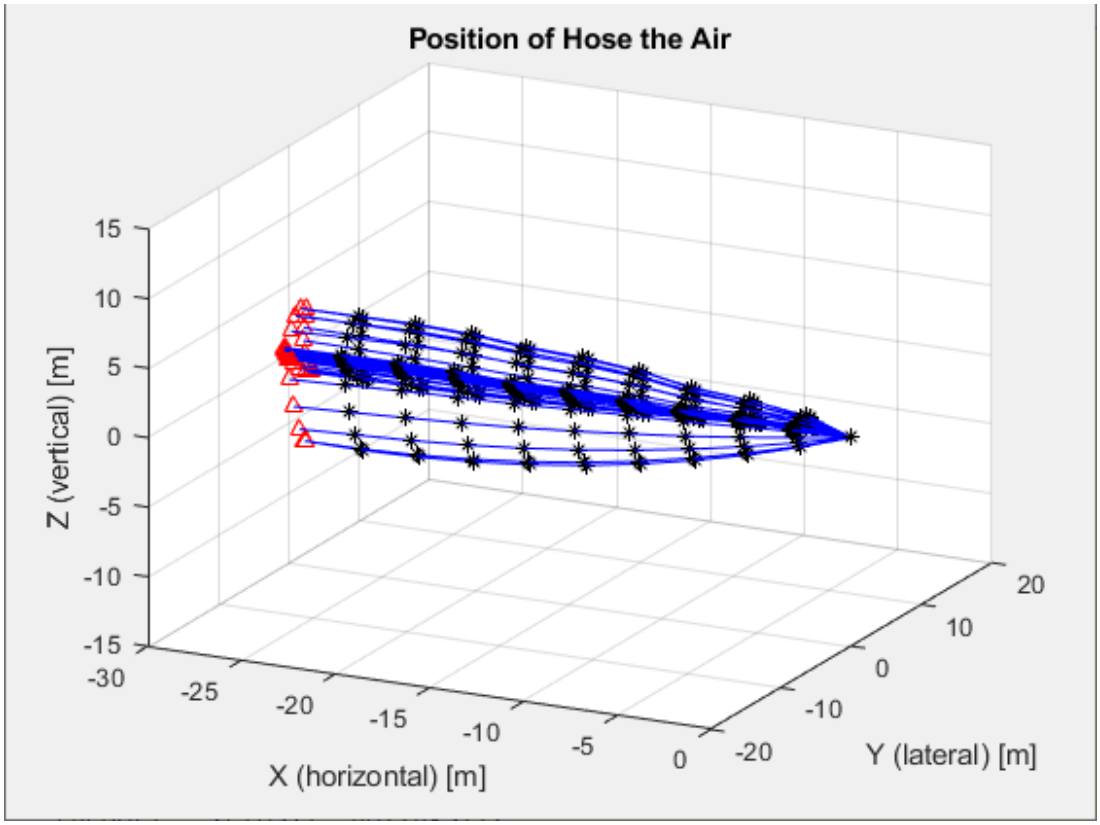
$$\vec{Q}_D = (m_N + m_{drogue})g + \frac{1}{2}\vec{D}_N + \vec{D}_D \tag{2.14}$$

The following algorithm summaries the operation procedures for system simulation using MATLAB:

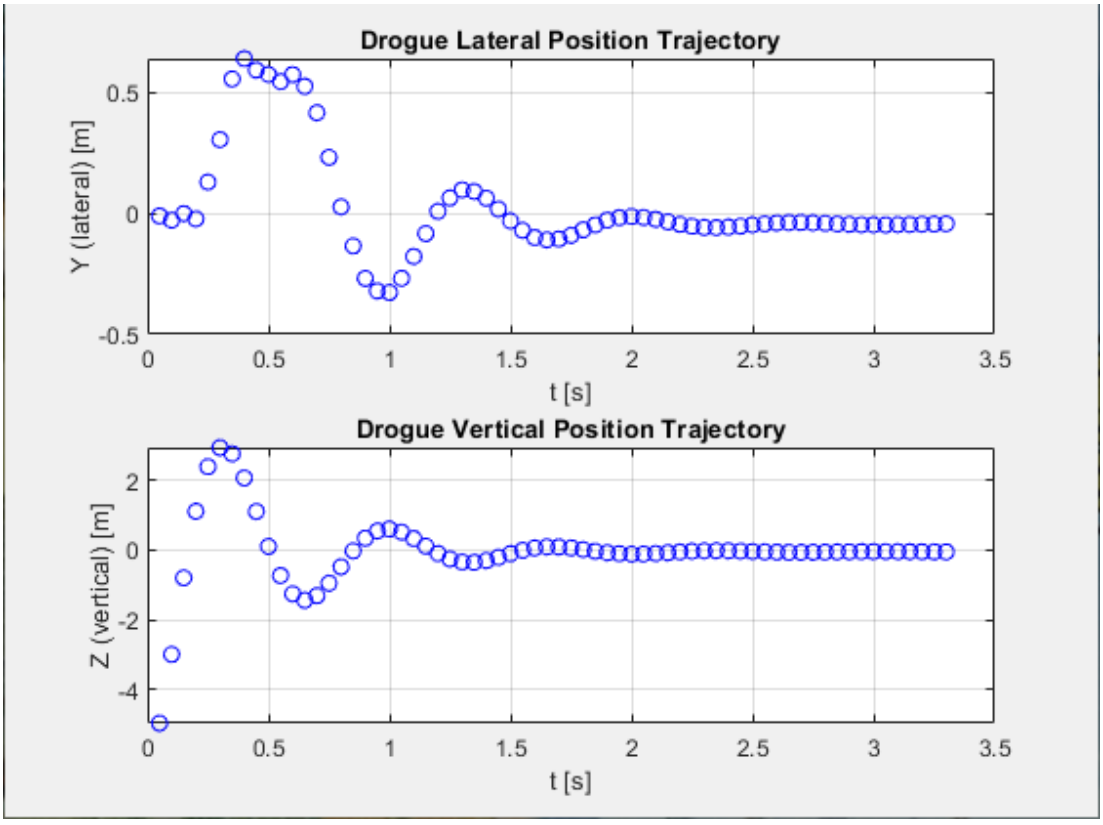
1. Recursively compute the position and velocity of each lumped mass using Equations (2.1) - (2.4).
2. Compute the external force vectors Q_K for each lumped mass using Equations (2.14) and (2.15).
3. Fill the Γ matrix and q vector using Equation (2.12) and solve $[\Gamma]\{T\} = \{q\}$ to find the link tension vector $\{T\}$.
4. Calculate the accelerations of each lumped mass, using Equation (2.8).
5. Compute the second derivative of each link angle, using Equation (2.7).
6. Estimate the link angles and their derivatives for the next time step using numerical integration.
7. Repeat Steps 1-6 to simulate dynamic response of a hose-drogue deployed in flight conditions behind an AT-802.

In simulation, the hose is 30m long with 10 links. The simulation runs until it is completed when drogue position converges within a set limit of 1 mm among three steps. An example of simulation results can be seen in Figure 2.16.

a)


Position of Hose the Air
Z (vertical) [m]
15
10
5
0
-5
-10
-15
-30
-25
-20
-15
-10
-5
0
X (horizontal) [m]
20
10
0
-10
-20
Y (lateral) [m]

b)


Drogue Lateral Position Trajectory
Y (lateral) [m]
0.5
0
-0.5
0
0.5
1
1.5
2
2.5
3
3.5
t [s]
Drogue Vertical Position Trajectory
Z (vertical) [m]
2
0
-2
-4
0
0.5
1
1.5
2
2.5
3
3.5
t [s]

c)

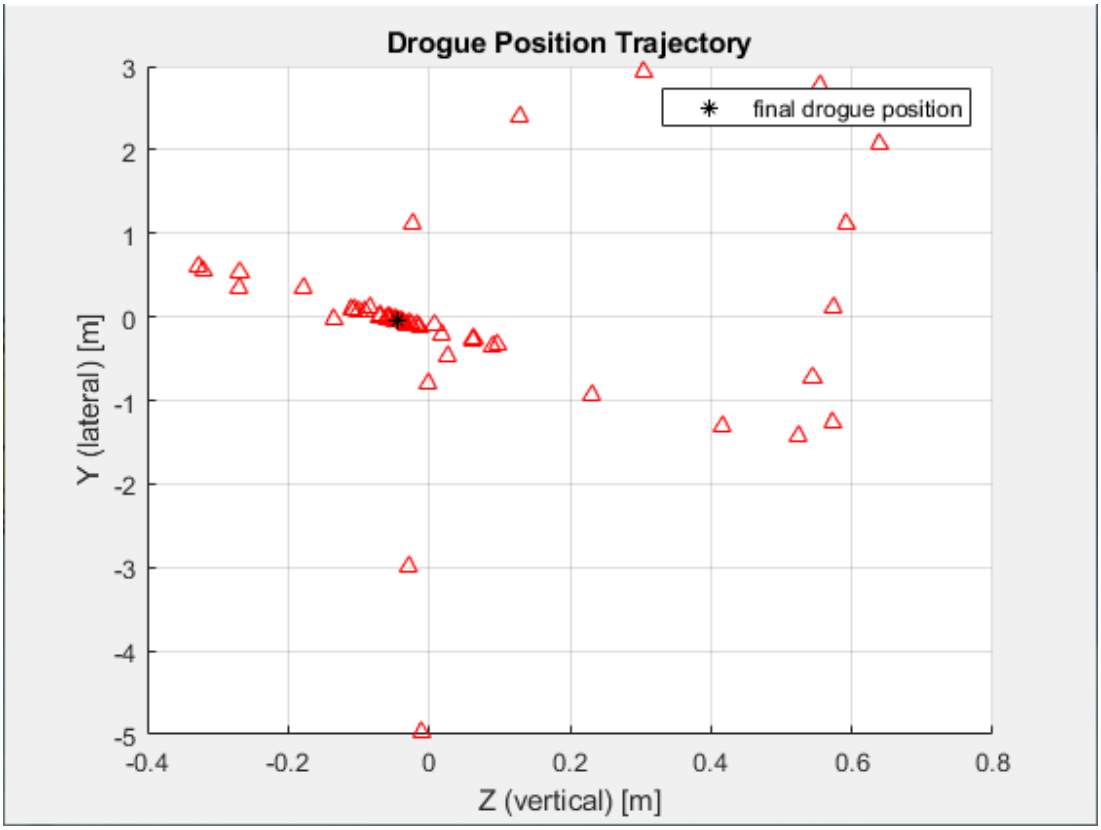


Figure 2.16: Results for 10 link test: a) 3d representation of simulation, b) lateral and vertical trajectories of drogue, c) drogue position (viewed from the rear)

The results show that the drogue tends to stabilize at a steady position when there is no turbulence. The drogue must be stable in order for refueling to proceed; therefore, control of the drogue would be a benefit to aid pilots in refueling. As it is at the initial stage of this AAR project, we have no hose-drogue prototypes for system control testing. A flexible beam structure in our lab will be used as a substitute for primary flexible hose system control under variable dynamitic conditions.

Chapter 3 – Development of the Reinforcement Learning Controller

3.1 Review of the Deep Deterministic Policy Gradient Strategy

Reinforcement learning control methods possess characteristics that make them attractive for drogue stabilization applications when compared to other areas of control design such as classical control and intelligent control. Reinforcement learning has been shown to excel at adaptation to changing environments [58], in the context of AAR, aerodynamic effects such as turbulence and wake are commonplace and change suddenly; an adaptable controller will better handle these conditions. Classical controllers are typically model-based and may struggle to adapt, whereas intelligent controllers, while flexible may require more human input to adapt from new data. Rapidly changing dynamic conditions while a drogue is in flight can make stabilization a challenge; reinforcement learning control implements exploration techniques that can be beneficial in these chaotic environments. Exploration during the training of a reinforcement learning controller can lead to the development of an action policy with unique strategies to combat these issues [59]. Classical and intelligent controllers often rely on predefined control rules and models, without the ability to discover less-obvious solutions. Potential environmental variations in AAR are unpredictable and difficult to consider in modeling or logical rules, by contrast reinforcement learning models can be expanded to consider more input features such as wind velocity and turbulence intensity without massively increasing controller design complexity [60]. A classical controller may require extensive redesign to accommodate more input variables and the models used may not allow for this increased complexity; intelligent controllers would require manual feature tuning to enable this and a vast collection of training data spanning different conditions to optimize for additional inputs.

There are several reinforcement learning strategies for control applications such as the Trust Region Policy Optimization algorithm, the Model-Based Policy Optimization algorithm, and the DDPG algorithm. While trust region methods are effective, they are very complex and computationally intensive to train successfully, mainly due to the application of constraint optimization [61]. Model-based methods can be effective while remaining sample and computationally efficient; the drawback is that a system model is required to learn optimal decision-making policy [62]. The DDPG meanwhile, is complex, but not so much as to be

computationally difficult to train while producing results that are very promising for control applications [63]. Also, the DDPG does not require a learned system model to be optimized [42]. In this work a reinforcement learning control method based on the deep deterministic policy gradient (DDPG) will be adopted for drogue system control. To investigate the merits of the proposed method, it is compared with a classical PD controller and an intelligent NF controller.

The DDPG technique uses deep-Q networks in conjunction with the deterministic policy gradient algorithm. The deep-Q network uses multiple densely connected hidden layers to create more complex interpretations of inputs [46]. A deep-Q network is model-free, or the exact policy π and reward R functions are formulated by learning [54] in order to approximate the optimal action-value function $Q^*(s,a)$:

$$Q^*(s,a) = \max(\mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^n r_{t+n} \mid s_t = s, a_t = a, \pi]) \tag{3.1}$$

Equation (3.1) represents the maximum sum of rewards, for a given state and action, where r_t is the reward at each time step t, and γ is the reward discount, and $\mathbb{E}$ represents the expected value of a finite number of random outcomes. The reward discount is applied to incentivise actions made sooner rather than those made later. Equation (3.1) is also known as the Bellman equation [64]. The applications of deep-Q networks for control are limited in that they can only generate an action function for systems with discrete action, and state spaces cannot be used for continuous control applications. To overcome these limitations, the deep-Q network is combined with an actor-critic method such as the deterministic policy gradient algorithm. An actor-critic algorithm uses two parallel neural networks (NNs): the actor, $u(s|\psi^u)$, which has NN parameters represented by ψ^u, and the critic NN, $Q(s,a|\psi^Q)$, where ψ^Qrepresent the NN parameters [63]. The actor NN takes the current state as inputs and generates an action output. The critic NN takes the current state and the actor action as inputs, and outputs a reward value referred to as a Q-value. Equation ((3.2)) indicates that the actor is updated by applying the chain rule to the expected return from the start distribution, J.

$$\begin{aligned} \nabla_{\psi_u} J &\approx \mathbb{E}\left[\nabla_{\psi^u} Q(s,a|\psi^Q)|_{s=s_t, a=u(s_t|\psi^u)}\right] \\ &= \mathbb{E}[\nabla_a Q(s,a|\psi^Q)|_{s=s_t, a=u(s_t)} \nabla_{\psi^u} u(s|\psi^u)|_{s=s_t}] \end{aligned} \tag{3.2}$$

where ∇_{ψ^u} and ∇_{ψ^Q} represent the gradients of the parameters for the actor and critic NNs respectively, and ∇_a is the gradient of the action output from the actor NN.

Equation (3.2) has been proven to be the gradient of the policy's performance, or the policy gradient [49]. The deterministic policy gradient integrates over the state space only, using much less computational power, and yielding the expected gradient of the approximated action-value function, represented by the critic NN. The DDPG algorithm is an integration of deterministic policy gradient algorithm with deep-Q networks; allowing the DDPG to learn in continuous action and state spaces, while taking advantage of the deep-Q network's ability to approximate non-linear functions as NNs.

3.2 Actor and Critic Networks

In reinforcement learning, the optimal action-value function, or the Q-function, represents the expected cumulative reward of taking a particular action for a given state and following an optimal policy thereafter. While the Q-function can be directly optimized to find the optimal policy, it can be computationally intensive or infeasible for large state and action spaces [65]. The critic NN is introduced to the DDPG technique to estimate the value function of a state-action pair, which can provide an approximation of the expected cumulative reward that an agent can achieve. The output of the critic NN is used by the agent's policy to update its actor NN. Specifically, the actor NN is updated in the direction that increases the expected cumulative reward, as estimated by the critic NN. This is done by minimizing the loss function given by:

$$L(\psi^Q) = \mathbb{E}[(Q(s_t, a_t) - y_t|\psi^Q)^2] \tag{3.3}$$

where $L(\psi^Q)$ represents the expected loss of the output of the critic $Q(s_t, a_t)$, at the current state s_t and action a_t, parameterized by ψ^Q.

y_t can be determined by:

$$y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, (u(s_{t+1})|\psi^u)|\psi^Q) \tag{3.4}$$

where $r(s_t, a_t)$ is the returned reward at the current state s_t and action a_t, γ is the discount factor for future rewards, and $Q(s_{t+1}, u(s_{t+1}|\psi^u)|\psi^Q)$ is the output given by the critic NN parameterized by ψ^Q at the next state s_{t+1} and at the action produced by the actor NN $(u(s_{t+1})|\psi^u)$ parameterized by ψ^u at the next state s_{t+1}.

By optimizing the value function, the agent can indirectly optimize its policy without having to compute the optimal action-value function directly [42]. In the DDPG technique this

value function is approximated by the critic NN and is optimized through the training process. However, reinforcement learning NNs have limitations such as slow divergence and instability when approximating non-linear functions such as Q-functions [66]. This can be corrected with two methods using off-policy training and the implementation of experience replay [67], as discussed in the following section

3.2 Implementation of the DDPG Algorithm

3.2.1 Off-Policy Training

Off-policy training uses additional NNs known as target networks. Target networks are initially created as copies of the actor and critic NNs. These target NNs are updated over each time step; the actor and critic NNs are updated using the error from the output of the target NNs by [68]:

$$\psi' \leftarrow \tau\psi + (1-\tau)\psi' \tag{3.5}$$

where ψ represents the parameters of either target network and τ is the update rate of the target networks. $\tau \subset (0, 1)$ is a typically small parameter as to change the targets slowly each time step to improve system stability when estimating non-linear functions.

The off-policy training can improve stability because the learning does not depend on the current policy and is also isolated from policy fluctuations. Furthermore, off-policy training allows for more efficient exploration of the state-action space. On-policy learning has limitations in exploring actions that are consistent with the current policy, which can lead to an inefficient search for optimal behavior. Conversely, off-policy learning can learn from experiences outside of the currently learned policy and facilitate the training operation.

3.2.2 Experience Replay

Experience replay is an algorithm that stores and reuses past experiences or transitions of an agent from its actions with an environment; it can break the temporal correlations between consecutive experiences and reuse them for learning. Inclusion of experience replay has been demonstrated to improve stability and efficiency in reinforcement learning [69]. Experience replay stores a buffer of previous states, actions, and rewards, as represented by:

$$\varkappa_t := (s_t, a_t, r_t, s_{t+1}) \tag{3.6}$$

where $\varkappa_t$ represents an experience replay sample at time step t, which includes the current state, action, and returned reward s_t, a_t, r_t, respectively as well as the next state s_{t+1}.

For each time step, the state, action, reward, and next state are stored in a large buffer. During training this buffer is sampled randomly in batches to avoid overfitting to the most recent experiences. Typically, the batching method employed is called minibatching, which is not the full buffer but rather a sample of N transitions. This makes it more computationally efficient and introduces more diversity to the training process [70]. Once the buffer is full, the oldest sample is replaced by the newest after each sample. Using off-policy training for the DDPG, the buffer will be large, and uniformly sample batches from this buffer allow learning across a set of uncorrelated transitions. The buffer acts similarly to training data but, rather than being labeled and provided to the algorithm, it is created and stored as the actor takes actions within the environment [66, 67]. This does not require external training data, but only an environment the agent can act in.

3.2.3 Gradient Descent using the Adaptive Moment Estimation Optimization Algorithm

The Adaptive Moment Estimation (ADAM) algorithm is a stochastic gradient descent algorithm that requires only the first order gradients and has a low memory requirement. It computes adaptive learning rates for all parameters, while keeping an exponentially decaying average of both the first and second order gradients [67]. ADAM has been applied for many reinforcement learning applications [68].

The process of the ADAM algorithm begins with initializing the first moment vector $\widehat{m}_0 \leftarrow 0$, the second moment vector $\hat{v}_0 \leftarrow 0$, and time step $t = 0$. The function $f(\psi)$ is assumed to be stochastic and differentiable with respect to its parameters. In this case because we use randomly sampled minibatches, the stochasticity requirement is satisfied. Equation (3.7) gives the gradient ∇_t, which is a vector of partial derivatives of the output of the function at time step t.

$$\nabla_t = \nabla_\psi f_t(\psi) \tag{3.7}$$

where $f(\psi)$ represents the generalized objective function to be minimized and ψ is the generalized NN parameters. This is a generalized form, for example to find the gradient of the actor output ∇_a, the equation would take the form given by:

$$\nabla_a = \nabla_{\psi^u}(u(s_{t+1})|\psi^u) \tag{3.8}$$

where ∇_{ψ^u} is the gradient of the parameters for the actor NN, $(u(s_{t+1})|\psi^u)$ is the action produced by the actor NN, parameterized by ψ^u at the next state s_{t+1}.

Next, the exponential moving averages of the gradient $\widehat{m}_t$, and the squared gradient $\hat{v}_t$, are updated as in Equation (3.9).

$$\widehat{m}_t = \beta_1 \widehat{m}_{t-1} + g_t(1-\beta_1) \tag{3.9}$$
$$\hat{v}_t = \beta_2 \widehat{m}_{t-1} + g_t(1-\beta_2)^2$$

where, β_1 and β_2 are hyperparameters; by trial and error β_1 is set to 0.9 and β_2 is set to 0.999.

The moving averages $\widehat{m}_t$ and $\hat{v}_t$ estimate the mean and the uncentered variance of the parameter gradient, respectively. This tracks past gradient behaviour for each parameter and allows for the parameter learning rate to be adjusted accordingly. A noisy gradient may require a low learning rate to ensure excessively large steps are not taken, whereas lower gradient variance for a parameter can enable a higher learning rate to achieve faster convergence. As they are initialized as vectors of zeros, the estimates contain biases corrected by:

$$\widehat{m}_t{}' = \frac{\widehat{m}_t}{(1-\beta_1)} \tag{3.10}$$

$$\hat{v}_t{}' = \frac{\hat{v}_t}{(1-\beta_2)} \tag{3.11}$$

where $\widehat{m}_t{}'$ and $\widehat{v}_t{}'$ are the respective bias corrected moving averages. Then, the parameters are updated by:

$$\psi_t = \psi_{t-1} - \frac{\varrho \hat{m}_t'}{\sqrt{\hat{v}_t'} + \varsigma} \tag{3.12}$$

where ϱ represents the learning rate and ς is a small constant selected to avoid division by zero, respectively.

Updating with the ADAM algorithm is performed at every time step of the training process, leading to convergence of the agent's policy over the course of training.

3.3 DDPG Training

3.3.1 Hyperparameters

Hyperparameters are parameters that are not updated during the training process of the reinforcement learning model but are instead set by the user prior to training. Hyperparameters can have a significant impact on the performance and convergence of a model, and as such, it is important to choose appropriate values for them. The hyperparameters used in this work are listed in Table .

Table 3.1: Hyperparameters for a DDPG algorithm

Symbol	Hyperparameter Description
γ	Discount factor for future rewards
τ	Target learning rate
ϕ^u, ϕ^Q	Actor and critic learning rates, respectively
μ, σ	Action noise hyperparameters, mean and standard deviation, respectively
b	Replay buffer size
N_E	Number of training episodes
ε_k	State observation noise scaling constant, $k = 1,2,\ldots,n$ where n is the number of inputs into the actor network
$\varrho, \beta_1, \beta_2$	Step size, parameters for use in ADAM gradient descent

Selecting the appropriate hyperparameters can be a challenging task. Manual tuning by trial and error will be employed. The selection of these specific hyperparameter in this work will be discussed in Chapter 4. The respective actor and critic learning rates, $\phi^\mu$and ϕ^Q, can have a

great effect on the learning process; a larger learning rate ϕ can lead to faster convergence, but increase the risk of overshooting and instability.

3.3.2 DDPG Algorithm

To begin the DDPG technique the experience replay is initialized as a large buffer $\mathbb{B}$, the actor, $u(s|\psi^u)$ and the critic, $Q(s,a|\psi^Q)$, are randomly initialized with weights ψ^u and ψ^Q. Next, the target actor u', and the target critic Q', are initialized as copies of the actor and critic networks as shown:

$$\psi^{Q'} \leftarrow \psi^Q, \qquad \psi^{\mu'} \leftarrow \psi^{\mu} \tag{3.13}$$

where $\psi^{Q'}$ and $\psi^{\mu'}$ are the parameters of the target critic NN and target actor NN, respectively.

Ensuring the agent can effectively explore the action space is a challenge in reinforcement learning for continuous problems [75]. The actor policy is modified to include a noise process to aid in exploration, as represented by Equation (3.14), where the noise component [76] is represented by $\mathcal{N}$ that is chosen based on the application. For this work, a normal distribution will be used to provide random action noise, as represented in Equation (3.15), where σ, is the standard deviation and $\bar{x}$, is the mean.

$$\mu'(s_t) = \mu\left(s_t|\psi_t^{\mu}\right) + \mathcal{N}_t \tag{3.14}$$

$$\mathcal{N}_t = \mathcal{N}(x,\sigma,\bar{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\bar{x})^2}{2\sigma^2}} \tag{3.15}$$

The following procedures are performed by the algorithm for each t^{th} time step:

1. Initialize state observation s_1. Select an action a_t, using Equation ((3.14)) to take in the environment from the current policy.
2. Perform action a_t in the environment and observe new state s_{t+1}, and reward r_t.
3. Store the transition $\varkappa_t$, in the replay buffer $\mathbb{B}$; if the buffer is full, overwrite the oldest transition.
4. Take a random mini-batch sample of N transitions, $\varkappa_i := (s_i, a_i, r_i, s_{i+1})$ from $\mathbb{B}$.
5. Solve for y_i using Equation ((3.16)) and compute the loss.

$$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\psi^{\mu'})|\psi^{Q'}) \tag{3.16}$$

where r_i is the reward at the i^{th} time step; γ is the discount factor; and $Q'(s_{i+1}, \mu'(s_{i+1}))$ is the Q-value given by the target critic at the next state as well as the next action at that state. This is performed in a matrix operation for all N sampled transitions from the batch and used to find the mean squared loss such that:

$$L_Q = \frac{1}{N}\sum_i \left(y_i - Q(s_i, a_i|\psi^Q)\right)^2, i = 1,2, \dots, N \tag{3.17}$$

where L_Q represents the loss function of the critic and N is the mini-batch sample size.

6. Update the critic by minimizing the loss L_Q, using the ADAM algorithm as described in Equations (3.7) - (3.12)
7. Update the actor policy with policy gradient from the N samples, using the ADAM algorithm as in Equations (3.7) - (3.12)
8. Update the target networks using Equation ((3.5).

These steps are repeated until the specified number of episodes has been completed for training. An episode is defined as a sequence of states, actions, and rewards that occur when the agent interacts with the environment until a terminal state is reached [77]. Each episode starts with an initial state and progresses through the training process until it reaches a defined terminal point. Generally, the terminal point is a fixed number of steps and/or a condition. Once an episode is completed, the next episode begins, by resetting the environment. Once all episodes are completed the training is finished. The critic is not necessary when training is not being performed; once trained the actor can be used as a state-action map for control.

3.4 Bridging the Sim-to-Real Gap

The sim-to-real gap problem in reinforcement learning arises when an agent trained in a simulated environment fails to transfer its learned policy to a real-world environment [73, 74]. This is a common problem, especially for robotics applications, where the agent is trained in a simulated environment before being deployed on a robot [75, 76]. This problem is caused by several factors, including differences between the simulated and real-world environments, modeling errors, and sensory and actuation discrepancies. Simulated environments may not

include all the relevant factors that affect the performance of the agent. For example, the model of the flexible beam in this work may not perfectly match the real dynamics of the flexible beam in the lab. As a result, the policy learned in a simulated environment may not be robust enough to handle the variations and uncertainties present in the real world, which will lead to poor performance or failure of the agent. To address this problem, researchers have suggested some techniques for closing the sim-to-real gap, such as domain randomization [55, 77], as discussed in the following subsection.

3.4.1 Domain Randomization

Domain randomization involves introducing random variations into the simulated environment during training to make the agent more robust to variations in the real-world environment. In this work this is done in the form of state observation noise, the perturbed state can be realized as:

$$s_t' \leftarrow \varepsilon_k s_t * (\mathcal{N}|_{\mu=0,\sigma=1}) \tag{3.18}$$

where s_t' is the perturbed value of state s_t, ε_k is the scaling constant for the k^{th} input, and $\mathcal{N}|_{\mu=0,\sigma=1}$, is the random output of a normal distribution with mean, $\mu = 0$ and standard deviation, $\sigma = 1$. These hyperparameters scale the noise to a size compatible with the domain of each observed input.

During training, after 50% of the episodes are complete, the domain randomization begins to perturb the state observations of the actor network before an action is taken. This will increase in intensity for another 25% of the episodes before reaching a maximum intensity at 75% of the training cycles. Then the training continues while the actor will undertake maximal state observation noise for the remaining episodes. The addition of domain randomization will allow the controller to operate on the physical apparatus more effectively. Details will be discussed in Chapter 4.

Chapter 4 – Experimental Setup and System Modeling

The primary objective of this work is to propose a novel control technique that can stabilize a refueling drogue when deployed during an AAR mission. Since access to real aircraft and equipment is limited for research and development (R&D) purposes, the control research will be undertaken using an apparatus that can simulate a similar dynamic response to that of a refueling drogue. For this reason, the Smart Structure experimental setup will be modified and used for this R&D work. This smart structure workstation could be a good approximation for a hose-drogue in 2D space and can be used to validate the performance of the proposed DDPG controller. To train a reinforcement learning agent this structure, a simulation is created. The equations of motions of the smart structure are derived and used to create a set of state-space equations. These state space equations are used to simulate the dynamics of the smart structure, where an agent can be trained, which will be discussed in detailed in the following sections.

4.1 Experimental Setup

Figure 4.1 shows the flexible beam experimental setup, or smart structure, used in this work. The tested flexible beam is a 1.5mm thick 110mm by 440 mm steel beam, which is clamped to the base. Another end of the flexible beam is connected with a rigid bar system. The shaft is driven by a servo motor mounted at the top of the link through a 70:1 gear ratio. This motor is powered by a power supply unit controlled by a computer. An encoder on the shaft has a 1024 count disc, which allows for a measurement the angular position of the shaft of θ with a resolution of 4096 counts/rev via a quadrature. Strain gauges are glued at the base of the flexible beam to measure the deflection of the top of the flexible beam x_b. The strain gauges are calibrated to 1 volt per 2.54 cm. The data from the encoder and the strain gauges is transmitted through the power supply unit into the data acquisition (DAQ) board connected to the computer. This structure will be used to approximate the hose-drogue system in 2D space as well as to validate the performance of the proposed DDPG control technique. The equations of motions of the smart structure are derived and used for state-space modeling and simulation of the dynamics of the smart structure, where an agent can be trained.

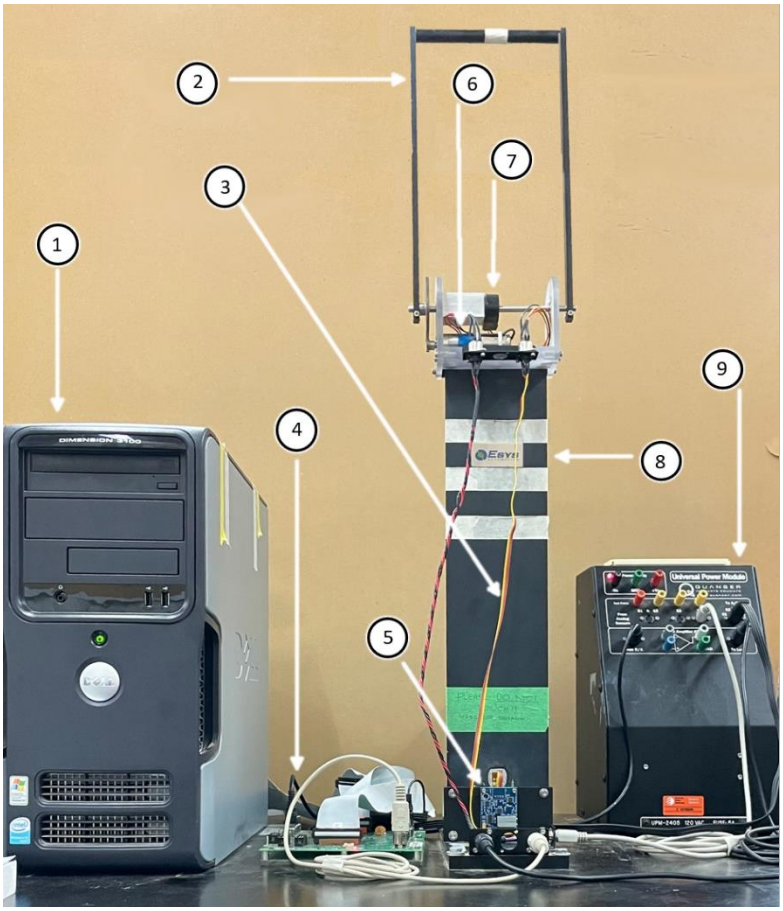


Figure 4.1: The smart structure workstation: (1) computer with to DAQ card. (2) rigid bar. (3) flexible beam. (4) Quanser DAQ board. (5) strain gauge signal conditioning board. (6) DC motor and gearbox. (7) rotary encoder. (8) mass block. (9) Quanser universal power module.

4.2 Smart Structure State-Space Modeling

4.2.1 Equations of Motion

To develop a model of the smart structure, the non-linear equations of motion are derived using the Euler-Lagrange method [83]. Firstly, a free body diagram is created as illustrated in Figure 4.2. The flexible beam has length l_b and is secured at its base, standing upright. The motor with mass m_m is attached at its free end. The motor drives the rigid bar through a gearbox, and the rigid bar can be modeled as a simple pendulum with length l_r and mass m_r. The deflection of the flexible beam is denoted by x_b where $x_b = 0$ when the flexible beam is completely upright and experiences no deflection.

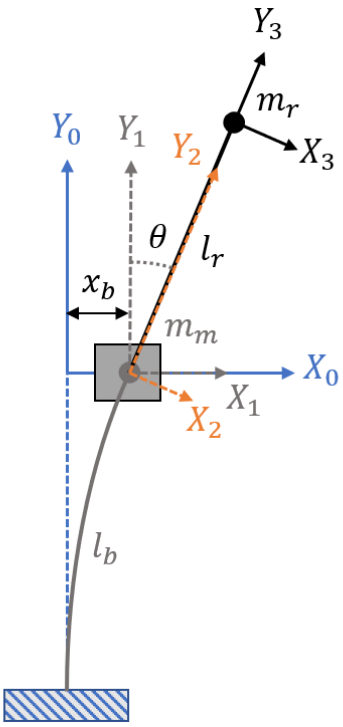


Figure 4.2: Free-body diagram of the smart structure

In modeling, a generalized coordinate system, also referred to as Lagrangian coordinates, is established and represented by:

$$s_t = [x_b(t), \theta(t), \dot{x}_b(t), \dot{\theta}(t)] \tag{4.1}$$

where s_t represents $x_b(t)$, the displacement of the motor at the top of the flexible beam, and $\theta(t)$, the angular position of the rigid bar, both at time moment t, as well as their derivatives $\dot{x}_b(t)$ and $\dot{\theta}(t)$.

It is assumed that $\theta(t)$ is changing positively when the rigid bar is moving clockwise when viewed from the left side, as shown in Figure 4.2. The first derivative of s_t with respect to time t is given by:

$$\dot{s}_t = [\dot{x}_b(t), \dot{\theta}(t), \ddot{x}_b(t), \ddot{\theta}(t)] \tag{4.2}$$

To simplify representation, we can drop time constant t from $x_b(t)$ and $\theta(t)$, which are simplified as x_b and θ. The translation matrix from the initial position to the point of deflection

at time t is represented by $T_{0,1}^{T}$ in Equation (4.3). Equation (4.4) is the rotation matrix for the rigid beam from an upright position, to the free end of the beam. Equation (4.5) is the translation matrix from the motor to the free end of the rigid beam.

$$T_{0,1}^{T} = \begin{bmatrix} 1 & 0 & 0 & x_b \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

$$T_{1,2}^{R} = \begin{bmatrix} cos\ (\theta) & 0 & sin\ (\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -sin\ (\theta) & 0 & cos\ (\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.4}$$

$$T_{2,3}^{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_r \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

The transformation matrix from the initial position to the free end of the rigid bar can be represented by substituting Equations (4.3), (4.4), and (4.5) into Equation (4.6).

$$T_{0,3} = T_{0,1}^{T} T_{1,2}^{R} T_{2,3}^{T} =$$

$$\begin{bmatrix} 1 & 0 & 0 & x_b \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_r \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\ (\theta) & 0 & sin\ (\theta) & sin\ (\theta) l_r \\ 0 & 1 & 0 & 0 \\ -sin\ (\theta) & 0 & cos\ (\theta) & cos\ (\theta) l_r \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

$$= \begin{bmatrix} cos\ (\theta) & 0 & sin\ (\theta) & sin\ (\theta) l_r + x_b \\ 0 & 1 & 0 & 0 \\ -sin\ (\theta) & 0 & cos\ (\theta) & cos\ (\theta) l_r \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The three-dimensional Cartesian coordinates of the free end of the rigid bar can be represented by:

$$\begin{aligned} x_r &= \sin(\theta)\, l_r + x_b \\ y_r &= 0 \\ z_r &= \cos(\theta)\, l_r \end{aligned} \tag{4.7}$$

where l_r is the length of the rigid beam.

Equations (4.7) are derived by the following representation:

$$\begin{aligned} \dot{x}_r &= \dot{\theta}\cos(\theta)\, l_r + \dot{x}_b \\ \dot{y}_r &= 0 \\ \dot{z}_r &= -\dot{\theta}\sin(\theta)\, l_r \end{aligned} \tag{4.8}$$

By the use of the Euler-Lagrange method, the Lagrangian, L_a is will be:

$$L_a = K - V \tag{4.9}$$

where K represents the total kinetic energy of the system; V is the total potential energy of the system represented by:

$$V = V_E + V_G \tag{4.10}$$

where, V_E is the elastic potential energy of the system of the flexible beam, determined by Equation (4.11); V_G is the gravitational potential energy of the system found with Equation (4.12):

$$V_E = \frac{K_s {x_b}^2}{2} \tag{4.11}$$

$$V_G = m_r g\, cos\ (\theta) l_r \tag{4.12}$$

where g is the gravitational acceleration constant and K_s is the spring constant of the flexible beam. The natural frequency of the beam in rad/sec can be represented by:

$$\omega_n = \sqrt{\frac{K_s}{m_m + m_r}} \tag{4.13}$$

where m_m is the mass of the motor mounted to the free end of the flexible beam and m_r is the mass of the rigid bar.

In general, the natural frequency ω_n can be determined experimentally. Then the spring constant of the flexible beam K_s can be estimated by:

$$K_s = \omega_n^2 (m_m + m_b) \tag{4.14}$$

then, the total potential energy becomes:

$$V = \frac{K_s {x_b}^2}{2} + m_r g \cos(\theta) l_r \tag{4.15}$$

The total kinetic energy of the system will be:

$$K = K_R + K_{T1} + K_{T2} \tag{4.16}$$

where K_R is the rotational kinetic energy, K_{T1} is the translational kinetic energy of the motor and gearbox, and K_{T2} is the translational kinetic energy of the flexible beam, which are determined by Equations (4.17), (4.18), and (4.19), respectively:

$$K_R = \frac{I_r \theta^2}{2} \tag{4.17}$$

$$K_{T1} = \frac{m_m {\dot{x}_b}^2}{2} \tag{4.18}$$

$$\begin{aligned} K_{T2} &= \frac{m_r v_r^2}{2} = \frac{m_r}{2}\sqrt{\dot{x}_r^2 + \dot{y}_r^2 + \dot{z}_r^2} \\ &= \frac{m_r}{2}\left[\left[\dot{\theta}\cos(\theta)\, l_r + \dot{x}_b\right]^2 + \left[\dot{\theta}\sin(\theta)\, l_r\right]^2\right] \end{aligned} \tag{4.19}$$

where I_r is the rotational moment of inertia of the rigid bar, $v_{r,}$ is the velocity of the free end of the rigid bar, and $\dot{x}_r, \dot{y}_r, \dot{z}_r$ are the velocities of the free end of the rigid in the Cartesian coordinate system.

The total kinetic energy can be determined by:

$$K = \frac{J_r \dot{\theta}^2}{2} + \frac{m_m {\dot{x}_b}^2}{2} + \frac{m_r}{2}\left[\left[\dot{\theta}\cos(\theta)\, l_r + \dot{x}_b\right]^2 + \left[\dot{\theta}\sin(\theta)\, l_r\right]^2\right] \tag{4.20}$$

In a N degree of freedom system Lagrange's Equations [83] l can be represented by:

$$\frac{\partial}{\partial t}\left(\frac{\partial}{\partial \dot{s}_i} L\right) - \left(\frac{\partial}{\partial s_i} L\right) = Q_i \tag{4.21}$$

where Q_i are the generalized forces acting on each coordinate system, $i = 1, 2, \ldots, N_c$ and N_C is the number of coordinate systems used to represent the system; $N_c = 2$ in this case. Then Lagrange's equations for this system become:

$$\begin{aligned}\frac{\partial}{\partial t}\left(\frac{\partial}{\partial \dot{x}_b}K - \frac{\partial}{\partial \dot{x}_b}V\right) - \left(\frac{\partial}{\partial x_b}K - \frac{\partial}{\partial x_b}V\right) &= Q_1 \\ \frac{\partial}{\partial t}\left(\frac{\partial}{\partial \dot{\theta}}K - \frac{\partial}{\partial \dot{\theta}}V\right) - \left(\frac{\partial}{\partial \theta}K - \frac{\partial}{\partial \theta}V\right) &= Q_2\end{aligned} \tag{4.22}$$

where Q_1 and Q_2, are the generalized forces, a summation of external forces acting on the system in each of the two coordinate systems.

Considering,

$$\frac{\partial}{\partial \dot{x}_b}V = 0, \qquad \frac{\partial}{\partial \dot{\theta}}V = 0 \tag{4.23}$$

Equation (4.20) can be simplified as:

$$\begin{aligned}\frac{\partial}{\partial t}\left(\frac{\partial}{\partial \dot{x}_b}K\right) - \left(\frac{\partial}{\partial x_b}K\right) + \left(\frac{\partial}{\partial x_b}V\right) &= Q_1 \\ \frac{\partial}{\partial t}\left(\frac{\partial}{\partial \dot{\theta}}K\right) - \left(\frac{\partial}{\partial \theta}K\right) + \left(\frac{\partial}{\partial \theta}V\right) &= Q_2\end{aligned} \tag{4.24}$$

The generalized forces Q_1 and Q_2, are represented by:

$$\begin{aligned}Q_1 &= B_b\dot{x}_b \\ Q_2 &= \tau_m - B_r\dot{\theta}\end{aligned} \tag{4.25}$$

where B_b is the viscous friction torque coefficient for the flexible beam; B_r is the viscous friction torque coefficient for the rigid bar. These coefficients are assumed to be zero because air resistance is negligible and does not affect the dynamics of the system meaningfully.

The torque produced by the motor τ_m can be estimated by:

$$\tau_m = \frac{\eta_g K_g \eta_m K_t\left(-K_g K_m \dot{\theta} + V_m\right)}{R_m} \tag{4.26}$$

where related motor parameters are summarized in Table 4.1.

Table 4.1: Motor parameters [84]

Symbol	Parameter	Value
η_g	Gearbox efficiency	$0.9 \pm 10\%$
K_g	Gearbox ratio	70:1
η_m	Motor efficiency	$0.69 \pm 5\%$
K_t	Motor torque constant	$7.68(10)^{-3}\ N \cdot m$
V_m	Motor applied voltage	[-4, 4] Volts
R_m	Motor armature resistance	2.6 Ω
K_m	Back-emf constant	$7.68(10)^{-3}\ \frac{V}{rad/s}$

The external force vector can be represented as [85]:

$$Q = \left[0, \frac{\eta_g K_g \eta_m K_t \left(-K_g K_m \dot{\theta} + V_m\right)}{R_m}\right] \tag{4.27}$$

where the motor parameters are listed in the Table 4.1.

Substitute Equations (4.15), (4.20), and (4.27) into Equation (4.24). By manipulation the equations of motion in Equations (4.28) and (4.29) can be obtained:

$$(m_m + m_r)\ddot{x}_b + m_r l_r \cos(\theta)\,\ddot{\theta} - m_r l_r \sin(\theta)\,\dot{\theta}^2 + K_s x_b = 0 \tag{4.28}$$

$$\begin{aligned}(J_r + m_r l_r^2)\ddot{\theta} + \ & m_r \cos(\theta)\, l_r \ddot{x}_b - g m_r \sin(\theta)\, l_r \\ &= \frac{\eta_g K_g \eta_m K_t}{R_m} V_m - \frac{\eta_g K_g \eta_m K_t K_g K_m}{R_m}\dot{\theta}\end{aligned} \tag{4.29}$$

Assume small amplitude oscillations, or:

$$\sin(\theta) \approx \theta,\ \ \cos(\theta) \approx 1,\ \ \theta^2 = 0 \tag{4.30}$$

The equations of motion in Equations (4.28) and (4.29) can be linearized and simplified as:

$$(m_m + m_r)\ddot{x}_b + m_r l_r \ddot{\theta} + K_s x_b = 0 \tag{4.31}$$

$$(J_r + m_r l_r^2)\ddot{\theta} + m_r l_r \ddot{x}_b = \frac{\eta_g K_g \eta_m K_t}{R_m} V_m - \frac{\eta_g K_g \eta_m K_t K_g K_m}{R_m} \dot{\theta} \tag{4.32}$$

4.3 State Space Equations

The linear state space equations can be written as:

$$\begin{aligned} \dot{s} &= As + Bu \\ r &= Cs + Du \end{aligned} \tag{4.33}$$

where u is the control input, s represents the state space variables, and $A, B, C,$ and D are state space matrices.

$$s = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} = \begin{bmatrix} x_b \\ \theta \\ \dot{x}_b \\ \dot{\theta} \end{bmatrix} \tag{4.34}$$

$$\dot{s} = \begin{bmatrix} \dot{s}_1 \\ \dot{s}_2 \\ \dot{s}_3 \\ \dot{s}_4 \end{bmatrix} = \begin{bmatrix} \dot{x}_b \\ \dot{\theta} \\ \ddot{x}_b \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} s_3 \\ s_4 \\ \dot{s}_3 \\ \dot{s}_4 \end{bmatrix} \tag{4.35}$$

here, Equations (4.31) and (4.32) are rearranged and substituted into Equations (4.34) and (4.35) and then isolated for $\dot{s}_3$ and $\dot{s}_4$:

$$\begin{aligned} \dot{s}_3 = &-\frac{(J_r + m_r l_r^2)K_s}{J_r(m_m + m_r) + l_r^2 m_r m_m} s_1 - \frac{g l_r^2 m_r^2}{J_r(m_m + m_r) + l_r^2 m_r m_m} s_2 \\ &+ \frac{K_g^2 K_m K_t \eta_g \eta_m l_r m_r}{(J_r(m_m + m_r) + l_r^2 m_r m_m)R_m} s_4 - \frac{K_g K_t V_m \eta_g \eta_m l_r m_r}{(J_r(m_m + m_r) + l_r^2 m_r m_m)R_m} \end{aligned} \tag{4.36}$$

$$\begin{aligned} \dot{s}_4 = &\frac{m_r l_r K_s}{J_r(m_m + m_r) + l_r^2 m_r m_m} s_1 + \frac{g m_r l_r (m_m + m_r)}{(J_r(m_m + m_r) + l_r^2 m_r m_m)R_m} s_2 \\ &- \frac{K_g^2 K_m K_t \eta_g \eta_m l_r (m_m + m_r)}{(J_r(m_m + m_r) + l_r^2 m_r m_m)R_m} s_4 + \frac{K_g K_t V_m \eta_g \eta_m l_r (m_m + m_r)}{(J_r(m_m + m_r) + l_r^2 m_r m_m)R_m} \end{aligned} \tag{4.37}$$

The state space matrices $A, B, C,$ and D can be defined as

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\dfrac{(J_r + m_r l_r^2)K_s}{J_r(m_m + m_r) + l_r^2 m_r m_m} & -\dfrac{g l_r^2 m_r^2}{J_r(m_m + m_r) + l_r^2 m_r m_m} & 0 & \dfrac{K_g^2 K_m K_t \eta_g \eta_m l_r m_r}{(J_r(m_m + m_r) + l_r^2 m_r m_m)R_m} \\ \dfrac{m_r l_r K_s}{J_r(m_m + m_r) + l_r^2 m_r m_m} & \dfrac{g m_r l_r (m_m + m_r)}{(J_r(m_m + m_r) + l_r^2 m_r m_m)R_m} & 0 & -\dfrac{K_g^2 K_m K_t \eta_g \eta_m l_r (m_m + m_r)}{(J_r(m_m + m_r) + l_r^2 m_r m_m)R_m} \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ -\dfrac{K_g K_t V_m \eta_g \eta_m l_r m_r}{(J_r(m_m + m_r) + l_r^2 m_r m_m)R_m} \\ \dfrac{K_g K_t V_m \eta_g \eta_m l_r (m_m + m_r)}{(J_r(m_m + m_r) + l_r^2 m_r m_m)R_m} \end{bmatrix} \tag{4.38}$$

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

$$D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Using the system parameters as listed in Table 4.1, the state space matrices can be represented as:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -69.4440 & -0.4007 & 0 & 0.1932 \\ 121.3376 & 17.8410 & 0 & -8.5994 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ -0.3594 \\ 16.0008 \end{bmatrix} \tag{4.39}$$

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

$$D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

The state space model can now be used to create a simulation environment to train a reinforcement learning agent.

4.3 Classical and Intelligent Controllers

4.3.1 PD Controller

The PD controller is used to evaluate the performance of the proposed DDPG control technique. The PD controller is selected a baseline for controller performance, PD is a robust and reliable control technique and results from the PD test will be used to verify the robustness of the DDPG under varied dynamic conditions. As reinforcement learning is a relatively new area of research a comparison with a classical and well understood control method such as PD can demonstrate the new potential of reinforcement learning control.

The desired output for the PD controller is given by:

$$u = -\left(k_{x_b}^p x_b + k_\theta^p \theta + k_x^d \dot{x}_b + k_\theta^d \dot{\theta}\right) \tag{4.40}$$

where u is the output control voltage; $k_{x_b}^p$ and k_θ^p are the proportional gains for x_b and θ, respectively; $k_{x_b}^d$ and k_θ^d are the derivative gains for $\dot{x}_b$ and $\dot{\theta}$, respectively.

A feedback controller using the linear quadratic regulator method will be designed, using the state space matrices A, B, C, and D in Equations (4.41) and (4.42). A set of weight matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ are used to determine the gains of the controller:

$$\boldsymbol{Q} = \begin{bmatrix} 8800 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 \\ 0 & 0 & 1.2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \boldsymbol{R} = [1] \tag{4.41}$$

The control gains can be computed by:

$$k = \begin{bmatrix} k_{x_b}^p \\ k_\theta^p \\ k_x^d \\ k_\theta^d \end{bmatrix} = \begin{bmatrix} -83.9958 \\ 8.2734 \\ 2.8734 \\ 1.1269 \end{bmatrix} \tag{4.42}$$

The designed PD controller can be represented as:

$$\boldsymbol{u} = -\left(-83.9958 x_b + 8.2734\theta + 2.8734\dot{x}_b + 1.1269\dot{\theta}\right) \tag{4.43}$$

4.3.2 NF Controller

The MATLAB Fuzzy Logic toolbox is used to design and train a NF controller for comparison. The NF controller is selected for comparison as it is a common machine learning

technique that is not in a reinforcement learning technique but rather it is a supervised learning technique, which requires supplied training data. Optimization using training data improves the performance of the NF controller, but properly capturing and selecting this data is a challenge. Incorrectly chosen training data can possibly leading to improved performance in some areas and losses in others, this problem is alleviated by the DDPG as it does not require supply of training data. PD control tests using the controller as described in section 4.3.1 are used to capture the training data for the proposed NF controller. Results are used to investigate the potential of the DDPG controller as a machine learning control technique for stabilization.

Figure 4.3 shows the NN architecture of the NF system. It has 5 layers:

Layer 1 is the input layer. The NF system has 4 input variables corresponding to each of the state space variables, $s_n = [x_b, \theta, \dot{x}_b, \dot{\theta}] = [s_1, s_2, \dots, s_N]$, where s_n is any of the 4 input variables, $n = 1, 2, \dots, N_s$ and $N_s = 4$.

Layer 2 is the fuzzification layer. Here each input variable is fuzzified using membership functions (MFs) as given by:

$$g(s_n) = e^{-\frac{(s_n - c_\lambda)^2}{2a_\lambda^2}} \tag{4.44}$$

where c_λ is the mean, and a_λ is the standard deviation, $\lambda = 1, 2, \dots, 10$; these are the non-linear parameters to be trained.

The positional inputs x_b and θ, are each defuzzied by 3 MFs, each representing the positive, negative, or neutral positions. The rate inputs $\dot{x}_b$ and $\dot{\theta}$, are each defuzzied by 2 MFs each representing positive and negative rates of change. This leads to a set of 36 fuzzy rules in layer 3.

Layer 3 is the fuzzy operation layer. Here, firing strengths of each fuzzy rule are computed using a T-norm operator as in:

$$W_h = g_\lambda(s_1) \times g_\lambda(s_2) \times g_\lambda(s_3) \times g_\lambda(s_4) \tag{4.45}$$

where W_h is the hth fuzzy rule and $h = 1, 2, \dots, 36$.

Layer 4 is the normalization layer. Firing strength output of each fuzzy rule is normalized such that all outputs add to 1.

$$W_h' = \frac{W_h}{\sum W} \tag{4.46}$$

Layer 5 is the defuzzification layer. Centroid defuzzification is performed in order to determine network output by:

$$u_h = W_h'\left(p_{h_1}s_1 + p_{h_2}s_2 + p_{h_3}s_3 + p_{h_4}s_4 + p_{h_5}\right) \tag{4.47}$$

where u_h is the defuzzied output of rule h and p_{h_n} represents the linear parameters of each rule.

Layer 6 is the output layer. The control action is taken as the sum of the outputs from each fuzzy rule as given by:

$$u = \sum_{h=1}^{36} u_h \tag{4.48}$$

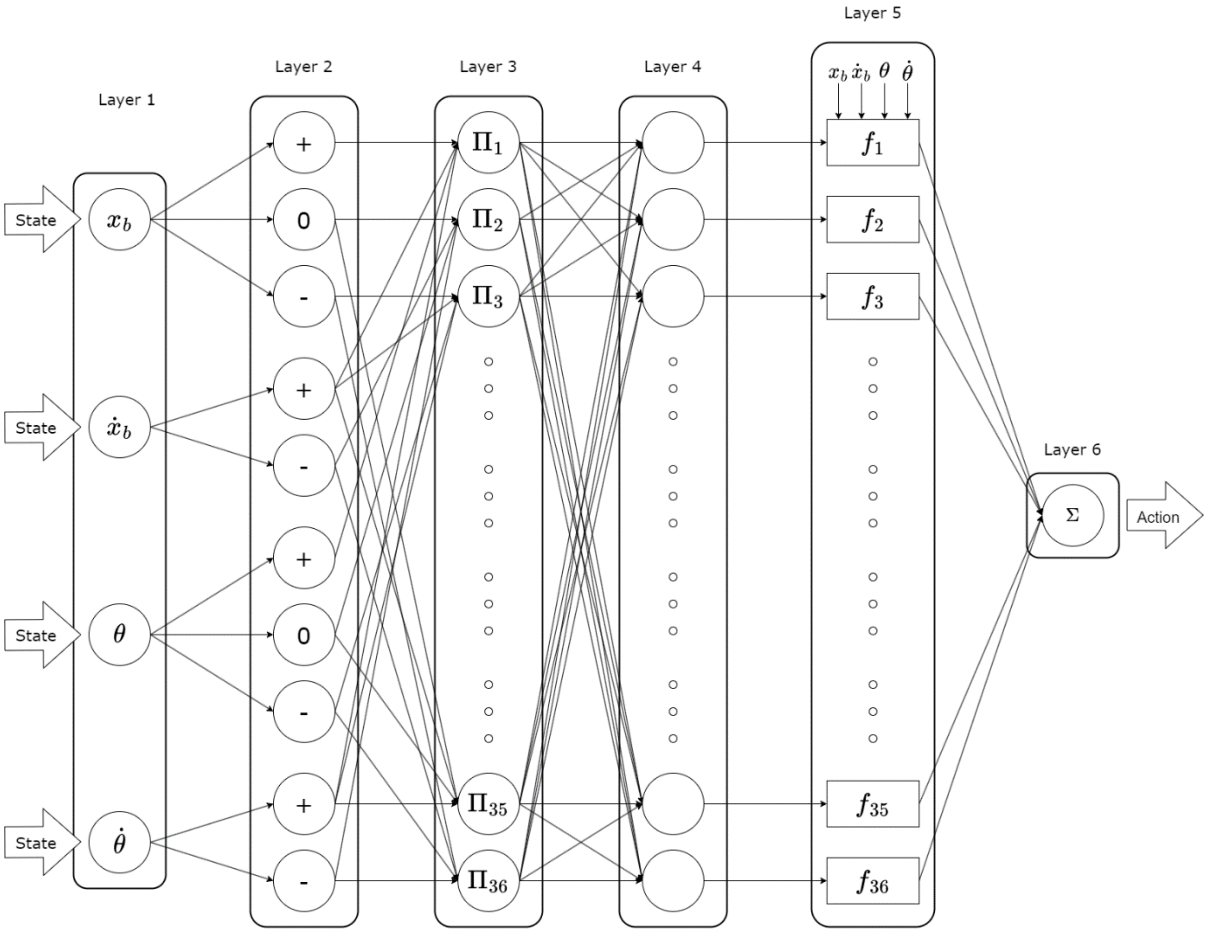


Figure 4.3: Network architecture of selected NF controller.

4.4 DDPG Controller

Reinforcement learning benefits from its large number of training steps, sometimes requiring millions of steps to achieve an optimal result. Performing training on the physical environment may take up to a minute per episode, where as a simulation can instantly reset and perform an order of magnitude more training episodes in the same amount of time. Open AI's Gym will be used as a platform for creating, running, and rendering simulation environments for reinforcement learning Python 3.9.0 will be used with this application program interface (API) as well as the TensorFlow and Keras APIs, where TensorFlow is a deep learning framework developed by Google, for building and training reinforcement learning algorithms. Furthermore, Gym adds a math library for use with NNs [87]. TensorFlow uses the Keras API for the creation of NNs. Keras allows the user to chose network features such as defining layers, number of nodes, inputs, outputs, connections, and functions [88]. These tools are free and commonly used in reinforcement learning research so they have large libraries of documentation and support online.

4.4.1 Actor and Critic NN Architecture

The architecture of the actor NN used by the agent can be seen in Figure 4.4, it has 4 layers:

Layer 1 is the input layer. The actor NN has 4 input variables corresponding to each of the state space variables, $s_n = [s_1, s_2, \dots, s_N]$, where s_n is any of the 4 input variables and $n = 1, 2, \dots, N_s$ and $N_s = 4$.

Layer 2 is a densely connected layer of 256 nodes. This means each output from the previous layer connects to all the nodes in this layer. The firing strength of each node is computed as the product of all inputs to the node and the weight parameter of the node as shown in:

$$O_j^i = \psi_{i,j}^u \prod_{n=1}^{N_s} s_n \tag{4.49}$$

where O_j^i is the firing strength of the node O, at position j, in layer i and $\psi_{i,j}^u$ is the weight parameter for that node. Here $j = 1, 2, \dots, N_i$, where N_i is the number of nodes in layer i.

Before firing strengths are output to the next layer they are passed through an activation function. In this network the Rectified Linear Units (ReLU) function is used due to it being fast to compute [89] and non-linear, it is given by:

$$U(x) = \max(0, x) \tag{4.50}$$

where $U(x)$ represents the ReLU function and $\max(0, x)$ returns the maximum between the input value and zero, turning negative inputs to zero, which makes this function non-linear. After firing strengths are activated they pass to the next layer of nodes

Layer 3 is a densely connected layer of 256 nodes. The general form of the firing strength of a densely connected node is given as:

$$O_j^i = \psi_{i,j}^u \prod_{j=1}^{N_{i-1}} O_j^{i-1} \tag{4.51}$$

where O_j^i is the firing strength of the node O, at position j, in layer i and $\psi_{i,j}^u$ is the linear parameter for that node. N_{i-1} is the number of nodes in the previous layer.

Layer 4 is the output layer of a single node. The hyperbolic tangent function is used as the activation function for the output layer, defined as [90]:

$$H(O_1^4) = \frac{e^{O_1^4} - e^{-O_1^4}}{e^{O_1^4} + e^{-O_1^4}} \tag{4.52}$$

where $H(O_1^4)$ is the hyperbolic tangent function output for the output node O_1^4, and e is Euler's number.

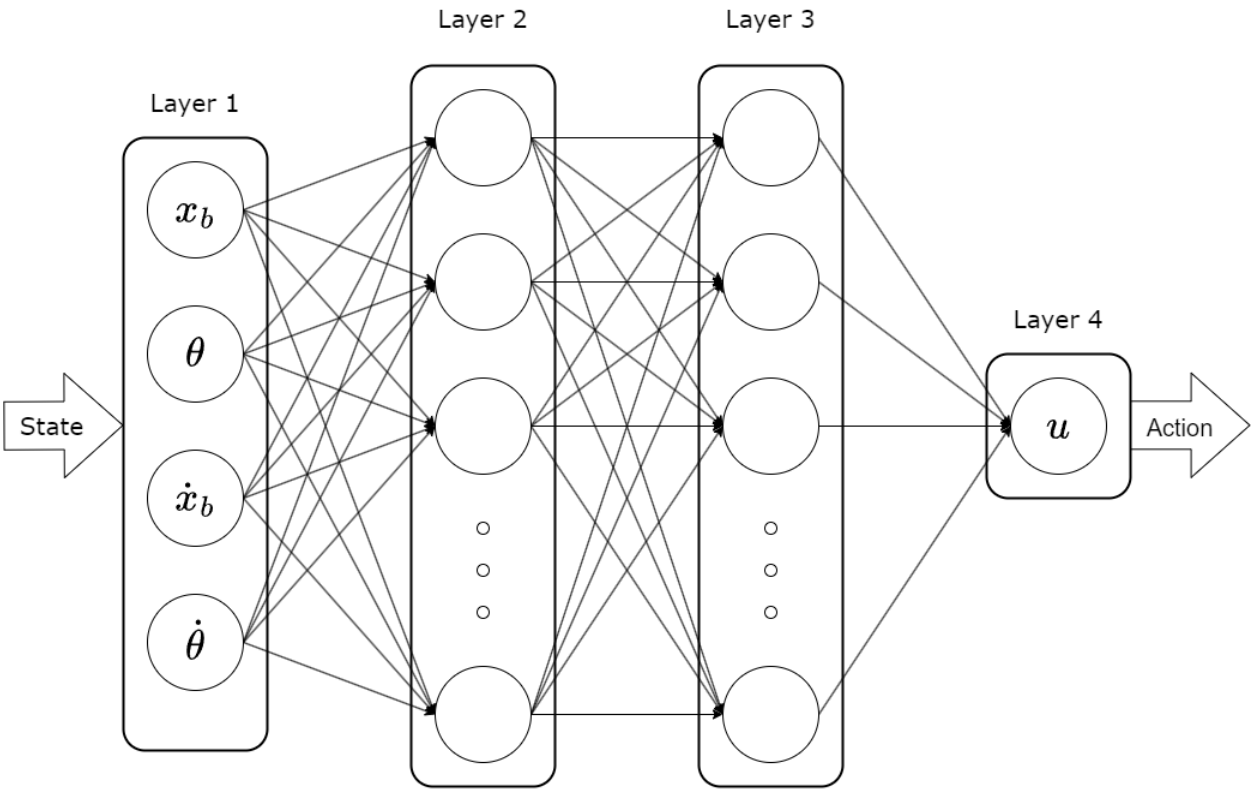


Figure 4.4: Actor network architecture

The architecture of the critic NN used by the agent can be seen in Figure 4.6, it has 7 layers:

Layer 1 is the input layer. The critic NN has 4 input variables corresponding to each of the state space variables, $s_n = [s_1, s_2, \dots, s_N]$, where s_n is any of the 4 input variables and $n = 1, 2, \dots, N_s$ and $N_s = 4$ as well as 1 input variable corresponding action taken by the actor. In this network layers 1 and 3 feature two parallel sets of nodes, separated connection to relation the action or state inputs.

Layer 2 is a densely connected layer of 128 nodes. The ReLU function is used for activation in all layers of this network besides the output. Layer 2 does not contain any nodes connected to the action input, rather the action input node connects directly to specific nodes in layer 3. The generalized form of the firing strength of any densely connected node in the critic NN is given by:

$$O_j^i = \psi_{i,j}^Q \prod_{j=1}^{N_{i-1}} O_j^{i-1} \tag{4.53}$$

where O_j^i is the firing strength of the node O, at position j, in layer i and $\psi_{i,j}^Q$ is the linear parameter for that node. N_{i-1} is the number of nodes in the previous layer connected to node O.

Layer 3 is a densely connected layer of 512 nodes, which are split into two groupings of 256 with one densely connected to the output from layer 2 and the other to the action input.

Layer 4 is a concatenation layer. The concatenation operation takes two vectors and combines them into one vector, this combines the outputs from the state and action groups of nodes. An example of the operation for layer 3 can be seen in:

$$[O_1^3, O_2^3, \dots, O_{255}^3] + [O_{256}^3, O_{257}^3, \dots, O_{512}^3] = [O_1^3, O_{256}^3, O_2^3, O_{257}^3, \dots, O_{255}^3, O_{512}^3] \tag{4.54}$$

Layer 5 is a densely connected layer of 256 nodes. Node connections follow the same general form as in Equation (4.53).

Layer 6 is a densely connected layer of 256 nodes. Node connections follow the same general form as in Equation (4.53).

Layer 7 is the output layer. The output is a summation operation given by:

$$Q = \sum_{j=1}^{N_{i=6}} O_j^{i=6} \tag{4.55}$$

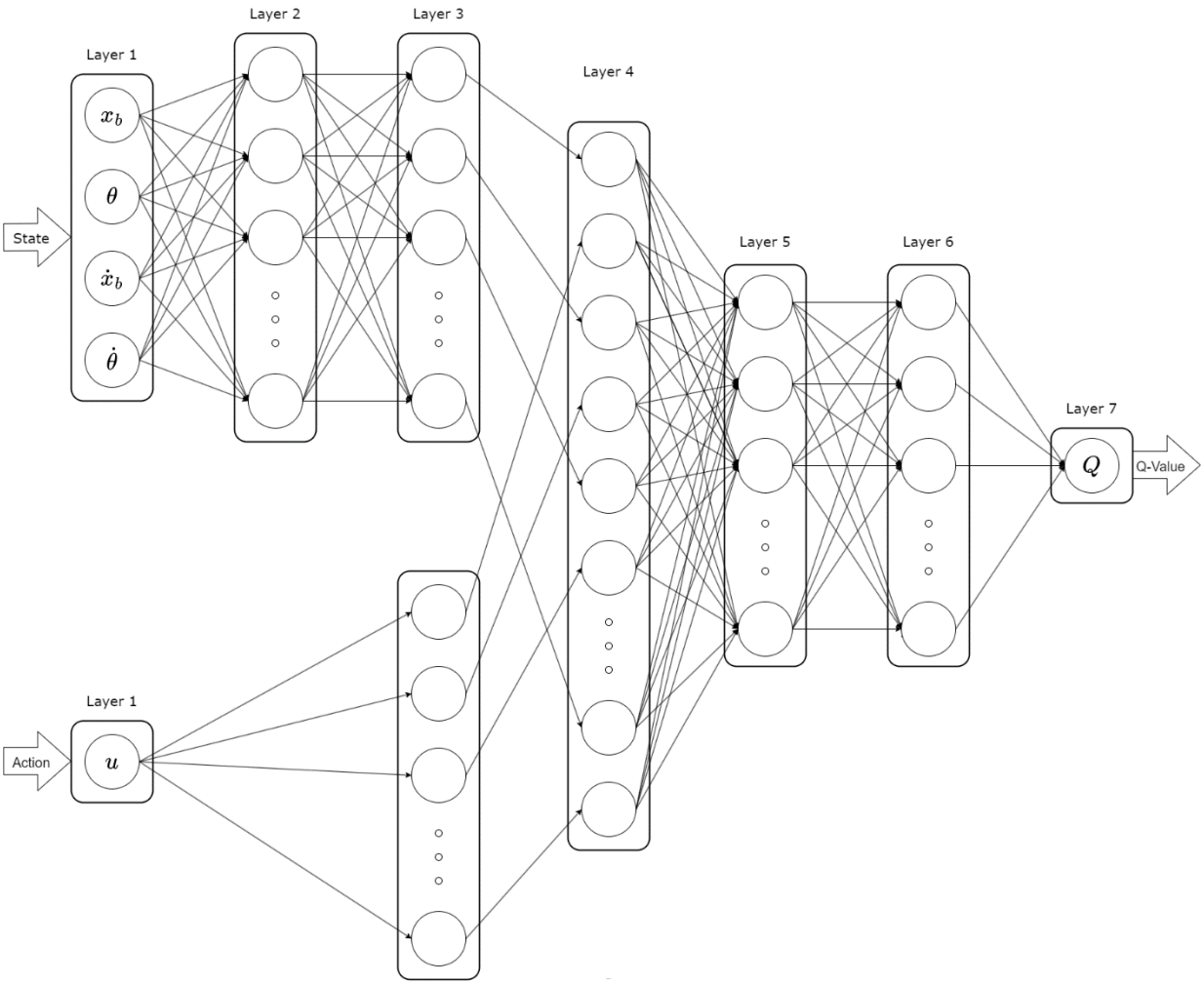


Figure 4.5: Critic network architecture

4.4.2 Smart Structure Simulation Environment

To create the smart structure simulation environment, first the environment must be initialized; Table 4.2 summarizes the hyperparameters used. When the environment is called by the agent for the first time the environment is initialized.

Table 4.2: Smart structure simulation environment parameters

Symbol	Parameter	Value
ζ	Step size for simulation	1/80 seconds
N	Maximum episode steps	500 steps
l_b, l_r	Length of beam, bar, respectively	$l_b = 1\frac{2}{3}$, $l_r = 1$ feet
A, B	State space matrices	See Equation 4.41-4.42
V_{max}	Maximum allowable voltage output	4V
x_{thresh}	Maximum allowable deflection	61 mm or 2.4 inches
θ_{thresh}	Maximum allowable angular position	45 degrees
$s_{t=0}$	Starting state	[0, 0, 0, 0]

When the agent calls the smart structure environment, it sends an action from the actor NN at the current state. The environment stores its current state between calls from the agent and returns the updated state, reward, and previous state to the agent after a step. When the environment is called the state space variables will be set equal to the current state of the model structure $s_t^T = [x_b(t), \theta(t), \dot{x}_b(t), \dot{\theta}(t)]$, if $t = 0$, $s_t = s_0$. Next the action is clipped to the maximum allowable voltage:

$$\begin{aligned} u(t) &= u(t), \quad if, \quad |u(t)| \le V_{max} \\ u(t) &= \frac{u(t)}{|u(t)|} V_{max}, \quad if, \quad |u(t)| > V_{max} \end{aligned} \tag{4.56}$$

where $u(t)$ is the action output from the controller and V_{max} is the maximum allowable voltage.

In calculating the reward for the current state and action, the reward, r is calculated as a sum of rewards for each state and the action such that:

$$r = -(r_{x_b} + r_\theta + r_{\dot{x}_b} + r_{\dot{\theta}} + r_u) \tag{4.57}$$

The reward functions for the DDPG controller in this work are tuned based on the expected size of each of the variables. The equations arrived at are given as:

$$r_{x_b} = 525x_b^2, \qquad r_\theta = \frac{32}{\pi^2}\theta^2, \qquad r_{\dot{x}_b} = 8\dot{x}^2, \qquad r_{\dot{\theta}} = \frac{1}{5}\ddot{\theta}^2, \qquad r_u = \frac{1}{10}u^2 \tag{4.58}$$

Next the state is transitioned to the updated state using the state spaces matrices A and B by solving $\dot{s}$ such as:

$$\begin{aligned} \dot{s} &= As + Bu \\ \dot{s} &= \begin{bmatrix} \dot{s}_1 \\ \dot{s}_2 \\ \dot{s}_3 \\ \dot{s}_4 \end{bmatrix} = \begin{bmatrix} \dot{x}_b \\ \dot{\theta} \\ \ddot{x}_b \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} s_3 \\ s_4 \\ \dot{s}_3 \\ \dot{s}_4 \end{bmatrix} \end{aligned} \tag{4.59}$$

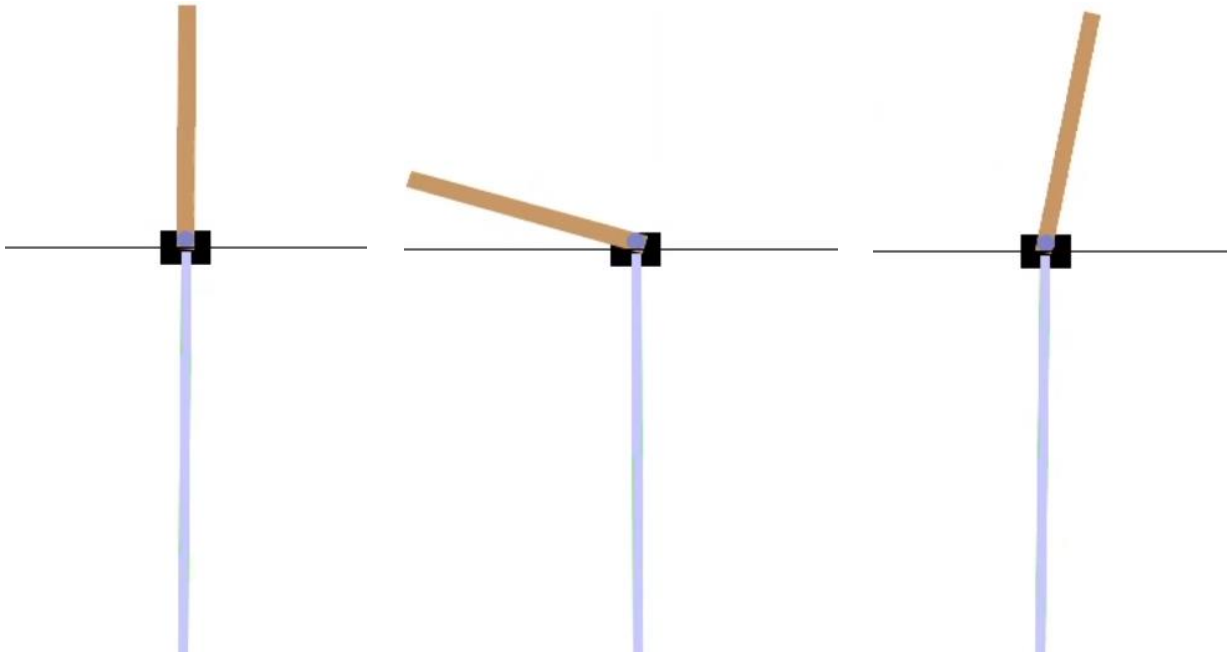
Then $\dot{s}$, can be used to determine the next state such that:

$$\begin{aligned} x_b(t+1) &= x_b(t) + \zeta \dot{x}_b(t) \\ \dot{x}_b(t+1) &= \dot{x}_b(t) + \zeta \ddot{x}_b(t) \\ \theta(t+1) &= \theta(t) + \zeta \dot{\theta}(t) \\ \dot{\theta}(t+1) &= \dot{\theta}(t) + \zeta \ddot{\theta}(t) \end{aligned} \tag{4.60}$$

where ζ is the step size.

After determining the next state, the step counter is increased by one and the environment returns the next state, reward, and a Boolean representing the episode completion check. The step is checked to see if the episode is complete, which happens when the maximum number of steps has been reached or the angle of the beam or position of deflection is within the set thresholds. If the episode is determined to be complete, the state will be reset to the starting state and the next episode begins from the beginning, which will be discussed in Section 4.4.2.

After every step, the environment is rendered as a visual indicator of the system state. Figure 4.6 shows an example of this rendering in a training episode. This rendering process is turned off during general system training to increase training speed.



(a) (b) (c)

Figure 4.6: Simulated training environment render - (a) starting state, (b) control begins after disturbance, (c) controller overshooting

4.4.3 Agent Training

In implementation of the reinforcement learning, the agent will follow the algorithm outline previously in Section 4.4.1. Firstly, the agent is initialized with the hyperparameters listed in Table 4.3. Selecting the appropriate hyperparameters can be a challenging task. Due to the circumstances of the control problem in this work, manual tuning by trial and error is employed.

Table 4.3: DDPG learning agent parameters

Symbol	Parameter	Value
b	Replay buffer size	1'000'000 steps
N_E	Number of training episodes	4000 episodes
N_b	Batch size for replay buffer sampling	64 samples
μ, σ	Mean and standard deviation for noise	$\mu = 0, \sigma = 1$
N_E	Number of episodes for training	4000 episodes

γ	Discount factor for future rewards	0.999
τ	Target learning rate	0.1
$\phi^{\theta}, \phi^{\mu}$	Critic and actor learning rates, respectively	$\phi^{\theta} = 0.001$, $\phi^{\mu} = 0.005$
ε_k	State observation noise scaling constant, $k = 1,2,\dots,n$ where n is equal to the number of inputs into the actor network	$\varepsilon_1 = 0.005, \varepsilon_2 = 0.05,$ $\varepsilon_3 = \frac{\pi}{2048}, \varepsilon_4 = 0.5$
$\varrho, \beta_1, \beta_2, \varsigma$	Step size, other constants for use in ADAM gradient descent	$\varrho = \zeta,$ $\varsigma = (10)^{-8},$ $\beta_1 = 0.9,$ $\beta_2 = 0.999$

The following summarizes the test operation procedures:

1) Firstly, the environment is initialized, and the buffer and NNs are created.

2) At the beginning of each episode, the initial state of the smart structure environment is set to zero flexible beam deflection and the bar at zero angular position (i.e., the flexible beam and rigid bar are standing straight up).

3) The agent provides a disturbance to the simulated structure by applying a negative voltage to the motor to generate an angular position of the rigid bar of [-0.8, -1.2] rad. The structure starts to vibrate.

3) The agent takes the control training actions on the structure. Random noise is added intentionally during training to better explore the action space of the environment and develop a more robust policy.

4) The noisy-action chosen by the agent is used to update the current state of the environment. This returns a vector called a state transition containing the previous states, new states, action taken, and the reward for that action. This transition is then stored in the replay buffer.

5) The networks are updated based on a minibatch of samples from the replay buffer; the critic is updated using the target critic and target actor networks, which can be used to estimate the effectiveness of the control action, compute the loss of the critic, and optimize the critic.

6) The actor network is updated using the loss from the critic calculated with Equation (3.17), and then the target actor and target critic networks are updated using Equation (3.5).

7) Steps 4)-7) are repeated until the episode is finished.

8) Steps 2)-8) are repeated until the specified training episodes are completed. Then the actor is fully trained, and performance can be evaluated to ensure training is appropriate.

Once 50% of the training episodes have been performed, domain randomization is initiated by applying state perturbation. Perturbed states are then fed to the actor network to determine an action for the current environment. This allows the agent to learn an effective policy and to increase the robustness of the policy. Each input is modulated by a noise scaling constant, as summarised in Table . Domain randomization noise increases linearly once it begins to affect the system states, which can reach its maximum after another half of the remaining episodes, or:

$$s_t \leftarrow \varepsilon_k s_t \times \left(\mathcal{N}|_{\mu=0,\sigma=1}\right) \times \frac{\max(0, E - 0.5N_E)}{0.25N_E} \tag{4.61}$$

where ε_k, is the scaling constant; $\mathcal{N}$ is the noise output from the noise function; E is the current episode; and N_E is the total number of episodes.

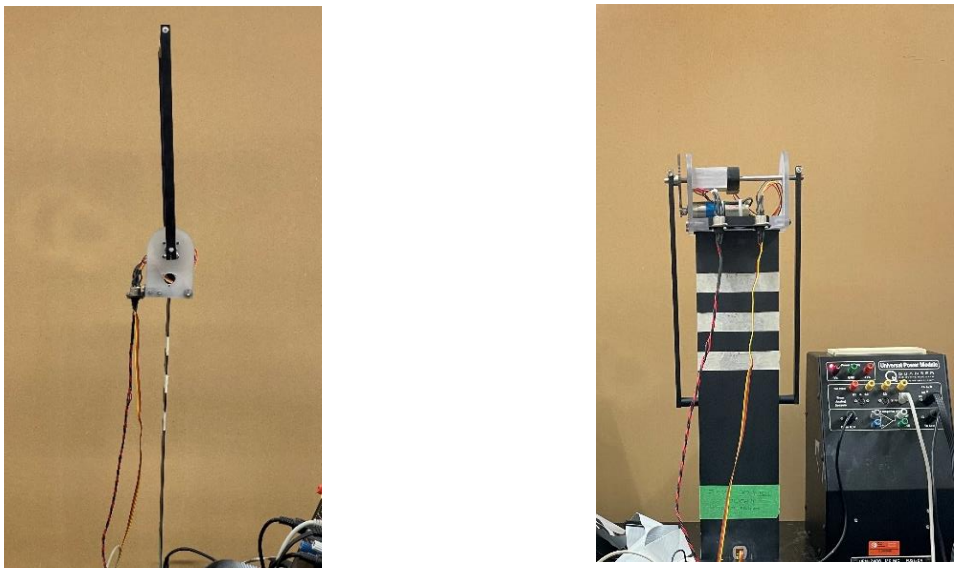
The effectiveness of related controllers for vibration suppression of the flexible beam structure will be discussed in Chapter 5.

Chapter 5 - Performance Verification

5.1 Proposed Methodology

Experiments are conducted using the smart structure workstation to verify the effectiveness of the proposed DDPG technique for vibration control of flexible beams with variable dynamics. The DDPG controller is evaluated using two other related controllers: the classical PD controller and an intelligent NF controller. The purpose of selecting the PD controller is to demonstrate the robustness of the performance of the DDPG. PD control is a classical and commonly used control technique for flexible beam vibration control, which can perform well even when conditions vary from the expected; in addition, the PD controller has been optimized and implemented in smart structure workstation by the manufacturer for demonstration purpose. The NF controller is selected to compare the ability of DDPG controller in reinforcement learning while generating a non-sensitive policy. The comparison tests will be undertaken to examine the effectiveness of the related controllers under variable dynamic conditions.

A testing procedure is devised and implemented as previously described in Section 4.1. The initial state for the smart structure is upright with no initial deflection, which is the position to begin a test. Figure 5.1(a) shows the structure in the resting position, and Figure 5.1(b) shows the structure in the initial state for testing.



(a) (b)

Figure 5.1: Smart structure in the - (a) upright position, (b) initial position

After the flexible beam has reached a steady state solution (i.e., completely upright), a disturbance is applied by the motor to the rigid bar. The maximum voltage is supplied until the rigid bar reaches a set position (1.0 rad in this case) to simulate a strong disturbance. After the disturbance is applied, the controllers are enabled to suppress the disturbance vibration.

The tests are undertaken to examine controller's abilities to suppress the vibration in the beam until it returns to a steady state solution. The performance indicators used to evaluate the performance of the related controllers in this test are settling time, overshoot, and mean error. The steady state condition is reached when the deflection over the test window is within ±1.5mm. The overshoot is the maximum deflection of the beam when it exceeds the steady state position. The mean error is the average absolute deflection of the beam over the control time. These indicators are chosen to simulate a drogue control scenario in AAR flight with disturbances. For this reason, settling time is chosen to represent the speed of vibration suppression or convergence. Overshoot indicator is selected to reflect the predictability of stabilization; and the mean error is used to represent the displacement over the control window.

5.2 Simulation of Variable System Dynamics

To simulate the effects of variable system dynamics of the flexible beam structure, a set of additional mass blocks are attached at three different positions on the flexible beam. The purpose is to test the robustness of the related controllers to accommodate with variable system dynamics. Each mass block is approximately 100 grams, and a pair of mass blocks are placed on each side of the beam. These three positions are selected as 100mm, 150mm, and 200mm from the rotational axis of the rigid bar, as illustrated in Figure 5.2.

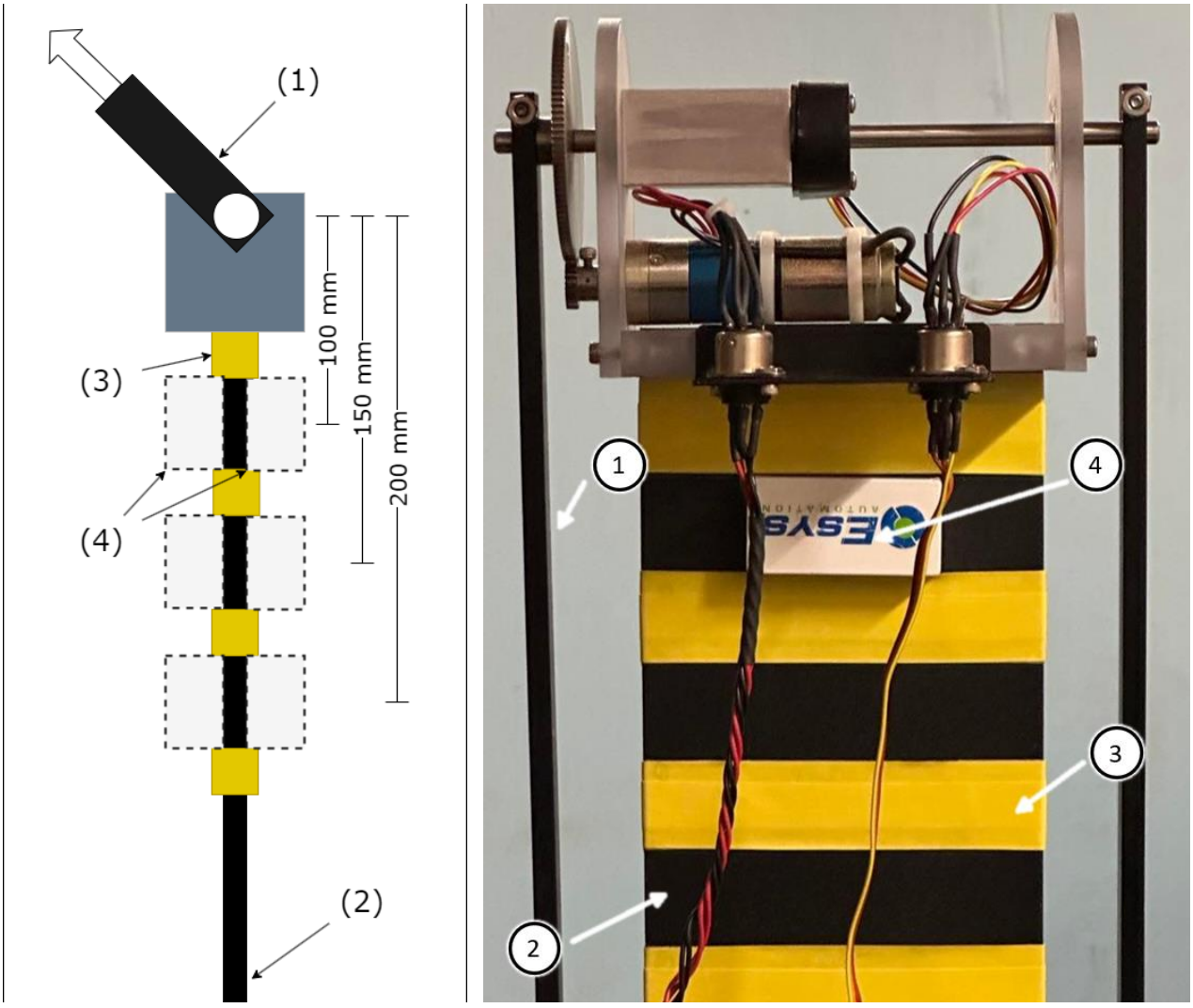


Figure 5.2: Dynamic loading of the smart structure workstation – (a) diagram (b) photo - (1) rigid bar, (2) flexible beam, (3) tape markers, (4) mass block positions

5.3 Experimental Test Result Analysis

Testing is comprised of 4 trials at each loading position for all controllers. The results in 48 tests are performed and recorded; detailed test indicators for all trials are summarised in Appendix A. For each dynamic loading condition, the response x_b is the deflection of the

flexible beam as illustrated in Figures 5.3-5.6. Typical test results are demonstrated below to represent the average results for each loading condition.

5.3.1 Test Results for the Flexible Beam without Extra Mass Blocks

Figure 5.3 shows the test results of the flexible beam using the related controllers with no mass blocks attached to flexible beam. The results of this trial are summarized in Table 5.1. The percentage change of each performance indicator is relative to the performance of the PD controller; a negative percentage indicates an increase in performance compared to the PD. The NF controller achieves a 19.87% reduction in settling time compared to the PD controller; however, this improved performance comes at the cost of an increased overshoot and mean error, by 18.22% and 24.27%, respectively. The proposed DDPG controller achieves the best performance with a faster settling time with a 23.84% reduction from the PD controller. It can also reduce the overshoot by 19.55% and the mean error by 2.57% in comparison with the reference PD control. The DDPG outperforms the PD and the NF controller due to its unique reasoning function and proper training by using the proposed domain randomization method.

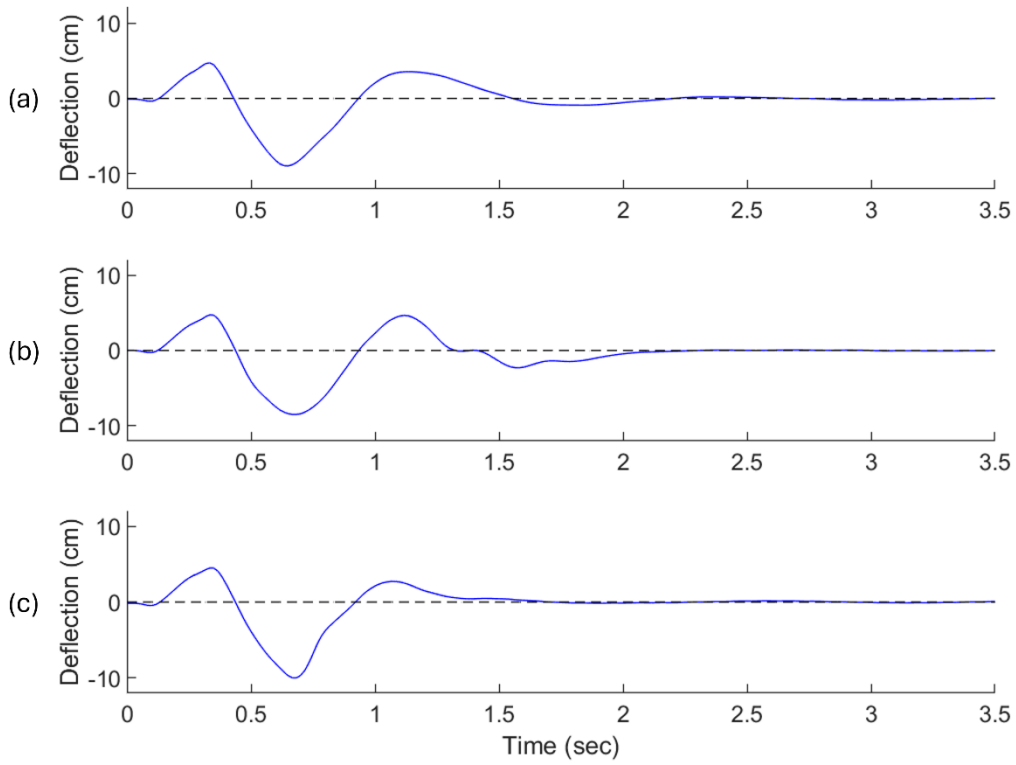


Figure 5.3: Deflections of the flexible beam without extra mass blocks, using the related controllers: (a) PD controller, (b) NF controller, (c) DDPG controller.

Table 5.1: Experimental results without extra mass blocks using the related controllers

Controller	Settling Time (s)	Change (%)	Overshoot (mm)	Change (%)	Mean Error (mm)	Change (%)
1 - PD	2.52	-	35.39	-	13.52	-
2 - NF	2.02	-19.87%	41.84	18.22%	16.79	24.27%
3 - DDPG	1.92	-23.84%	28.47	-19.55%	13.17	-2.57%

5.3.2 Test Results for the Flexible Beam with Additional Mass Blocks at a Top Position

Mass blocks are then placed in the top position of the flexible beam, with the center of the mass blocks 100mm below the rigid bar's rotational axis. Figure 5.4 shows the testing results using the related controllers, and the results of the tests are summarized in Table 5.2. In this loading condition the PD controller performs similarly to the PD control without additional mass blocks loading condition above. This is general indicator the reliability of the use of the

optimized PD controller for vibration suppression in this flexible beam. The NF control does not perform as well as the PD control in this case with a slightly slower settling time and poor overshoot (41.8%); it is mainly due to the reason that the NF controller is trained using data sets generated from the flexible beam without no extra mass blocks, but controlled using the default PD controller. The DDPG is not affected by this because it does not require any externally provided training data. The proposed DDPG control technique can provide the best performance in this case, with an improved settling time (22.01%), overshoot (27.14%) and mean error (4.66%), with reference to the PD control due to the use of domain randomization and the unique reinforcement learning.

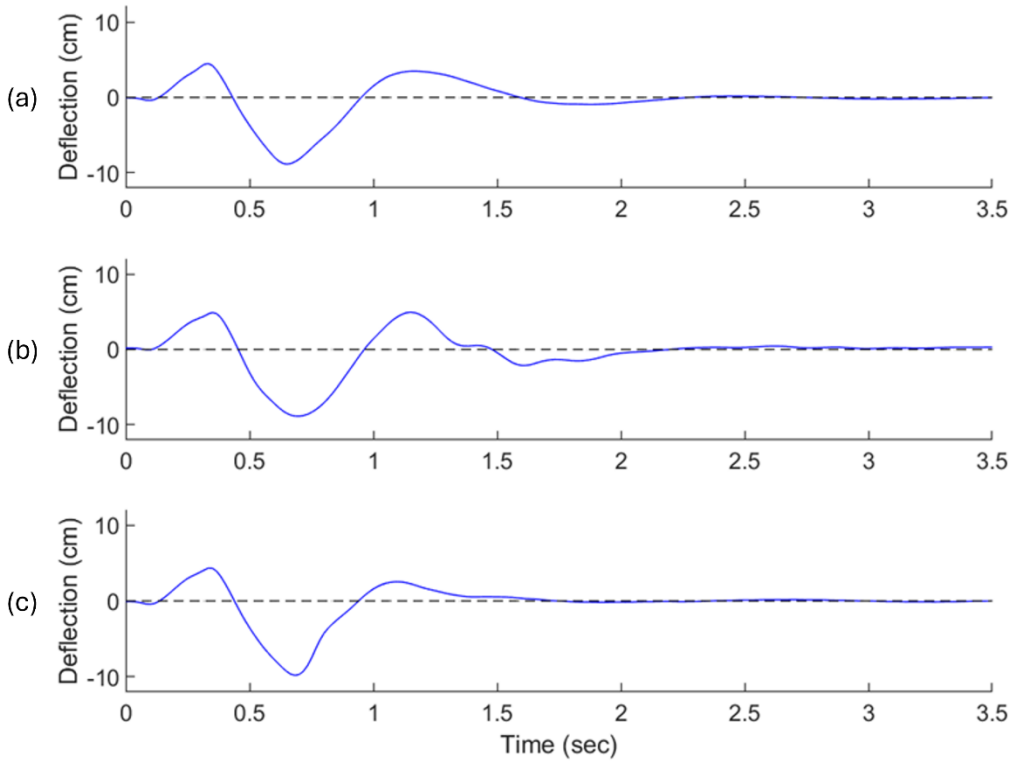


Figure 5.4: Deflections of the flexible beam with mass blocks at the top position (100mm below the rotational axis of the rigid bar), using the related controllers: (a) PD controller, (b) NF controller, (c) DDPG controller.

Table 5.2: Experimental results with mass blocks at the top position using the related controllers

Controller	Settling Time (s)	Change (%)	Overshoot (mm)	Change (%)	Mean Error (mm)	Change (%)
1 - PD	2.65	-	34.79	-	13.44	-

2 - NF	2.68	1.26%	49.33	41.80%	15.58	15.94%
3 - DDPG	2.07	-22.01%	25.34	-27.14%	12.81	-4.66%

5.3.2 Test Results for the Flexible Beam with Additional Mass Blocks at the Middle Position

The mass blocks are placed to the second position, 150mm from the rigid bar's rotational axis to simulate a different dynamics condition of the flexible beam. Figure 5.5 shows the test results using the related control techniques, and Table 5.3 summaries the results with the reference of the PD control. The PD controller performs reliably reasonably well in this case. The NF controller performs reasonably well under this loading condition; although it has a longer settling time (21.88%) compared to the PD controller, the NF control has a better response in, the overshoot and mean error than the PD control. The DDPG controller outperforms both the PD and NF controllers, with a (24.38%) decrease in settling time, a (21.79%) decrease in overshoot, and a minimally improved (0.72%) reduction in mean error. Although the DDPG controller does not perform as good as in other beam dynamic conditions, with only a minor reduction in mean error, it still can provide the best performance due to the application of domain randomization during training.

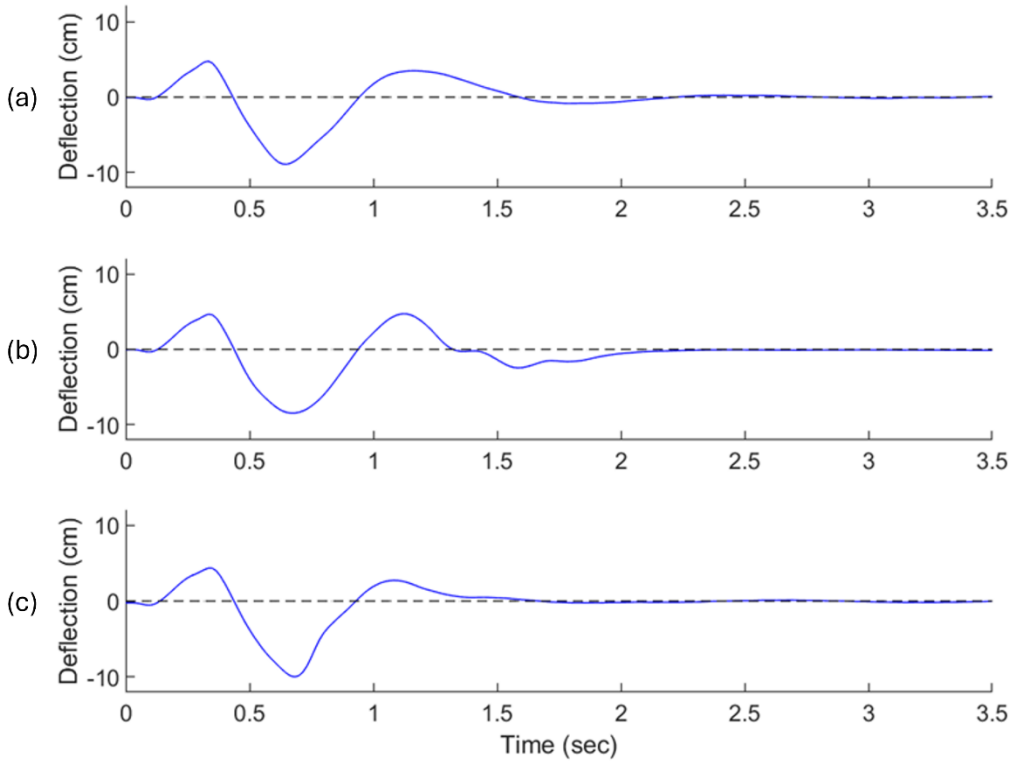


Figure 5.5: Deflections of the flexible beam with mass blocks at the middle position (150 mm below the rotational axis of the rigid bar) using the related controllers: (a) PD controller, (b) NF controller, (c) DDPG controller.

Table 5.3: Experimental results with mass blocks at the middle position using the related controllers

Controller	Settling Time (s)	Change (%)	Overshoot (mm)	Change (%)	Mean Error (mm)	Change (%)
1 - PD	2.67	-	34.89	-	13.30	-
2 - NF	2.08	-21.88%	47.11	35.00%	16.49	23.95%
3 - DDPG	2.02	-24.38%	27.29	-21.79%	13.20	-0.72%

5.3.2 Test Results for the Flexible Beam with Additional Mass Blocks at the Low Position

The mass blocks are then moved to the lower position, 200mm below the rotational axis of the rigid bar. Figure 5.6 shows the performance under this loading condition using the related controllers, which are summarized in Table 5.4. Here, the PD controller can still provide a reasonably well performance similarly to the previous dynamic conditions. The NF controller outperforms the PD in settling time while sacrificing the overshoot and mean error. The DDPG controller shows better performance when compared to both the PD controller and NF controller.

The DDPG settles (22.44%) faster than the PD, reduces the overshoot by almost half with a (45.15%) decrease and improves the mean error by (7.84%). Examining the results from each loading condition for the DDPG controller has demonstrated its fast convergence and more reliable performance corresponding to different flexible beam dynamics. The reinforcement learning algorithm allows the agent to generate an optimized policy for the given control problem, and the policy evolves without relying on any external data. The DDPG reinforcement learning agent generates an effective but sensitive policy and the domain randomization can correct the sensitivity so as to improve control convergence and accuracy. This demonstrates that a DDPG reinforcement learning controller has potential to be applied for continuous control problems, such as controlling a drogue in an aerial refuelling mission.

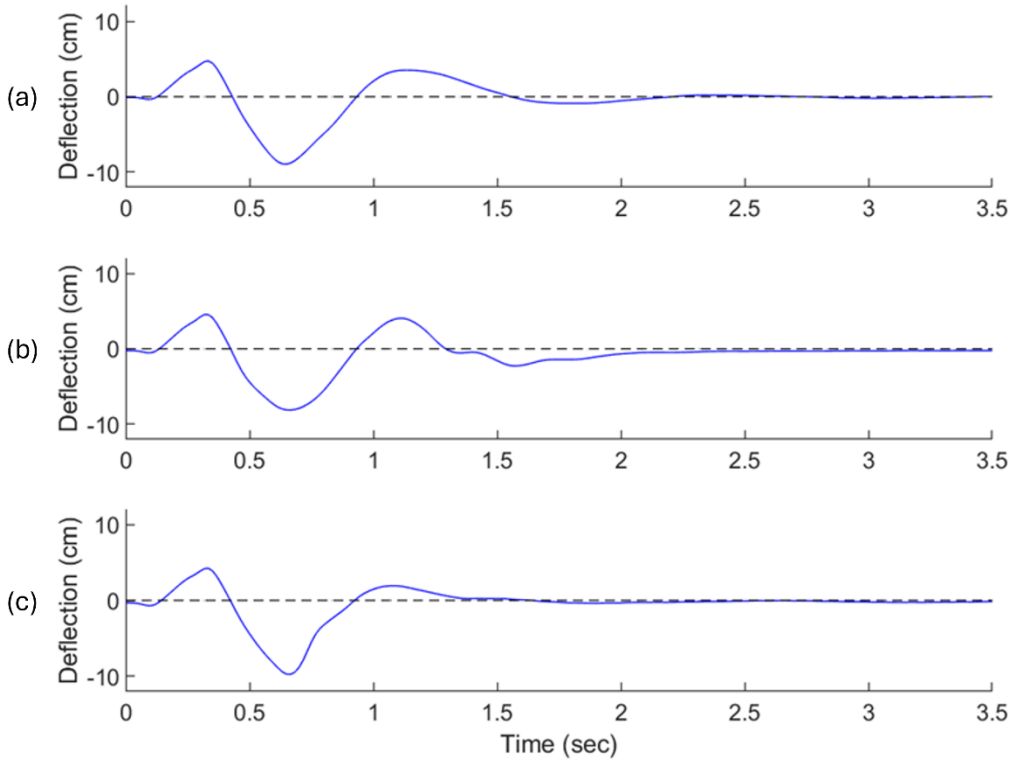


Figure 5.6: Deflections of the flexible beam with mass blocks at the low position (200 mm below the rotational axis of the rigid bar) using the related controllers: (a) PD controller, (b) NF controller, (c) DDPG controller.

Table 5.4: Experimental results with mass blocks at the low position using the related controllers

Controller	Settling Time (s)	Change (%)	Overshoot (mm)	Change (%)	Mean Error (mm)	Change (%)
1 - PD	2.60	-	35.15	-	13.43	-

2 - NF	2.07	-20.51%	40.56	15.38%	16.61	23.62%
3 - DDPG	2.02	-22.44%	19.28	-45.15%	12.38	-7.84%

Chapter 6 - Conclusion and Future Work

6.1 Conclusion

Historically, AAR is a technology that been limited to military use due to the cost of maintaining the capability and the risk involved. There are many possible benefits of bringing this technology to civilian air operations such as reduced costs, increased payload capacity, and less aircraft wear and tear. This work investigates the feasibility of converting a civilian tanker into an aerial refueling tanker. To address the issue of cost with civilian AAR the Air Tractor 802 is considered due to it's relatively low-cost compared to large tanker platforms. Risk involved in AAR is largely due to the chaotic aerodynamic conditions that can occur during refueling; to mitigate this an actively stabilized drogue is considered.

The first objective of this thesis was to model the AT-802 and a hose-drogue aerial refueling system that could be mounted onboard. A three-dimensional model of the aircraft was produced and used in computational fluid dynamics simulations. These results were analysed to determine if there may be any regions of wake behind the AT-802 that may make a drogue respond erratically. Simulations showed that the wake behind the aircraft was relatively homogenous and would not significantly affect a drogues stable flight. A dynamic model of the airframe was developed using a lumped mass model of the hose and drogue. The hose-drogue system in flight was simulated to investigate possible drogue behaviour in the flight conditions expected for refueling from the AT-802. Results showed that the drogue stabilized on a single point and did not oscillate dramatically during the simulation. Results indicate that a drogue flying behind an AT-802 should behave at least as well as it would behind a large tanker.

The second object of this thesis was to develop a Deep Deterministic Policy Gradient (DDPG) technology for vibration control of a drogue in flight. Since the hose-drogue prototype could not be developed in time to support this research, while access to military equipment of this kind is also restricted, an equivalent flexible beam workstation was used for the research of drogue vibration control. Extra mass blocks were used in the flexible smart structure to simulate variable dynamics conditions. The proposed vibration control technique uses an agent trained using reinforcement learning with the DDPG strategy. Additionally, domain randomization was used to increase the adaptive capability of the controller and to transfer system characteristics from the simulated training environment to the experimental apparatus it is tested on. The

effectiveness of the proposed DDPG technique was examined systematically by experimental tests; its performance was compared with two related controllers, the optimal PD control implemented in the flexible beam workstation, and the NF controller. The result results showed that the NF controller could outperform the PD control when the test conditions were ideal, but in some other conditions the NF controller performed similarly to the PD control. However, the proposed DDPG controller could provide the best control performance under all these test and dynamics conditions with reduced settling time, lower overshoot and smallest mean errors. The best performance of the DDPG technique was due to its unique control policy that was developed with reinforcement learning, combined with domain randomization to improve robustness. It has potential to be applied for drogue control in the real aerial refueling prototyping.

6.2 Future Work

This project was the preliminary work in the research and development of exploring Air Tractor 802 for an aerial refueling of helicopters.

In the future this will be expanded on in the following areas:

- Prototype systems will be manufactured and installed into a FuelBoss aircraft to test and improve the control technology, as well as to pursue Transport Canada approval.
- The aeroelastic model of the hose-drogue will be improved using real flight-testing data such as turbulence, wind gusts, tanker wake.
- The DDPG controller will be improved by optimizing the network architecture and including real environmental conditions, hose length, and receiver aircraft conditions.

References

[1] S. Lamarche, “The backbone of reach & power: air-to-air refueling in the RCAF,” M.S. thesis, Department of National Defence, Canadian Forces College, Toronto, ON, Canada, 2015.

[2] Royal United Services Institute of Nova Scotia, “Refuelling RCAF aircraft post-2020”, 2020. [Online] Available: https://rusi-ns.ca/refuelling-rcaf-aircraft-post-2020/.

[3] FuelBoss. (2020). *Purpose-built for elevated results*. [Online] Available: https://fuelboss.ca/

[4] Air Tractor, *Aircraft overview - Air Tractor® AT-802F*, 2020. [Online] Available: https://at802f.com/aircraft-overview/

[5] R. K. Smith, “75 years of inflight refueling”. *Air Force History and Museums Program*, 1998.

[6] S. J. Dougherty, “Air refueling: the cornerstone of global reach-global power,” Air War College, Alabama, Report. 1996.

[7] B. J. Theiss, “A comparison of in-flight refueling methods for fighter aircraft: boom-receptacle vs. probe-and-drogue,” *College Aviation Review International*, vol. 25, no. 2, pp. 51–72, 2007.

[8] C. Bolkcom, “Air force aerial refueling methods: flying boom versus hose-and-drogue,” *CRS Report to Congress*, p. 11, 2006, [Online]. Available: https://www.fas.org/sgp/crs/weapons/RL32910.pdf

[9] P. S. Killingsworth, “Multipoint aerial refueling - a review and assessment.” RAND National Defense Research Institute, Santa Monica, CA, p. 86, 1996.

[10] D. H. Kalt, R. Tipton, L. Martin, D. Ferwereda, D. A. Benson, and A. Ezfa, “Aerial refueling probe / drogue system," Aerial Refueling Systems Advisory Group, Guidance Document, 2018, [Online] Available: https://apps.dtic.mil/sti/pdfs/AD1064517.pdf

[11] X. Dai, Q. Quan, J. Ren, and K. Y. Cai, “Iterative learning control and initial value estimation for probe–drogue autonomous aerial refueling of UAVs,” *Aerospace Science and Technology*, vol. 82–83, pp. 583–593, 2018.

[12] D. Silva, *Photograph of wingtip vortices created by a U.S. Coast Guard C-130J Super Hercules aircraft on 16 April 2009*, [Online]. Available: http://chamorrobible.org/gpw/gpw-201305.htm

[13] X. Dai, Z. Wei, and Q. Quan, "Modeling and simulation of bow wave effect in probe and drogue aerial refueling," *Chinese Journal of Aeronautics*, vol. 29, no. 2, pp. 448–461, 2016.

[14] Y. Sun, H. Duan, and N. Xian, "Fractional-order controllers optimized via heterogeneous comprehensive learning pigeon-inspired optimization for autonomous aerial refueling hose–drogue system," *Aerospace Science and Technology*, vol. 81, pp. 1–13, 2018.

[15] A. Dogan and W. Blake, "Modeling of bow wave effect in aerial refueling," *AIAA Atmospheric Flight Mechanics Conference 2010*, Toronto, Ontario, August, 2010.

[16] U. Bhandari, P. R. Thomas, S. Bullock, T. S. Richardson, and J. L. du Bois, "Bow wave effect in probe and drogue aerial refuelling," *AIAA Guidance, Navigation, and Control (GNC) Conference*, pp. 1–21, 2013.

[17] H. Wang, X. Dong, J. Xue, and J. Liu, "Dynamic modeling of a hose-drogue aerial refueling system and integral sliding mode backstepping control for the hose whipping phenomenon," *Chinese Journal of Aeronautics*, vol. 27, no. 4, pp. 930–946, 2014.

[18] Q. He, H. Wang, Y. Chen, M. Xu, and W. Jin, "Command filtered backstepping sliding mode control for the hose whipping phenomenon in aerial refueling," *Aerospace Science and Technology*, vol. 67, pp. 495–505, 2017.

[19] Z. H. Zhu and S. A. Meguid, "Elastodynamic analysis of aerial refueling hose using curved beam element," *AIAA Journal*, vol. 44, no. 6, pp. 1317–1324, 2006.

[20] J. Eichler, "Dynamic analysis of an in-flight refueling system," *Journal of Aircraft*, vol. 15, no. 5, pp. 311–318, 1978.

[21] J. Yan, "Modeling of probe-and-drogue part of an in-flight refueling system", Thesis, Rochester Institute of Technology, 2004.

[22] Z. H. Zhu and S. A. Meguid, "Modeling and simulation of aerial refueling by finite element method," *International Journal of Solids and Structures*, vol. 44, no. 24, pp.

8057–8073, 2007.

[23] K. Ro, J. W. Kamman, S. Ki, and C. Ki, “Modeling and simulation of hose-paradrogue aerial refueling systems,” *Journal of Fluids and Structures*, vol. 33, no. 1, 2010.

[24] K. Ro, T. Kuk, and J. Kamman, “Dynamics and control of hose-drogue refueling systems during coupling,” *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 6, 2011.

[25] K. Salehi Paniagua, P. García-Fogeda, F. Arévalo, and J. Barrera Rodriguez, “Aeroelastic analysis of an air-to-air refueling hose–drogue system through an efficient novel mathematical model,” *Journal of Fluids and Structures*, vol. 100, p. 103164, 2021.

[26] K. Ro, T. Kuk, and J. W. Kamman, “Active control of aerial refueling hose-drogue systems,” *AIAA Guidance, Navigation, and Control Conference*, no. August, pp. 1–12, 2010.

[27] T. Kuk, K. Ro, and J. W. Kamman, “Design, test and evaluation of an actively stabilized drogue refueling system,” *AIAA Infotech at Aerospace Conference and Exhibit 2011*, no. March, pp. 1–13, 2011.

[28] T. Kuk and K. Ro, “Design, test and evaluation of an actively stabilised drogue refuelling system,” *Aeronautical Journal*, vol. 117, no. 1197, pp. 1103–1118, 2013.

[29] T. Kuk, “Active Control of Aerial Refueling Drogue,” *Disertations* vol 256, 2014.

[30] Z. Liu, J. Liu, and W. He, “Modeling and vibration control of a flexible aerial refueling hose with variable lengths and input constraint,” *Automatica*, vol. 77, pp. 302–310, 2017.

[31] D. Yuan *et al.*, “Study on the controllability of a drogue for hose-drogue aerial refueling system,” *2017 Asian Control Conference, ASCC 2017*, vol. 2018-Janua, no. 1, pp. 2592–2595, 2018.

[32] P. García-Fogeda, J. Esteban Molina, and F. Arévalo, “Dynamic response of aerial refueling hose-drogue system with automated control surfaces,” *Journal of Aerospace Engineering*, vol. 31, no. 6, p. 1-13, 2018.

[33] J. Cheng, F. Deng, X. Liu, and K. Ji, “Active control of aerial refueling hose-drogue dynamics with the improved reel take-up system,” *International Journal of Aerospace Engineering,* vol. 2022, 2022.

[34] Z. Su and H. Wang, “Antidisturbance vibration suppression of the aerial refueling hose during the coupling process,” *International Journal of Aerospace Engineering*, vol. 2017, 2017.

[35] J. Ariss and S. Rabat, “A comparison between a traditional PID controller and an Artificial Neural Network controller in manipulating a robotic arm,” Thesis KTH Royal Institue of Technology, Stockholm, Sweden, 2019.

[36] M. Aamir, “On replacing PID controller with ANN controller for DC motor position control,” *International Journal of Research Studies in Computing*, vol. 2, no. 1, pp. 21–29, 2013.

[37] Y. Dote, “Introduction to fuzzy logic,” *IECON Proceedings (Industrial Electronics Conference)*, vol. 1, no. October, pp. 50–56, 1995.

[38] J. H. Lilly, *Fuzzy Control and Identification*. Hoboken, New Jersey: Wiley 2010.

[39] Y. S. Maslennikova and V. V Bochkarev, “Training algorithm for neuro-fuzzy network based on singular spectrum analysis,” *Cornell arXiv:1410.1151*, 2014, [Online]. Available: http://www.arxiv.org/pdf/1410.1151.pdf

[40] F. O. Karray and de C. Silva, “Soft computing and intelligent systems design,” p. 139, Essex, England: Pearson 2004.

[41] X. Ji and W. Wang, “A neural fuzzy system for vibration control in flexible structures,” *Intelligent Control and Automation*, vol. 2, no. 3, pp. 258–266, 2011.

[42] T. P. Lillicrap, *et al.*, “Continuous control with deep reinforcement learning,” *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.

[43] M. Kearns, Y. Mansour, and A. Y. Ng, “A sparse sampling algorithm for near-optimal planning in large Markov decision processes,” *IJCAI International Joint Conference on Artificial Intelligence*, vol. 2, pp. 1324–1331, 1999.

[44] R. Bellman, “A Markovian decision process,” *Mathematics in Science and Engineering*, vol. 130, no. 3. pp. 172–187, 1977.

[45] V. Mnih *et al.*, “Playing atari with deep reinforcement learning,” *Cornell*

arXiv:1312.5602, pp. 1–9, 2013, [Online]. Available: http://arxiv.org/abs/1312.5602

[46] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[47] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[48] D. Silver *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *Cornell arXiv:1712.1815,* pp. 1–19, 2017, [Online]. Available: http://arxiv.org/abs/1712.01815

[49] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," *31st International Conference on Machine Learning, ICML 2014*, vol. 1, pp. 605–619, 2014.

[50] A. Hassan, A. U. Rehman, N. Shabbir, S. R. Hassan, M. T. Sadiq, and J. Arshad, "Impact of inertial response for the variable speed wind turbine," *2019 International Conference on Engineering and Emerging Technologies, ICEET 2019*, no. 10, pp. 1–6, 2019.

[51] P. Chen, Z. He, C. Chen, and J. Xu, "Control strategy of speed servo systems based on deep reinforcement learning," *Algorithms*, vol. 11, no. 5, 2018.

[52] R. Yu, Z. Shi, C. Huang, T. Li, and Q. Ma, "Deep reinforcement learning based optimal trajectory tracking control of autonomous underwater vehicle," *Chinese Control Conference, CCC*, no. July 2017, pp. 4958–4965, 2017.

[53] M. Gheisarnejad and M. H. Khooban, "An intelligent non-integer pid controller-based deep reinforcement learning: implementation and experimental results," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 4, pp. 3609–3618, 2021.

[54] S. Fujimoto, H. Van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *35th International Conference on Machine Learning, ICML 2018*, vol. 4, pp. 2587–2601, 2018.

[55] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2017, no. 9, pp.

23–30, 2017.

[56] F. Muro, *Air Tractor AT-802 | Aviation Photo #6176597 | Airliners.net.* [Online]. Available: https://www.airliners.net/photo/Untitled/Air-Tractor-AT-802/6176597/L?qsp=eJxtjTEOwjAQBP9ytZskAiF3pKGEgg%2BczgtYCol1viJRlL9jHImKbjW7ml1JptEw231JIE8ZrPIiR4mV35n8ShxVlB/Wc46yz05N4378ghH6tyHftt3BUZ7U%2BqXYAxvOIkiGQDu/aoB%2BK2Spt89iKY4EvdVM3bHwEHMauDpgHAfatg9TUT36 (accessed Aug. 25, 2022).

[57] A. H. León, *Air Tractor AT-802 - FAASA Chile | Aviation Photo #2101778 | Airliners.net.* [Online]. Available: https://www.airliners.net/photo/FAASA-Chile/Air-Tractor-AT-802/2101778 (accessed Aug. 25, 2022).

[58] M. Attar and M. Dabirian, "Reinforcement Learning for Learning of Dynamical Systems in Uncertain Environment: a Tutorial," *Cornell arXiv:1905.07727* pp. 1–27, 2019, [Online]. Available: http://arxiv.org/abs/1905.07727

[59] H. Modares and F. L. Lewis, "Linear quadratic tracking control of partially-unknown continuous-time systems using reinforcement learning," *IEEE Transactions Automation and Control*, vol. 59, no. 11, pp. 3051–3056, 2014.

[60] A. M. Annaswamy, A. Guha, Y. Cui, S. Tang, P. A. Fisher, and J. E. Gaudio, "Integration of adaptive control and reinforcement learning for real-time control and learning," *Cornell arXiv:*2105.06577*,* 2021, pp 1-39 [Online]. Available: http://arxiv.org/abs/2105.06577

[61] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel, "Trust region policy optimization," *32nd International Conference on Machine Learning ICML 2015*, vol. 3, pp. 1889–1897, 2015.

[62] S. Curi, F. Berkenkamp, and A. Krause, "Efficient model-based reinforcement learning through optimistic policy search and planning," *Advances in Neural Information Processing Systems*, vol. 2020-December, no. NeurIPS, 2020.

[63] J. Arshad *et al.*, "Deep deterministic policy gradient to regulate feedback control systems using reinforcement learning," *Computers. Materials. Continua*, vol. 71, no. 1, pp. 1153–1169, 2022.

[64] B. O'Donoghue, I. Osband, R. Munos, and V. Mnih, "The uncertainty Bellman equation

and exploration," *35th International Conference on Machine Learning, ICML 2018*, vol. 9, pp. 6154–6173, 2018.

[65] J. Arshad *et al.*, "Deep deterministic policy gradient to regulate feedback control systems using reinforcement learning," *Computers, Materials and Continua*, vol. 71, no. 1, pp. 1153–1169, 2022.

[66] A. Agostini and E. Celaya, "Exploiting domain symmetries in reinforcement learning with continuous state and action spaces," *8th International Conference on Machine Learning and Applications, ICMLA 2009*, pp. 331–336, 2009.

[67] John N. Tsitsiklis, "An analysis of temporal-difference learning with function approximation," *IEEE Transactions on Automatic Control*, vol. 42, no. 5, pp. 32–33, 37, 1997.

[68] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[69] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, Massachusetts: MIT Press 2020.

[70] D. Zha, K. H. Lai, K. Zhou, and X. Hu, "Experience replay optimization," *IJCAI International Joint Conference on Artificial Intelligence*, vol. 2019, no. 8, pp. 4243–4249, 2019.

[71] X. Peng, L. Li, and F. Y. Wang, "Accelerating minibatch stochastic gradient descent using typicality sampling," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 11, pp. 4649–4659, 2020.

[72] W. Fedus *et al.*, "Revisiting fundamentals of experience replay," *37th International Conference on Machine Learning, ICML 2020*, vol. PartF16814, pp. 3042–3052, 2020.

[73] D. Horgan *et al.*, "Distributed prioritized experience replay," *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, pp. 1–19, 2018.

[74] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *3rd*

International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, pp. 1–15, 2015.

[75] S. Ruder, "An overview of gradient descent optimization algorithms," *Cornell arXiv:1609.04747,* pp. 1–14, 2016, [Online]. Available: http://arxiv.org/abs/1609.04747

[76] A. Vemula, W. Sun, and J. A. Bagnell, "Exploration in action space," 2020, [Online]. Available: http://arxiv.org/abs/2004.00500

[77] J. Hollenstein, S. Auddy, M. Saveriano, E. Renaudo, and J. Piater, "Action noise in off-policy deep reinforcement learning: impact on exploration and performance," *Cornell arXiv:* 2206.03787*,* pp. 1–27, 2022, [Online]. Available: http://arxiv.org/abs/2206.03787

[78] F. Pardo, A. Tavakoli, V. Levdik, and P. Kormushev, "Time limits in reinforcement learning," *35th International Conference on Machine Learning, ICML 2018*, vol. 9, pp. 6443–6452, 2018.

[79] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," *2020 IEEE Symposium Series on Computational Intelligence, SSCI 2020*, pp. 737–744, 2020.

[80] R. Ramakrishnan, E. Kamar, D. Dey, E. Horvitz, and J. Shah, "Blind spot detection for safe sim-to-real transfer," *Journal of Artificial Intelligence Research*, vol. 67, pp. 191–234, 2020.

[81] C. Rizzardo, F. Chen, and D. Caldwell, "Sim-to-real via latent prediction: Transferring visual non-prehensile manipulation policies," *Frontiers in Robotics and AI*, vol. 9, no. 1, pp. 1–14, 2023.

[82] R. Vrabič *et al.*, "An architecture for sim-to-real and real-to-sim experimentation in robotic systems," *Procedia CIRP*, vol. 104, no. March, pp. 336–341, 2021.

[83] X. Bin Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3803–3810, 2018.

[84] W. H. Lee, "Lagrangian method," *Computer simulation of shaped charge problems*, pp. 5–30, 2006.

[85] Manual - Quanser Inc., *Smart Structure Control*, 2005.

[86] Manual - Quanser Inc., *Modeling and Vibration Control,* 2005.

[87] Manual - OpenAI, *AIGym - Release 0.24.1*, 2022. [Online] Available: https://github.com/openai/gym/releases/tag/0.24.1

[88] Manual - Tensorflow, *TensorFlow Core*. [Online]. Available: https://www.tensorflow.org/guide (accessed Aug. 22, 2022).

[89] Manual - Keras, *Keras API reference*. [Online]. Available: https://keras.io/api/ (accessed Aug. 22, 2022).

[90] Y.-H. Wu, F.-Y. Sun, Y.-Y. Chang, and S.-D. Lin, "ANS: adaptive network scaling for deep rectifier reinforcement learning models," *Cornell arXiv:1809.02112,* 2018, [Online]. Available: http://arxiv.org/abs/1809.02112

[91] D. Lee, "Comparison of reinforcement learning activation functions to improve the performance of the racing game learning agent," *Journal of Information Processing Systems*, vol. 16, no. 5, pp. 1074–1082, 2020.

Appendix A: Experimental Data

Table A.1: Complete experimental results - No mass blocks

Mass @ 0 mm		Settling Time (s)	Change (%)	Overshoot (mm)	Change (%)	Mean Error (mm)	Change (%)
Trial 1	Controller 1	2.52	-	35.39	-	13.52	-
	Controller 2	2.02	-19.87%	41.84	18.22%	16.79	24.27%
	Controller 3	1.92	-23.84%	28.47	-19.55%	13.17	-2.57%
Trial 2	Controller 1	2.62	-	35.34	-	13.22	-
	Controller 2	2.00	-23.57%	41.51	17.48%	16.81	27.14%
	Controller 3	2.07	-21.02%	20.92	-40.81%	12.07	-8.67%
Trial 3	Controller 1	2.52	-	35.78	-	13.65	-
	Controller 2	2.07	-17.88%	46.94	31.20%	16.89	23.74%
	Controller 3	1.67	-33.77%	26.55	-25.79%	14.24	4.31%
Trial 4	Controller 1	2.58	-	35.90	-	13.03	-
	Controller 2	2.08	-19.35%	47.49	32.27%	16.37	25.60%
	Controller 3	1.68	-34.84%	25.72	-28.36%	14.33	9.95%

Table A.2: Complete experimental results – Mass blocks @ 100mm position

Mass @ 100 mm		Settling Time (s)	Change (%)	Overshoot (mm)	Change (%)	Mean Error (mm)	Change (%)
Trial 1	Controller 1	2.67	-	34.51	-	13.34	-
	Controller 2	2.12	-20.63%	45.64	32.24%	15.87	19.03%
	Controller 3	2.73	2.50%	25.27	-26.77%	11.37	-14.71%
Trial 2	Controller 1	2.65	-	34.79	-	13.44	-
	Controller 2	2.68	1.26%	49.33	41.80%	15.58	15.94%
	Controller 3	2.07	-22.01%	25.34	-27.14%	12.81	-4.66%
Trial 3	Controller 1	2.65	-	35.16	-	13.37	-
	Controller 2	2.20	-16.98%	45.72	30.05%	17.55	31.31%
	Controller 3	2.13	-19.50%	24.51	-30.28%	13.30	-0.48%
Trial 4	Controller 1	2.70	-	35.84	-	13.41	-
	Controller 2	2.18	-19.14%	45.89	28.06%	16.95	26.38%
	Controller 3	2.05	-24.07%	25.60	-28.57%	12.97	-3.29%

Table A.3: Complete experimental results – Mass blocks @ 150mm position

Mass @ 150 mm		Settling Time (s)	Change (%)	Overshoot (mm)	Change (%)	Mean Error (mm)	Change (%)
Trial 1	Controller 1	2.65	-	34.57	-	13.10	-
	Controller 2	2.12	-20.13%	46.72	35.12%	16.52	26.07%
	Controller 3	2.03	-23.27%	27.21	-21.31%	13.44	2.57%
Trial 2	Controller 1	2.67	-	34.89	-	13.30	-
	Controller 2	2.08	-21.88%	47.11	35.00%	16.49	23.95%
	Controller 3	2.02	-24.38%	27.29	-21.79%	13.20	-0.72%
Trial 3	Controller 1	2.60	-	34.80	-	13.54	-
	Controller 2	2.08	-19.87%	41.12	18.18%	16.87	24.60%
	Controller 3	2.08	-19.87%	23.81	-31.58%	13.04	-3.72%
Trial 4	Controller 1	2.60	-	34.98	-	13.54	-
	Controller 2	2.15	-17.31%	46.51	32.96%	16.82	24.25%
	Controller 3	2.70	3.85%	19.89	-43.14%	10.60	-21.69%

Table A.4: Complete experimental results – Mass blocks @ 200mm position

Mass @ 200 mm		Settling Time (s)	Change (%)	Overshoot (mm)	Change (%)	Mean Error (mm)	Change (%)
Trial 1	Controller 1	2.60	-	35.41	-	13.47	-
	Controller 2	2.08	-19.87%	46.48	31.28%	15.82	17.48%
	Controller 3	1.93	-25.64%	27.23	-23.09%	13.15	-2.34%
Trial 2	Controller 1	2.60	-	35.15	-	13.43	-
	Controller 2	2.07	-20.51%	40.56	15.38%	16.61	23.62%
	Controller 3	2.02	-22.44%	19.28	-45.15%	12.38	-7.84%
Trial 3	Controller 1	2.62	-	35.64	-	13.23	-
	Controller 2	2.12	-19.11%	39.78	11.63%	18.14	37.14%
	Controller 3	1.93	-26.11%	27.44	-23.01%	13.19	-0.30%
Trial 4	Controller 1	2.65	-	35.69	-	13.21	-
	Controller 2	2.12	-20.13%	47.37	32.71%	16.43	24.36%
	Controller 3	2.02	-23.90%	26.89	-24.67%	12.92	-2.19%