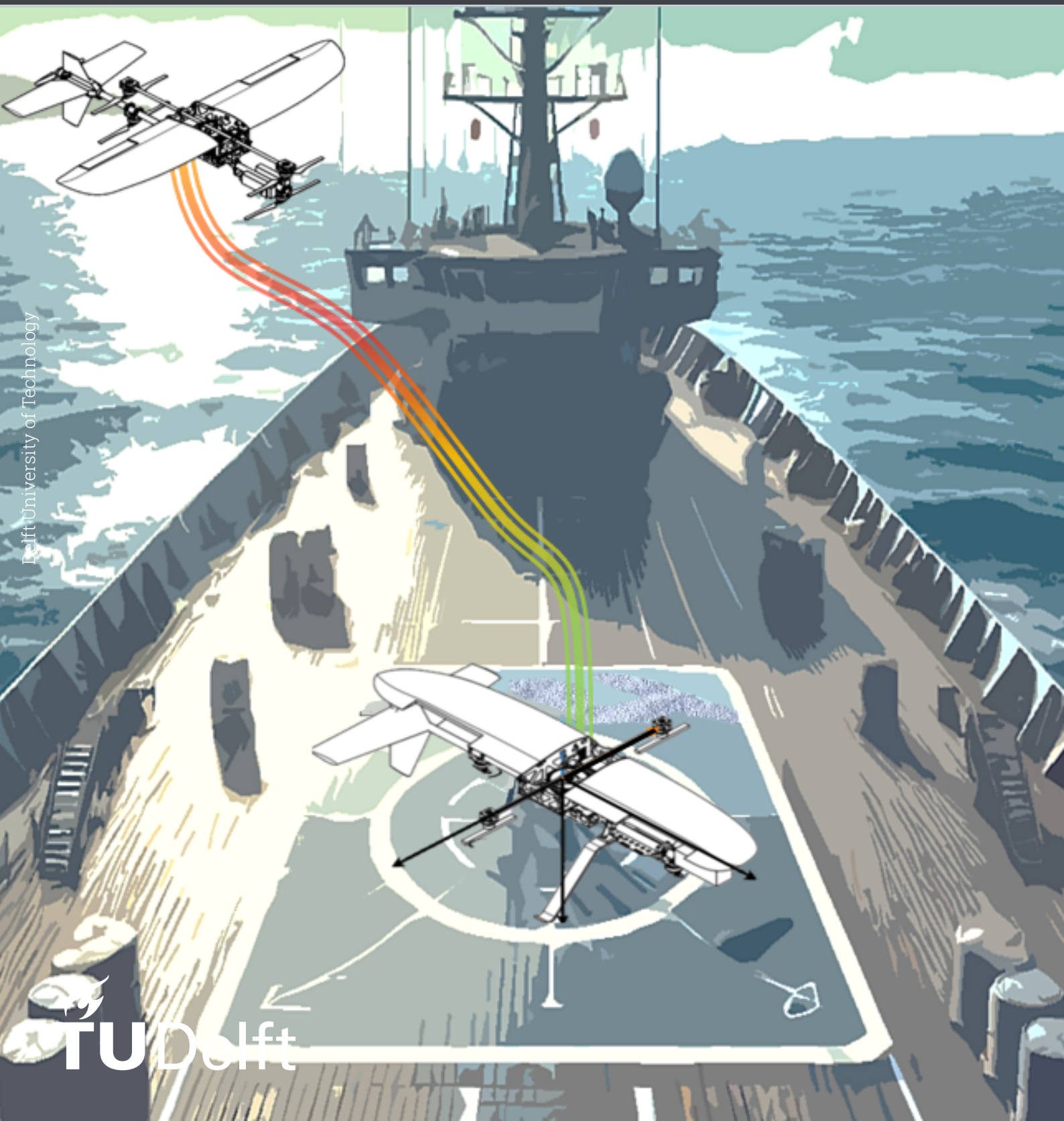
Reinforcement Learning for Landing the Variable Skew Quad Plane on a Moving Platform

Achieving Optimal Guidance for Ship Landings

Cansu Yıkılmaz



Delft University of Technology

Reinforcement Learning for Landing the Variable Skew Quad Plane on a Moving Platform

Achieving Optimal Guidance for Ship Landings

by

Cansu Yıkılmaz

to obtain the degree of Master of Science
at the Delft University of Technology
to be defended publicly on August 14, 2024

Supervisor:	Dr.ir. Christophe de Wagter
Chair:	Dr.ir. Ewoud J.J. Smeur
External Examiner:	Dr.ir. Francesca De Domenico
Project Duration:	December, 2023 - August, 2024
Faculty:	Faculty of Aerospace Engineering, Delft

TU Delft

Acknowledgements

With this work, I put an end to one of the most chaotic, unpredictable, emotional, and hardest journeys of my life. Looking back, it would have been impossible to foresee all the things that happened in these two years. Standing at the end of it, closing this chapter hits so hard but also so soft at the same time.

I am proud to have completed the master's program with this thesis, especially given the limited time and challenging circumstances. Studying this particular topic was both difficult and immensely rewarding. I must admit that the landing animations exceeded my expectations, and I often found myself admiring them as if they were pieces of art. This achievement is thanks to my supervisor, Dr. Christophe De Wagter, who has been very understanding of my constraints and flexible with the outcomes, always kind and respectful towards me.

I also want thank Dr. Ewoud Smeur for his valuable feedback on the model, and PhD students Robin Ferede and Tomaso de Ponti for their help in implementation.

On a personal note, I want to first acknowledge my family, who are exhausted from watching me go through all this. I guess the summer when I will have no assignments to deal with has finally arrived. Thank you for supporting and believing in me no matter what.

My good friend Ege, thank you for being with me on this journey, always ready to listen and help. Thank you for convincing me to apply when I was about to give up and maybe being the reason for my presence here. Thank you for being the only person who knows and understands why I inverted that axis at that time and why I once acted frozen in the most random meeting. Sharing life's cringiest moments with you is always a pleasure. I will always look up to you as an engineer and be proud of everything you do.

Finally, my dear Ata, I want to thank you from the deepest parts of my heart for being there for me and cooking my favourite dessert at the hardest times. Thank you for being my home and family in Delft.

I want to conclude this chapter with words for the future me since I will probably be the only one reading this. These two years were everything but easy. I just hope that I can find the same strength again to go through life when the time necessitates.

Cansu Yıkılmaz
Delft, August 2024

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

Abbreviation	Definition
ANDI	Adaptive Nonlinear Dynamic Inversion
ANN	Artificial Neural Networks
DDPG	Deep Deterministic Policy Gradient
DNN	Deep Neural Networks
DQN	Deep Q-Networks
G&NC	Guidance and Control Networks
GPS	Global Positioning System
INDI	Incremental Nonlinear Dynamic Inversion
INS	Inertial Navigation System
LMS	Least Mean Squares
MAVLab	Micro Air Vehicle Laboratory
MDP	Markov Decision Process
MOC	Mass Optimal Control
MPC	Model Predictive Control
NLP	Nonlinear Programming
PID	Proportional - Integral - Derivative
PID_{abs}	Proportional - Integral - Derivative for absolute velocity
PID_{rel}	Proportional - Integral - Derivative for relative velocity
PPO	Proximal Policy Optimization
POMDP	Partially Observable Markov Decision Process
OFW	Oblique Flying Wing
QC	Quadratic Control
QUAD	Quadcopter
RL	Reinforcement Learning
RMSE	Root Mean Square Error
RQ-AL	Research Questions for Autonomous Landing
RQ-ML	Research Questions for Machine Learning
RQ-PF	Research Questions for Problem Formulation
RQ-EV	Research Questions for Evaluation
RWSC	Reaction Wheel Spacecraft
SAC	Soft Actor Critic
SQP	Sequential Quadratic Programming
SSC	Simple Spacecraft
SAC	Soft Actor Critic
TPBVP	Two Point Boundary Value Problem
TRPO	Trust Region Policy Optimization
TOC	Time Optimal Control
TSR	Tracking Success Rate
TVR	Thrust Vectoring Rocket
VSQP	Variable Skew Quad Plane
VTOL	Vertical Take-off and Landing
UAV	Unmanned Aerial Vehicle
WLS	Weighted Least Squares

Symbols

Symbol	Definition	Unit
A_{p_i}	Amplitude of the platform for each axis i	[m]
a_n	Acceleration in north	[m/s^2]
a_e	Acceleration in east	[m/s^2]
a_d	Acceleration in down	[m/s^2]
$a_{n_{des}}$	Desired acceleration in north	[m/s^2]
$a_{e_{des}}$	Desired acceleration in east	[m/s^2]
$a_{d_{des}}$	Desired acceleration in down	[m/s^2]
$\boldsymbol{a}_d$	Drone acceleration vector	[m/s^2]
$\dot{\boldsymbol{a}}_d$	Drone 1st order acceleration vector	[m/s^3]
$\ddot{\boldsymbol{a}}_d$	Drone 2nd order acceleration vector	[m/s^4]
$\dddot{\boldsymbol{a}}_d$	Drone 3rd order acceleration vector	[m/s^5]
h	Height	[m]
h_t	Reference height	[m]
k_1	The vertical plane coefficient	[-]
k_2	The horizontal plane coefficient	[-]
k_3	The vertical plane coefficient for the relative velocity collision penalty	[-]
k_4	The horizontal plane coefficient for the relative velocity collision penalty	[-]
k_5	The vertical plane coefficient for the absolute velocity collision penalty	[-]
k_6	The horizontal plane coefficient for the absolute velocity collision penalty	[-]
p_n	Position in north	[m]
p_e	Position in east	[m]
p_d	Position in down	[m]
$p_{r_{\text{old}_v}}$	Old relative position in vertical plane	[m]
$p_{r_{\text{new}_v}}$	New relative position in vertical plane	[m]
$p_{r_{\text{old}_h}}$	Old relative position in horizontal plane	[m]
$p_{r_{\text{new}_h}}$	New relative position in horizontal plane	[m]
$\boldsymbol{p_d}$	Drone position vector	[m]
$\boldsymbol{p_r}$	Relative position vector	[m]
$\boldsymbol{p_s}$	Ship position vector	[m]
R_p	Position reward	[m]
R_v	Velocity reward	[m/s]
R_{c1}	First collision penalty	[m/s]
R_{c2}	Second collision penalty	[m/s]
v_n	Velocity in north	[m/s]
v_e	Velocity in east	[m/s]
v_d	Velocity in down	[m/s]
$\boldsymbol{v_d}$	Drone velocity vector	[m/s]
$\boldsymbol{v_r}$	Relative velocity vector	[m/s]
$\boldsymbol{v_s}$	Ship velocity vector	[m/s]
$v_{r_{\text{old}_v}}$	Old relative velocity in vertical plane	[m/s]
$v_{r_{\text{new}_v}}$	New relative velocity in vertical plane	[m/s]
$v_{r_{\text{old}_h}}$	Old relative velocity in horizontal plane	[m/s]
$v_{r_{\text{new}_h}}$	New relative velocity in horizontal plane	[m/s]
w_i	Angular frequency for each axis i	[Hz]
ϕ	Roll angle	[rad]
ψ	Yaw angle	[rad]
θ	Pitch angle	[rad]
μ_i	Random walk component	[-]

1

Introduction

Unmanned Aerial Vehicles have become increasingly popular and found applications in many areas including military operations, search and rescue, delivery services, wireless communication, and aerial surveillance. The usability of UAVs depends on their flight capabilities across various flight phases: take-off, climb, cruise, descent, and notably, landing. Most UAVs are capable of autonomous take-off and cruise, but autonomous landing, especially on moving targets like ships, is a more challenging task. To achieve a successful autonomous landing, the UAV system must operate close to perfection in a limited time and space to avoid the risks of crashes or operational failures. The risks associated with landing mainly depend on factors such as the type of landing (indoor or outdoor), the design of the UAV (fixed-wing, rotary-wing, or hybrid), and the characteristics of the landing target (stationary or moving). Overall, the landing problem is defined under considerations and limitations specific to the system and operation.

This report studies the problem of landing on a moving target (ship) for the Variable Skew Quad Plane (VSQP), a hybrid UAV developed by the Micro Air Vehicle Laboratory at TU Delft. The VSQP combines the capabilities of rotary-wing UAVs and fixed-wing planes into one, capable of operating in hover mode like a quadrotor and in forward flight mode like a fixed-wing plane. The hybrid design of the VSQP provides several operational advantages, such as improved cruise efficiency, reduced drag in forward flight, and enhanced control authority in hover mode.

The main research objective is to develop an optimal guidance policy for trajectory generation and tracking to land the VSQP precisely on a moving ship. This is a highly challenging task as it involves landing on a moving target for a hybrid drone. Landing on a ship involves dealing with a constantly moving target affected by arbitrary factors like sea state, wind, and the ship's maneuvers. These factors introduce additional variables that must be accounted for in the VSQP's guidance system. The unpredictable movements of the ship, along with strong wing gusts acting on the UAV, require a highly adaptive yet robust approach to landing.

The current guidance and control scheme of the VSQP consists of a one-loop architecture that applies the Adaptive Incremental Nonlinear Dynamic Inversion approach (ANDI). ANDI based control includes a Weighted Least Squares (WLS) routine for allocating the control signals such that the overall control effort and the difference between the demanded and achieved control are minimized while satisfying the priorities set by the weighting factors. The inputs to the WLS are sent from the reference models through error controllers defined for both the position and attitude of the drone. The overall system accepts the position waypoints as inputs and tries to find the optimal control inputs via ANDI accordingly. Although ANDI is highly efficient in terms of tracking and disturbance avoidance, the new hard-to-predict parameters introduced by the ship require more effective guidance algorithm that considers the ship's specific movements before determining the desired position.

Given the challenges related to the landing problem and the limitations of the current guidance and control scheme, this study explores the potential of reinforcement learning as a solution for calculating the optimal guidance inputs to the inner controller. In recent years, reinforcement learning has emerged as a faster and more computationally efficient option for a wide variety of optimal control problems. Instead of relying on intense numerical solvers for computing an optimal trajectory at every time step, a more adaptable and flexible reinforcement learning architecture offers solutions to both the trajectory

generation and tracking problems. This thesis aims to employ the capabilities of reinforcement learning to develop an optimal guidance strategy for the VSQP, ensuring its safe and precise landing on a moving ship, which is highly critical in extending the operational versatility of UAVs.

The report is structured as follows: Chapter 2, outlines the overall research framework, research proposal, research objective and research questions under the topic of the thesis. This chapter is particularly important as it also outlines the overall structure of the report, with each section corresponding to the related research questions. Chapter 4 introduces the landing problem for UAVs, with a specific focus on ship landings. The challenges and key phases of autonomous landing are given in the sub-sections. This section investigates the landing phase from both a general perspective and in terms of its sub-phases. In Chapter 5, machine learning techniques, including supervised, unsupervised, and reinforcement learning, are investigated as solutions to optimal control problems. In this context, reinforcement learning has been the focus for landing problems, with state-of-art algorithms and their corresponding real life applications discussed. With the details of the guidance and control structure of the Variable Skew Quad Plane taken into consideration in Chapter 6, literature review part is completed. The preliminary analysis and the implementation details are given in Chapter 7. The thesis concludes with the conclusion Chapter 8 and additional work is provided in Appendix.

2

Research Plan

This section provides an overview of the research plan, including the research proposal, research objective, and research questions. The literature review will be based on these research questions and have the purpose of answering them.

Research Proposal

The Variable Skew Quad Plane (VSQP) is a hybrid drone designed by the Micro Air Vehicle Laboratory (MAVLab) at TU Delft. It can function as both a fixed-wing plane and a quadcopter, depending on its skew angle. The current model of the VSQP uses a one-loop adaptive incremental nonlinear dynamic inversion-based controller, which employs a Weighted Least Squares (WLS) routine for optimally allocating control tasks among actuators. The drone's task is to land precisely on moving targets, particularly ship platforms. Achieving this requires real-time trajectory planning, necessitating the formulation of an optimal guidance policy. GN&C networks have emerged as faster, more computationally efficient structures for calculating optimal control inputs compared to traditional techniques, which often rely on computationally intensive numerical solvers. In this thesis, GN&C networks will be explored for achieving optimal guidance for landing the VSQP.

Research Objective

Constructing a good research objective is crucial as it provides a general but clear purpose for the study. The research objective for this work is given as follows:

Research Objective

This research aims to develop an optimal guidance policy by exploring the use of learning based approaches for a hybrid drone (VSQP). By doing so, it is expected to have optimal control inputs (accelerations) that will be fed back to the existing adaptive incremental nonlinear dynamic inversion based controller and improve the overall performance.

Research Questions

This section outlines the research questions that will be answered throughout the report. The questions are formulated to define the scope of the project by partitioning the research objective into several sub-tasks. It is expected that answering these questions will achieve the research objective.

Autonomous landing is a significant part of a standard UAV flight. While it has great importance as a whole, landing for a UAV can also be divided into several phases such as target detection, state estimation, tracking, and landing. Understanding these phases is crucial to clearly indicate what the potential solution addresses. Therefore, the first set of research questions will define the landing problem in general, classify landing in terms of platform characteristics, discuss the challenges associated with landing, and highlight the limitations of conventional methods to motivate learning-based approaches. These questions are labelled as **RQ-AL**.

Research Questions (RQ-AL) - Autonomous landing for UAVs

RQ-AL 1. How is the autonomous landing procedure is defined in the literature?
RQ-AL 2. What are biggest challenges associated with autonomous landing?
RQ-AL 3. What type of methods are used for the landing problem and what conventional methods lack so that it creates a specific need for the learning based approaches?

The second set of questions, labelled as **RQ-ML**, investigates the applicability of different machine learning algorithms to the landing problem. Specifically, GN&C networks have been used for certain guidance and control problems, employing supervised and reinforcement learning techniques. For this thesis, the methodology regarding how these networks should learn remains an open question. While linked to the nature of the problem, these questions are expected to shape the primary methodology and its details. This set of questions with the previous one lays the foundation for the literature review.

Research Questions (RQ-ML) - Machine Learning for Optimal Control

RQ-ML 1. Which machine learning technique suits better for landing the VSQP on a moving platform?
RQ-ML 2. How is an objective function is structured for the landing problem?
RQ-ML 3 What metrics are used to asses the performance of the landing?

The third set of questions determines the methodology and its specifics. These questions develop the learning-based problem formulation for landing the VSQP, specifically questioning how this new black-box model would be placed in the loop and the necessary inputs and outputs for it. Since the landing problem requires consideration of two different dynamics (the drone and the platform), it is important to answer questions regarding the states of these dynamics for the problem's formulation. These questions are given under the category of *"Learning based Problem Formulation for the VSQP"* labelled as **RQ-PF**.

Research Questions (RQ-PF) - Learning based Problem Formulation for the VSQP

RQ-PF 1. How the black box model should be integrated into the loop?
RQ-PF 1.1. Considering the existing control architecture and characteristics of the given landing problem, what should be the inputs and outputs to the system?
RQ-PF 1.2. How the response of the controller will be modelled for an overactuated system such as the VSQP, especially for training purposes?
RQ-PF 2. How the landing platform will be modelled for the simulations?
RQ-PF 2.1. What states will be considered for the platform?
RQ-PF 2.2. How is the model for the target platform will be integrated with the rest of the system (drone)?

The final set of questions verifies and validates the proposed technique. They specifically inquire about the parameters used to evaluate the drone's performance for the defined landing problem. Moreover, they set a benchmark for the proposed solution for further validation purposes. These questions are given under the category of "*Evaluation of the proposed framework*" and labelled as **RQ-EV**..

Research Questions - Evaluation of the proposed framework

RQ-EV 1. How could the proposed solution (ex.reinforcement learning) be verified and validated?
RQ-EV 2. What type of performance metrics would be applicable to the designed framework?
RQ-EV 3. Can the proposed solution be compared to a benchmark controller and if so, what would be the choice for it?

Research Plan

Considering the research objective and the related research questions, the comprehensive thesis plan is outlined in the Gantt chart shown in Figure 2.1 .

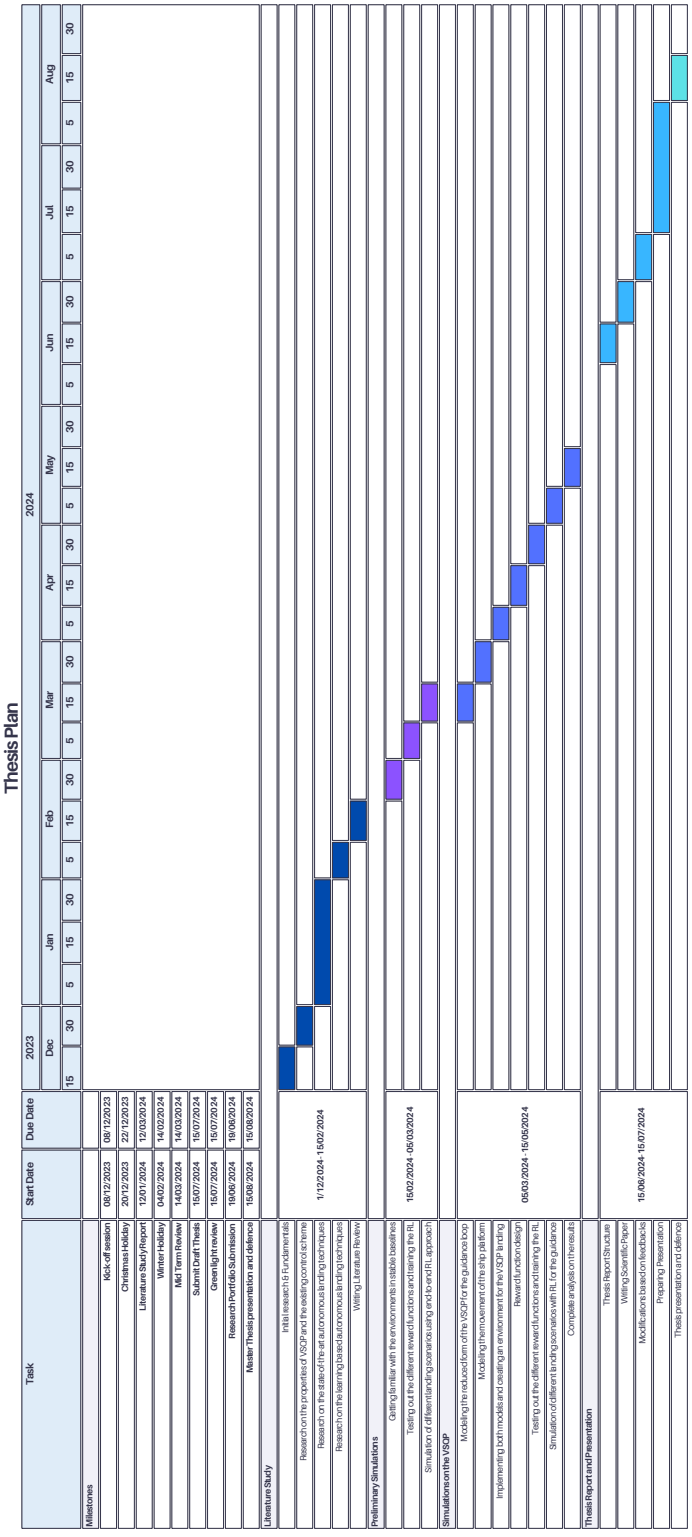


Figure 2.1: Gantt chart for the thesis plan

3

Scientific Article

Reinforcement Learning based Optimal Guidance for Landing the Variable Skew Quad Plane on a Ship

Cansu Yıkılmaz
Delft University of Technology, Delft, The Netherlands
Faculty of Aerospace Engineering, Control & Simulation

ABSTRACT

In this work, we present a reinforcement learning based approach for optimal guidance in landing a Variable Skew Quad Plane (VSQP) on a moving ship platform. We develop a reinforcement learning framework that computes the optimal acceleration inputs for the inner adaptive incremental nonlinear dynamic inversion based controller. Through extensive simulations, we assess the performance of different reward function combinations based on key performance metrics: touchdown velocity, deviation, and duration. Having identified the most optimal form of the reward for the given landing problem, we further validate our approach on real ship data that incorporates dynamics that do not exist in the training set. Our results indicate that the reinforcement learning based approach outperforms the benchmark PID controller in achieving smoother and safer landings even under complex motion characteristics.

1 INTRODUCTION

Unmanned Aerial Vehicles have found use in a wide variety of applications including environmental monitoring, search and rescue missions, and surveillance [1, 2] due to their capability to collect real-time data while being able to operate in remote and hazardous locations. However, the operational range of UAVs, particularly in highly dangerous sea environments, is often limited. This limitation necessitates the recovery of the UAV on the ship which in turn requires an advanced autonomous landing technology that is a combination of highly accurate sensing, guidance, and control techniques.

The complexity of autonomous landing significantly exceeds that of other flight phases such as takeoff, climb, and cruise. This is due to the fact that obtaining accurate measurements or close to optimal estimates of both the UAV and the target while ensuring a robust trajectory following is an extremely challenging task. The complexity is even further amplified in dynamic scenes where the target is moving in an unpredictable manner under the effect of external disturbances and uncertainties [3]. This becomes particularly evident in the case of ship landing, where the unpredictability of the sea environment adds an extra layer of difficulty to the problem.

An autonomous landing problem has been extensively studied [4, 5, 6, 7, 8, 3] in terms of its sub-phases: target detection, relative state estimation, tracking and landing. The majority of studies relied on using vision-based systems for target detection and relative state estimation tasks with methods differing according to the characteristics of the landing platform (static vs dynamic). These systems are specifically preferred in maritime operations [9, 10] since reliance on inertial navigation is not feasible due to concerns related to the accuracy of the measurements and security.

In the context of tracking and landing, the classical control techniques for autonomous landing have relied on a variety of methods. These include PID controllers [8, 11, 12], adaptive control techniques [13, 14], robust flight [15] and model predictive control [16, 17] methods. While these techniques are effective in certain cases, they come with their own set of challenges.

PID controllers often face problems with robustness and flexibility due to dependency on fixed gains and perform poorly under the effect of external disturbances [18]. More advanced controllers like model predictive and robust flight control techniques require a detailed model and obtaining a perfect representation of the real-world system is often challenging. Moreover, the full formulation of the landing as an optimal control problem to be solved with numerical methods introduces additional complexities. Such a formulation is not only computationally intense but also quite challenging due to the dynamic motion characteristics of the target. These facts highlight the need for more efficient and robust solutions in autonomous landing.

Reinforcement learning as a machine learning algorithm has gained significant popularity in recent years and found use in solving guidance and control problems [19, 20]. In the context of optimality, the progress of reinforcement learning has been towards filling the existing gaps in traditional methods. Optimal control techniques often require to have complete knowledge of the system dynamics which may not always be feasible when uncertainties and disturbances are involved. However, reinforcement learning offers a robust framework for dealing with such changes due to its ability to

learn from raw environmental data without having prior information on the system and adapt to unknown dynamics and unpredictable changes in the environment. This attribute of reinforcement learning makes it highly suitable for complex control problems as in the case of ship landing.

In this work, we use reinforcement learning, particularly Proximal Policy Optimization (PPO) to develop an optimal guidance policy for landing the Variable Skew Quad Plane (VSQP) on a moving platform (ship). By doing so, we compute the optimal control inputs (accelerations) and feed them back into the existing adaptive incremental nonlinear dynamic inversion based controller which includes a Weighted Least Square (WLS) based optimization routine for control allocation. Against most approaches in the literature, we account for the motion in the z direction and search for the best objective function through a comparison between different reward combinations. Finally, we test our approach with real-world ship data and compare it with a PID controller.

2 Related Work

Reinforcement learning has recently become an effective approach for solving autonomous landing problems. One of the earliest studies by Polvara et al. [21] utilized Deep Q-Networks (DQN) for landing on a static platform using low-resolution images coming from a downward facing camera. They have divided the landing task into landmark detection and vertical descent, with each managed by separate DQNs that can communicate with each other via an internal trigger. Later on, Lee et al. [22] developed an actor-critic framework where they included a PID based inner attitude controller, a ground-looking camera model, and a laser rangefinder. Their approach focused on generating appropriate roll and pitch commands for successful tracking and landing while for altitude and heading they provided constant control commands.

These studies considered the platform to be stationary, however, real-life situations (ship landing) require UAV landing to be on moving platforms. The challenge of landing on a dynamic target was addressed by Rodriguez et al. in [23] where they used Deep Deterministic Policy Gradient (DDPG) in combination with the Gazebo platform for simulations. Although their study was successful in the implementation, it is claimed in [24] that not considering motion in the z-axis in the problem definition and weak generalization capability of the Gazebo platform result in autonomously incapable agents. In their study, Xie et al. divide the problem into two parts: perception and relative pose estimation; trajectory optimization and control, and only consider the latter. In contrast to the previous studies, they have also accounted for sensor noise, intermittent measurements, randomness in UAV movement, and, incomplete or inaccurate observations. Moreover, they explored a hybrid approach that combined reinforcement learning with heuristic methods for tracking and landing tasks. While tracking part of the hybrid strategy is accomplished with DDPG in an end-to-end way, the landing task is assumed to be successful if the vertical distance is less than 0.1m while the horizontal distance is less than 0.8m.

In the context of ship landing, Saj et al. [25] claim that previous studies lack the robustness component which leads to the development of not fully efficient algorithms for ship landing problems. Considering the associated challenges, they adapted the twin delayed DDPG (TD3) algorithm for a vision-based ship landing. To overcome the reality gap problem, they have applied the domain randomization approach [26] through a variation of environment parameters. They showcased the superior performance of their framework over a PID controller in the real-life tests made under varying wind conditions.

These studies collectively highlight the capabilities of reinforcement learning in solving a variety of autonomous landing problems.

3 Variable Skew Quad Plane (VSQP)

The Variable Skew Quad Plane is a hybrid UAV designed by the Micro Air Vehicle Laboratory (MAVLab) at TU Delft. The VSQP has two modes of operation - hover and cruise which is controlled through skew angle Λ. In hover mode ($\Lambda = 0\,\mathrm{deg}$), the drone operates as a quadrotor and is controlled using differential thrust attitude. In the cruise mode ($\lambda = 90$), it operates as a quad-plane and achieves forward speed with a push propeller located at the tail. The VSQP, with changing skew angle uses the concept of Oblique Flying Wing (OFW) plane. This design offers better gust rejection in hover mode and lower drug during cruise mode, providing improved packability and ease of operation. The configuration modes of the VSQP are shown in Figure 1. Note that during landing, the VSQP operates as a quadrotor with its skew angle set to zero, aligning its wings with the body, making it more stable in case of external disturbances and allowing a safer landing.

The current guidance and control model of the VSQP incorporates an Adaptive Incremental Nonlinear Dynamic Inversion (ANDI) controller with the Weighted Least Square based control allocation as the VSQP is an overactuated system [27]. This structure allows one-loop architecture representation of the controller as opposed to previous studies where the inner and outer controllers were designed separately running two different optimization routines [28]. In this way, it is expected to have a robust controller that can distribute the control load optimally throughout the control surfaces of the VSQP. The diagram showing the overall guidance and control scheme of the VSQP is given in Figure 3.

Within the $G\&C$ structure, the WLS block takes the virtual control parameters of which values are calculated in the error controllers for the position and attitude with the gains set according to the desired response of the drone. While error controllers ensure the reference models are being tracked closely, reference models are used to shape the given desired set point into reference signals that are compatible with the

(a) VSQP in quad-mode configuration. (b) VSQP in transition (c) VSQP in forward flight configuration.

Figure 1: The configuration modes of the VSQP

characteristics of the controller.

Although the controller is robust and adaptive to changes in the states of the UAV and the environment, the inputs to be tracked are dependent on the given desired waypoint position and the reference model parameters. In the case of a ship landing, it may not be desired to only track the position of the ship with fixed response characteristics since ship motion exhibits characteristics that are continuously changing under the effect of external disturbances and uncertainties. Additionally, it is not possible to assess how optimal the guidance inputs are by considering the whole system. Therefore, the guidance scheme of the VSQP is replaced by a reinforcement learning-based model to learn the optimal acceleration inputs entering into the controller. With reinforcement learning in the loop, it now becomes possible to assess the optimality of the guidance model over different objectives while exploring different landing trajectories that are generated during landing.

The simplified diagram for the reinforcement learning framework is shown in Figure 3. Box number 1 denotes the reinforcement learning framework and contains deep neural network structure whereas box number 2 illustrates the response of the vehicle to given control inputs by the controller. Box number 1 takes the states of the VSQP as inputs and generates the corresponding optimal acceleration inputs which are denoted as a_{des}. The response of the vehicle, symbolized as a_{real} corresponds to the actual accelerations resulting from the applied control inputs. To effectively map the given commands to the actual accelerations and use them in the training process, a simplified model for both the UAV and the controller is necessary. In a previous study [29] where reinforcement learning is used in combination with an Incremental Nonlinear Dynamic Inversion (INDI), a first-order transfer function was used to model the controller's response to given attitude control signals. However, this approach is not directly applicable to the VSQP as it is an overactuated system that incorporates a separate routine for control allocation. This optimization routine complicates the overall framework as all actuators affect the acceleration response of the VSQP.

In addition to the existing actuators of the VSQP, the WLS also uses pitch and roll angles as its virtual control inputs within its optimization routine which later are fed back to the attitude reference model. These Euler angles have the slowest response among all actuators, therefore limiting the vehicle's response to the given control input. Taking this factor into consideration, the simplified model for the VSQP is obtained by only considering the attitude dynamics in the loop. As a result, a transfer function that maps the desired accelerations to the actual ones is given in Eq.1.

$$acc_{tf} = \frac{206s^3+861.1s^2+287s+95.68}{s^6+20.22s^5+143.3s^4+682s^3+1021s^2+340.2s+95.68} \quad (1)$$

The response of the transfer function to a step input is shown in Figure 2. Note that although the drone model is represented with a sixth-order transfer function an equivalent fourth-order model is used to avoid the numerical issues previously encountered in the initial phase of the training.

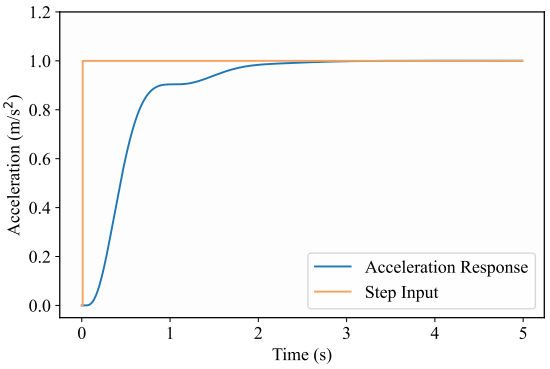


Figure 2: Response of the VSQP to a step input

4 Reinforcement Learning Based Problem Formulation

4.1 Background in Reinforcement Learning

Reinforcement learning is an area of machine learning focusing on how an agent should interact with the environment to take actions that maximize the cumulative reward over time. At each time step t, the agent receives a state s_t from a state space $\mathcal{S}$ and chooses an action a_t from an action space $\mathcal{A}$, based on a policy $\pi(a_t \mid s_t)$. The policy could be stochastic $\pi(a \mid s)$, with a probability associated with each possible action, or deterministic $\pi(s)$, indicating the agent's

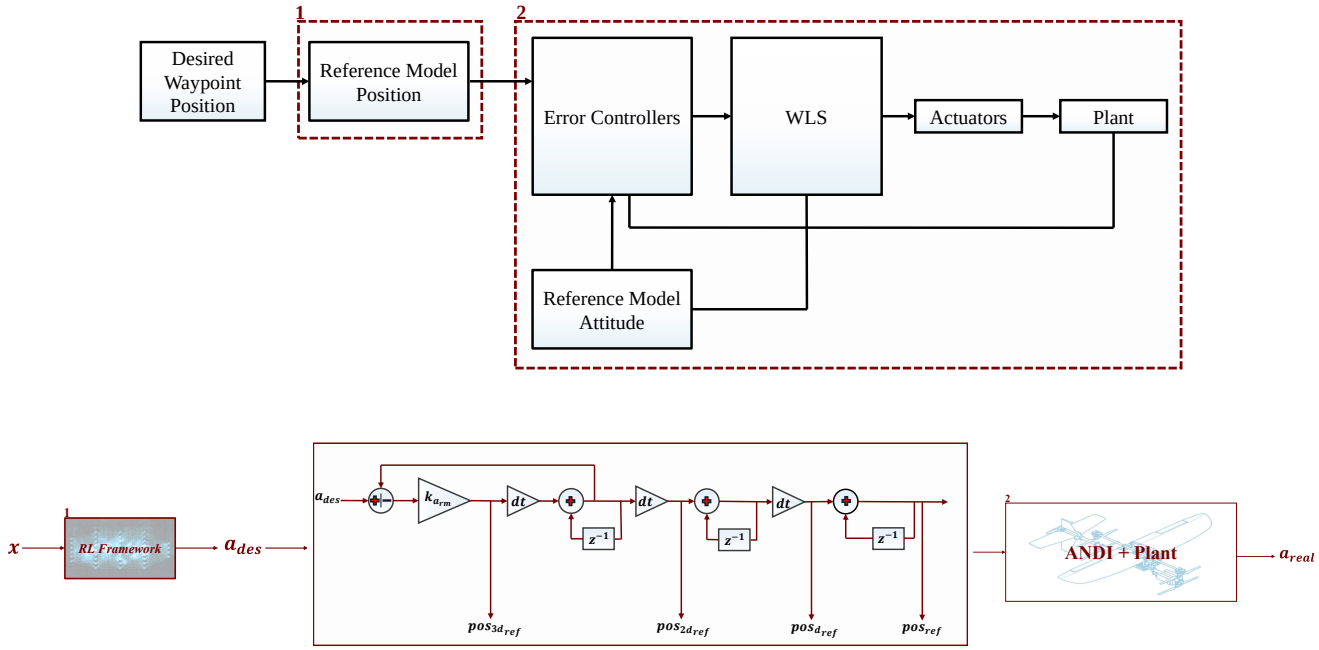


Figure 3: Reinforcement learning framework integrated into the existing control stucture

decision-making process from states to actions. Upon taking an action, the agent receives a reward r_t and makes the transition to the next state s_{t+1}, according to the reward function $\mathcal{R}(s,a)$ and state transition probability $\mathcal{P}(s_{t+1} \mid s_t, a_t)$. This process continues until the agent reaches a terminal state and it restarts. The cumulative reward, defined as the return function $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ discounted with a factor $\gamma \in (0,1]$.

The expectation of cumulative reward for a specific policy is represented as V^π, the value function, and is given in Eq. 2.

$$V^\pi(s_t) = \mathbb{E}\left[R_t \mid s_t, a_t = \pi(s_t)\right] \tag{2}$$

The action-value function Q^π in Eq.3, is equivalent to the value function for every action-state pair (s_t, a_t).

$$Q^\pi(s_t, a_t) = r(s_t, a_t) + \gamma \sum p(s_{t+1} \mid s_t, a_t) V^\pi(s_{t+1}) \tag{3}$$

The goal is to find the optimal policy π^* as in Eq. 4 that maximizes the value function and the corresponding value function is called the optimal value function.

$$\pi^* = \arg\max_\pi V^\pi(s_t) \tag{4}$$

Policy gradient methods are a class of reinforcement learning algorithms that are particularly effective in environments with continuous state and action spaces. In a policy gradient algorithm, the policy is presented as π_θ where θ are the neural network parameters. Given that the expected return is $J(\theta) = \mathbb{E}_s\left[V_{\pi_\theta}(s)\right]$, the goal is to find optimal parameters $\theta^* = \arg\max_\theta J(\theta)$ through a gradient ascent update $\theta_{k+1} = \theta_k + \alpha_k \nabla J(\theta_k)$ where α_k is the learning rate. The gradient of $J(\theta)$ is calculated using the policy gradient theorem which is given by:

$$\nabla J(\theta) = \mathbb{E}_{\pi_\theta}\left[Q_{\pi_\theta}(s,a) \nabla \log \pi_\theta(s,a)\right] \tag{5}$$

There are many types of policy gradient algorithms such as TRPO [30], SAC [31], and PPO [32]. In this work, we adapt the PPO algorithm to solve the landing problem of the VSQP on a moving platform. PPO is a highly stable and efficient policy gradient algorithm that introduces techniques like clipping the objective and using an adaptive penalty coefficient to improve the stability and efficiency of the learning process [32]. It employs an actor-critic model where the actor takes outputs actions and the critic evaluates their performance.

PPO uses the following objective:

$$L^{CLIP}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right] \tag{6}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ represents the probability ratio of the action under the new and old policies, $\hat{A}_t$ is advantage at timestep t, and ϵ is a hyperparameter that defines the clipping range to keep the ratio within reasonable bounds.

4.2 *Problem Formulation*

The UAV landing problem can be described as a Markow decision process (MDP). Typical MDP can be defined as a

four tuple $(\mathcal{S}, \mathcal{A}, R, \Omega)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $R(s, a)$ represents the immediate rewards, and $\otimes$ is the observation space. Based on this model, the landing problem is formulated as follows:

1. *State Space* $\mathbb{S}_d$: The state space $\mathbb{S}_d$ represents the states of the UAV. Previously, it was mentioned that a fourth-order transfer function is used to model the response of the VSQP and controller. Therefore, the state space for the drone consists of extra three acceleration derivatives $\dot{\boldsymbol{a}}_d, \ddot{\boldsymbol{a}}_d, \dddot{\boldsymbol{a}}_d$ on top of the position $\boldsymbol{p}_d$, velocity $\boldsymbol{v}_d$, and acceleration $\boldsymbol{a}_d$ components as in Eq. 7.

$$\mathbb{S} = \{\boldsymbol{p}_d, \boldsymbol{v}_d, \boldsymbol{a}_d, \dot{\boldsymbol{a}}_d, \ddot{\boldsymbol{a}}_d, \dddot{\boldsymbol{a}}_d\} \tag{7}$$

2. *State Space* $\mathbb{S}_s$: The state space $\mathbb{S}_s$ represents the states of the ship platform. It consists of platform position p_s and velocity v_s components as given in Eq. 5

$$\mathbb{S}_s = \{\boldsymbol{p}_s, \boldsymbol{v}_s\} \tag{8}$$

The motion of the ship is characterized by sinusoidal signals that are governed by specific frequencies, amplitudes, and initial conditions. The definitions for position and velocity is expressed in the following equations Eq.9 and Eq. 10.

$$p_i(t) = A_{\mathrm{p},i} \sin(\omega_i t + \phi_i) + p_{\mathrm{init},i} + \xi_i(t) \tag{9}$$

$$v_i(t) = \omega_i A_{\mathrm{p},i} \cos(\omega_i t + \phi_i) + \eta_i(t) \tag{10}$$

where $A_{\mathrm{p},i}$ is the amplitude of the sinusoidal motion for each axis i, ω_i denotes the angular frequency, ϕ_i is the phase shift, and finally $\eta_i(t)$ captures the random walk component. This random component helps to simulate unpredictable movements or external disturbances affecting the ship's motion.

3. *Observation Space* Ω: Although the state space for the drone and ship has the full state information, the input to the reinforcement learning algorithm is given as the relative position p_r and velocity v_r components of the drone with respect to the moving platform along with the *real acceleration* $\boldsymbol{a}_d$ of the drone. As a result, observation space is defined in Eq. 11

$$\mathbb{S}_s = \{\boldsymbol{p}_r, \boldsymbol{v}_r, \boldsymbol{a}_d\} \tag{11}$$

4. *Action Space* $\mathbb{A}$: The action space Eq. 12 $\mathbb{A}$ consists of *desired acceleration* components $a_{n_{des}}, a_{e_{des}}, a_{d_{des}}$ determined based on the inputs of the reference model given in Figure 6.

$$\mathbb{A} = \{a_{n_{des}}, a_{e_{des}}, a_{d_{des}}\} \tag{12}$$

Within the reinforcement learning framework, the control input parameters a_n and a_e are used to steer the drone onto the platform while a_d is mainly used to find the right time to accomplish landing.

5. *Reward Function* R: Designing a reward function that is suitable for a *landing on a moving target* problem is not a straightforward task, requiring many iterations over different objective functions. Considering the subphases of the landing problem, we classify the reward function in terms of three components: position tracking, velocity tracking, and collision penalty.

   (a) Position Tracking: The reward function R_p in Eq. 14 is designed for accurate tracking of the moving platform by the minimization of the deviation in both vertical and horizontal planes relative to the target.

$$R_p = k_1 \left| p_{r_{\mathrm{old}_v}} \right| - k_1 \left| p_{r_{\mathrm{new}_v}} \right| + k_2 \left\| \left\| \mathbf{p}_{r_{\mathrm{old}_h}} \right\| - k_2 \left\| \mathbf{p}_{r_{\mathrm{new}_h}} \right\| \right\| \tag{13}$$

   The parameters $p_{r_{\mathrm{old}_v}}$ and $p_{r_{\mathrm{new}_v}}$ represent the old and new vertical distances; $\mathbf{p}_{r_{\mathrm{old}_h}}$ and $\mathbf{p}_{r_{\mathrm{new}_h}}$ represent the horizontal distances between the UAV and the target, respectively. The term $\left|p_{r_{\mathrm{old}_v}}\right| - \left|p_{r_{\mathrm{new}_v}}\right|$ rewards the agent based on the absolute difference in vertical position and $\|\mathbf{p}_{r_{\mathrm{new}_v}}\| - \|\mathbf{p}_{r_{\mathrm{new}_h}}\|$ uses the norm of both x and y components together to compute the difference. The importance of one motion with respect to another is measured by the weights k_1 and k_2 which are determined to be 2 and 1, respectively.

   (b) Velocity Tracking: The velocity reward R_v is designed to ensure that the speed of the UAV is well adjusted in a way that it gets higher rewards as the UAV slows down at every time step while also getting close to the platform. Similarly to the position reward R_p, the parameters $v_{r_{\mathrm{old_v}}}$ and $v_{r_{\mathrm{old_h}}}$ represent the old and new relative velocities in vertical plane whereas $\mathbf{v}_{r_{\mathrm{old_v}}}$ and $\mathbf{v}_{r_{\mathrm{old_h}}}$ represent the corresponding parameters in the horizontal plane. The reward function is dynamically scaled by a factor that depends on the drone's height. The scaling factor is important since it fine-tunes the UAV's speed so that the touchdown velocity does not get too big while also making sure that velocity tracking only takes place close to the platform.

$$R_v = k_1 \left(v_{r_{\text{old}_\text{v}}} - v_{r_{\text{new}_\text{v}}}\right) \left(\frac{h_t + h}{h_t}\right) + k_2 \left(v_{r_{\text{old}_\text{h}}} - v_{r_{\text{new}_\text{h}}}\right) \left(\frac{h_t + h}{h_t}\right) \quad (14)$$

(c) Collision Penalty: The collision penalty is designed to be the most critical part of the reward function since it directly penalizes the UAV based on its speed at the touchdown point on the platform. To assess the performance of the landing, two different reward functions, based on relative velocities and absolute velocities are considered for the collision penalty.

The first reward function R_{c_1} as given in Eq.15 is defined as the *relative velocity collision penalty* that penalizes the relative velocity of the UAV with respect to the platform's. The constants k_3 and k_4 again allow for tunning the sensitivity of the penalty to the different directional velocities, avoiding lateral collisions as well as maintaining a safe descent speed.

$$R_{c_1} = k_3 \left|v_{r_{\text{new}_v}}\right|^2 + k_4 \left\|\mathbf{v}_{r_{\text{new}_h}}\right\|^2 \quad (15)$$

The second collision reward R_{c_2}, *absolute velocity collision penalty*, focuses on the vertical velocity components at the point of landing. The $v_{d_{\text{new}_z}}$ indicates the vertical velocity of the drone and $v_{p_{\text{new}_z}}$ refers to the platform's vertical velocity.

These collision rewards only get active at the final point of landing, returning a reward with the corresponding velocities of the drone and the platform. This approach resembles a form of Cliffwalk [33] since the agent does not receive intermediate rewards till the point of touchdown. While this makes the search for an optimal policy more challenging, the agent gains greater freedom in its actions as opposed to the reward functions used in the previous studies [23, 25, 24], that defined a separate reward function for each altitude range. In case this reward is combined with the reward for velocity tracking 5, the agent receives intermediate rewards on slowing its velocity down beforehand.

$$R_{c_2} = k_5 \left|v_{d_{\text{new}_z}}\right|^2 + k_6 \left|v_{p_{\text{new}_z}}\right|^2 \quad (16)$$

These reward functions are strategically combined to identify the most effective objective for the given problem. The table 1 outlines various combinations used in our study. Reward number R1 only consists of the position reward R_p encouraging the agent to minimize the difference between the old and new positions. With this reward, the agent behavior is examined without any limitation on its velocity. R2 and R3 incorporate the collision penalties R_{c_1} and R_{c_2} along with the position reward. These combinations aim to explore the agent's response in terms of speed reduction in the absence of intermediate rewards, thereby assessing the necessity of such rewards. Lastly, rewards R4 and R5 combine everything including the velocity reward R_v with the collision penalties. This is crucial for determining which type of collision penalty is most effective for the specific problem at hand.

Table 1: Reward combinations used in the study

Reward Number	Combination
R1	R_p
R2	$\mathrm{R}_p + \mathrm{R}_{c_1}$
R3	$\mathrm{R}_p + \mathrm{R}_{c_2}$
R4	$\mathrm{R}_p + \mathrm{R}_v + \mathrm{R}_{c_1}$
R5	$\mathrm{R}_p + \mathrm{R}_v + \mathrm{R}_{c_2}$

5 TRAINING

For training the reinforcement learning framework, a gym environment including classes for both the UAV and the ship platform dynamics, in accordance with their state space models, has been created in Python. The UAV model is derived from converting the transfer function provided in Eq.5 to equivalent differential equations and solving it through the Euler method. The fourth-order differential of accelerations is propagated until the position, velocity, and acceleration components of the UAV are computed.

The ship model uses sinusoidal signals with mixed frequencies and random walk components to simulate a wide variety of signals that resemble real ship motion. This variety in the ship motion is critical to prevent overfitting during the training process, as the validation will be done with real ship data, which incorporates dynamics that are more complex than standard sinusoidal signals.

In alignment with the observation space Ω (see Eq.11), the environment is set to receive information on relative position, relative velocity, and real drone accelerations at every time step. The outputs are determined to be the desired acceleration inputs which are later used to propagate the states of the drone to obtain the velocity and position information.

While the initial simulations incorporated motion in north and east for the ship, this is later limited to only motion in the z-direction. This is due to the fact that vertical tracking is paramount to achieving a successful landing, given the ship's constant exposure to wave-induced hard-to-predict random motions. For horizontal tracking, the performance is evaluated by selecting different initial conditions. The training process is repeated for all the combinations indicated in

Table 1 until a stopping condition is satisfied. The simulations are ended when any of the following conditions are met: a ground collision occurs, the drone exists the designated area, the maximum number of steps is reached, or the drone achieves a landing condition, specifically being 0.1m above the platform.

5.1 Initializing the drone dynamics

For training, a vectorized environment is used to run several environments in parallel. To do so, initial drone states are uniformly sampled from the following integrals given in Eq. 17.

$$\begin{array}{lll} p_n \in [-1,1] + p_{n_i} & p_e \in [-1,1] + p_{e_i} & p_d \in [-1,1] + p_{d_i} \\ v_n \in [-0.5,0.5] & v_e \in [-0.5,0.5] & v_d \in [-0.5,0.5] \\ a_n \in [-0.05,0.05] & a_e \in [-0.05,0.05] & a_d \in [-0.05,0.05] \end{array} \quad (17)$$

Here, $p_{n_i}, p_{e_i}, p_{d_i}$ represent the initial position elements of the drone. The parameters p_{n_i} and p_{e_i} are chosen in a way that the drone aligns itself with the platform and p_{d_i} is determined to be 10 meters as the starting altitude at the beginning of the training procedure. Additionally, nine additional state elements which are the derivatives of accelerations ($\dot{\boldsymbol{a}}_d, \ddot{\boldsymbol{a}}_d, \dddot{\boldsymbol{a}}_d$) up to a third degree are also initialized with a value of zero with the given state elements. This is because the drone model uses a fourth-order transfer function to compute its actual accelerations and all the numerical derivations take place within the model.

5.2 Initializing the ship dynamics

Similarly to the drone model, ship states are also uniformly sampled from the given intervals in Eq. 18.

For the ship, wave parameters amplitude A and frequency f are used to initialize the sinusoidal motion characteristics. To create variation in the generated sinusoidal motions additional parameters such as mix frequency f_{mix}, random walk step size rw_{ts}, and low pass filter coefficient α are also given as an input to the model. The values for these parameters are determined as 0.2, 0.01, and 0, respectively.

$$\begin{array}{lll} A_x \in [-0.1,0.1] & A_y \in [-0.1,0.1] & A_z \in [-0.1,0.1] + A_z \\ f_x \in [-0.1,0.1] & f_y \in [-0.1,0.1] & f_z \in [-0.1,0.1] + f_z \end{array} \quad (18)$$

An example of noise characteristics incorporated in the ship model is shown in Figure 4.

6 Evaluation

In this section, the performance of given reward combinations will be assessed based on three key metrics: touchdown velocity, deviation, and duration. This evaluation will be completed for a specific test case scenario.

As a result of this assessment, the best combination of reward elements will be identified. Once this combination is determined, an additional analysis by adjusting the coefficients

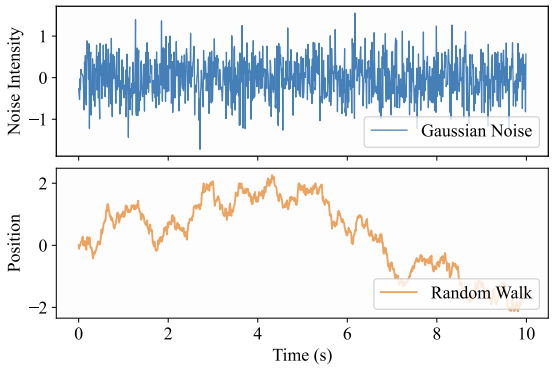


Figure 4: Noise characteristics for the ship motion

k_5 and k_6 will be completed to refine the reward function further. The results of these analyses will be presented using box plots that allow for a clear comparison of the reward functions across the considered parameters.

6.1 Test Case

As is indicated in Section 5, the frequency and amplitude of the platform were randomized within the intervals in Eq.18 to cover a wide range of scenarios in the training process. In this section, a test case scenario is chosen for a fair comparison of the given reward functions. A sinusoidal wave in z-direction with the characteristics given in Eq.19 is generated for testing the reward functions in the simulations.

$$A_z \in [-0.1,0.1] + A_z, \quad f_z \in [-0.1,0.1] + f_z \quad (19)$$

While generating the test case scenario, the motion of the ship in the north and east is ignored since the vertical motion has the greatest importance in mimicking the response of the waves. Still, the associated frequency f_z and amplitude A_z are randomized, though now the intervals are kept smaller for a more stable performance. The figure showing the generated ship motion in the z direction for a number of 15 simulations is given in Figure 5.

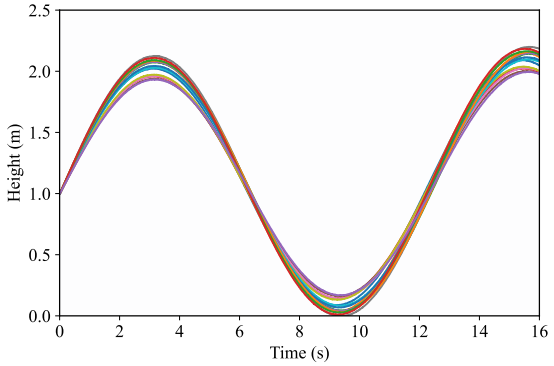


Figure 5: Ship motion for the test case

Note that while the generated wave is 15 seconds long,

the simulations are stopped at the point where any of the end conditions are satisfied.

6.2 Testing reward functions for different sinusoidal waves

An analysis has been performed to understand the influence of given reward combinations on the drone's performance and identify the one that best addresses the landing problem through three key criteria: touchdown velocity, deviation from the target center, and time to collision.

The touchdown velocity is a critical metric for assessing the safety and the smoothness of the UAV's landing. While a lower touchdown velocity signifies a softer landing, higher ones point out that a landing resulted in a crash. The deviation metric measures the accuracy of the UAV's landing by evaluating how close the UAV lands to the intended target point. Lastly, time to collision represents the duration that a landing takes.

The assessment of the drone based on these performance metrics is visualized with box plots in Figure 6. The top plots aim to determine which reward combination (R1 to R5) yields the best results, especially regarding touchdown velocity for further analysis. On the other hand, the bottom figures result from an additional analysis to determine the coefficients (k_5 and k_6) for the best reward combination chosen, which in this case, the *final* indicates the reward R5 with tuned coefficients.

The top-left subplot shows the distribution of touchdown velocities for each reward combination. The y-axis represents the vertical descent velocity v_{d_z} in meters per second (m/s), and the x-axis lists the reward combinations (R1 to R5) and the final (R5 with tuned coefficients). R1 has the highest median touchdown velocity at around 3.0 m/s, with a range of values extending from about 2.5 to 3.5 m/s. This indicates that with only a position element in the reward, the landing is less controlled, leading to higher impact speeds. In contrast, the value of the median velocity drops significantly for R2 and R3, while the latter has a slightly lower median value. This shows that having a collision penalty, either absolute or relative, has a significant effect in reducing touchdown velocity. By the addition of the velocity element in R4 and R5, the velocity values further decrease, with touchdown velocities close to zero, indicating the landing is extremely smooth and controlled.

The middle box plot shows the deviation from the target center (d_{xy}) in meters for each reward combination. While the deviation values are easily negligible for rewards R2, R3, and R4, R1 and R5 show that the drone slightly deviates from the center. Finally, the top-right shows the landing duration in seconds. As expected, the duration gets longer as the velocity values decrease, resulting in a less aggressive flight.

While different performance parameters are used, more weight is put into having lower velocities. Based on the results given in the top-left figure, the reward function R5, which uses the absolute velocities of the drone, has been identified as having the most optimal form for the given problem. Further analysis was conducted to determine the effect of coefficients (k_5 and k_6) on the UAV's performance that was previously given a value of 1. Both coefficients k_5 and k_6 are tested with the values of 10, 25, 50, 75, and 100, respectively. The results are given in the bottom box plots which shows the effect of of different coefficients on the UAV's performance for the chosen reward combination.

The first plot on the left again shows the distribution of touchdown velocities, but this time for the same combination R5. Generally, all coefficient values result in relatively low touchdown velocities, showing stable behavior. However, there are visible differences in the value distribution. Among the coefficients, k:100 achieves the lowest median value of almost 0.0 m/s. Even though the coefficients do not strongly show a linear relationship, it was expected that high penalty values would lead to extremely small velocities. While this is proved to be true in the simulations, it was also observed that very high coefficients either lead to extremely long flights or no landing at all. Specifically, high penalty values for the collision stop the drone way earlier, making it extremely hesitant to take any action. Therefore, the coefficients above 100 are not shown in the graph as the number of successful cases was pretty low.

Despite the lowest touchdown velocity achieved by k:100, this coefficient was not chosen due to its relatively larger deviation from the center, which compromises landing accuracy. The given deviation and duration plots generally show similar results and thus do not suggest a strong relationship between these coefficient values. The graphs indicate that coefficients between 10 and 100 provide a good balance in terms of performance metrics. Considering the negative effect of higher penalties on performance and the low deviation values, the coefficient k:25 was chosen to for R5. The final form of the reward function is given in Eq. 5.

$$
\begin{aligned}
R_{final} = & \left|p_{r_{\mathrm{old}_v}}\right| - \left|p_{r_{\mathrm{new}_v}}\right| + 2\left\|\mathbf{p}_{r_{\mathrm{old}_h}}\right\| - 2\left\|\mathbf{p}_{r_{\mathrm{new}_h}}\right\| \\
& + \left(v_{r_{\mathrm{old_v}}} - v_{r_{\mathrm{new_v}}}\right)\left(\frac{h_t + h}{h_t}\right) \\
& + 2\left(v_{r_{\mathrm{old_h}}} - v_{r_{\mathrm{new_h}}}\right)\left(\frac{h_t + h}{h_t}\right) \\
& + 25\left|v_{d_{\mathrm{new}_z}}\right|^2 + 25\left|v_{p_{\mathrm{new}_z}}\right|^2
\end{aligned}
\tag{20}
$$

It is important to clarify that these simulations only valid for the given test case scenario and may vary slightly in case a different sinusoidal signal is chosen. The objective here is not to pinpoint an exact value for the coefficients, but rather to demonstrate that within a certain range, the chosen reward combination yields consistent performance. Therefore, the resultant reward form would still result in successful landings even if a different coefficient had been selected.

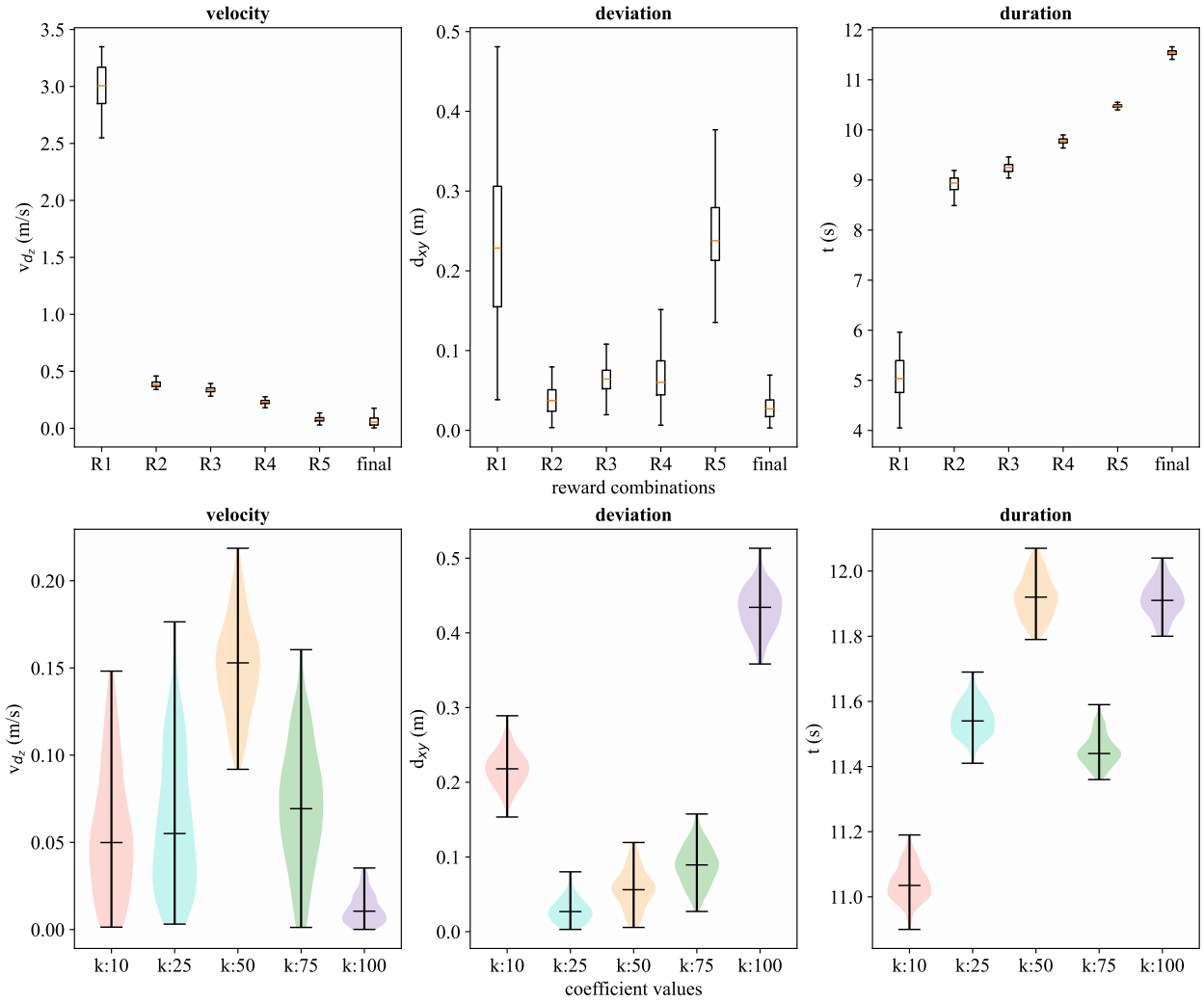


Figure 6: The analysis of reward combinations in terms of performance metrics

7 Validation

Although the assessment of different reward functions was completed using randomized sinusoidal motion, real ship movements often not carry the characteristics of a sine wave. To determine if the reinforcement framework could successfully land on waves that are not sinusoidal-like, validation with real ship data is both valuable and necessary. For this purpose, real ship data is gathered at a frequency of 5 Hz was used for the simulations. Since the simulations run at a different frequency (10 Hz), the data was interpolated using a quadratic fit and sampled for the validation.

Similar to the analysis conducted in the training section, the performance of the reinforcement learning framework was tested on the real ship data using the final form of the reward function (see Eq. 20). The assessment was again completed in terms of previously given performance metrics: downward velocity, deviation, and duration.

Additionally, the RL framework was tested against a PID based benchmark controller. Figure 8 shows the considered PID-based controllers that were designed for position and velocity tracking namely, PID$_{rel}$ and PID$_{abs}$. The former takes the relative velocity of the drone with respect to the platform, while the latter generates velocity commands based on the position tracking only, as it may not be desired to track the platform's velocity from the beginning.

The left plot in Figure 6 compares the vertical descent velocities (v_{d_z}) of the RL framework and the two PID controllers. While the figures show that PID$_{rel}$ works slightly better than the PID$_{abs}$ in terms of downward velocity, the RL framework achieves the lowest median velocity among them, with values close to 0.1 m/s. Setting the limit at around 0.5 m/s for a successful landing, these results indicate that RL shows a superior performance with extremely smooth land-

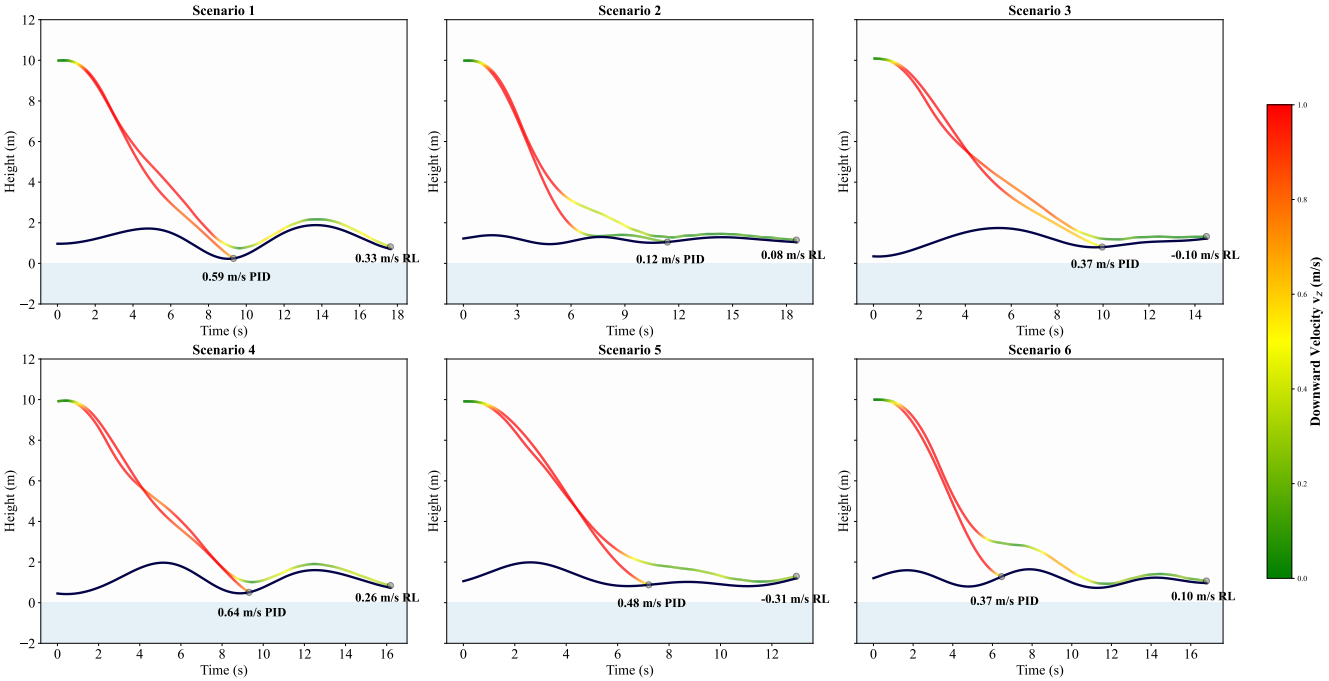


Figure 7: Validation trajectories

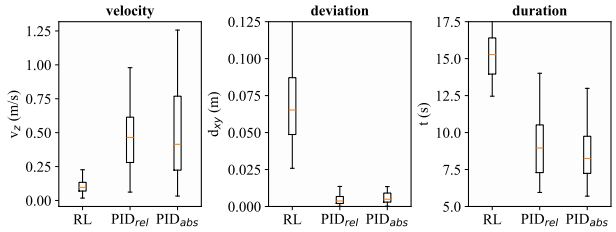


Figure 8: Validation box plots

Table 2: Summary Statistics for RL and PID Controllers

Label	Min	Max	Mean	Std Dev
RL-vz	0.0176	0.2265	0.1017	0.0453
PID$_{rel}$-vz	0.0617	1.3303	0.4779	0.2459
PID$_{abs}$-vz	0.0325	1.2563	0.4828	0.3082
RL-xy	0.0258	0.1320	0.0692	0.0243
PID$_{rel}$-xy	0.0001	0.0433	0.0073	0.0091
PID$_{abs}$-xy	0.0002	0.0453	0.0084	0.0097
RL-ttc	12.4600	19.7600	15.3107	1.6582
PID$_{rel}$-ttc	5.9500	14.0100	9.0416	1.9637
PID$_{abs}$-ttc	5.700	12.990	8.5610	1.7039

ings, even leaving some margin in velocity.

Reinforcement learning and PID frameworks were also tested for the deviation and the duration parameters. While PID controllers make a precise landing almost at the center point, the RL framework show slight deviation from the center. Also for the duration, RL takes a little bit more time to land which is consistent with its low touchdown velocities. The statics for the reinforcement learning and PID controllers are given in Table 2.

The Figure 7 visualizes the trajectories of a drone controlled both PID (PID$_{rel}$) and reinforcement learning framework over real ship data, across six different scenarios. The y-axis indicates the height (in meters) and the x-axis shows the duration of the flight (in seconds). The trajectories are color-coded to indicate the downward velocity (v_{d_z}) of the drone, with green representing low velocities and red representing high velocities. The color coding has the purpose of showing how different the velocity varies along the trajectory for RL compared to the benchmark PID controller. Additionally, the impact velocities are indicated in circles for both controllers.

The trajectories controlled by the PID consistently result in more aggressive landings with high touchdown velocities. In each scenario, PID makes a relatively rapid descent as its velocity is characterized by the almost fully red-colored trajectories shown in the figure. In contrast, the reinforcement learning framework displays significantly different landings. It closely tracks the wave patterns of the ship and wait for optimal conditions before landing, with the velocities making a smooth transition from red to green along the descent path. It consistently show lower touchdown velocities leading to a stable and controlled flights as opposed to the PID control that often results in crashes instead of successful landings.

8 Conclusion

This study demonstrated the effectiveness of a reinforcement learning-based framework for the autonomous landing of the Variable Skew Quad Plane on a moving platform (ship). Several simulations performed with randomized sinusoidal signals revealed that the reinforcement learning could adapt to a wide variety of signals, sinusoidal or non-sinusoidal, and generate control inputs that effectively adjust the drone's speed and trajectory to achieve successful landings.

The validation with real ship data which incorporated non-sinusoidal and unpredictable movements, confirmed the robustness and adaptability of our framework. Compared to the benchmark controller, reinforcement learning achieved significantly lower impact velocities that resulted in controlled, safer landings by closely tracking the wave patterns of the ship and waiting for the optimal landing point.

This work highlighted the capabilities of reinforcement learning for a challenging autonomous landing task which requires an enhanced performance against environmental variability and operational uncertainty. The integration of reinforcement learning as a guidance model within the control loop promotes safer and more reliable autonomous operations.

Future work may expand upon this study to target the landing problem fully with the given platform detection methodologies in the literature. By formulating the whole landing process, real-life experiments could be completed and especially the robustness property of the proposed algorithm would be tested to its limits. In addition to that, a different network structure (ex. Recurrent Neural Networks) that has ability to learn the patterns between the data may be suitable for this kind of landing problem since real data shows a pattern of mixed sinusoidal signals. Lastly, a more detailed analysis of the reward elements may be performed to incorporate ship characteristics further into the objective.

References

[1] JeongWoon Kim, Yeondeuk Jung, Dasol Lee, and David Hyunchul Shim. Outdoor autonomous landing on a moving platform for quadrotors using an omnidirectional camera. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1243–1252. IEEE, 2014.

[2] Kun Li, Peidong Liu, Tao Pang, Zhaolin Yang, and Ben M Chen. Development of an unmanned aerial vehicle for rooftop landing and surveillance. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 832–838. IEEE, 2015.

[3] Yi Feng, Cong Zhang, Stanley Baek, Samir Rawashdeh, and Alireza Mohammadi. Autonomous landing of a uav on a moving platform using model predictive control. *Drones*, 2(4):34, 2018.

[4] Oualid Araar, Nabil Aouf, and Ivan Vitanov. Vision based autonomous landing of multirotor uav on moving platform. *Journal of Intelligent & Robotic Systems*, 85:369–384, 2017.

[5] Marcin Skoczylas. Vision analysis system for autonomous landing of micro drone. *acta mechanica et automatica*, 8(4):199–203, 2014.

[6] Liguo Tan, Juncheng Wu, Xiaoyan Yang, and Senmin Song. Research on optimal landing trajectory planning method between an uav and a moving vessel. *Applied Sciences*, 9(18):3708, 2019.

[7] Pablo R Palafox, Mario Garzón, João Valente, Juan Jesús Roldán, and Antonio Barrientos. Robust visual-aided autonomous takeoff, tracking, and landing of a small uav on a moving landing platform for lifelong operation. *Applied Sciences*, 9(13):2661, 2019.

[8] Ajmal Hinas, Jonathan M Roberts, and Felipe Gonzalez. Vision-based target finding and inspection of a ground target using a multirotor uav system. *Sensors*, 17(12):2929, 2017.

[9] Bochan Lee, Vishnu Saj, Dileep Kalathil, and Moble Benedict. Intelligent vision-based autonomous ship landing of vtol uavs. *Journal of the American Helicopter Society*, 68(2):113–126, 2023.

[10] Xuancen Liu, Shifeng Zhang, Jiayi Tian, and Longbin Liu. An onboard vision-based system for autonomous landing of a low-cost quadrotor on a novel landing pad. *Sensors*, 19(21):4703, 2019.

[11] Lizhen Wu, Chang Wang, Pengpeng Zhang, and Changyun Wei. Deep reinforcement learning with corrective feedback for autonomous uav landing on a mobile platform. *Drones*, 6(9):238, 2022.

[12] Lirong Zhou, Anton Pljonkin, and Pradeep Kumar Singh. Modeling and pid control of quadrotor uav based on machine learning. *Journal of Intelligent Systems*, 31(1):1112–1122, 2022.

[13] Botao Hu, Lu Lu, and Sandipan Mishra. Fast, safe and precise landing of a quadrotor on an oscillating platform. In *2015 American Control Conference (ACC)*, pages 3836–3841. IEEE, 2015.

[14] JeongWoon Kim, YeonDeuk Jung, DaSol Lee, and David Hyunchul Shim. Landing control on a mobile platform for multi-copters using an omnidirectional image sensor. *Journal of Intelligent & Robotic Systems*, 84:529–541, 2016.

[15] Shyh-Pyng Shue and Ramesh K. Agarwal. Design of automatic landing systems using mixed h2/h1 control.

Journal of Guidance, Control, and Dynamics, 22(1), 1999.

[16] Yi Feng, Cong Zhang, Stanley Baek, Samir Rawashdeh, and Alireza Mohammadi. Autonomous landing of a uav on a moving platform using model predictive control. *Drones*, 2(4), 2018.

[17] Alireza Mohammadi, Yi Feng, Cong Zhang, Samir Rawashdeh, and Stanley Baek. Vision-based autonomous landing using an mpc-controlled micro uav on a moving platform. In *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 771–780, 2020.

[18] Man Yuan, Chang Wang, Pengpeng Zhang, and Changyun Wei. *PID with Deep Reinforcement Learning and Heuristic Rules for Autonomous UAV Landing*, pages 1876–1884. 03 2023.

[19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[20] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[21] Riccardo Polvara, Massimiliano Patacchiola, Sanjay Sharma, Jian Wan, Andrew Manning, Robert Sutton, and Angelo Cangelosi. Toward end-to-end control for uav autonomous landing via deep reinforcement learning. In *2018 International conference on unmanned aircraft systems (ICUAS)*, pages 115–123. IEEE, 2018.

[22] Seongheon Lee, Taemin Shim, Sungjoong Kim, Junwoo Park, Kyungwoo Hong, and Hyochoong Bang. Vision-based autonomous landing of a multi-copter unmanned aerial vehicle using reinforcement learning. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 108–114. IEEE, 2018.

[23] Alejandro Rodriguez-Ramos, Carlos Sampedro, Hriday Bavle, Paloma de la Puente, and Pascual Campoy. A deep reinforcement learning strategy for uav autonomous landing on a moving platform. *Journal of Intelligent Robotic Systems*, 93:351–366, 2019.

[24] Jingyi Xie, Xiaodong Peng, Haijiao Wang, Wenlong Niu, and Xiao Zheng. Uav autonomous tracking and landing based on deep reinforcement learning strategy. *Sensors*, 20:5630, 2020.

[25] Vishnu Saj, Bochan Lee, Dileep Kalathil, and Moble Benedict. Robust reinforcement learning algorithm for vision-based ship landing of uavs. *arXiv preprint arXiv:2209.08381*, 2022.

[26] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.

[27] TML De Ponti, EJJ Smeur, and BWD Remes. Incremental nonlinear dynamic inversion controller for a variable skew quad plane. In *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 241–248. IEEE, 2023.

[28] HJ Karssies and C De Wagter. Extended incremental non-linear control allocation (xinca) for quadplanes. *International Journal of Micro Air Vehicles*, 14:17568293211070825, 2022.

[29] End-to-end neural network based optimal quadcopter control. *Robotics and Autonomous Systems*, 172:104588, 2024.

[30] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[31] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[32] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[33] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay, 2016.

APPENDIX: ERROR GRAPHS FOR THE VALIDATION SCENARIOS

In this section, error graphs for the given validation scenarios are presented in Figure 9. The graphs illustrate the variations in the position and the velocity errors of the drone with respect to the platform. The groups of parameters e_{pn}, e_{pe}, e_{pd} for position and e_{vn}, e_{ve}, e_{vd} for velocity represent the errors in the north, east, and down directions respectively. Each scenario shows an initial high error, particularly in the down direction, which significantly decreases over time. This pattern demonstrates the drone's ability to adapt and correct its trajectory towards the desired path.

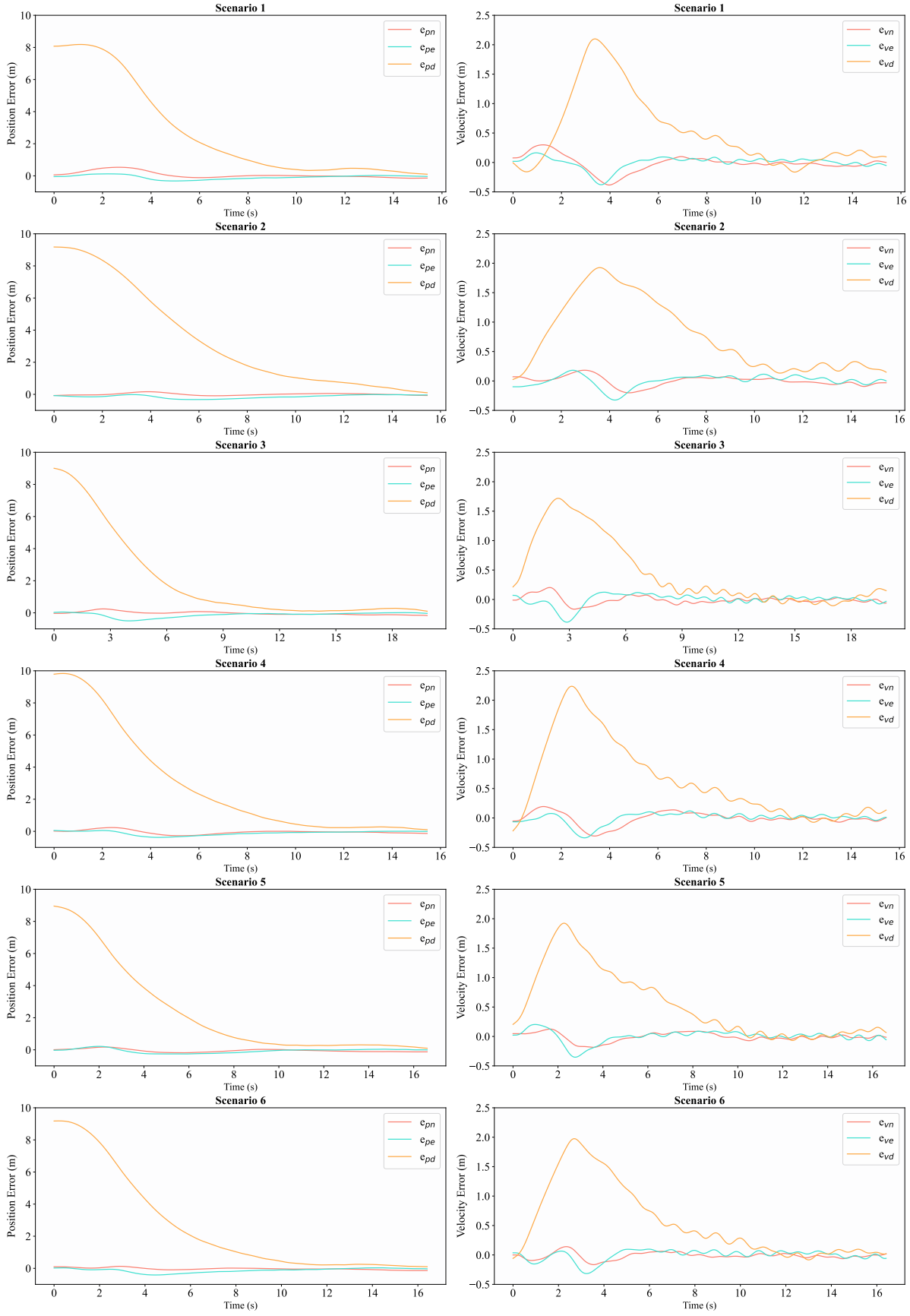


Figure 9: Error graphs for the validation scenarios

4

Autonomous Landing Problem for UAVs

The autonomous landing is one of the most important and complex phases of UAV operations. It involves navigating through several distinct phases, each with specific tasks and requirements. This chapter aims to understand the nature of the autonomous landing problem by addressing the first set of research questions **RQ-AL**. In the beginning, a comprehensive definition of the autonomous landing problem will be provided, with a particular focus on the challenges of ship landing problems. Later, the challenges associated with autonomous landing, including the dynamic and unpredictable maritime environment will be introduced. The key phases of a landing operation will be explored, highlighting the steps involved from target detection to the final landing maneuver. By examining the tracking and landing phases in the context of traditional techniques, the limitations of the existing approaches will be discussed and set the stage for exploring learning-based techniques.

4.1. The landing problem

Unmanned Aerial Vehicles (UAVs) have gained significant popularity in recent years so that UAVs are now found in a variety of applications, from environmental monitoring to military operations. Regardless of the task, a standard UAV flight consists of various phases, including take-off, climb, cruise, descent, and, most critically, landing. While certain phases, such as take-off and cruise, are relatively easy to achieve in UAV operations, autonomous landing is one of the most challenging due to the risks involved. Moreover, it has been found that most UAV crashes occur during the landing stage [19]. This is due to the fact that while landing, a strong air current is generated under the drone, which can cause a crash with any minor faulty input. This situation gets much worse if the system is subjected to strong gusts, such as in the case of ship landings. Therefore, the autonomous landing of UAVs demands near-perfect system operation, requiring a combination of highly accurate sensing, guidance, and control techniques.

The specific definition of a landing problem depends on the parameters such as the type of vehicle, assumptions made in the problem design, available resources, techniques, and the applicability of the problem to real world situations. In the context of this report, the particular type of landing problem considered is ship landing a hybrid drone (VSQP).

4.1.1. Landing on a ship

The use of UAVs in maritime operations has become increasingly common, including oceanographic data collection and search and rescue missions. This is because the characteristics of UAVs have a significant contribution to the operational efficiency and effectiveness, offering unique capabilities such as advanced surveillance, real-time data collection, and the ability to operate in remote and hazardous locations. However, UAVs are limited in range to operate in highly dangerous sea-environments, making the recovery of the UAV on the ship deck a necessity [2].

Landing a UAV on a ship involves several phases, each requiring precise tracking and control. Figure 4.1 illustrates the landing on a ship in terms of both drone and ship dynamics, along with the four

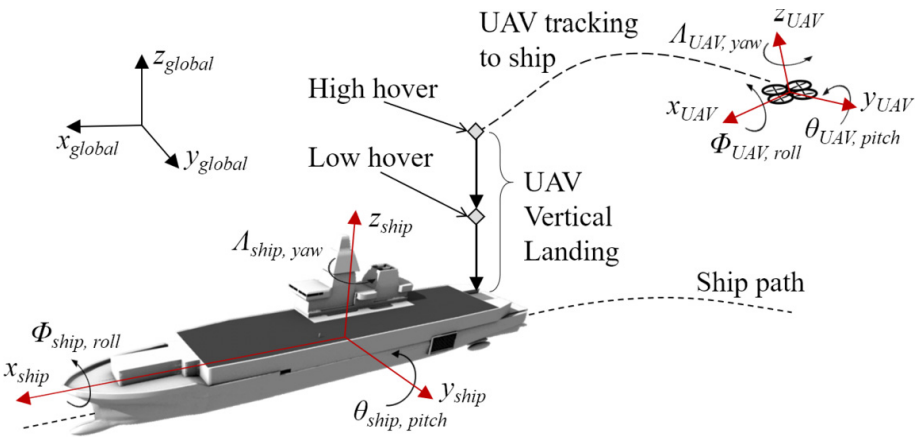


Figure 4.1: Recovery phases of a ship landing [2]

phases, tracking, high hover, low hover, and final descent [2]. Understanding the necessity of these phases is critical for a safe and successful landing.

The first phase, known as the tracking or homing phase, involves the UAV approaching the ship at its cruise altitude. In this phase, the UAV maintains its altitude while homing in on the vessel to ensure that it is on the correct path toward the landing area. As it gets closer to the ship, it transitions to the high hover phase. During this stage, the UAV reduces its altitude to a position clear of the ship's structures while still hovering above the deck. This phase acts as a preparatory stage, where final adjustments are made to ensure the UAV is correctly aligned and ready for descent.

Once the UAV is positioned itself correctly in the high hover phase, it can move into the other: low hover phase. Here, the UAV shows a similar behaviour response to the previous phase while maintaining a safe distance from the ship. This phase is important since the UAV must ready in terms of orientation and height for the final landing preparations. The final phase, where the UAV makes the last descent, accounts for the duration of controlled landing on the deck of the ship.

The UAV must be able to navigate through all these phases while correcting its position and orientation against the dynamic and unpredictable maritime environment. Especially the continuously changing ship dynamics (e.g., ship pitch and roll angles) and sudden waves turn the standard autonomous landing into an extremely complex problem. The complexity of ship landing requires highly advanced flight guidance and control systems onboard, allowing the UAV to make real-time decisions regarding the flight phases.

4.2. Challenges in Autonomous Landing

The major challenges involved with autonomous landing can summarized as obtaining accurate measurements or close-to-optimal estimates of the landing platform and the UAV, as well as achieving trajectory following that is robust to the presence of disturbances and uncertainties [14].

Going into further detail, these two challenges are connected to the following characteristics of autonomous landing:

- **Real-time massive information processing:** Autonomous landing requires considering the interaction between the environment and the UAV, including detecting the target and computing the relative states of the UAV even in complex scenarios such as moving platforms.
- **Limited onboard resources** As previously mentioned, the necessary computing power is high, specifically for vision-oriented algorithms due to the time required for image processing. Thus, the limited capacity of available resources adds another layer of complexity.
- **High manoeuvrability of UAV platforms:** UAVs, especially rotary-wing types, have high maneuvering capability, which requires obtaining fast and accurate feedback of the state information.
- **Limited image processing algorithms:** Vision based systems rely on traditional image processing algorithms for target tracking. However, different scenarios require different designs, making

it challenging to transition from one scenario to another, especially in terms of stability and robustness.

- **Dynamic scenes:** Dynamic scenes introduce a unique set of challenges to the autonomous landing problem. The motion of the platform requires more advanced navigation and control systems capable of reacting not only to the UAV but also to the platform itself. Although techniques used in static scenes still apply to dynamic ones, they often need some level of adaptation to cope with the newly added dynamics. For instance, current target detection and recognition algorithms struggle with recognizing the moving platform, thus needing cooperative targets having distinctive features.

4.2.1. Challenges specific to ship landing problems

The autonomous landing problem of UAVs on a ship is a highly complex engineering issue [33]. Compared to land-based UAV operations, the sea environment introduces challenges that extend beyond these encountered during typical UAV operations. That is because the maritime environment lacks necessary landmarks and space for landing [60], complicating the overall problem further.

The challenges specific to ship landing can be broadly categorized into the dynamic operational environment, guidance and navigation difficulties, and technical requirements and hardware limitations.

Dynamic Operational Environment

The dynamic nature of the sea environment is one of the significant challenges of ship landing. Ships are subject to continuously changing, hard-to-predict waves, creating motions such as rolling, pitching and heaving. These continuous external effects not only complicate the ship motion but also the relative motion of the drone with respect to the ship. With the risk of the UAV falling into the sea in case of a system failure, maintaining a stable trajectory during the UAV's approach and landing sequences becomes extremely challenging.

Guidance and Navigation Difficulties

Guidance and navigation are particularly challenging in maritime environments. Unlike land-based operations, the sea environment does not provide static reference commands for precise positioning and navigation. For instance, vision based systems struggle in adverse conditions such as fog, rain or night operations. These systems often require specific markers or objects for a successful landing, which may not be always feasible. Additionally, unpredictable dynamics of the sea necessitates advanced real-time processing and decision making capabilities.

Technical Requirements and Hardware Limitations

To process real-time data, respond to rapid changes in environment, and make quick decisions, UAVs must be equipped with advanced sensors along with robust and sophisticated algorithms. While there is a need for autonomous landing technology, it is also important that the hardware is not specialized only for a certain vehicle type of operation, ensuring flexibility across different platforms. This indicates that the mission mostly relies on the UAV's onboard systems which must be highly reliable and capable of operating under various environmental conditions. Additionally, the limited space on a ship's deck adds another layer of complexity, leaving the recycling of UAVs as an open discussion point, still.

4.3. Key phases in autonomous landing

Autonomous landing consists of multi sub-phases to achieve a successful operation. The key phases in autonomous landing are target detection, relative state estimation, and finally, tracking and landing.

The first phase, *the target detection*, aims to identify the landing target. Depending on the mission, the target could be a landing pad, a moving vehicle, or a natural landmark. Vision-based system are commonly used in the detection process. Once the target is detected, the next step involves estimating the relative state of the UAV with respect to the target. The second phase, *relative state estimation*, involves determining the position, velocity, and orientation of the UAV relative to the target. This process is crucial in steering the UAV towards the target. It often uses the Global Positioning System (GPS) in combination with the Inertial Navigation System (INS) via state estimation techniques such as Kalman filters or algorithms for target localization [14]. The final phase, *tracking and landing*, focuses on closely tracking the target and finding a position to execute the landing maneuver. This is achieved by the vehicle's guidance and control scheme, which generates the required control inputs.

A generic diagram showing the main features of an autonomous landing system is shown in Figure 4.2.Following the key phases in landing, the system typically includes four blocks: sensors/navigation system, guidance and flight controller, and the UAV to be controlled. Sensors/navigation block is responsible of providing the state information of the UAV either via state estimation techniques through sensor fusion or extracting features using vision-based techniques. Combining the sensory input with the desired trajectory, guidance system produces the commands for the controller, which are the corresponding required changes in state variables. Finally, the controller computes the necessary increments in control surfaces. This diagram includes the common techniques used for achieving the given sub-landing tasks, which will be explained in more detail in the following sections.

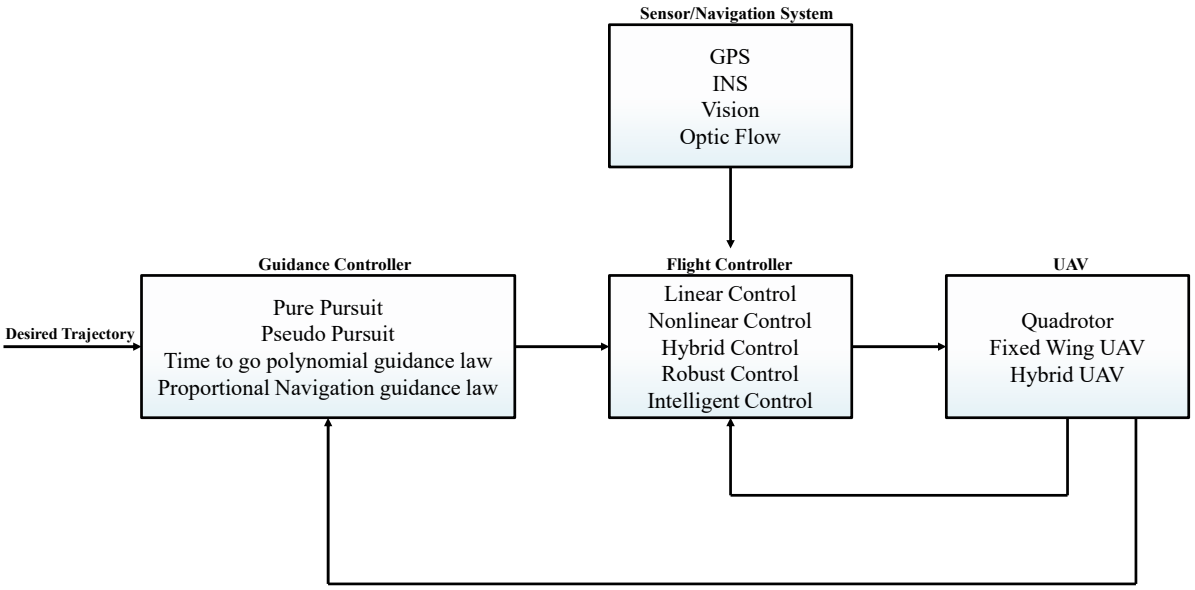


Figure 4.2: A generic landing control diagram (adapted from [18])

4.3.1. Target Detection

Target detection is a crucial phase in the autonomous landing process. It involves identifying the landing target so that the extracted information can be used for the subsequent steps of the landing. The detection process is often achieved by the use of vision based system, the details of which are provided in the following section.

Vision Based Systems

Vision based systems are commonly found in autonomous landing systems due to the characteristics such as strong autonomy, low cost, and anti-interference [66]. Especially the problems in which the exact location of the platform is either unknown or involves random movement (e.g., ship landings) require an algorithm based on computer vision. Moreover, in naval operations, vision-based algorithms enhance target detection without compromising security, as ships may not want to take a risk of jeopardizing its security by informing the UAV about its position.

Vision-based target detection involves different feature extraction techniques. These techniques analyse visual data to identify and track the landing target. This information is crucial for the guidance and control structure of the UAV, since the UAV positions itself relative the landing target based on this real-time information. Feature extraction is achieved by the use of sensory systems, which can be either active or passive. One of the most commonly used sensors in these systems is the infrared camera, a passive sensor detecting and measuring radiation naturally emitted by the landing target [67]. This makes them an obvious choice as they do not rely on visible light and can operate in various weather conditions. Similarly to the infrared cameras, other sensors like ultrawideband radar and lidar can also

be used in vision based systems. When combined with infrared cameras, these sensors improve the accuracy and reliability of the target detection process [53].

Xin et al. [66] classified the landing problem into three categories based on the characteristics of the target: static, dynamic, and complex scenes. Within static scenes, cooperative targets and natural scenarios are considered. Dynamic scenes are examined in terms of vehicle and ship based systems and lastly, complex scenes involve the consisted selections of safe landing areas for UAVs. The classification of UAV autonomous landing based on the target area is shown in Figure 4.3.

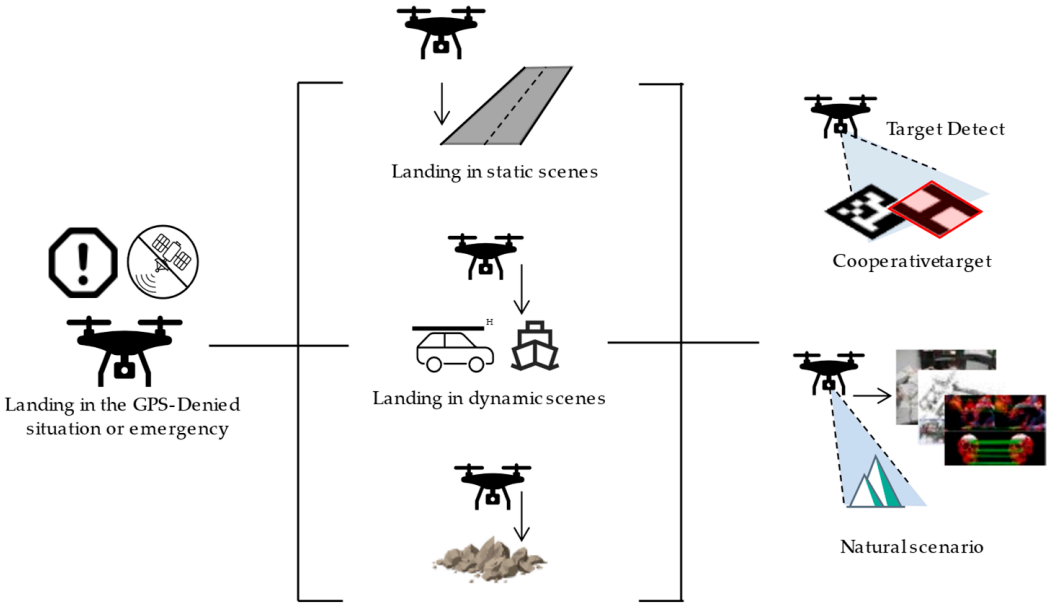


Figure 4.3: Autonomous landing classification for vision based systems [66]

Vision methodologies for target detection

In this section, the main vision methodologies for both static and dynamic scenes are discussed by providing an overview of cooperative targets, classical feature based solutions, machine learning based approaches, and natural scenario based autonomous landing techniques.

- **Cooperative Targets based solutions:** In these solutions, artificial markers are used to help the UAV orient itself for landing by identifying specific features. Different shapes like "T", "H", circular markers are used based on the problem definition. However, some landmarks are unsuitable for placing those such markers, making it highly challenging to land using these methods.
- **Classical Feature-Based Solutions** Feature based solutions involve artificial markets of which their design is solely on geometric patterns. Similar to cooperative targets based solutions, various shapes are used for target detection and UAV attitude estimation. However, these methods some limitations, such as the dependence on the quality of the visual markers, which are easily affected by noise and other environmental conditions.
- **Machine Learning-Based Solutions** Machine learning techniques focus on increasing the adaptability and robustness of the solutions beyond what traditional techniques propose. Most common machine learning techniques include classifier based methods, like Support Vector Machine (SVM), deep learning approaches such as convolutional neural networks for the purposes of enhancing the target detection and landing performance in various scenarios.
- **Natural Scenario Based Autonomous Landing** Unlike the previously mentioned techniques, the natural scenario based method does not rely on markers. It is particularly useful in situations like rescue missions where the UAV needs to operate without prior setup. This involves scene matching and 3D reconstruction via sensor and Simultaneous Localization and Mapping (SLAM) technology.

4.3.2. Relative state estimation

GPS-INS Systems

Autonomous landing of UAVs requires the accurate measurement of the state parameters of both the drone and the landing site to obtain relative position and attitude. The key technologies used in this process involve a combination of various sensors and navigation techniques. The Global Positioning System (GPS) and Inertial Navigation System (INS) are two of the most common techniques used as part of a sensor navigation system. During the autonomous landing of UAVs, the INS provides a set of navigation parameters, including position, velocity, and attitude [64] at a high rate of transmission. Although GPS transmits data at a slower rate, providing more accurate sensory information regarding the geological and time data. As both systems are essential elements of the sensory navigation system, they are often used together through various sensor estimation techniques. Combining the characteristics of both systems to obtain highly accurate data is crucial for making the necessary adjustments for a precise landing.

4.3.3. Tracking and Landing

Tracking the target closely and searching for a point to execute the landing manoeuvre are the final steps for achieving a successfully landing procedure. This final phase is accomplished with an advanced guidance and control scheme. Various guidance and control techniques have been applied to landing problems in the literature, ranging from simple PID controllers to advanced learning based adaptive approaches. While the classical techniques and how they are used for the landing will not be given in this section, the limitations that create a strong need for learning based approaches will be discussed in detail.

Addressing the limitations of classical techniques in autonomous landing

Classical techniques for autonomous landing have mostly relied on Proportional Integral Derivative (PID), robust flight, and model predictive controllers. While these methods have their specific contribution to the problem, each have certain limitations which may cause them to fail in specific situations.

PID controllers have been a simple yet efficient choice for landing problems in many studies. Although they are widely used, they face many challenges. One significant challenge the need for manual tuning of its parameters [68, 65]. In the context of UAV landing, the parameters of PID controllers need to be tuned to operate effectively in highly dynamic environments. A PID controller with fixed gains may not respond well to external, nonlinear dynamic disturbances, which are highly common in especially maritime operations. To address the challenges, recent studies have proposed to use machine learning techniques such as a reinforcement learning. This approach allows the controller to adapt to autonomously adapt to various scenarios. This combination of PID controllers with deep reinforcement learning has shown great promise in improving landing performance.

More advanced techniques such as Model Predictive Controllers (MPCs) and Robust Flight Control techniques are also commonly used for autonomous landing of UAVs. Compared to PID controllers, MPC offers additional advantages, such as the ability to predict and calculate the landing trajectory in advance [14]. This planning process continues until the UAV gets very close to the platform, at which point the sampling time decreases, leading to re-planning at high frequency. While being computationally expensive, this high frequency results leads to high accuracy trajectory planning. MPC can also be used in conjunction with other controllers, such as Incremental Nonlinear Dynamic Inversion (INDI) [23], to improve robustness against external disturbances. Robust flight controllers, such as fuzzy logic, H2 and Hinf, are also common methods for autonomous landing. These controllers can adapt to varying environmental conditions, making them suitable for a wide range of applications.

However, both methods have struggles with the models used for the problem. While MPC relies on high-fidelity models that may not be always available, robust techniques face issues with model uncertainties. Additionally, both methods, especially MPC, are computationally expensive. MPC generally tries to solve a generic, possibly non-convex, nonlinear program which can lead to numerical issues [41]. Furthermore, due to the complex nature of the landing process, it may be slow and extremely power-consuming. These limitations highlight the need for more efficient and adaptive control techniques for autonomous landing.

4.4. Conclusion and Discussion

This chapter has provided an in-depth exploration of the autonomous landing problem for UAVs, particularly focusing on the unique challenges and requirements of ship landing scenarios. By addressing the first set of research questions, **RQ-AL**, a comprehensive understanding of the nature of autonomous landing and a motivation adopting advanced, learning-based solutions have been established.

To summarize the key points discussed in the chapter, the key phases and challenges in autonomous landing and the limitations of classical techniques are given in Table 4.1, 4.2, and 4.3.

Table 4.1: Key Phases in Autonomous Landing

Phase	Description
Target Detection	Identifying the landing target using vision-based systems.
Relative State Estimation	Determining the UAV's position, velocity, and orientation relative to the target using GPS-INS systems.
Tracking and Landing	Closely tracking the target and executing the landing maneuver through advanced guidance and control schemes.

Table 4.2: Challenges in Autonomous Landing

Challenge	Description
Real-time Massive Information Processing	Handling large amounts of data in real-time, considering the interaction between the UAV and its environment.
Limited Onboard Resources	High computing power required for vision-oriented algorithms and limited onboard resources.
High Maneuverability of UAV Platforms	Fast and accurate feedback of state information required due to high maneuvering capability of rotary-wing UAVs.
Limited Image Processing Algorithms	Traditional algorithms face challenges in adapting to different scenarios, impacting stability and robustness.
Dynamic Scenes	Advanced navigation and control systems needed to react to both the UAV and the moving platform.

Table 4.3: Limitations of Classical Techniques

Technique	Limitations
PID Controllers	Requires manual tuning of parameters and struggles with nonlinear disturbances.
Model Predictive Controllers	Computationally expensive, relies on high-fidelity models, and can lead to numerical issues.
Robust Flight Controllers	Faces challenges with model uncertainties and high computational costs.

This comprehensive overview provides a solid foundation for the subsequent chapters, which will dive into the implementation and evaluation of learning based solutions for the autonomous landing of UAVs.

5

Machine Learning for Optimal Guidance

The application of machine learning in UAV operations has been transformative as UAVs become more commonly used in various areas, from environmental monitoring to military operations, necessitating more sophisticated adaptive guidance and control techniques.

Machine learning techniques offer promising solutions to address the existing challenges of the autonomous landing problem and shortcomings of the traditional techniques, as detailed in Chapter 4. In line with the given research questions **RQ -ML**, this chapter aims to discuss the machine learning techniques in the context of optimal control and guidance problems. The discussion begins with an overview of optimal control theory, specifically for defining the role of objective functions and corresponding optimal inputs. This is followed by an in-depth look at different machine learning techniques, including supervised learning, unsupervised learning, and reinforcement learning. Each method's specifics and potential applications in UAV operations are examined, highlighting their strengths and limitations.

A significant portion of this chapter is dedicated to reinforcement learning due to its ability to learn optimal policies through interactions with the environment (trial and error) while being highly adaptive and robust at the same time. This makes reinforcement learning an effective solution, as these abilities are critical during ship landing under extreme conditions. Within the chapter, the state-of-the-art techniques of reinforcement learning for landing problems are discussed in detail, and real life applications where reinforcement learning outperforms the classical methods are provided.

5.1. Optimal Control Theory

Classical control systems make use of several methods to determine the namely "acceptable system" through a trial and error process. The performance of this system is judged by parameters such as rise time, settling time, peak overshoot, gain and phase margin, and bandwidth. However, as these systems are constrained by the requirements (ex. min fuel) of the real world classical control techniques often fall short. Optimal control theory as a more direct approach to such complex problems has been made feasible with the development of digital computers. The main objective of optimal control theory is defined as a process to satisfy these physical constrains while minimizing (or maximizing) some kind of performance criteria [34].

5.1.1. Formulation of the optimal control problem

Optimal control problems can be formulated in different ways depending on the system's equations, objectives, and constraints.

Consider the following dynamical system:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{5.1}$$

where $\mathbf{x} \in R^n$ is the state vector and u is the control vector.

In optimal control, the purpose is to find the "admissible control" inputs that satisfy the constraints during the entire time interval. To find the state-control trajectory $\{\mathbf{x}(t), \mathbf{u}(t) : 0 \leq t \leq T\}$, the general optimal control formulation could be defined as follows:

$$\begin{aligned} \underset{\mathbf{u}(\mathbf{t}),T}{\text{minimize}} \quad & J(\mathbf{x}(t), \mathbf{u}(t), T) \\ \text{subject to} \quad & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \quad \forall t, \\ & \mathbf{x}(0) = \mathbf{x}_o \\ & \mathbf{x}(T) = \mathbf{x}_f \end{aligned} \tag{5.2}$$

where $\mathbf{x}_o$ is the initial state, $\mathbf{x}_f$ is the target state, J is the objective function determining the path cost and T is the total trajectory time. Objective functions are defined based on the selected performance measures which are limited by the system and environment.

The value function that represents the minimal cost to reach the goal location for the cases where the final time T is left free is defined as

$$v\left(\mathbf{x}_0\right) = \min_{\mathbf{u}} J(\mathbf{x}(t), \mathbf{u}(t)) \tag{5.3}$$

Note that the finite horizon control problem carries characteristics similar to an infinite horizon problem. The value function 5.3 can be introduced as the solution to the partial differential equation [52]:

$$\min_{\mathbf{u}} \{\mathcal{L}(\mathbf{x}, \mathbf{u}) + \mathbf{f}(\mathbf{x}, \mathbf{u}) \cdot \nabla_{\mathbf{x}} v(\mathbf{x})\} = 0 \tag{5.4}$$

which is subject to the boundary conditions $v\left(\mathbf{x}_t\right) = h\left(\mathbf{x}\left(t_f\right)\right), \forall \mathbf{x}_t \in \mathcal{S}$. The optimal control policy is then defined as:

$$\mathbf{u}^*(\mathbf{x}) = \text{argmin}_{\mathbf{u}} \{\mathcal{L}(\mathbf{x}, \mathbf{u}) + \mathbf{f}(\mathbf{x}, \mathbf{u}) \cdot \nabla_{\mathbf{x}} v(\mathbf{x})\} \tag{5.5}$$

Equations 5.4 and 5.5 represent the Hamilton-Jacobi-Bellman (HJB) equations for the free time, deterministic, optimal control problem. These equations are important as they indicate the uniqueness of an optimal state-feedback $u^*(x)$.

5.2. Solving Continuous Optimal Control Problems

Solving an optimal control problem is the determination of a policy that satisfies a certain objective. The main methods to solve specifically continuous optimal control problems are given in the diagram shown in Figure 5.1.

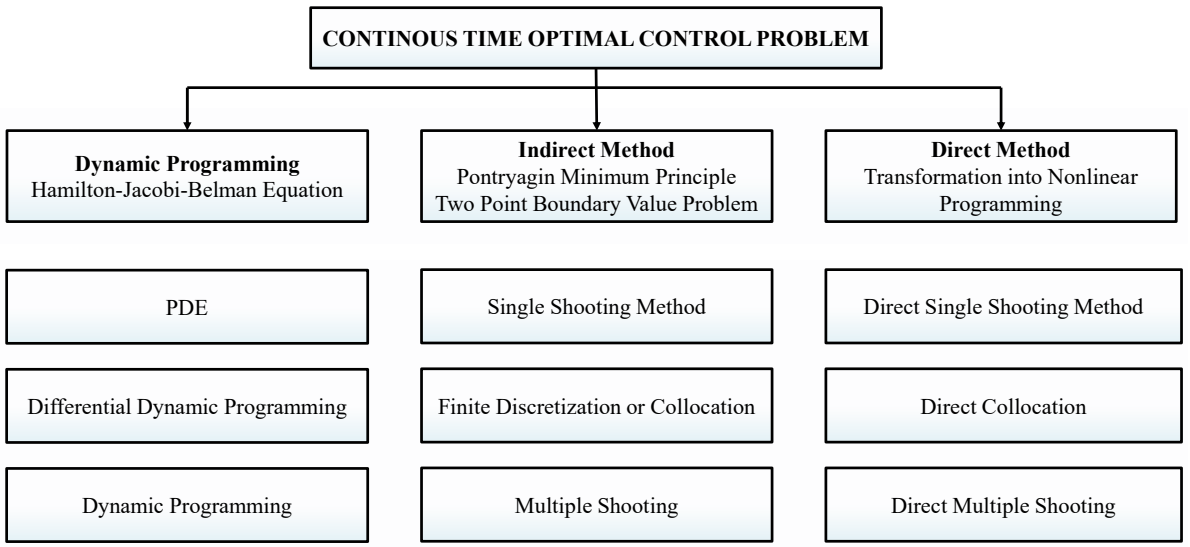


Figure 5.1: Classification of methods for continuous time optimal control problems

These methods are used based on the specific requirements of the problem such as the dynamics involved, the presence of uncertainty, and the computing resources available. In this section, only numerical methods are considered since they stand out with their efficiency, applicability and flexibility to the real-world applications when dealing with continuous time optimal control problems.

5.2.1. Numerical Methods for Optimal Control Problems

Optimal control problems are often quite complex, nonlinear problems for which coming up with an analytic solution is not straightforward or possible at all. Numerical methods, on the other hand, can efficiently handle the the complexity and nonlinearity of such problems and find approximate solutions. With the use of advanced technology, they can be used for solving a wide range of problems that were previously too computationally intensive to solve.

Numerical methods incorporate a certain type of iteration process with a finite set of unknowns [7] for solving optimal control problems. The common numerical techniques with the corresponding application areas are given in the following.

Direct Shooting

Direct shooting method is mostly used for launch vehicle and orbit transfer applications as these problems could be parameterized with a relatively small number of NLP variables. The programs namely POST and GST both make use of direct shooting technique. Earyl versions of POST utilized a reduced gradient optimization algorithm whereas recent releases have implemented the Sequential Quadratic Programming (SQP) method. GST on the other hand, uses a modified form of the reduced gradient algorithm that incorporates quasi-Newton updates for constraint elimination and Hessian approximation [9].

Indirect shooting

Indirect shooting is another numerical technique that transforms the original problem into a Two-Point Boundary-Value Problem (TPBVP) based on Pontryagin's maximum principle. However, it focuses on special cases (ex. launch vehicle trajectory design) due to the sensitivity of this method to the initial guess. As an example, The DUKSUP program [57] uses this technique to solve high thrust launch vehicle trajectory design and optimization problem.

Multiple Shooting

Direct and indirect shooting methods both suffer from small variations that are made early in the trajectory and propagate into nonlinear changes at the end of the trajectory. To overcome this issue, multiple shooting is introduced to solve two point boundary problems. Multiple shooting divides the trajectory into several pieces to enhance the robustness of the solution. Additionally, different phases could be run on individual process. This method is mostly used for examples with high difficulties such optimal interplanetary orbit transfer planning or landing in the presence of wind shear [8].

Direct Transcription

Direct methods could be used without deriving the necessary conditions such as adjoint, transversality, and maximum principle. On top of this, these method do not require an prior specification of the arc sequence for problems with path inequalities.

5.2.2. Optimal control applied to landing problems

Optimal control techniques, particularly numerical methods, have been used to address the landing problems. In this section, optimal control for landing is only examined in terms of the resultant control profiles and associated objective functions for specific landing problems.

In [51], Sanchez and Izzo builds on their previous work where they have used direct methods and failed to address complex problems such as pinpoint and thrust vector landing. Since direct methods caused numerical instabilities in the solution, the optimal solutions were obtained with indirect methods instead. In their work, they have considered different type of landing problems which are classified based on the dynamic models including a quadcopter, simple spacecraft, reaction wheel spacecraft and thrust vectoring rocket. The details of the models and the corresponding optimal control profiles is given in Table 5.1 and in Figure 5.6.

Time optimal control (TOC) and quadratic control (QC) are considered for the quadcopter models and mass optimal control (MOC) and quadratic control are considered for the spacecraft models.

Table 5.1: The considered models and corresponding optimal control problems

Model	$n_{\mathbf{x}}$	u_1	u_2	Variable mass	$\mathbf{g}$	Optimization problems
Quadcopter (QUAD)	5	N	rad/s	No	Earth	TOC, QC
Simple Sc. (SSC)	4	N	rad	Yes	Moon	MOC, QC
Reaction Wheel Sc. (RWSC)	5	N	rad/s	Yes	Moon	MOC, QC
Thrust Vectoring Rocket (TVR)	6	N	rad	Yes	Moon	MOC, QC

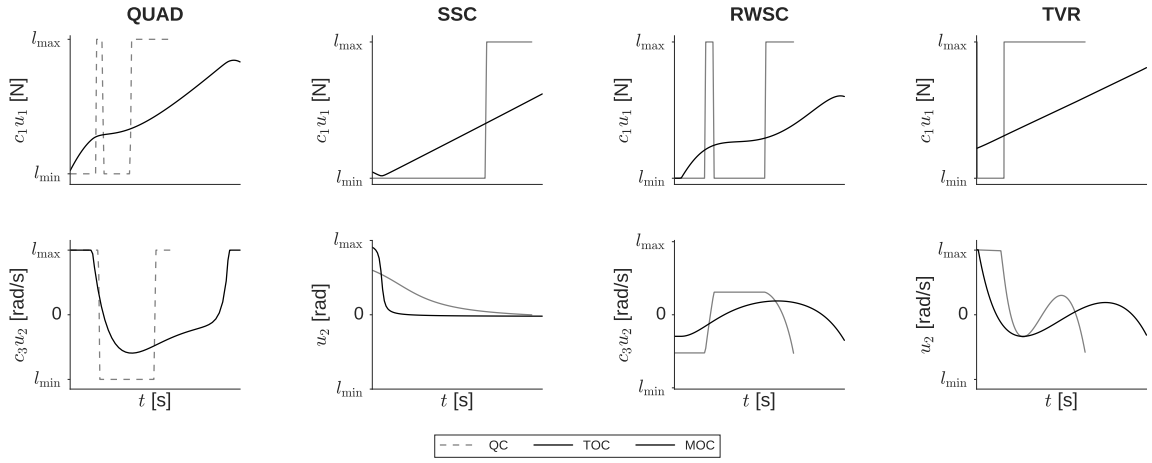


Figure 5.2: Optimal control profiles for different models and objective functions considered [52]

Figure 5.6 illustrates different control profiles including continuous control, discontinous control, bang-off-bang control and saturated control.

Sanchez et at all. distinguishes between different optimal control problems through the parameters in the cost function. For the quadcopter, the following cost function is given:

$$J = (1-\alpha)\int_0^{t_f} \left(\gamma_1 c_1^2 u_1^2 + \gamma_2 c_2^2 u_2^2\right) dt + \alpha \int_0^{t_f} dt \tag{5.6}$$

where u_1 and u_2 are the control and α is the continuation parameter. The parameter α lies in the range of $\in [0,1]$ and defines the transition between quadratic optimal control problem (QC), $\alpha = 0$ and a time optimal control problem (TOC), $\alpha = 1$. Similarly, the same α parameter is used to make the transition between a quadratic optimal control problem (QC) and a mass optimal control problem (MOC) for a spacecraft model via the following cost function:

$$J = \frac{1}{c_2}\left[(1-\alpha)\int_0^{t_f} \gamma_1 c_1^2 u_1^2 dt + \alpha \int_0^{t_f} c_1 u_1 dt\right] \tag{5.7}$$

In previous studies that considered the optimal control problem for drone racing [16] where the requirement is to finish the racing by passing through some obstacles, the trade-off is mostly done between energy and time optimality. Considering the given cost functions in Eq. 5.6 and Eq. 5.7 and generalizing them to this case, the example cost function that covers both situations in that case is given as follows:

$$J = (1-\epsilon)T + \epsilon \int_0^T \|\mathbf{u}(t)\|^2 dt \tag{5.8}$$

Here, the objective function is weighted by using the hybridization parameter ϵ with $\epsilon = 1$ corresponds to energy optimal flight and $\epsilon = 0$ corresponds to time optimal flight.

The choice of a good objective function is highly critical not only in optimal control problems but also in machine learning. In the context of optimal control, the objective function helps define the desired outcomes and be the determining factor for computing optimal control inputs. The accuracy and stability of control profiles that are generated are directly influenced by how well the objective function is designed for the problem. A poorly chosen objective can lead to sub-optimal or even unstable control operation which is a risk that cannot be taken in applications such as autonomous landing.

5.3. Machine Learning

Machine learning, particularly its field of deep learning, has extended to be used for guidance and control problems traditionally dominated by classical control techniques. These traditional techniques, while effective, of have their limitations in terms of model precision, disturbance rejection, and computational cost [46]. Since the pioneering work of Ivakhenko [30], which demonstrated the capabilities of multi-layer neural networks, machine learning has evolved to address these limitations.

The advantage of machine learning in optimal control comes from its ability to approximate and learn nonlinear functions and handle large datasets without explicitly considering or programming every possible outcome. This is particularly useful in control problems where a wide rang of initial conditions and external disturbances must be considered [52].

Machine learning techniques can be categorized into three groups: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning is for classification and regression where the models are trained to match the data with the predefined label and predict values, respectively. Common techniques could be given as support vector machines, decision trees etc. On the other hand, unsupervised learning tries to find patterns in the data without having labels on. Algorithms like K-means clustering and Mean Shift clustering are popular techniques used to group data into clusters. Lastly, reinforcement learning is another decision making branch of machine learning where the learning happens through maximizing a specific reward function over many steps. The diagram showing an overview of machine learning techniques are given in Figure 5.3.

5.3.1. Neural Networks

Neural networks (NNs) are models inspired by the biological neurons in the human brain. In a similar way, ANNs have interconnected nodes that are linked together and operate according to a set of learning rules. Each node in the neuron uses an activation function to process input signals and give outputs based on weight and bias parameters. Learning rules set the way the networks weights are adjusted adn changing thruohgut the training. These components together allow the neural network to learn and adapt to the given input. In the following Figure 5.4 a structure of a single neuron is shown.

Building upon the concepts introduced by ANNs, deep neural networks extend the structure of ANNs by incorporating multiple hidden layers between input and output layers. This multi layered structure allows neural network to have a more depth understanding of the input data and draw complex pattern out of it. This is especially useful for more sophisticated tasks such as speech recognition, predictive analysis etc. A representation of a deep neural network is given in Figure 5.5.

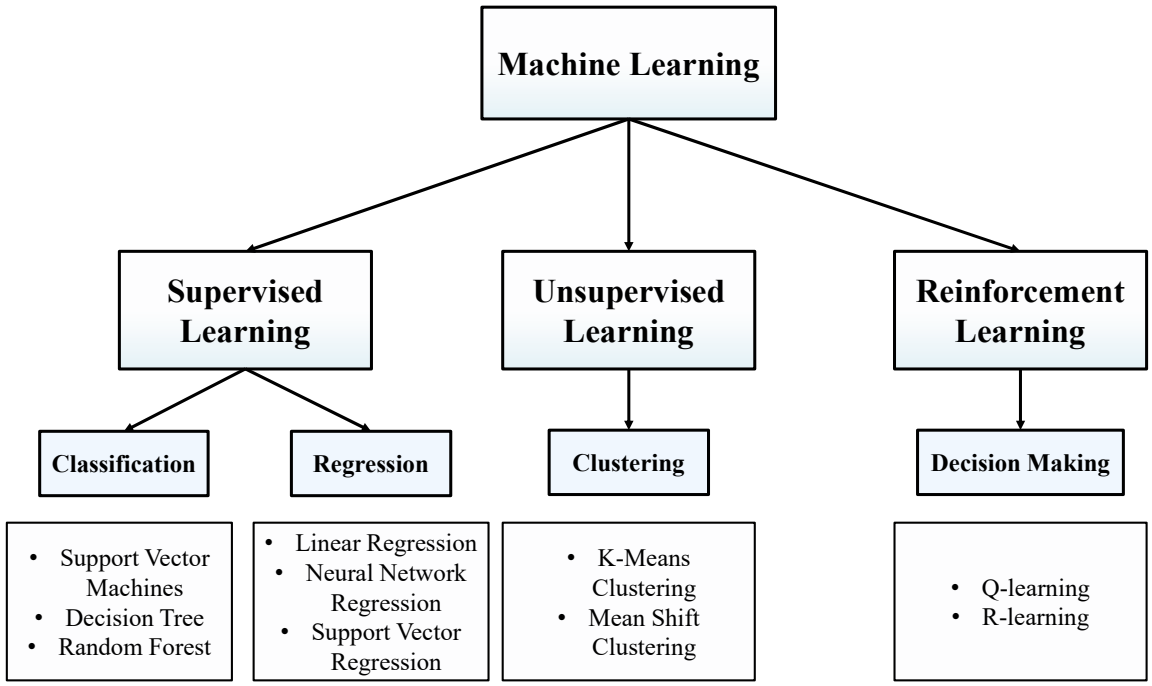


Figure 5.3: Machine learning algorithms classification [40]

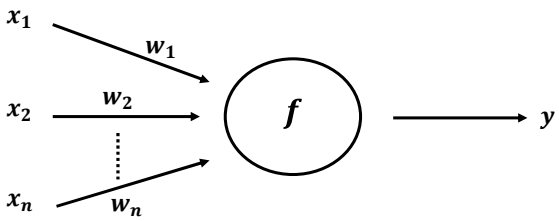


Figure 5.4: The structure of a single neuron

Activation Functions

Activation functions are mathematical operations that affect the output from ANNs. They allow network to capture complex model interdependences by introducing non-linearity into the network. By doing that, they influence the network's ability to express target functions[35]. Activation functions play a crucial role in neural networks and the choice of a certain activation function has a large impact on the learning capability and performance of the neural network. Expressions for common activation functions like Sigmoid, Hyperbolic Tangent, Rectified Linear Unit are given below in Eq. 5.9.

$$\begin{aligned} \text{Sigmoid}(x) &= \frac{1}{1+e^{-x}}, & \text{ReLU}x &= \max(0,x), \\ \text{Tanh}(x) &= \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}, & \text{Softmax}(x_i) &= \frac{e^{x_i}}{\sum_k e^{x_k}}. \end{aligned} \tag{5.9}$$

5.3.2. Supervised Learning

Supervised learning is one of the fundamental approaches in machine learning where a model learns from a set of labelled data which is provided by an external supervisor [10]. This supervisor is generally a human expert who can label the data correctly. Each sample in the data set includes the situation

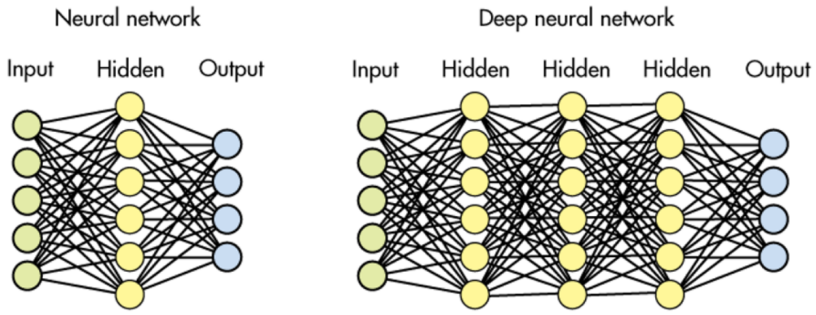


Figure 5.5: Neural network vs Deep neural network representation

description with the corresponding label that happens to be either an action or category for that situation.

The main goal in supervised learning is to find patters in the existing training data so that accurate predictions could also be made when faced new, unseen situations. However, supervised learning faces challenges in complex and dynamic environments. This is due to the fact that obtaining a data set that covers all possibilities that can happen in that environment is nearly impossible. For instance, interactive environments require the agent to discover and learn from its own experiences and in such cases, supervised learning would not work in an efficient way.

5.3.3. Unsupervised Learning

Unsupervised learning is mostly about discovering the pattern hidden within unlabeled data. Unlike supervised learning, unsupervised learning does not need an external supervisor to learn thus no prior human intervention is required. The nature of the supervised learning is more suitable tasks like clustering where the aim is group similar data together. Another application areas include dimensionality reduction, anomaly detection [21] etc.

5.3.4. Deep Neural Networks for Optimal Control

Solving an optimal control problem comes with a high computational cost which often makes it unfeasible for onboard real-time applications. Hence, in current applications, the optimal trajectory is precomputed while in real time it is tracked by an additional controller to correct the deviations from flight trajectory.

Although the application areas of DNNs were more parallel with perception related tasks, it has been just recently discovered its potential in control problems [37]. Some studies [38, 17] explored the use DNNs in deterministic continuous time nonlinear systems, however these were limited to simple scenarios involving linear systems or unbounded control. Others [22] have focused on solving Hamilton-Jacobi-Belmann equations and two point boundary value problem that arise from Pontryagin's optimal control theory. Furthermore, a lot of research used NNs to approximate the value function $v(t, x)$ while also looking solutions to the states, co-states and the controls to make sure that the assembled Hamiltonian respects Pontryagin's conditions [13]. Ongoing research indicates that deep architectures has the potential to replace all or some parts of the onboard decision-making system for the navigation and control. In the following section, the applications of DNNs to landing problems will be examined.

Deep Neural Networks for landing problems

In this section, studies that combine optimal control with deep neural for landing problems will be given and discussed in detail.

Sanchez et. all first make use of deep neural networks with supervised learning in the domain of optimal landing [52] to represent the optimal control structure for multicopter and spacecraft models which are deterministic, non-linear continuous systems. Optimal control problems differed in complexity and dimension are considered, namely the pinpoint landing of a multicopter and the landing for two different models of spacecraft. Additionally, different cost functions resulting in variety of control variables, i.e. smooth, saturated and bang-bang are considered. For each problem, only initial conditions are taken into account for the generation of training data.

Training and Data generation

For deep neural networks to mimic the solution to the optimal control problem, it is necessary to obtain the optimal state feedback pair with one of the numerical optimal control techniques.

Sanchez et all. use the direct method "Hermite-Simpson transcription" to solve the corresponding non-linear programming via a sequential quadratic programming NLP solver, namely SNOPT [20]. They generate 150,000 different trajectories starting from a point that is randomly sampled within the initialization areas A. They used 90% of the data to train the model while the rest is kept for validation.

Due to the problems cause by the use of direct method with saturated controls inputs, they have encountered chattering effects that negatively affected the learning process. This problem is tried be solved by the addition of a regularization term to the objective function however as seen from the results, this by itself was not enough to remove chattering effects completely which in turn made it not possible to study complex problems such as pinpoint landing and thrust vectoring.

In their later work [51], state action pairs for data generation were calculated with indirect methods and continuation techniques (i.e. single shooting) instead of direct methods to avoid such problems. Continuation methods were used to have a better initial guess as especially TOC and MOC problems require relatively precise guesses.

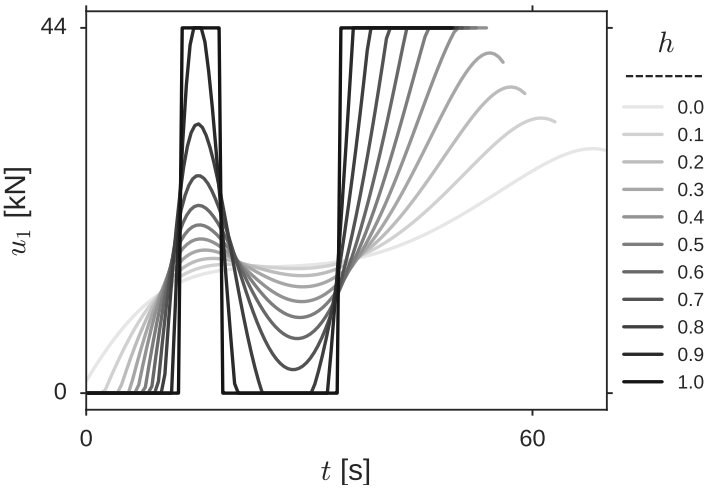


Figure 5.6: Transition from quadratic control (h = 0) to mass optimal control (h = 1)[52]

Sanchez and Izzo [52] uses DNNs to represent the solution to the corresponding Hamilton-Jacobi equation for four different cases of pinpoint landing that involved a quadcopter model, a mass varying spacecraft with bounded thrust, a mass varying spacecraft equipped with a reaction wheel and finally a mass varying rocket with thrust vector control.

Given the assumptions in the model, DNNs can be directly trained in a supervised way using the optimal state-action pairs previously calculated. These trained networks make an ideal choice for real-life applications as they require minimal CPU resources. In addition to that, the training process is completed offline thus not affecting the real-time optimal control architecture.

The trained networks as a result of this process enable the capabilities of real-time control without having an extra need for on-board optimal control methods (direct or indirect). Furthermore, they can operate even in situations they are not trained to handle which contribute to their robustness and potential applications.

5.4. Reinforcement Learning

Reinforcement learning can be defined as the process of agents interacting with an environment to maximize cumulative rewards. To do so, agents must go through a trial and error process to discover which actions return the most rewarded. In most cases, actions do not only affect the immediate reward but all subsequent ones. As a result, delayed reward and trial and error process are known as the most important features of reinforcement learning [58].

Reinforcement learning is fundamentally different from supervised and unsupervised learning. Unlike supervised learning where an external supervisor provides the actions that are associated certain situations, RL operate in a unsupervised way in unfamiliar environment without having any guidance. Furthermore, reinforcement learning is not about finding out the hidden structure within data like unsupervised learning. Instead, it's maximizing a reward function through interactions with the environment.

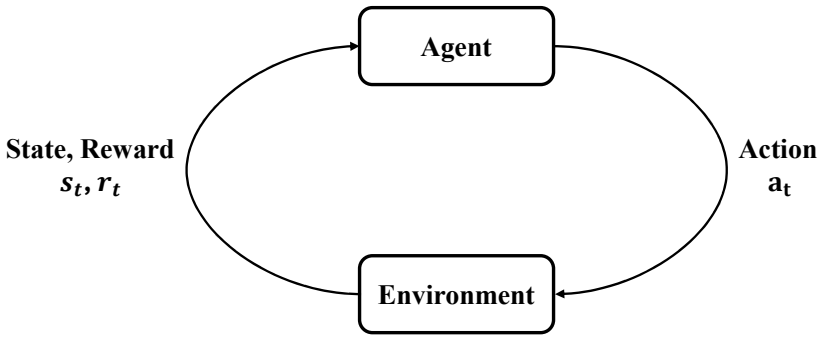


Figure 5.7: Agent-environment interaction

5.4.1. Elements of Reinforcement Learning

There are four elements that are defined as the main concepts in reinforcement learning: a policy, a reward, a value function and, a model of the environment [3].

Policy

In RL framework, a policy is defined as the way the agent behaving at a time. It is basically a mapping from inputs (states from the environment) to actions that are associated with those states. The policy may be as simple as a look up table or it may involve an extensive search. In either situiation, it is alone sufficient to define the agent's behaviour thus could also be defined as the core of RL.

Policies can be deterministic or stochastic, either specifying a single action or probabilities for each action, respectively.

Deterministic policy is denoted by μ:

$$a_t = \mu(s_t), \tag{5.10}$$

where stochastic policy is denoted by π

$$a_t \sim \pi(\cdot \mid s_t) \tag{5.11}$$

If the agent is following the policy π at time t, the expression $\pi(a \mid s)$ defines the probability that $A_t = a$ when $S_t = s$ [58].

Reward

Reward function in reinforcement learning is the objective of agent. The main goal for the agent to maximize the cumulative reward over time. Based on the received reward signal at a time, the agent could understand which actions are simply good and bad. The reward function is fundamental criteria to modify the policy. For instance, an agent received a low reward while following a certain policy may change it afterwards to choose a different action in similar situations in the future.

The reward function R depend on the current state, the current action and the next state and is represented as follows in Eq.5.12

$$R_t = R(s_t, a_t, s_{t+1}) \tag{5.12}$$

Return

Return is one of the central concepts in RL which is defined based on the cumulative reward received after certain time steps. The important parameter in calculating the return is *discounting* which impacts the selection of future actions. Eventually, the agent tries to take the actions that will return the maximum cumulative reward in time.

In mathematical terms, the return parameter G_t is expressed in terms of sum of the rewards received after time t, with each reward adjusted with the discount factor γ raised to the power of how many steps in the future the reward is received. The expression for the discounted return is given in Eq. 5.13.

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{5.13}$$

The discount factor γ varies from 0 to 1, representing the degree to which future rewards are considered less valuable then immediate ones. If the value of γ is lower, the less value the future rewards will hold and vice versa. As γ approaches 1, the agent becomes more farsighted [58] giving nearly equal weights to immediate and distant future rewards.

Value Function

Value function is similar to a reward function in a sense that it assess the performance of a certain policy but considering long term advantages instead of immediate benefits as it happens in the reward function. Essentially, the value of a state is a representation of the total rewards an agent expect to gather in the future, starting from that state. For instance, a state that often returns a low reward may still not be discarded in case it leads to other states with higher rewards. To express it in terms of human analogy, while rewards are like the feelings of pleasure or pain, values are more of refined and farsighted judgements of the relationship we have with the environment [58].

The value function of a state s under a policy π is expressed as $v_\pi(s)$ follows:

$$v_\pi(s) \doteq \mathbb{E}_\pi [G_t \mid S_t = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \right], \text{ for all } s \in \mathcal{S}, \tag{5.14}$$

where $v_\pi(s)$ is the expected value of random variable for the agent under the policy π at time step t. This is defined as the *state-value function for policy* π.

In a similar way, the value of taking action a in state s under a policy π is defined as $q_\pi(s, a)$:

$$q_\pi(s, a) \doteq \mathbb{E}_\pi [G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right] \tag{5.15}$$

q_π is the *action-value function for policy* π.

Optimal Policies and Optimal Value Functions

The solution to a reinforcement learning problem comes from finding a policy that will maximize the cumulative reward over time. A policy π is better or equal to another policy π' if the expected return is also better or equal to that of π' for all states. Mathematically, $\pi \geq \pi'$ if and only if $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in \mathcal{S}$.

The optimal policy is defined as the one that works better than all other policies. There may be more than one optimal policy, but they have the same *same optimal state-value function*, defined as v_*

$$v_*(s) = \max_\pi v_\pi(s) \quad \text{for all } s \in S. \tag{5.16}$$

Optimal policies also share the same *optimal action-value function* q_*:

$$q_*(s, a) = \max_\pi q_\pi(s, a) \quad \text{for all } s \in S \text{ and } a \in A(s). \tag{5.17}$$

We can write q_* in terms of v_* as

$$q_*(s,a) = E\left[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a\right]. \tag{5.18}$$

This expression represents the expected return for taking action a in state s after following an optimal policy.

Exploration vs Exploitation

The most famous challenge known as only specific to RL is known as the exploration - exploitation dilemma. This dilemma leads agent to create balance between exploiting what has been discovered and exploring to find better action selections in the future. To be able to maximize the cumulative reward, the agent should prefer the actions that had previously found effective. However, to have those actions it should also keep discovering and try the actions that is has not selected before. Neither of these two actions can be done without failing at a task hence, the agent must keep trying and progress what appears to be at the conditions. Even though there are certain approaches that work better for deterministic or stochastic cases, the exact solution to this dilemma still does not exist [58].

Overall, reinforcement learning is a unique framework that focuses on the entire system of a goal directed agent interacting with an uncertain environment. By definition, it includes explicit goals associated with the environments and agents, environmental perception and action selection under uncertainty and distinguishes it from methods that overlook between the interior elements or only address isolated components without considering their integration into a broader system.

Policy and Value functions

- RL consists of three important parameters: policy, value function, and model.
- Deterministic policy $a^* = \arg_a \max \pi(a \mid s)$ outputs the action that has the highest probability.
- There are two kinds of value functions as $V_\pi(s)$ and $Q_\pi(s,a)$, The first kind of value function $V_\pi(s) = E_\pi\left[G_t \mid s_t = s\right] = E_\pi\left[\sum_{k=0}^{\infty} \Upsilon^k r_{t+k+1} \mid s_t = s\right]$ means when we use policy π, how much value the agent can accumulate from status s to the end status; the second value function can be called Q function, $\mathrm{Q_\pi(s,a)} = E_\pi\left[\mathrm{G_t} \mid \mathrm{S_t = s, A_t = a}\right] = E_\pi\left[\sum_{\mathrm{k=0}}^{\infty} \gamma^{\mathrm{k}} \mathrm{R_{t+k+1}} \mid \mathrm{S_t = s, A_t = a}\right]$, this value function use the current state and current action to estimate expectations for future rewards.

On-policy vs Off-policy Learning

In an reinforcement learning framework, two approaches: on and off policy methods define how the agent learn from the environment.

On-policy: On policy methods are about learning from what the agent is doing currently. They do not make use of old data, thus they are weaker in terms of sample efficiency. However, this leads the algorithm to directly optimize the objective by trading off sample efficiency for a stable performance. This method allows the agent to react and adapt through direct engagements with the environments. As an example, on policy learning could be a robot trying to find its way in a maze by trying and learning from its own actions.

Off-policy: In contrast to on-policy methods, off-policy methods improve a policy by using a different one that is used the generate the data.This way, they are able to utilize very old data. This gives the chance to exploit the Bellman's equation however, there is no guarantee that satisfying Bellman's equations will lead to having a great policy performance. Hence, compared to on-policy methods, this class of algorithms are more brittle and unstable [62].

5.5. State of the Art Deep Reinforcement Learning Techniques

Deep reinforcement learning extend upon the traditional reinforcmeent learning by incorporating deep neural networks to handle high dimensional, continuous state and action spaces. Deep RL algorithms are categorized into two groups as model-free and model based approaches. Model free methods make no assumption about the model and directly learn from interactions with the environment. These techniques have the advantage of simplicity in implementation however, they may require large number of samples to obtain an efficient policy.

On the other hand, model-based approaches learn through the model of the environment. These techniques may benefit from the models ability to generate data for training however, there is also a possibility of of suffering from model bias in case the model is not an accurate representation of the environment. The diagram showing the model free and model based approaches exist in deep reinforcement learning is given in Figure 5.8.

In this section, only model-free approaches are considered due to their applicability to continuos control problems where the state and action spaces are high dimensional and nonlinear dynamics are involved.

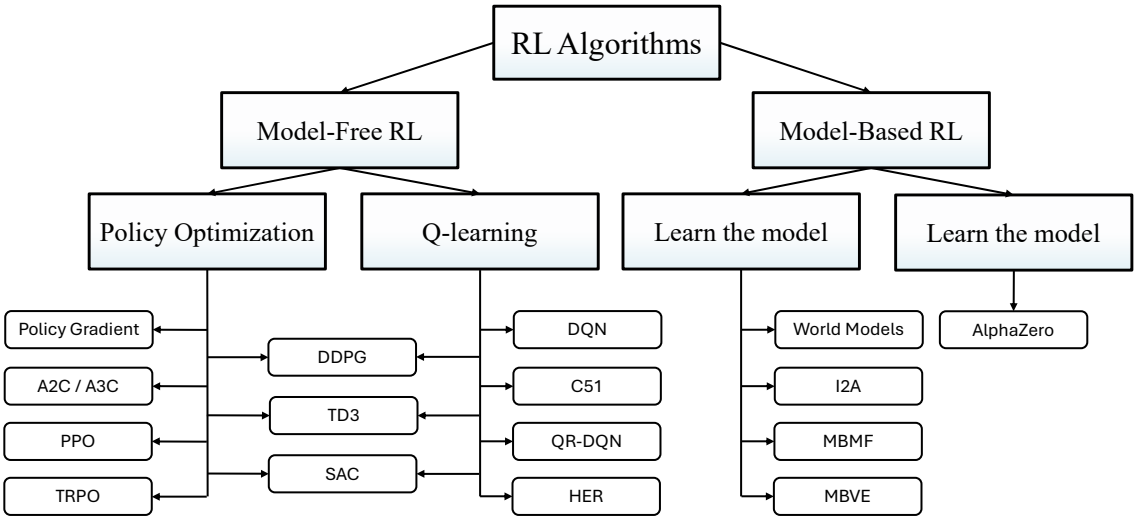


Figure 5.8: State-of-the-art reinforcement learning algorithms [42]

5.5.1. Model-Free RL

In model-free reinforcement learning, agent train without modeling the dynamics of the environment. The focus in on learning directly from the experience gathered from the environment. There two approaches involved with model free RL for training agents: policy optimization and Q learning.

Policy Optimization

Policy optimization methods are centered around directly adjusting the policy $\pi_\theta(a \mid s)$. The optimization of the parameters θ is done either through gradient descent on the performance objective or maximizing local approximations of $J(\pi_\theta)$. This optimization is often performed on-policy, showing that the policy updates are done with data collected only from the current policy not from a different policy's actions. In policy optimization, an approximator $V_\phi(s)$ is used to update the policy in consideration by indicating if it is preferred or not to stay in the current state. Common examples of Policy Optimization algorithms are given as follows:

- **A2C/A3C**: This method has an actor-critic architecture in which the actor directly updates the policy according to the gradient and the critic estimates the value function.
- **PPO (Proximal Policy Optimization)**: PPO makes the learning proces more stable by employing a surrogate objective function which gives a rather conservative estimate for the amount $J(\pi_\theta)$.

PPO limits the size of policy updates, avoiding destructive large updates. It makes use of trust regions to ensure that the deviations from the current policy are not significantly large, making the learning process more reliable and stable.

Q learning

Q-learning is another model-free reinforcement learning technique that learns an approximator $Q_\theta(s,a)$ for the optimal-action value function $Q^*(s,a)$ which indicates the maximum expted value of the total reward, in state s with the action a.

Q-learning typically operates off-policy which means that the agent learns indirectly by using data that may be collected from different policies. The associated policy is obtained from the connection of $Q*$ with $\pi*$. The actions taken by the agent in Q-learning is given as follows:

$$a(s) = \arg\max_a Q(s,a). \tag{5.19}$$

Common examples of Q-learning methods are given in the following.

- **Deep Q Network (DQN)**: DQN uses neural network to approximate the state-value function in Q-learning framework. It is generally used in combination with *Experience Replay* to store the episodes in memory for off policy learning where samples are randomly drawn from the buffer later in the process. On top of this, the network is generally optimized towards a frozen target network of which paramaters are updated periodically. These two properties together propose a solution to the autocorrelation problem in on-line learning and have the purpose of making the training more stable [28].
- **Categorical DQN (C51)**: Categorical DQN builds upon the DQN network by modelling the value function as a continous probability distribution. This way, the entire distribution of possible outcomes are predicted rather than just an average. Some of the hyperparameters included in C51 are the number of atoms, minimum Q-value, maximum Q-value etc. [11].

5.5.2. Reinforcement Learning for optimal control

Fundamentally, reinforcement learning acts as a bridge bringing together the concepts like traditional optimal control, adaptive control and bio-inspired methods [**bibid**]. From this perspective, RL could be defined as a heuristic process where an agent tries to maximize its future rewards, in equivalence, minimizing the overall control cost. This process of learning could be seen as development of the optimal policy.

Reinforcement learning, particularly q-learning, was recognized as a technique for achieving adaptive control by Sutton et al. in [59]. The authors further discussed in 1995 that dynamic programming based learning could actually bridge the gap between artificial intelligence and real time control and planning [4]

The nature of adaptive control in reinforcement learning could be traced back to the development of the adaptive actor-critic model by Barto and Sutton [5] for which later Xin and Balakrishnan provided a convergence proof. Later on, Bertsekas provided the details of the optimal adaptive control theory via Adaptive Dynamic Programming in his work [6]. Adaptive actor-critic methods are recognized as one of the most effective approaches in reinforcement learning for achieving optimal adaptive control [26].

The progress of RL has been towards to filling the gaps in optimal control by being able to adapt to unknown dynamics and unpredictable changes (ex. unpredictable motions) in the environment which traditional techniques often fail to accomplish [1, 39]. In conventional optimal control, it is required to have complete knowledge of the system dynamics with the predefined models of the environment thus making it not suitable for more dynamic environments. On the other hand, reinforcement learning offers a robust framework for dealing with such changes thanks to its ability to learn from raw environmental data without having prior information of the system dynamics. This attribute makes RL highly suitable for robotics and complex control tasks where more sophisticated methods are required to use.

Reality gap problem

Despite being highly efficient in solving control problems, RL faces a challenge known as the reality gap problem in real world applications [50]. The reality gap occurs due to relying too much on simulations and having a performance gap between simulation and real life. Even though these

simulation environments are often good representatives of real world situations, as the complexity of the problem increases, the reality gap becomes also bigger and performance of the RL framework degrades. Due to this reason, studies often adapt an approach where they combine the high level RL framework with a low level controller [44].

However, it is indicated that [32] these combined approaches limit the optimality of the controller with margins put for potential disturbances. Recent studies show that end to end RL frameworks that directly map raw sensor data to control inputs and get rid of the intermediate steps in between could actually outperform the combined approaches [29]. The further implementations included modelling the thrust and moment models externally and then putting in the control loop [16].

5.5.3. Applications of Reinforcement Learning for autonomous landing

The use of reinforcement learning in autonomous landing problems is relatively new and research in this topic is highly limited in the literature. Only a few studies have made attempts to solve this problem in which the considered landing problems differed in terms of application area, the type of sensory input (low resolution images [45], monocular camera [49]) received and the motion characteristics of the platform (moving or stationary) targeted. In this section, the approaches used in these studies will be given and examined in detail to understand how the choice of environment, RL algorithm, and reward function relate to the type of landing problem.

The authors in [45] made one of the first attempts and proposed a Deep Q-Network based algorithm for UAV autonomous landing on a static pad using only low-resolution images coming from a down-looking camera. Taking the complexity of the problem into account, two sub-tasks: landmark detection, and vertical descent are considered which are both handled by two independent DQNs that can communicate with each other via an internal trigger. For the landmark detection, the UAV moved only in the xy plane until it aligned itself with the marker. In the vertical descent phase, it decreased its altitude by keeping the marker in the center. The representation of these sub-tasks is shown in Figure 5.14.

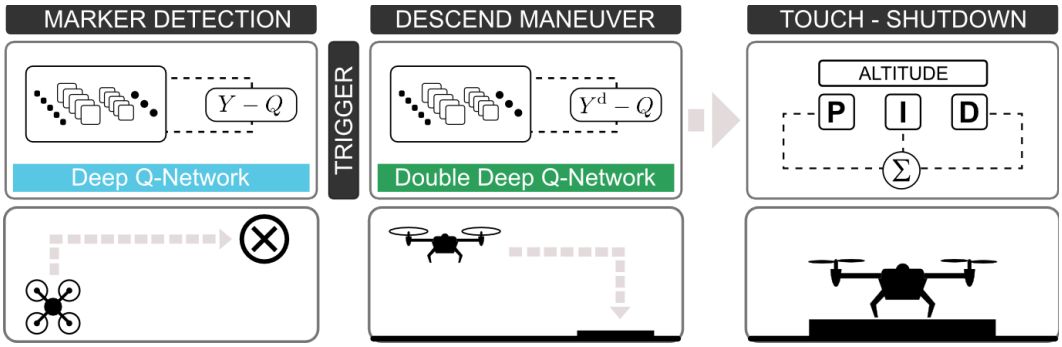


Figure 5.9: Landmark detection and vertical descent [45]

The given figure shows the component of the proposed landing system where the first DQN handles the marker detection and the second DQN is for the descending maneuver that takes place between 20 to 1.5 meters. For the last phase of the landing, they made use of a closed loop controller working from the height of 1.5 m to ground level.

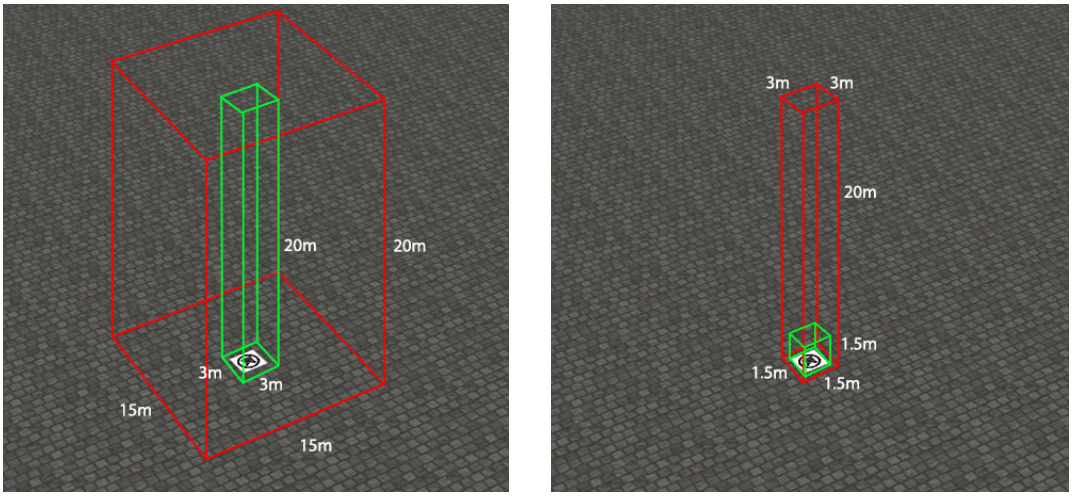


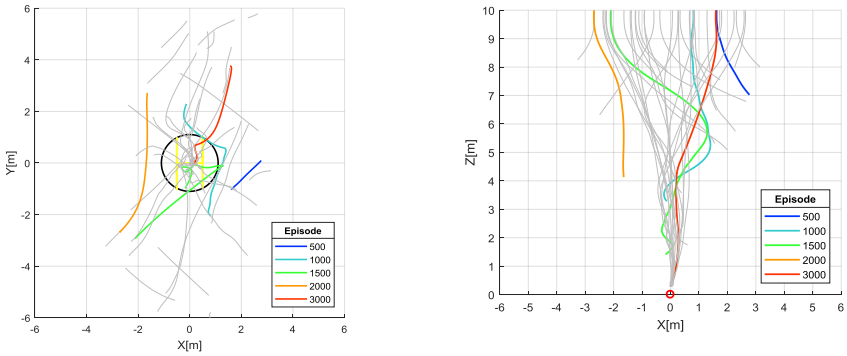
Figure 5.10: Bounding boxes for target detection [45]

The authors indicate that highly challenging nature of the landing problem is caused by its being a form of Blind Cliffwalk where the agent gets a negative or positive reward at the end of the landing process while it has to pass through all intermediate phases successfully. This problem is extra challenging cause the target area is only a small portion of the state space [45]. As a solution, they propose the use of double DQN approach [63] with partitioned buffer replay to guarantee a fair sampling between experiences.

Actor-Critic Framework for the Autonomous Landing

Lee et all. [36] developed an actor-critic framework for solving the vision based autonomous landing problem. Their approach included a PID based inner attitude controller, a ground looking camera model and a laser rangefinder. The system is trained to generate suitable roll and pith commands for successful tracking and landing while altitude and heading was not controlled by the agent but instead given constant control commands.

The actor-critic framework used for the corresponding system is given in Figure 5.12. The networks included two fully connected hidden layers with different weights. Reward function is used to update the critic network which is only activated during training. After the training is completed, only actor network is used for the landing in case there is a need from the human pilot.



(a) Flight trajectories (top view)

(b) Flight trajectories (side view)

Figure 5.11: Flight trajectories from training [36]

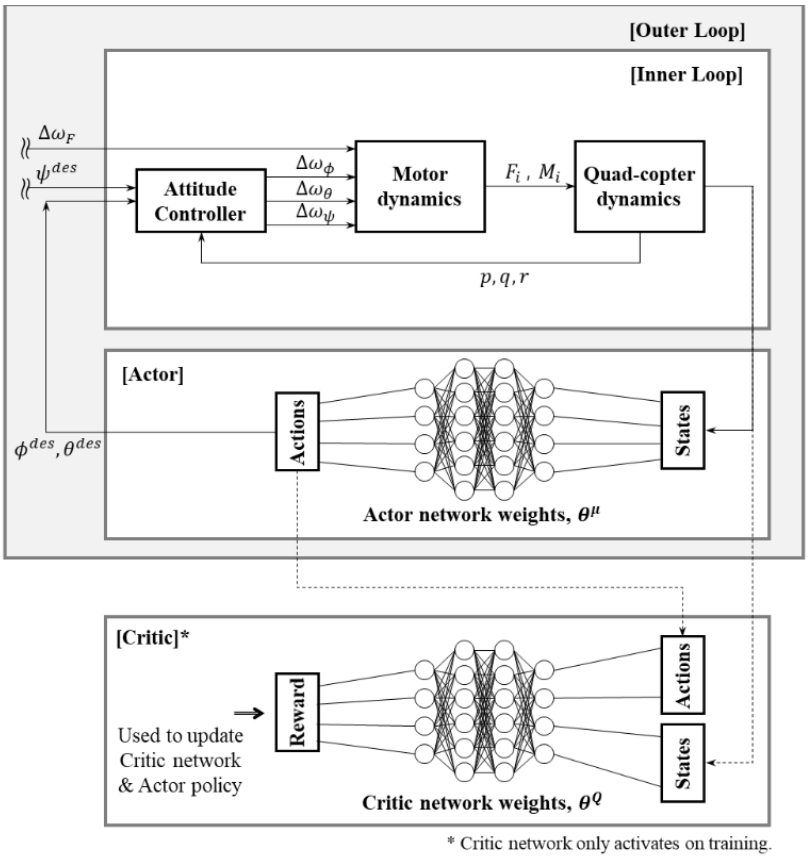


Figure 5.12: Actor-critic framework for vision based autonomous landing [36]

Resultant flight trajectories from the training process are shown in Figure 5.11. As a final note, the authors mentioned that the real life experiments resulted in non-smooth trajectories for which delay in sensor systems and external unmodelled disturbances are given as the main reasons.

Autonomous landing on moving platforms

Studies that are mentioned so far considered the target as a stationary platform however, certain real life situations (ex. ship landing) require UAV landing to be on moving platforms. [46] carries the importance of being the first study accomplished RL based landing on a moving platform. The study utilized the Deep Deterministic Policy Gradients (DDPG) for the landing problem and for simulations they used the Gazebo platform. The considered framework is given in Figure 5.13

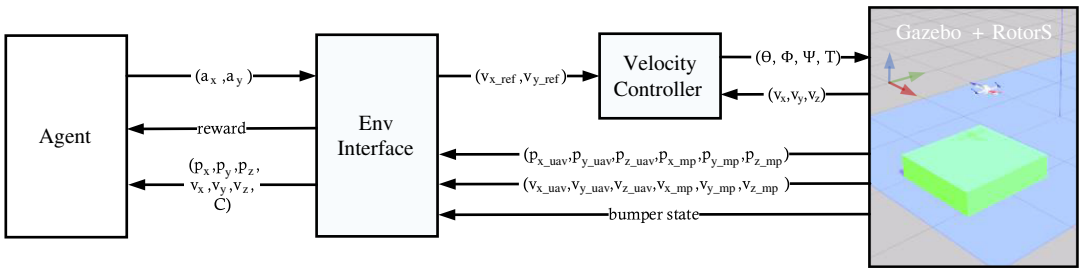


Figure 5.13: Reinforcement Learning Framework [46]

The formulation of the reinforcement learning framework is given through the following state 5.20 and action space 5.21 definitions.

$$\mathbb{S} = \{p_x, p_y, p_z, v_x, v_y, C\} \tag{5.20}$$

$$\mathbb{A} = \{a_x, a_y\} \tag{5.21}$$

Here, p_x, p_y and p_z are defined as the relative positions of the UAV with respect to the platform, v_x and v_y are the relative velocities and finally, C is the state of a pressure sensors located on the platform. a_x, a_y are determined as reference velocities and given as inputs to the controller (see Fig 5.13). However, the work did not consider the z-motion due to reasons given for the higher complexity of the vertical motion. Instead, a constant velocity commands were feeded at every time step.

To be able to generate smooth and continuous control actions, the following reward function was designed.

$$\begin{aligned} \text{shaping}_{\,t} = &-100\sqrt{p_x^2 + p_y^2} - 10\sqrt{v_x^2 + v_y^2} - \sqrt{a_x^2 + a_y^2} \\ &+ 10C\,(1 - |a_x|) + 10C\,\left(1 - \left|a_y\right|\right) \end{aligned} \tag{5.22}$$

$$r = \text{shaping}_{\,t} - \text{shaping}_{\,t-1}$$

In the design of the reward function, relative states of the UAV with respect to the platform are weighted with different coefficients to effectively vary their contribution to the overall reward function. Additionally, *shaping* technique was used to speed up the simulations. In the study, a chosen network has been tested in two scenarios, namely slow and fast where the maximum velocity is given as 0.4 m/s and 1.2 m/s, respectively. The success condition is defined as if the UAV touched the platform surface within an area of 1.0 m x 1.0 m. As indicated by the authors, unsuccessful cases were mostly due to the fact that the UAV was getting out of range and the given velocity was constant during the flight.

Although the study was successful in the implementation, [65] claims that non-existency of z axis in the problem definition and weak generalization capability of Gazebo platform result in autonomously incapable agents. In their study, Xie et all. divides the problem into two different parts: perception and relative pose estimation; trajectory optimization, and control of which they only considered the latter. They take the sensor noise, intermittent measurements and, randomness in the UAV movement into account in their definition of the landing problem. Additionally, as opposed to previous research [45, 46], incomplete and inaccurate observations were also taken into account. With these in consideration, a dynamic model based partially observable Markov decision process (POMDP) [56] was defined to present the UAV tracking and landing problem.

The state space considered in the paper is given as:

$$s = \{X_u, Y_u, Z_u, v_{ux}, v_{uy}, v_{uz}, X_t, Y_t, Z_t, v_{tx}, v_{ty}, v_{tz}\} \tag{5.23}$$

where $v_{ux}, v_{uy} \in [-10, 10], v_{uz} \in [-1, 3], v_{tx}, v_{ty} \in [-5, 5], v_{tz} = 0$.

The action space A used the speeds of the UAV as $A = \{v_x, v_y, v_z\}$. Finally, the observation space is defined as

$$\Omega = \{X_u', Y_u', Z_u', v_{ux}', v_{uy}', v_{uz}', X_t', Y_t', Z_t', v_{tx}', v_{ty}', v_{tz}'\}\,. \tag{5.24}$$

Since one of the main goals is to minimize the distance of the UAV to the moving platform, reward function is defined based on the distance parameters as given in 5.25

$$R = \begin{cases} -10, \text{ dist } > 6 \\ -0.1 \times \text{ dist }, 0.8 \leq \text{ dist } \leq 6 \\ +10, \text{ dist } < 0.8 \end{cases} \tag{5.25}$$

where the variable *dist* is

$$\text{dist } = \sqrt{(X_u - Y_t)^2 + (X_u - Y_t)^2}.$$

Here, reward distribution is made considering the problem of agent not receiving enough rewards till it reaches the target location thus requiring intermediate steps to make the landing.

Hybrid Strategy

Similar to the previous studies, Xie et all. also considered a hybrid approach to solve the landing problem. In their approach, tracking and landing is handled differently, the former by making use of reinforcement learning and the latter with heuristic rules defined by the authors themselves. The schematic for the hybrid strategy is shown in Figure C.10

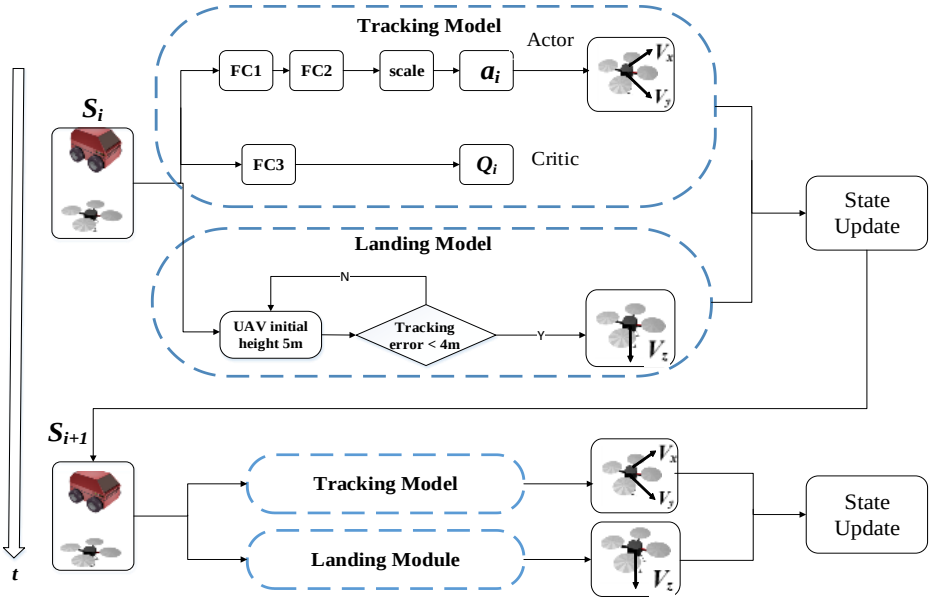


Figure 5.14: Hybrid strategy used in [65]

Tracking part of the hybrid strategy is accomplished with the algorithm DDPG in an end to end way whereas the landing part is assumed to be successful if the vertical distance is less than 0.1m while the horizontal distance error is less than 0.8 m.

The performance of RL was evaluated using the root mean square error (RMSE) and a parameter called tracking success rate (TSR) which was only considered during tracking tests [65]. The definition of the TSR variable is given as follows:

$$\text{TSR} = \sum_{i=0}^{N} \frac{D_i}{N} \times 100\%, D_i = \left\{ \begin{array}{l} 1, \text{ dist } < 3 \\ 0, \text{ dist } \geq 3 \end{array} \right\} \tag{5.26}$$

where D_i is assessed either 1 or 0 based on the success of landing, and N is the number of total time steps.

Given these performance parameters, Xie et all. made a comparison between RL and PID control. During the tests, the platform was moved in four different ways: linear, circular, and random motions. It was found that the motion of the platform has an effect on which algorithm outperforms the other one. According to the results given in Table 5.2, the proposed algorithm was being outperfermed in a linear and circulate motion while there is a significant performance upgrade in random motions. The reason for this was indicated as the PID control could not find the patterns of nonlinearity in its control inputs thus leading to a suboptimal performance.

Movement Type	Linear	Circular	Random1	Random2
PID method	97 %	85 %	56 %	41 %
Proposed method	94 %	83 %	70 %	74 %

Table 5.2: Landing performance comparison of PID and RL for different movement types

Reinforcement Learning for Ship landing

Ship landing is a challenging problem for RL to solve due to small landing space, hard-to-predict ship movements, limited resources for localization and harsh environment conditions such as wind gusts. Saj et all. [49] claim that previous studies on using RL for landing problems lack the robustness component which lead to development of not fully efficient algorithms that are not useful for ship landing problems. Considering all these challenges, they adapt the state of the art twin delayed DDPG (TD3) algorithm to land an UAV on a ship of which position to UAV was previously computed with vision based tracking.

They define the state space for the given problems as follows:

$$s_t = (p_t, v_t, p_{t-1}, v_{t-1}, \ldots, p_{t-5}, v_{t-5}) \tag{5.27}$$

where p_t and v_t are the relative position and velocity of the UAV with respect to the ship.

Action space was determined based on the characteristics of the Parrot Anafi drone they used for real world demonstrations. The drone has four different control inputs: roll, pitch, yaw and heave. According to the experiments they have completed, they saw that wind disturbances mostly affect the angles pitch and roll thus, these were the only two angles considered for the vertical landing.

In terms of the reward function variables, the authors take a different direction and use action values instead. Again, the reward is designed depending on the region the drone is in 5.28.

$$\text{Reward} = \begin{cases} -\frac{1}{20}\,|a_{\text{diff}}| - \frac{1}{10}|a| & \text{if } |d| \leq 0.1 \\ & \text{(Region-1)} \\ -2|d| - \frac{1}{20}\,|a_{\text{diff}}| - \frac{1}{10}|a| & \text{if } 0.1 \leq |d| \leq 0.4 \\ & \text{(Region-2)} \\ -1 & \text{if } 0.4 \leq |d| \leq 2 \\ -\,(T_{\max} - T_{\text{inside}}) & \text{(Region-3)} \\ - & \text{if } |d| > 2 \end{cases} \tag{5.28}$$

As it is mentioned previously, reality gap problem is a serious problem that often occurs within RL frameworks. To overcome this, Saj and et all. applied the domain randomization approach [47, 61, 43] by varying the parameters of the simulation environment. In their case, they used the Gazebo simulator to create several wind conditions. Wind conditions were chosen as constant, sudden and, sinusoidal wind which they have been applied for time periods of 10 to 50 seconds. It was observed that the headwind has an effect on the forward drift and the crosswind affects the sideward drift hence, they have applied them for roll and pitch controller training, respectively.

The tests were made for hovering and landing conditions and compared the performance of the RL with PID controller as similarly done in. They observed that while RL has the ability to make a landing PID controller was not able to make the deviations in the forward and sideward distance to go zero. The results of both algorithms showing their reaction to the given disturbances are shown in Figure 5.15

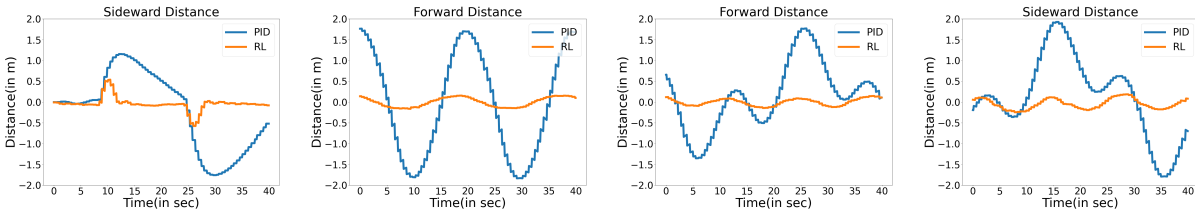


Figure 5.15: Deviation values for RL and PID [49]

5.6. Conclusion and Discussion

In this chapter, an overview of machine learning techniques for use in optimal control and guidance problems has been given. The chapter started with a very general description of the definition of an optimal control problem. The underlying optimal control theory helps to formulate the structure of the problem at hand and understand where the actual optimality element comes from. After this description, existing techniques to solve a continuous optimal control problem are provided with a special focus on numerical methods. This classification is important because the majority of studies and engineering applications have been using numerical methods to come up with solutions to optimal control problem.

Furthermore, the literature on using machine learning techniques for landing problems first shows the use of supervised learning with different numerical methods, thus providing a description of them. The initial studies addressing optimal control problems introduced what is known as GNC networks, where typical guidance and control tasks are solved using learning based approaches. This includes first computing the optimal inputs with a numerical technique and then trying to learn the input-output relationship in a supervised manner. This combined approach allows for fully exploiting the capabilities of neural networks and numerical methods together. However, the progress in obtaining optimal control inputs is restricted by the numerical method used, as each one of them has specific characteristics that may lead to unstable behaviour.

Thus, later studies have focused on using reinforcement learning in continuous optimal control and guidance problems. Unlike other techniques, reinforcement learning has made progress in filling the gaps in optimal control by being highly adaptable and flexible to variations in system dynamics and environment, while at the same time requiring no prior model. These attributes of reinforcement learning make it highly suitable for landing problems, as the challenges associated with them can be compensated for with what reinforcement learning offers. In the context of autonomous landing, state-of-the-art deep learning techniques are discussed in terms of target motion characteristics, sensor integration methods, and vehicle capabilities. While the literature highlights the efficiency and generalizability of reinforcement learning, further validation was also provided with benchmark controllers.

The progress in this chapter has been made to how reinforcement learning stands out compared to the other techniques for autonomous landing problems.

Table 5.3: Machine Learning Techniques for UAV Guidance and Control

Technique	Applications	Strengths and Limitations
Supervised Learning	Used for classification and regression tasks in UAV operations.	Effective for labeled data; limited in dynamic and complex environments.
Unsupervised Learning	Finds patterns in unlabeled data; used for clustering and anomaly detection.	No prior human intervention needed; less effective for interactive environments.
Reinforcement Learning	Learns optimal policies through trial and error; highly adaptable.	Effective for dynamic environments; requires extensive training and computational resources.

Table 5.4: Reinforcement Learning Algorithms for UAV Landing

Algorithm	Description	Applications and Results
Deep Q-Network (DQN)	Uses neural networks to approximate state-value function; combines experience replay for stability.	Applied to static pad landing; handles low-resolution images; shows successful implementation in simplified environments.
Proximal Policy Optimization (PPO)	Employs a surrogate objective function; uses trust regions to ensure stable learning.	Used for continuous control tasks; effective in complex and dynamic environments; stable and reliable performance.
Deep Deterministic Policy Gradients (DDPG)	Combines policy gradient with Q-learning; suitable for continuous action spaces.	Applied to landing on moving platforms; shows improved performance in handling unpredictable dynamics.

6

Variable Skew Quad Plane

The diverse requirements of flight operations have driven the evolution of UAVs in terms of their shape, size, and configuration. Typically, UAVs can be broadly classified into fixed-wing and rotary-wing categories, each serving offers unique advantages and faces limitations.

In this work, the vehicle in consideration is the Variable Skew Quad Plane, a hybrid drone that combines the capabilities of both fixed-wing and rotary-wing UAVs, offering a flexible and adaptable structure suitable for different flight modes. This section introduces the modelling details, the role of actuators, and the guidance and control scheme of the VSQP. By understanding this one-loop adaptive controller, questions regarding the learning-based problem formulation **RQ-PF** have been addressed.

6.1. Background on hybrid UAVs

Unmanned aerial vehicles (UAVs) have become increasingly popular for their ease of deployment, low maintenance costs and high mobility. Today, UAVs are used in a wide range of applications from military operations to delivery of goods. The most common two types of UAV platforms that are currently in use are known as fixed wing UAVs and rotorcraft UAVs. Both designs have their specific capabilities and limitations finding use in operations that align with their specifications. The fixed-wing UAV is faster, can carry heavier payloads, and has a longer flight range and endurance. However, it requires special equipment for takeoff and landing and is not suitable for missions that require a low flight speed or confined environment. The rotorcraft UAV, on the other hand, is more flexible when it comes to takeoff and landing requirements and can hover in place. However, it has limitations on speed and endurance, which restricts its capabilities for wide range coverage or long endurance missions [48]. The aim to combine the capabilities of both designs for a much broader range of applications under additional requirements such as good wing rejection capacities and better performance in gusty environments has led to the innovation of hybrid UAVs. Hybrid UAVs make use of the VTOL capabilities of multicopter in vertical phases but are also be able to have the efficiency of a wing in cruise. These are achieved either with a change in attitude and control system or a modification to the standard geometry of the drone. With hybrid UAVs, easier and more efficient operation in different flight phases becomes possible.

6.2. Variable Skew Quad Plane (VSQP)

The Variable Skew Quad plane is a hybrid UAV designed by the Micro Air Vehicle Laboratory (MAVLab) at TU Delft. The VSQP has two modes of operation - hover and cruise. In hover mode, the drone operates as a quadrotor and is controlled using differential thrust attitude. In the cruise mode, the drone operates as a quad-plane and achieves forward speed with a push propeller located at the tail. The VSQP does not have a fixed wing configuration but instead utilizes the rotations concept seen in the Oblique Flying Wing (OFW) plane [27]. To deploy the wings, a central rotating pivot is used while the lateral rotors are folded into the fuselage. This design offers better gust rejection in hover mode and lower drag during cruise mode, while also providing improved packability and ease of operation.

6.2.1. Model Configuration

In this section, configuration of the VSQP with different skew angles are given along with the actuator placements on the body. Figure 6.1 shows the VSQP in hover mode on the left with a skew angle equal to 0 degrees. In this mode, wings align with the body making it more stable in case of external disturbances and allowing it to accomplish a safer landing. Two of the motors are mounted longitudinally on the fuselage, and two are on the arms extending outwards, placed perpendicular to the wing. The configuration for the skew angle set to 90 degrees is shown on the right in Figure 6.1, corresponding to forward flight mode. In opposite to the hover mode, the motors that were previously extending outwards are folded into the fuselage. This is done for the purposes of reducing aerodynamic drag, making the flight in forward mode more efficient.

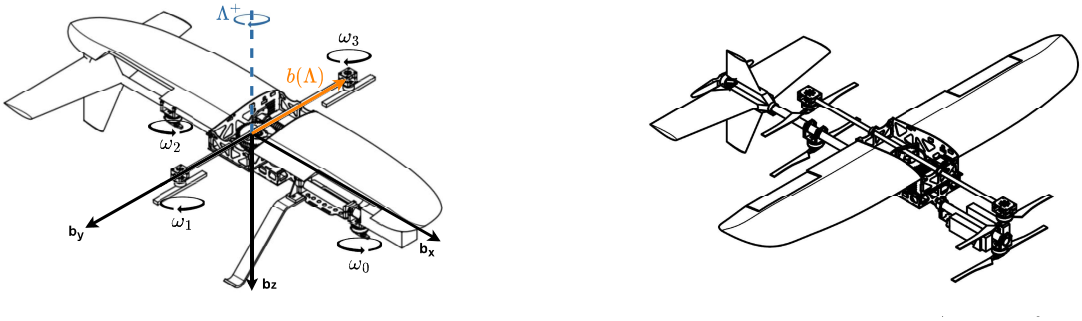


Figure 6.1: Configuration of the VSQP with different skew angles

For the guidance and stabilization of the VSQP, a total of 10 actuators are employed in the model. The placement of these actuators is shown in Figure 6.2 and the table 6.1 accompanying this figure listed the actuators with indications for the ones capable of rotations.

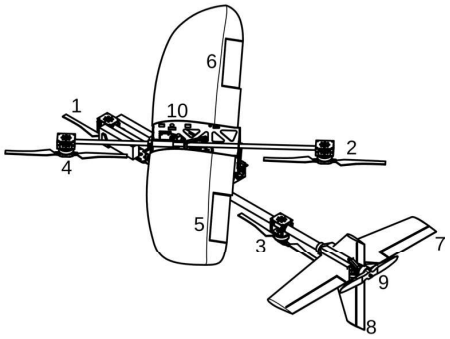


Figure 6.2: Actuators of the VSQP

Table 6.1: Actuators and Their Functions

Actuator number	Actuator name	Rotating
1	Front Motor	
2	Right Motor	✓
3	Back Motor	
4	Left Motor	✓
5	Left Aileron	✓
6	Right Aileron	✓
7	Elevator	
8	Rudder	
9	Push Motor	
10	Wing Rotation Servo	

6.3. Guidance and Control Scheme of the VSQP

This section describes the guidance and control architecture of the VSQP by providing the formulation of the ANDI controller with the details of the control allocation algorithm for the Variable Skew Quad Plane.

6.3.1. INDI

The control scheme of the VSQP applies the main principles of Incremental Nonlinear Dynamic Inversion (INDI) for its stabilization and guidance due to the robustness and low model dependency of INDI compared to other conventional control methods. INDI is an incremental control approach that is derived from Nonlinear Dynamic Inversion (NDI) which inverts the nonlinear dynamics to obtain a linear representation of the system. However, NDI highly relies on the model, and its accuracy is affected by how well in detail the model is designed. Hence, to compensate for the inaccuracies in the model and consider the unmodeled disturbances, INDI as a sensor-based approach is commonly used for controlling UAVs.

Formulation of INDI

Consider the following general nonlinear system:

$$\dot{x} = f(x, u) \tag{6.1}$$

Using Taylor's expansion, the system can be linearized at the current time step indicated by the subscript '0' as it is shown in Eq. 6.2 and Eq. 6.3.

$$\dot{x} \simeq f(x_0, u_0) + \left.\frac{\partial f(x,u)}{\partial x}\right|_{x=x_0, u=u_0} (x - x_0) + \left.\frac{\partial f(x,u)}{\partial u}\right|_{x=x_0, u=u_0} (u - u_0) \tag{6.2}$$

$$\dot{x} \simeq \dot{x}_0 + F(x_0, u_0)(x - x_0) + G(x_0, u_0)(u - u_0) \tag{6.3}$$

where the variables $x_0, \dot{x}_0$ and u_0 are obtained by the available measurements coming from the sensors, the parameter F represents the dynamics of the system and the parameter G is defined as the control effectiveness matrix.

Assuming that the time separation principle holds for the system indicating that the actuators are fast enough such that the system dynamics $F(x_0, u_0)(x - x_0)$ can easily be ignored, the following relationship in Eq. 6.4 could be written:

$$\dot{x} \simeq \dot{x}_0 + G(x_0, u_0)(u - u_0) \tag{6.4}$$

To calculate the control inputs, a virtual control input v should be defined and information regarding the control effectiveness matrix should be available. Replacing the virtual control input with the derivative of the state $\dot{x}$ and inverting the control effectiveness matrix G, the following incremental control input is found.

$$u = G^{-1}(v - \dot{x}_0) + u_0 \tag{6.5}$$

6.3.2. Adaptive INDI (ANDI)

Although INDI is highly efficient, it has certain limitations especially in dealing with delays within the system that are often caused by measurements and actuators and adapting to varying control effectieveness matrix, especially changes in parameters such as moment of inertia of the vehicle, the type of motors and propellers etc. To account for these parameters and overcome the issues associated with INDI, adaptive scheme for estimating the control effectiveness online based on a Least Mean Squares (LMS) is proposed by Ewoud et all. [55].

In the LMS formulation, the difference between the expected acceleration and measured acceleration is first calculated and based on the calculated error the control effectiveness is incremented accordingly. An example implementation of the LMS is shown in Eq. 6.6 and Eq. 6.7.

$$\boldsymbol{G}(k) = G(k-1) - \boldsymbol{\mu_2}\left(\boldsymbol{G}(k-1)\begin{bmatrix} \Delta\omega_f \\ \Delta\dot{\omega}_f \end{bmatrix} - \Delta\dot{\Omega}_f\right)\begin{bmatrix} \Delta\omega_f \\ \Delta\dot{\omega}_f \end{bmatrix}^T \mu_1 \tag{6.6}$$

$$G = \begin{bmatrix} G_1 & G_2 \end{bmatrix} \tag{6.7}$$

Here, $\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$ are the adaptation constants that determine the stability and the rate of adaptation of the system. Faster convergence could be obtained by making these parameters larger until the theoretical limit [25].

As it is clear from the implementation of the LMS, control effectiveness will not change if the inputs are constant and in case there is more excitation of the system, faster adaptation will be achieved.

Results in [55] showed that adaptive INDI performs better by reacting faster to the uncertainties in the model and having a high disturbance rejection performance.

Control Allocation

Control allocation is defined as the distribution of the control effort over actuators which are higher in number that than the number of controlled variable. This is a critical problem for the VSQP since it is an overactuated system that necessitates the use of control allocation for optimal use of its resources (control surfaces). Smeur and Höppener [54] underlines the importance of control allocation for systems that are under the restriction of control saturation as these saturations may often lead to undesired behaviour.

Previously, INDI control law is given as follows:

$$u = G^{-1}\left(v - \dot{x}_0\right) + u_0 \tag{6.8}$$

.

While this control law is highly effective, it does not guarentee the desired behaviour in case there is saturation exists on the control inputs. This may lead to situations such as insufficient moment generation which is of a high importance. To overcome this and similar issues that arise within simple INDI control, a control allocation based on Weighted Least Squares formulation from Harkegard [24] is adapted by Ewoud et all. Two separate cost functions for the purpose of solving two separate objectives which are minimizing the error in the control objective and penalizing the inputs of the actuators are given in Eq. 6.9.

$$\begin{aligned} C(u) &= \|W_u\left(u - u_d\right)\|^2 + \gamma\,\|W_v(Gu - v)\|^2 \\ &= \left\| \begin{pmatrix} \gamma^{\frac{1}{2}} W_v G \\ W_u \end{pmatrix} u - \begin{pmatrix} \gamma^{\frac{1}{2}} W_v v \\ W_u u_d \end{pmatrix} \right\|^2, \end{aligned} \tag{6.9}$$

where W_v is the diagonal weighting matrix for the control objective, and W_u is the diagonal weighting matrix for the inputs. The distinction between these two objective functions is made via the scale factor $\gamma >> 1$. For conveience, the given objective objective function is reformulated as a quadratic programming problem as follows:

$$A = \begin{bmatrix} \gamma^{\frac{1}{2}} W_v\left(G_1 + G_2\right) \\ W_u \end{bmatrix} \text{ and } b = \begin{bmatrix} \gamma^{\frac{1}{2}} W_v v \\ W_u u_d \end{bmatrix}. \tag{6.10}$$

Now that the problem is formulated in the form of a quadratic function the solution could be obtained through the active set method for which the detailed explanation could be found in [24].

6.4. One-loop ANDI for the VSQP

Previously, INDI-based approach has been proposed for controlling the VSQP [12]. Models for the actuator effectiveness and lift are developed as a function of the skew angle. An automatic controller that can adjust the skew angle is derived and the whole system was tested and verified through wind tunnel tests completed at the Open Jet Facility (OJF) at TU Delft.

The current model of the VSQP incorporates a structure that is one step advanced than the INDI based system. The model combines the adaptive INDI 6.3.2 with the WLS based control allocation 6.3.2 and apply them as a one-loop structure instead of repeating the inner optimization routing in the outer loop as previously done in [31]. By doing so, it is expected to have a robust controller that can distribute the control load optimally throughout the control surfaces of the VSQP. The diagram showing the main elements of the guidance and control loop is given in Figure 6.3 below.

The WLS block takes the virtual control parameters of which values are calculated in the error controllers for the position and attitude with the gains set according to the desired response of the drone. Error controllers ensure that the reference models are being tracked well. In addition to the error controllers, reference models are used to shape the given desired set point into reference signals that are compatible with the characteristics of the controller.

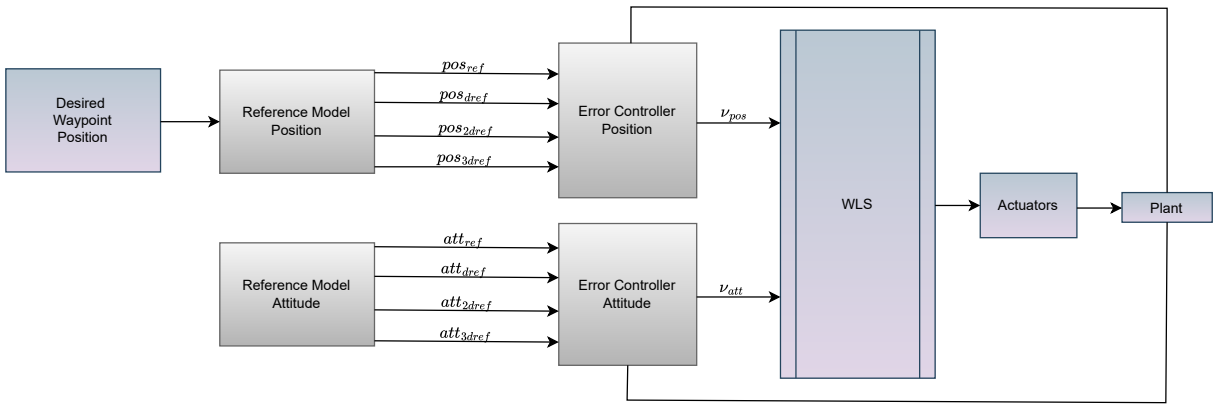


Figure 6.3: Guidance and Control Scheme of the VSQP

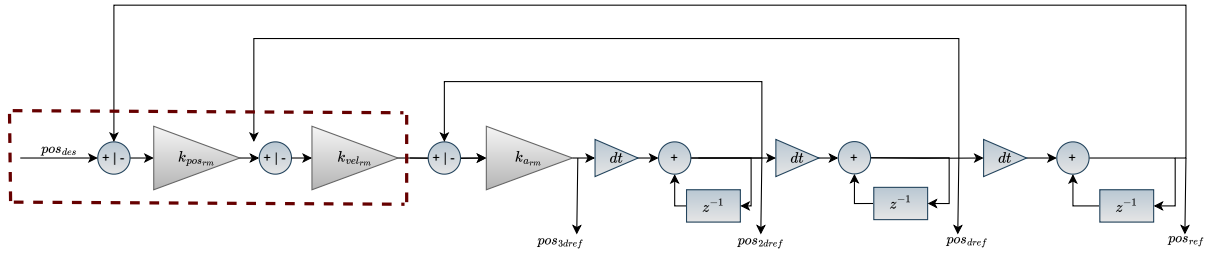


Figure 6.4: Reference model for position

6.5. Conclusion and Discussion

In this chapter, the details regarding the Variable Skew Quad Plane including the model configuration, guidance and control scheme, and the one loop ANDI structure has been given. Understanding the model configuration is crucial for identifying the geometry changes during flight. The VSQP lands with its skew angle set to zero, operating in a complete hover mode. This is important for identifying the active actuators during flight, which in turn affects the modeling procedure.

The VSQP incorporates a one-loop ANDI controller, which consists of several sub-elements such as reference models, error controllers, an optimization routine block and the drone itself. The main motivation for ANDI is to address system delays and adapt to varying effectiveness matrix parameters. With ANDI, a more compact and robust controller is achieved for the VSQP. While the controller is highly effective in control allocation and robust to external disturbances, the guidance part is open to further advancements since it heavily relies on the existing fixed-structure of the reference models and error controllers. Moreover, the input to the controller is entirely determined based on the given position waypoint. In the context of landing on moving platforms, this would push the drone to follow a pattern similar to that of the platform, which may not always be the most desirable approach. Understanding the existing guidance and control scheme of the VSQP is important to identify areas for further improvement.

Based on given the information provided in the previous chapters, the dynamic nature of landing on a moving platform requires flexibility in the action space and the ability to make smarter decisions. As discussed in Chapter 5, this can be achieved through the use of learning based approaches. Reinforcement learning, in this context, becomes an effective alternative due to its ability to changing environmental

conditions and make real-time adjustments to improve landing accuracy and safety.

For the given model, reinforcement learning based guidance system could offer the followings:

- **Adaptability:** Unlike the fixed gain structure of reference models and error controllers, reinforcement learning learns from the response of the whole system, thus dynamically adjusting itself to environmental changes.
- **Smartness:** During the training process, neural networks are exposed to thousands of scenarios with varying parameters. The reinforcement learning algorithm can optimize the entire landing process, leading to more efficient landings.
- **Robustness:** A combined approach utilizing the reinforcement learning framework with the inner controller still makes use of the controller's capabilities while reinforcement learning in guidance also adapts based on how the controller reacts to given inputs. Although the system's optimality level is currently restricted by the controller, the robustness property is preserved.

7

Preliminary Analysis

The preceding Chapters 4 and 5 have discussed the autonomous landing problem for UAVs in the context of its sub-phases and introduced the applications of machine learning for optimal control problems, with a focus on reinforcement learning. The literature review has shown that reinforcement learning has a great potential to replace the existing guidance model of the VSQP with a more advanced architecture that is capable of providing further flexibility and robustness.

The current guidance and control structure of the VSQP, while fully implemented in Simulink, is not feasible to use for training purposes. As given in Chapter 6, the control structure incorporates a Weighted Least Squares Algorithm within the ANDI controller. This optimally distributes the control load among the actuators of the VSQP and stabilize the drone in all flight conditions. While the model is already highly detailed, this separate optimization routine running inside the loop adds another layer of complexity, necessitating a simpler model.

To incorporate the reinforcement learning framework in the loop, we must first identify the system's inputs and outputs. Given the general guidance and control scheme of the VSQP and specifics of the reference model in the previous section, both the response of the VSQP and the ANDI controller will be simplified as a first step in the preliminary analysis. Consequently, the states of the VSQP will be fed back to the reinforcement learning framework and the computed optimal control inputs are going to be the corresponding acceleration inputs. The overall diagram for the black box model incorporated system is shown in Figure 7.1.

Having obtained the associated models for the drone and the controller, the next step is fully implement these models in Python and compare the response of the system with the output from the Simulink. This step is important since there is possibility of encountering numerical issues while solving the related differential equations.

The modelling of the VSQP only makes up to half of the framework since the landing platform should also be represented with its corresponding dynamics. While the literature offers various approaches for ship modelling, ranging from high-fidelity models to simple sinusoidal signals, and this study opts for the latter. Sinusoidal signals, while not an exact representation, capture the general characteristics of real ship motion, making the problem applicable to a wide variety of cases that may involve different models for the ship. This approach also enhances the flexibility and adaptability of the proposed reinforcement learning framework, which is crucial for its generalizability.

Note that, as part of the preliminary analysis, the reinforcement learning-based framework was initially conducted for the Parrot Bebop 1 with a stationary platform consideration. However, these details are not included in this chapter. For a better flow in the report, the details and results of the simulations are provided in Appendix A.

7.1. Drone Model

The model details of the VSQP with its guidance and control structure is given in Chapter 6. As is mentioned in the section introduction, certain level of simplification is required to model the input-output relationships of the black box model and ANDI controller. The next section will describe the main steps into incorporating the black box model in the loop.

7.1.1. Incorporating the black box model in the loop

The simplified diagram for the reinforcement learning framework is shown in Figure 7.1. Here box number 1 denotes the black box model and contains the deep neural network structure whereas box number 2 illustrates the response of the vehicle to given control inputs by the controller. The black box model takes the states of the VSQP as inputs and generates the corresponding optimal acceleration inputs which are denoted as a_{des}. The response of the vehicle, symbolized as a_{real} corresponds to the actual accelerations resulting from the applied control inputs. To effectively map the given commands to the actual accelerations for the training, a simplified model is obtained.

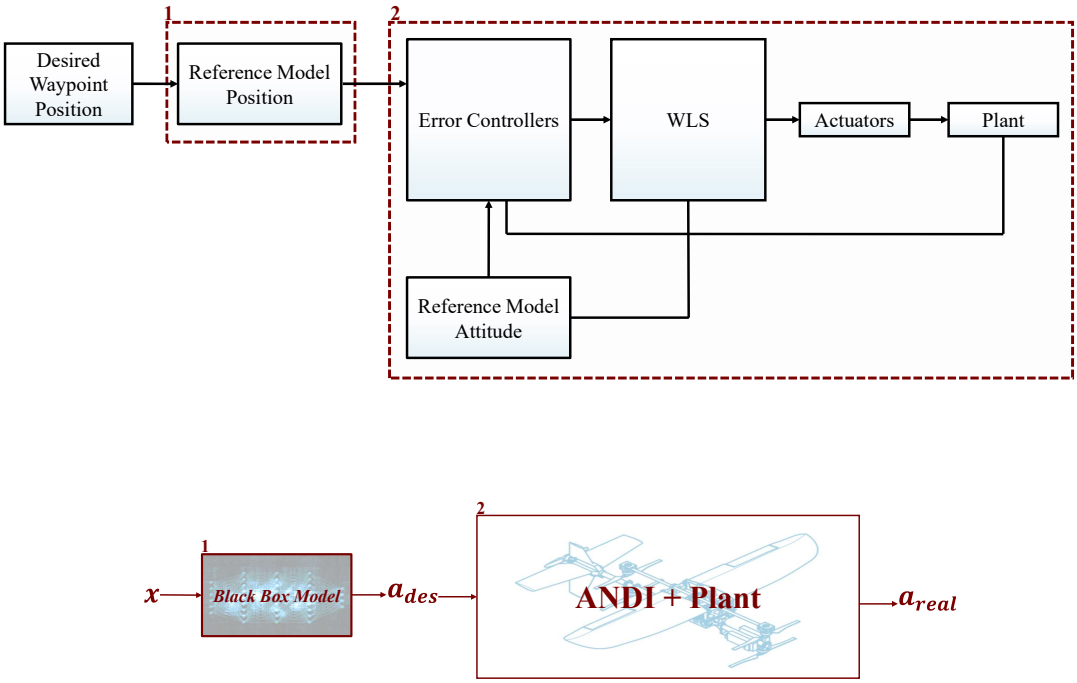


Figure 7.1: Incorporating the black box model in the loop

7.1.2. Transfer function for the acceleration response

To obtain the transfer function from a_{des} to a_{real}, a representation of the reference model along with the controller input-output relationship is required. In a previous study [15] where reinforcement learning is used in combination with an Incremental Nonlinear Dynamic Inversion (INDI) based controller, a first-order transfer function was used to model the controller's response to given attitude control signals. However, this approach is not directly applicable to the VSQP as it is an overactuated system that incorporates a separate optimization routine for control allocation. This routine complicates the overall framework as all actuators have an effect on the acceleration response of the VSQP.

The Weighted Least Squares algorithm based control allocation is explained in detailed in Chapter 6. The main purpose of the WLS is to distribute the overall control load among the actuators while being objective to the equations Eq.6.9.

In contrast to the conventional approach, the WLS also uses the Euler angles ϕ and θ as its virtual actuators. This approach allows a greater control over the response of the drone as the "optimal" reference values continuously fed back to the attitude reference model at each time step. However, while these angles within the loop get rid of the necessity to use a cascaded structure but instead an one loop one, they are also the slowest actuators among the others. This implies that the response of the controller is inherently limited by the dynamics of these angles. Therefore, only attitude dynamics are gonna be used in the simplification of the controller's response.

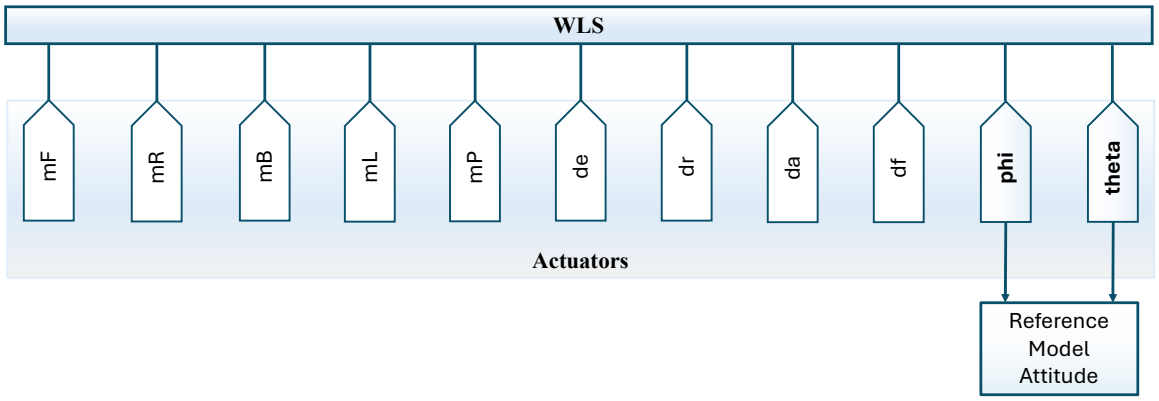


Figure 7.2: Actuators used in the controller

The actuators of the VSQP including the angles ϕ and θ are shown in Figure 7.2 and the description of the actuators are indicated in Table 7.1.

Table 7.1: The VSQP Actuator Descriptions

Actuator	Description
mF	Front Motor
mR	Right Motor
mB	Back Motor
mL	Left Motor
mP	Pusher
de	Elevator Deflection
dr	Rudder Deflection
da	Aileron Deflection
df	Flap Deflection
phi	Roll Angle
theta	Pitch Angle

Table 7.2: Coefficients of the reference models and the error controllers

Actuator	Description	Value
$k_{pos_{ec}}$	Position Error Controller Gain	2.8
$k_{vel_{ec}}$	Velocity Error Controller Gain	5.9
$k_{acc_{ec}}$	Acceleration Error Controller Gain	4.23
$k_{pos_{rm}}$	Position Reference Model Gain	0.2
$k_{vel_{rm}}$	Velocity Reference Model Gain	0.6
$k_{vel_{rm}}$	Acceleration Reference Model Gain	1.8
$k_{a_{rm}}$	Attitude Reference Model Gain	2.0
$k_{r_{rm}}$	Angular Rates Reference Model Gain	6.1
$k_{rdot_{rm}}$	Angular Accelerations Reference Model Gain	18.4
$k_{a_{ec}}$	Attitude Error Controller Gain	2.0
$k_{r_{ec}}$	Angular Rates Error Controller Gain	6.1
$k_{rdot_{ec}}$	Angular Accelerations Error Controller Gain	18.4

With the attitude reference model, transfer function from desired to real dynamics are represented in Eq. 7.3, where parameters p_1, p_2 and p_3 are all determined as 6.14. This reference model is obtained

under the assumption of first order dynamics actuators, correct modelling of actuator dynamics and effectiveness, non-saturated actuators and adaptive INDI controller.

$$H_{3rd}(s) = \frac{att(s)}{att_{des}(s)} = \frac{p_1}{s+p_1}\frac{p_2}{s+p_2}\frac{p_3}{s+p_3} = \frac{p_1p_2p_3}{s^3 + (p_1+p_2+p_3)\,s^2 + (p_1p_2+p_1p_3+p_2p_3)\,s + p_1p_2p_3} \tag{7.1}$$

The simplified control diagram that includes the attitude dynamics for the actuators is provided in Figure 7.3. Here $pos_{ref}, pos_{dref}, pos_{2dref}, pos_{3dref}$ represent the position, velocity, acceleration and jerk reference signals, $k_{pos_{ec}}$ is the corresponding error controller gain, respectively. In order to use the attitude dynamics in the control law, the corresponding angular frequency ω_T is calculated as follows.

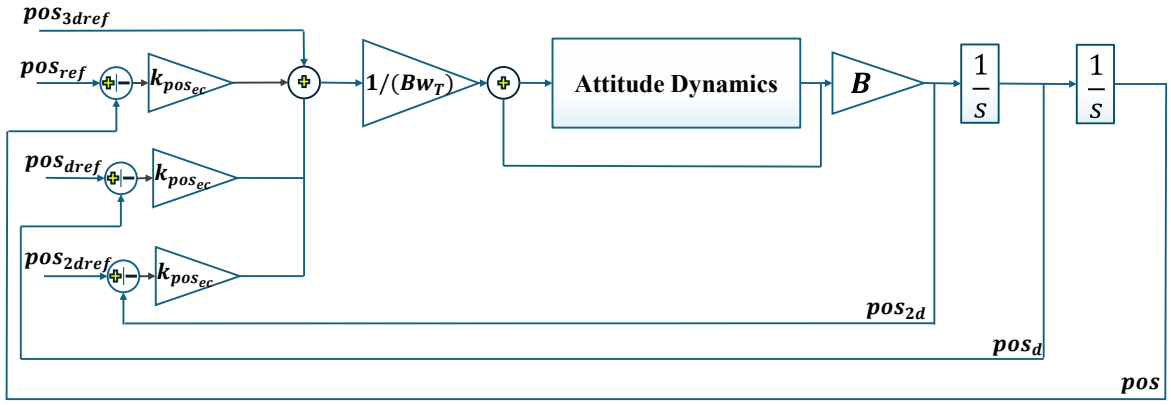


Figure 7.3: Simplified control diagram with attitude dynamics

Using Taylor's series approximation for small s:

$$e^{-\tau s} \approx 1 - \tau s = \frac{1}{e^{\tau s}} = \frac{1}{1+e^{\tau s}} \tag{7.2}$$

The attitude transfer function could be written as in Eq. 7.3 with the information that all coefficients are equal to each other.

$$H_{3rd}(s) = e^{\frac{-1}{p_1}s} e^{\frac{-1}{p_1}s} e^{\frac{-1}{p_1}s} = e^{\frac{-3}{p_1}s} = \frac{1}{1+\frac{3}{p_1}s} \tag{7.3}$$

Considering the general form of a first order system in Eq. 7.4

$$G(s) = \frac{b}{s+a} \rightarrow \frac{K}{\tau s+1} \tag{7.4}$$

where the coefficients a and b are given as

$$a = \frac{1}{\tau} \quad b = \frac{K}{\tau}. \tag{7.5}$$

Replacing p_1 with p, The cut-off frequency for the third order system is found as:

$$\omega_T = \frac{p_1}{3} = \frac{6.14}{3} = 2.14Hz \tag{7.6}$$

This value corresponds to the in the given simplified control diagram Figure 7.3 and allows attitude dynamics to be used in the control law.

With coefficients of the error controllers and the reference models given in Table 7.2, the transfer function for the VSQP is written in Eq. 7.7.

This transfer function maps the desired acceleration to the actual values and provide a way to represent the response of the controller and the drone altogether. The intermediate steps for obtaining the given transfer function is given in Appendix B.

$$acc_{tf} = \frac{206s^3 + 861.1s^2 + 287s + 95.68}{s^6 + 20.22s^5 + 143.3s^4 + 682s^3 + 1021s^2 + 340.2s + 95.68} \tag{7.7}$$

The step response and bode diagram of the transfer function are given in Figure 7.4.

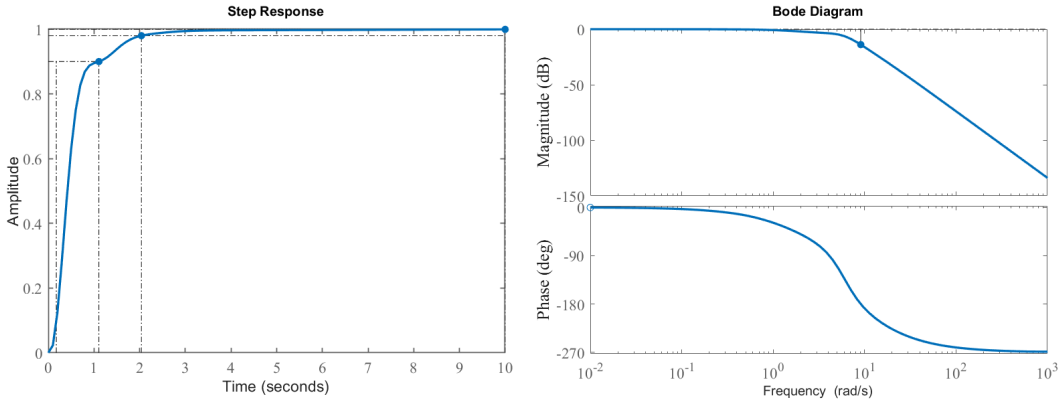


Figure 7.4: Step Response and Bode diagrams for the acceleration transfer function

Upon applying a step input, the system reacts quickly in the beginning of the step response graph. This is due to the higher-order terms in the numerator of the transfer function. The response then continuous to rise to the final value of 1 without having any overshoot but a slight change in the response characteristics. Overall, this indicates that the system is well-damped, meaning that is does not oscillate or have an exceeding over the top before settling down.

The initial set of simulations completed with the sixth-order transfer function led to some numerical issues in the beginning of the training which lead to a certain level simplification in the transfer function. Through *pole-zero simplification* method which is shown in Figure 7.5 , an equivalent fourth-order transfer function obtained as given in Eq 7.8.

$$acc_{4tf} = \frac{-9.477e-07s^3 + 0.0008596s^2 + 203.6s + 793}{s^4 + 19.89s^3 + 139.5s^2 + 633s + 793.2} \tag{7.8}$$

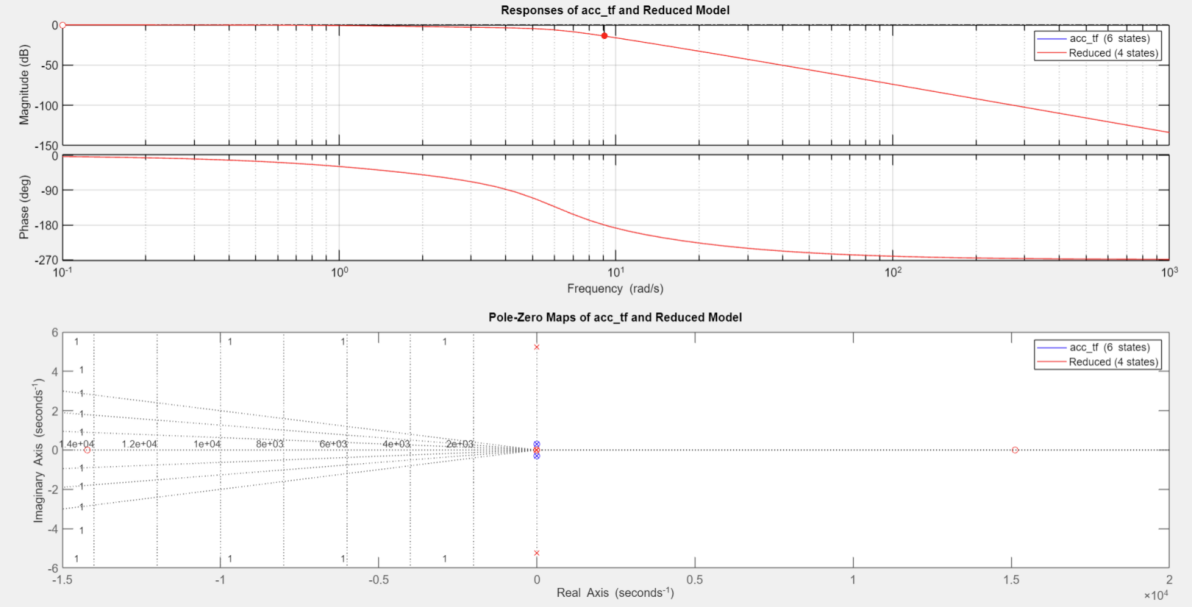


Figure 7.5: Step Response and Bode diagrams for the acceleration transfer function

The comparison between the fourth and sixth order transfer functions in step response is provided in Figure 7.6.

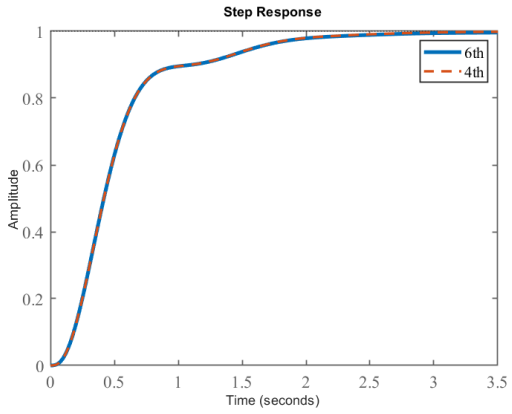


Figure 7.6: Step Response comparison of the 4th order with the 6th order transfer function

7.2. Ship Model

A ship model is required to represent the characteristics of a moving platform. To have a better flexibility and adaptability in the representation of the platform motion, sinusoidal signals that are governed by specific frequencies, amplitudes, and initial conditions are used to in the simulations.

The following state elements in Eq. 7.9 are modelled with the corresponding sinusoidal signals.

$$
\begin{aligned}
\mathrm{p}_i(t) &= \sin(\omega_i t) \cdot A_{\mathrm{pos},i} + P_{\mathrm{start},i} \\
\mathrm{v}_i(t) &= \cos(\omega_i t) \cdot A_{\mathrm{pos},i} \cdot \omega_i \\
\phi(t) &= \sin(\omega_{\mathrm{roll}}\, t) \cdot \pi / A_{\mathrm{roll}} \\
\theta(t) &= \sin(\omega_{\mathrm{pitch}}\, t) \cdot \pi / A_{\mathrm{pitch}} \\
\psi(t) &= \sin(\omega_{\mathrm{yaw}}\, t) \cdot \pi / A_{\mathrm{yaw}}
\end{aligned}
\tag{7.9}
$$

The given equations describe the dynamic behaviour of an object's position, velocity and orientation in terms of time. Here, ω_i is the angular frequency of oscillation, $A_{pos,i}$ is the amplitude of this oscillation, and $P_{start,i}$ represents the initial position. This equation indicates that the position oscillates sinusoidally over time around an initial starting point.

For the velocity, the equation $\cos(\omega_i t) \cdot A_{\text{pos},i} \cdot \omega_i$ indicates the velocity oscillates over time in a cosine pattern with its phase shifted by $\pi/2$ from the wave of the position. And the amplitude of the velocity is further scaled by the term ω_i.

The attitude of the ship is characterized by its roll, pitch and yaw angles which are also represented with a set of sinusoidal signals. For the Euler angles, A and ω represents the angular frequencies and amplitude factors, respectively. These equations also indicate that the attitude of the drone may also oscillate sinusoidally over time, each with its respective frequency and amplitude to describe the rotational motion of the platform.

These parameters allow to select a range of frequencies and amplitudes to obtain a wide motion range for the platform. While this is extremely beneficial during training which is exposing the network to different kind of signals, the real ship motion incorporates characteristics that are more complex than the regular sinusoidal signals. Therefore, a further modelling on the noise characteristics of the platform is required.

Noise Characteristics

To resemble the real-ship motion, Gaussian noise and random walk were added to the modelled signals with the addition of mixed frequency component f_{mix} and low pass filter coefficients α. While the f_{mix} adds a variation over the base sinusoidal that breaks the predictable pattern of a signal, the random components introduced band limited noise to the data. A sample for the gaussian noise and the random walk variation is given in Figure 7.7.

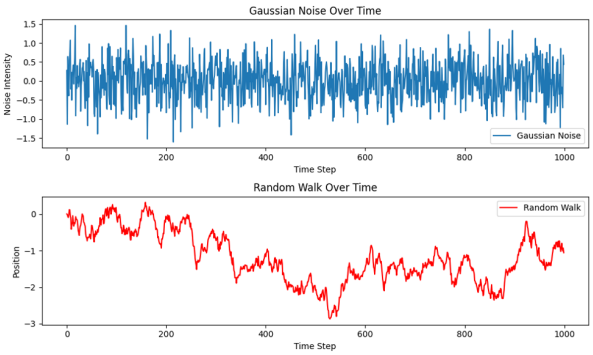


Figure 7.7: Gaussian Noise and Random Walk

7.3. Conclusion and Discussion

In this chapter, we provided a preliminary analysis for integrating a reinforcement learning framework into the Variable Skew Quad Plane (VSQP) control system. The current guidance and control structure of the VSQP, implemented in Simulink, incorporates a Weighted Least Squares (WLS) algorithm within an Adaptive Nonlinear Dynamic Inversion (ANDI) controller. While this structure optimally distributes control loads among the VSQP's actuators, it introduces complexity that makes it impractical for training reinforcement learning models. Therefore, a simplified model of the VSQP and its controller was developed for preliminary analysis.

To incorporate reinforcement learning, we identified the system's inputs and outputs, simplifying the ANDI controller and the VSQP model. This simplification involved representing the complex control dynamics with a transfer function, making the system feasible for training reinforcement learning models in Python. Initial comparisons between the simplified model and the detailed Simulink implementation were conducted to ensure accuracy and stability.

The ship model, representing the landing platform's dynamics, was also simplified using sinusoidal signals. While these signals do not fully capture the complexity of real ship motion, they provide a flexible and adaptable representation suitable for training the reinforcement learning framework. The

introduction of noise characteristics, including Gaussian noise and random walk variations, further enhanced the model's realism.

Overall, this preliminary analysis lays the foundation for integrating reinforcement learning into the VSQP control system. The next steps involve fully implementing these models in Python, conducting detailed simulations, and refining the reinforcement learning framework to achieve optimal performance in dynamic and uncertain environments. Through this approach, we aim to develop a robust and adaptable control system for the VSQP, capable of autonomous landing on moving platforms.

8

Conclusion

In this work, the main objective was to develop an optimal guidance policy for landing the Variable Skew Quad Plane by exploring learning-based approaches. Research questions were formulated to shape the structure of the thesis, including the literature review, methodology and analysis.

Research Questions (RQ-AL) - Autonomous landing for UAVs

RQ-AL 1. How is the autonomous landing procedure is defined in the literature?
RQ-AL 2. What are biggest challenges associated with autonomous landing?
RQ-AL 3. What type of methods are used for the landing problem and what conventional methods lack so that it creates a specific need for the learning based approaches?

The first set of questions were addressed in Chapter 4, focusing on understanding the autonomous landing problem through its sub-phases and specific challenges, particularly in ship landings. The literature review revealed that autonomous landing is significantly more complex than other flight phases, involving several sub-phases, each with specific requirements.The landing operation typically begins with a target detection algorithm to identify the platform and extract useful information for the guidance and control architecture. Common studies in the literature included vision-based systems incorporating cooperative, feature-based, and machine learning-based solutions. In some cases, vision-based systems were combined with GPS/INS systems for higher accuracy in state estimation. Although target detection and relative state estimation were not covered in this thesis, understanding these phases provided insights into the essential state information needed for a fully defined landing problem.

The challenges of autonomous landing were particularly highlighted in the context of maritime operations. Generally, these challenges arise from real-time massive information processing, limited onboard resources, and the need for higher maneuverability in the air. While ship landings share similar characteristics with other landing problems, the dynamic operational environment plays a critical role in distinguishing them. As discussed in the subsequent chapters, this specificity has strongly influenced the solution methods proposed for landing on moving platforms.

Studies have shown that classical guidance and control algorithms for tracking and landing tasks primarily include PID controllers, model predictive control, and robust flight techniques. While each method effectively handles the landing task in specific ways, they also have several drawbacks, making them challenging to apply to dynamic landing problems. These drawbacks include the fixed gain structure of PID controllers, modeling challenges, and the computational expense of more advanced approaches.

With these research questions, an introduction to the problem has been provided, and a motivation for more advanced learning-based approaches has been established.

Research Questions (RQ-ML) - Machine Learning for Optimal Control

RQ-ML 1. Which machine learning technique suits better for landing the VSQP on a moving platform?
RQ-ML 2. How is an objective function is structured for the landing problem?
RQ-ML 3 What metrics are used to asses the performance of the landing?

The second set of questions was addressed in Chapter 5, which thoroughly explained the machine learning techniques and their relation to optimal control and autonomous landing problems. These questions aimed to determine the best machine learning technique for this type of landing problem, focusing on the methodology part of the thesis. The methods investigated were primarily supervised and reinforcement learning, as these were the most commonly used in the literature. The progression moved from a general description of optimal control theory to supervised learning and finally to reinforcement learning.

It was observed that supervised learning approaches heavily depended on numerical methods, requiring a complete formulation of the optimal control problem. This reliance sometimes led to issues with instability and initial conditions. Additionally, the dynamic nature of the problem made it extremely challenging to develop a model that accurately represents the complex mixed dynamics in the final phase of landing. Consequently, studies have leaned towards using reinforcement learning due to its ability to adapt to varying and unknown dynamics and disturbances. Although there were not many studies available for this specific combination, they were helpful in understanding the design process of an objective function for the landing problem. Each study proposed a unique objective function, and it was observed that they followed a certain pattern where the specification of different elements within the reward was highly dependent on the height element. While this approach yielded successful results, it also restricted the problem and conflicted with the main idea of defining the reward function for what to solve not how to solve it. In this study, reward combinations contained similar parameters to those in the literature but did not strictly limit the drone's behaviour based on its altitude.

For the assessment of the landing performance, different performance metrics were used. The first metric checked was the success rate of the landing, essentially to see if the drone landed or not. Then, trajectory comparisons were made with a benchmark controller to indicate that the errors for the reinforcement learning-based framework were smaller in variations and eventually stabilized around zero. Some studies further classified the error element and analysed the performance in the horizontal and vertical planes separately. This classification was crucial to our study as it shaped the way we designed the reward function in terms of how we weighted different motion elements.

Research Questions - Learning based Problem Formulation for the VSQP

RQ-PF 1. How the black box model should be integrated into the loop?
RQ-PF 1.1. Considering the existing control architecture and characteristics of the given landing problem, what should be the inputs and outputs to the system?
RQ-PF 1.2. How the response of the controller will be modelled for an overactuated system such as the VSQP, especially for training purposes?
RQ-PF 2. How the landing platform will be modelled for the simulations?
RQ-PF 2.1. What states will be considered for the platform?
RQ-PF 2.2. How is the model for the target platform will be integrated with the rest of the system (drone)?

With the research questions RQ-PF, the purpose was to define the methodology and its specifications. The first question aimed to identify the inputs and outputs of the system and integrate this relationship with the rest of the control loop. At the beginning of the study, one of the main objectives was to ensure that the outputs from the black box model were the optimal acceleration inputs. This required obtaining the actual response of the vehicle along with the response of the controller. This was a highly challenging task due to the overactuated nature of the VSQP and the existing control allocation algorithm. It was essential to have the state parameters, particularly position, velocity, and acceleration, which all depended on the internal structure of the ANDI controller. The model of the VSQP was simplified

under the assumption that attitude dynamics act as the only actuators since they are the slowest among all, limiting the overall response of the vehicle. The obtained model was further simplified to avoid numerical instabilities during the training process.

The motion of the ship is characterized by sinusoidal signals governed by specific frequencies, amplitudes, and initial conditions. Using sinusoidal signals instead of one specific dynamic model helped to generalize the problem and added more value to the results in terms of generalizability..

Research Questions - Evaluation of the proposed framework

RQ-EV 1. How could the proposed solution (ex.reinforcement learning) be verified and validated?
RQ-EV 2. What type of performance metrics would be applicable to the designed framework?
RQ-EV 3. Can the proposed solution be compared to a benchmark controller and if so, what would be the choice for it?

The last set of questions addressed the verification and validation of the proposed approach. The verification of the reinforcement learning framework was conducted through statistical analysis of the training data set, while validation was completed using real ship data. These analyses demonstrated that the proposed approach could still achieve successful landings, even though the real data incorporated dynamics not present in the training set. To assess performance, three metrics were designed and used: touchdown velocity, deviation from the center, and flight duration. These parameters highlighted how different reward functions relate to each other and shaped the final, most effective version among them.

Finally, the performance of the reinforcement learning framework was compared with PID controllers for further validation. While PID controllers are simple and highly effective in a variety of scenarios, they often resulted in crashes rather than successful landings due to their direct approach. In contrast, the reinforcement learning-based framework was able to learn the pattern of the problem and act effectively by incorporating the nonlinear relationship between the drone and the platform. Landings with reinforcement learning showed lower touchdown velocities, often leaving a margin for error. In almost all cases, landings were completed within the desired duration, with the agent waiting for optimal conditions to perform the landing.

Future work for this topic may involve several prospects. Firstly, this thesis only addresses the tracking and landing part of the entire autonomous landing process. This means that target detection or obtaining relative state estimation are not considered. Integrating vision-based systems with GPS/INS, as suggested in the literature, would add complexity to the problem and may require a different approach to input modeling and training.

Furthermore, complete simulation and real data integration into the system would allow for real-life experiments and provide an opportunity to test the system's robustness under external disturbances such as wind gusts

While reinforcement learning was proposed as a solution to the problem, the focus of the thesis was not solely on this. It first addressed questions regarding incorporating a complex model into the structure, modeling ship wave motions with generalizable sinusoidal signals for different motion considerations using only one specific type of reinforcement learning algorithm. As the technique was effective from the beginning for both stationary and moving platforms, there was no need to test and compare state-of-the-art deep reinforcement learning algorithms, as this was not one of the study's initial objectives.

Modifications to the framework would not only involve changing the main algorithm but also the structure of the neural network. Real ship motion incorporates dynamics that could be mimicked with mixed frequency sinusoidal signals. Therefore, a recurrent neural network structure could identify the motion pattern and make predictions from that point. Predicting the states of the ship would allow future states to be incorporated into the model, giving reinforcement learning a chance to develop a smarter policy. Lastly, the proposed structure was validated with PID controllers due to their simple yet efficient structure. By setting aside complexity and computational expense considerations, the proposed approach could be further tested against more advanced controllers such as model predictive control.

This thesis approached the autonomous landing problem from a learning-based perspective and addressed the challenges associated with the dynamic nature of landing using reinforcement learning. With this work, reinforcement learning showed great promise as an optimal guidance strategy, achieving

safe and precise landings of the VSQP and contributing to extending the operational capability of UAVs for autonomous landing.

References

[1] David Abel. "A theory of abstraction in reinforcement learning". In: *arXiv preprint arXiv:2203.00397* (2022).

[2] Shadi Abujoub, Johanna McPhee, and Rishad A Irani. "Methodologies for landing autonomous aerial vehicles on maritime vessels". In: *Aerospace Science and Technology* 106 (2020), p. 106169.

[3] Kai Arulkumaran et al. "Deep reinforcement learning: A brief survey". In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 26–38.

[4] Andrew G Barto, Steven J Bradtke, and Satinder P Singh. "Learning to act using real-time dynamic programming". In: *Artificial intelligence* 72.1-2 (1995), pp. 81–138.

[5] Andrew G Barto, Richard S Sutton, and Charles W Anderson. "Neuronlike adaptive elements that can solve difficult learning control problems". In: *IEEE transactions on systems, man, and cybernetics* 5 (1983), pp. 834–846.

[6] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*. Vol. 4. Athena scientific, 2012.

[7] John T Betts. "Survey of numerical methods for trajectory optimization". In: *Journal of guidance, control, and dynamics* 21.2 (1998), pp. 193–207.

[8] Hans Georg Bock and Karl-Josef Plitt. "A multiple shooting algorithm for direct solution of optimal control problems". In: *IFAC Proceedings Volumes* 17.2 (1984), pp. 1603–1608.

[9] Paul T Boggs and Jon W Tolle. "Sequential quadratic programming". In: *Acta numerica* 4 (1995), pp. 1–51.

[10] Pádraig Cunningham, Matthieu Cord, and Sarah Jane Delany. "Supervised learning". In: *Machine learning techniques for multimedia: case studies on organization and retrieval*. Springer, 2008, pp. 21–49.

[11] Will Dabney et al. "Distributional reinforcement learning with quantile regression". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[12] T.M.L. De Ponti, E.J.J. Smeur, and B.W.D. Remes. "Incremental Nonlinear Dynamic Inversion controller for a Variable Skew Quad Plane". In: *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2023, pp. 241–248. DOI: 10.1109/ICUAS57906.2023.10156289.

[13] Sohrab Effati and Morteza Pakdaman. "Optimal control problem via neural networks". In: *Neural Computing and Applications* 23 (2013), pp. 2093–2100.

[14] Yi Feng et al. "Autonomous landing of a UAV on a moving platform using model predictive control". In: *Drones* 2.4 (2018), p. 34.

[15] Robin Ferede et al. "End-to-end neural network based optimal quadcopter control". In: *Robotics and Autonomous Systems* 172 (2024), p. 104588.

[16] Robin Ferede et al. "End-to-end Reinforcement Learning for Time-Optimal Quadcopter Flight". In: *arXiv preprint arXiv:2311.16948* (2023).

[17] Clara Lucía Galimberti et al. "Hamiltonian deep neural networks guaranteeing nonvanishing gradients by design". In: *IEEE Transactions on Automatic Control* 68.5 (2023), pp. 3155–3162.

[18] Alvika Gautam, PB Sujit, and Srikanth Saripalli. "A survey of autonomous landing techniques for UAVs". In: *2014 international conference on unmanned aircraft systems (ICUAS)*. IEEE. 2014, pp. 1210–1218.

[19] Zijian Ge et al. "Vision-based UAV landing with guaranteed reliability in adverse environment". In: *Electronics* 12.4 (2023), p. 967.

[20] Philip E Gill, Walter Murray, and Michael A Saunders. "SNOPT: An SQP algorithm for large-scale constrained optimization". In: *SIAM review* 47.1 (2005), pp. 99–131.

[21] Derek Greene, Pádraig Cunningham, and Rudolf Mayer. "Unsupervised learning and clustering". In: *Machine learning techniques for multimedia: Case studies on organization and retrieval* (2008), pp. 51–90.

[22] Philipp Grohs and Lukas Herrmann. "Deep neural network approximation for high-dimensional parabolic Hamilton-Jacobi-Bellman equations". In: *arXiv preprint arXiv:2103.05744* (2021).

[23] Kaiyang Guo et al. "Autonomous landing of a quadrotor on a moving platform via model predictive control". In: *Aerospace* 9.1 (2022), p. 34.

[24] Ola Harkegard. "Efficient active set algorithms for solving constrained least squares problems in aircraft control allocation". In: *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.* Vol. 2. IEEE. 2002, pp. 1295–1300.

[25] S Haykin and B Widrow. "Least-Mean-Square Adaptive Filters". In: (2003).

[26] Pingan He and Sarangapani Jagannathan. "Reinforcement learning neural-network-based controller for nonlinear discrete-time systems with input constraints". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 37.2 (2007), pp. 425–436.

[27] Michael Hirschberg, David Hart, and Thomas Beutner. "A summary of a half-century of oblique wing research". In: *45th AIAA Aerospace Sciences Meeting and Exhibit*. 2007, p. 150.

[28] Yanhua Huang. "Deep Q-Networks". In: *Deep Reinforcement Learning: Fundamentals, Research and Applications*. Ed. by Hao Dong, Zihan Ding, and Shanghang Zhang. Singapore: Springer Singapore, 2020, pp. 135–160. ISBN: 978-981-15-4095-0. DOI: 10.1007/978-981-15-4095-0_4. URL: https://doi.org/10.1007/978-981-15-4095-0_4.

[29] Jemin Hwangbo et al. "Control of a quadrotor with reinforcement learning". In: *IEEE Robotics and Automation Letters* 2.4 (2017), pp. 2096–2103.

[30] Alexey Grigorevich Ivakhnenko. "Polynomial theory of complex systems". In: *IEEE transactions on Systems, Man, and Cybernetics* 4 (1971), pp. 364–378.

[31] HJ Karssies and C De Wagter. "Extended incremental non-linear control allocation (XINCA) for quadplanes". In: *International Journal of Micro Air Vehicles* 14 (2022), p. 17568293211070825.

[32] Elia Kaufmann, Leonard Bauersfeld, and Davide Scaramuzza. "A benchmark comparison of learned control policies for agile quadrotor flight". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 10504–10510.

[33] Nikolay V Kim et al. "Selecting a Flight Path of an UAV to the Ship in Preparation of Deck Landing". In: *Indian Journal of Science and Technology* (2016).

[34] Donald E Kirk. *Optimal control theory: an introduction*. Courier Corporation, 2004.

[35] Johannes Lederer. "Activation functions in artificial neural networks: A systematic overview". In: *arXiv preprint arXiv:2101.09957* (2021).

[36] Seongheon Lee et al. "Vision-based autonomous landing of a multi-copter unmanned aerial vehicle using reinforcement learning". In: *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2018, pp. 108–114.

[37] Sergey Levine. "Exploring deep and recurrent architectures for optimal control". In: *arXiv preprint arXiv:1311.1761* (2013).

[38] Haochen Li et al. "Fast Trajectory Generation with a Deep Neural Network for Hypersonic Entry Flight". In: *Aerospace* 10.11 (2023), p. 931.

[39] Marlos C Machado et al. "Temporal abstraction in reinforcement learning with the successor representation". In: *Journal of Machine Learning Research* 24.80 (2023), pp. 1–69.

[40] Priya R Maidamwar, Mahip M Bartere, and Prasad P Lokulwar. "A survey on machine learning approaches for developing intrusion detection system". In: *Proceedings of the international conference on innovative computing & communication (ICICC)*. 2021.

[41] Siri Mathisen et al. "Precision deep-stall landing of fixed-wing UAVs using nonlinear model predictive control". In: *Journal of Intelligent & Robotic Systems* 101 (2021), pp. 1–15.

[42] Deepanshu Mehta. “State-of-the-Art Reinforcement Learning Algorithms”. In: *International Journal of Engineering Research and* (2020). URL: https://api.semanticscholar.org/CorpusID:212589933.

[43] Xue Bin Peng et al. “Sim-to-real transfer of robotic control with dynamics randomization”. In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2018, pp. 3803–3810.

[44] Robert Penicka et al. “Learning minimum-time flight in cluttered environments”. In: *IEEE Robotics and Automation Letters* 7.3 (2022), pp. 7209–7216.

[45] Riccardo Polvara et al. “Toward end-to-end control for UAV autonomous landing via deep reinforcement learning”. In: *2018 International conference on unmanned aircraft systems (ICUAS)*. IEEE. 2018, pp. 115–123.

[46] Alejandro Rodriguez-Ramos et al. “A deep reinforcement learning strategy for UAV autonomous landing on a moving platform”. In: *Journal of Intelligent & Robotic Systems* 93 (2019), pp. 351–366.

[47] Fereshteh Sadeghi and Sergey Levine. “Cad2rl: Real single-image flight without a single real image”. In: *arXiv preprint arXiv:1611.04201* (2016).

[48] Adnan S Saeed et al. “A survey of hybrid unmanned aerial vehicles”. In: *Progress in Aerospace Sciences* 98 (2018), pp. 91–105.

[49] Vishnu Saj et al. “Robust Reinforcement Learning Algorithm for Vision-based Ship Landing of UAVs”. In: *arXiv preprint arXiv:2209.08381* (2022).

[50] Erica Salvato et al. “Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning”. In: *IEEE Access* 9 (2021), pp. 153171–153187.

[51] Carlos Sánchez-Sánchez and Dario Izzo. “Real-time optimal control via deep neural networks: study on landing problems”. In: *Journal of Guidance, Control, and Dynamics* 41.5 (2018), pp. 1122–1135.

[52] Carlos Sánchez-Sánchez, Dario Izzo, and Daniel Hennes. “Optimal real-time landing using deep networks”. In: *Proceedings of the Sixth International Conference on Astrodynamics Tools and Techniques, ICATT*. Vol. 12. 2016, pp. 2493–2537.

[53] Joao Leonardo Silva Cotta et al. “High-Altitude Precision Landing by Smartphone Video Guidance Sensor and Sensor Fusion”. In: *Drones* 8.2 (2024), p. 37.

[54] Ewoud Smeur, Daan Höppener, and Christophe De Wagter. “Prioritized control allocation for quadrotors subject to saturation”. In: *International Micro Air Vehicle Conference and Flight Competition*. September. 2017, pp. 37–43.

[55] Ewoud JJ Smeur, Qiping Chu, and Guido CHE De Croon. “Adaptive incremental nonlinear dynamic inversion for attitude control of micro air vehicles”. In: *Journal of Guidance, Control, and Dynamics* 39.3 (2016), pp. 450–461.

[56] Edward J Sondik. “The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs”. In: *Operations research* 26.2 (1978), pp. 282–304.

[57] O. Frank Spurlock and Craig H. Williams. *DUKSUP: A Computer Program for High Thrust Launch Vehicle Trajectory Design and Optimization*. NASA Technical Memorandum. Available at NASA STI Repository. 2015.

[58] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[59] Richard S Sutton, Andrew G Barto, and Ronald J Williams. “Reinforcement learning is direct adaptive optimal control”. In: *IEEE control systems magazine* 12.2 (1992), pp. 19–22.

[60] Liguo Tan et al. “Research on optimal landing trajectory planning method between an UAV and a moving vessel”. In: *Applied Sciences* 9.18 (2019), p. 3708.

[61] Josh Tobin et al. “Domain randomization for transferring deep neural networks from simulation to the real world”. In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2017, pp. 23–30.

[62] Masatoshi Uehara, Chengchun Shi, and Nathan Kallus. “A review of off-policy evaluation in reinforcement learning”. In: *arXiv preprint arXiv:2212.06355* (2022).

[63] Hado Van Hasselt, Arthur Guez, and David Silver. "Deep reinforcement learning with double q-learning". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 30. 1. 2016.

[64] Bingkun Wang et al. "An Autonomous Tracking and Landing Method for Unmanned Aerial Vehicles Based on Visual Navigation". In: *Drones* 7.12 (2023), p. 703.

[65] Jingyi Xie et al. "UAV autonomous tracking and landing based on deep reinforcement learning strategy". In: *Sensors* 20.19 (2020), p. 5630.

[66] Long Xin et al. "Vision-based autonomous landing for the UAV: A review". In: *Aerospace* 9.11 (2022), p. 634.

[67] Tao Yang et al. "A ground-based near infrared camera array system for UAV auto-landing in GPS-denied environment". In: *Sensors* 16.9 (2016), p. 1393.

[68] Man Yuan et al. "PID with deep reinforcement learning and heuristic rules for autonomous UAV landing". In: *International Conference on Autonomous Unmanned Systems*. Springer. 2022, pp. 1876–1884.

A

Simulations for Landing the Parrot Bebop 1 on a stationary platform

As a part of the preliminary analysis, the simulations are first conducted for the Parrot Bebop 1 drone for a landing on *a stationary* platform task. In this chapter, the process of designing an effective reward function for this type of a landing problem is explained and the contribution of each element is examined.

Parrot Bebop 1 Equations of Motions

The equations of motions for the Parrot Bebop 1 are given in the following equations A.1, A.2, A.3 and A.4. Since the problem considering a stationary platform not a moving one, these were the only dynamics included in the vectorized environment.

$$\dot{\mathbf{x}} = \mathbf{v} \tag{A.1}$$

$$\dot{\mathbf{v}} = \mathbf{g} + \frac{1}{m} R(\boldsymbol{\lambda}) \begin{bmatrix} -k_x u \sum_{i=0}^{4} \omega_i \\ -k_y v \sum_{i=0}^{4} \omega_i \\ k_w \sum_{i=0}^{4} \omega_i^2 + k_z w \sum_{i=0}^{4} \omega_i + k_h (u^2 + v^2) \end{bmatrix} \quad \text{where} \quad \begin{bmatrix} u \\ v \\ w \end{bmatrix} = R(\boldsymbol{\lambda})^T \mathbf{v} \tag{A.2}$$

$$\dot{\boldsymbol{\lambda}} = Q(\boldsymbol{\lambda}) \boldsymbol{\Omega} \tag{A.3}$$

$$I\dot{\boldsymbol{\Omega}} = -\boldsymbol{\Omega} \times I\boldsymbol{\Omega} + \begin{bmatrix} k_p(\omega_1^2 - \omega_2^2 - \omega_3^2 + \omega_4^2) \\ k_q(\omega_1^2 + \omega_2^2 - \omega_3^2 - \omega_4^2) \\ k_r(\tau_1(\omega_1 + \omega_2 + \omega_3 + \omega_4) + k_{r_2}(\dot{\omega}_1 + \dot{\omega}_2 + \dot{\omega}_3 + \dot{\omega}_4) - k_{r_\tau} T) \end{bmatrix} \tag{A.4}$$

$$\dot{\omega}_i = \frac{(u_i - \omega_i)}{\tau} \tag{A.5}$$

The parameters included in the code are given in Table A.1, A.3, and A.2. Reader can check the reference paper `https://arxiv.org/pdf/2304.13460.pdf` for further information on the model.

Table A.1: Model parameters

Parameter	Value
k_x	$1.07933887 \times 10^{-5}$
k_y	$9.65250793 \times 10^{-6}$
k_z	2.7862899×10^{-5}
k_w	$4.36301076 \times 10^{-8}$
k_h	0.0625501332
k_p	1.4119331×10^{-9}
k_{pv}	−0.00797101848
k_q	$1.21601884 \times 10^{-9}$
k_{qv}	0.0129263739
k_{r1}	$2.57035545 \times 10^{-6}$
k_{r2}	$4.10923364 \times 10^{-7}$
k_{rr}	0.000812932607

Parameter	Value
g	9.81
I_{xx}	0.000906
I_{yy}	0.001242
I_{zz}	0.002054

Table A.2: Inertia and Gravity Parameters

Parameter	Value
τ	0.06
w_{min}	3000
w_{max}	11000

Table A.3: Motor parameters

Reward Function Design

Reward design process has completed with consideration of different tasks that were part of the whole landing process. In the following subsections, the elements included in the reward/objective function are explained individually.

Distance to goal

The primary objective was to reach the target platform while ignoring variations in other state elements such as velocity and acceleration of the drone. Consequently, the simulations were labelled as successful in cases where the drone tracked and landed on the platform. The corresponding reward element is provided in Eq. A.6.

$$R_p = 10(|p_{r_{\text{old}}}| - |p_{r_{\text{new}}}|) \tag{A.6}$$

The parameters $p_{r_{\text{old}}}$ and $p_{r_{\text{new}}}$ represent the relative old and new distances between the drone and the platform, respectively. These terms reward the agent based on the decrease in the relative distance and motivates if further to reach the target.

While the landings were completely successful within the trained set and even with deviations up to 4 meters, the landings concluded with extremely high velocities. This necessitated an element to adjust the drone's velocity during landing.

Velocity reward

For successful landings, the touchdown velocity must be below a certain threshold. Therefore, the drone's velocity was penalized using two types of reward functions, as shown in Eq. A.7 and Eq. A.8. Here the parameters $v_{d_{\text{old}}}$ and $v_{d_{\text{new}}}$ represent the old and new velocities of the drone. The latter function imposed stricter penalties on the drone, resulting in lower accelerations compared to the former.

$$R_{v1} = (|v_{d_{\text{old}}}| - |v_{d_{\text{new}}}|) \tag{A.7}$$

$$R_{v2} = -0.01 \times |\mathbf{v}_{\text{new}}| \tag{A.8}$$

Landing classification

The provided landing classification equation A.9 contributes significantly to the autonomous landing problem by focusing on vertical velocity considerations. It assigns a positive reward (+10) if the UAV's vertical velocity ($v_{2g_{\text{new}}}$) is less than 0.5 m/s, promoting gentle and safe landings, while penalizing higher velocities to discourage hard landings. This reward system is integral to a reinforcement learning (RL) framework, guiding the UAV to learn optimal landing maneuvers through trial and error. By incorporating vertical velocity into the reward function, the UAV is encouraged to minimize its descent speed, ensuring smoother touchdowns and reducing the risk of damage.

$$R = \begin{cases} +10, & v_{2g_{\text{new}}} < 0.5 \\ -10, & v_{2g_{\text{new}}} < 0.5 \end{cases} \tag{A.9}$$

Penalty for rotation

Rotation penalty given in Eq. A.11 is designed to discourage excessive angular velocity during the landing phase of the UAV. By assigning a negative value proportional to the absolute angular velocity, the system incourages the dron to maintain a stable and controlled orientation. This is curucial for ensuring smooth landing, as high rotational speeds can lead to instability and increased risk of damage.

$$penalty_{\text{rotation}} = -0.01 \times |\Omega_{\text{new}}| \tag{A.10}$$

Penalty for ground collision/ out of bounds

Penalties for collisions and out-of-bounds conditions are part of the end conditions that terminate the simulations when met. In this work, a penalty is applied if the UAV either collides with the ground or lands outside the designated area. This function ensures that both the state space and action space are bounded.

$$R_{rp} = \begin{cases} +10, & v_{2g_{\text{new}}} < 0.5 \\ -10, & v_{2g_{\text{new}}} < 0.5 \end{cases} \tag{A.11}$$

Table A.4: Reward combinations used for the Parrot Bebop 1

Reward Number	Combination
R1	R_p
R2	$R_p + R_{v1} + R_{rp}$
R3	$R_p + R_{v1} + R_{rp} + R_c$
R4	$R_p + R_{v2} + R_{rp} + R_c$

Results

The reward combinations used in the study are presented in Table A.4. The performance of each reward function was tested under nominal conditions, with deviations of 2 meters and 4 meters. While nominal conditions served to verify the occurrence of landings, the deviation tests assessed the robustness of the reinforcement learning. For the given reward combinations, the variations in state parameters p_n, p_e, p_d (position), v_n, v_e, v_d (velocity) and ϕ, θ, ψ (Euler angles) are illustrated in the following figures: nominal conditions in Figures A.1,A.7,A.13; deviation of 2 meters in Figures A.3,A.9,A.15, A.21; and deviation of 4 meters in A.5,A.11,A.17. These figures demonstrate how the UAV reaches and lands (if applicable) on the platform with varying reward functions, highlighting the effectiveness of each reward in guiding the UAV's behavior during the landing phase.

As a result of this progressive analysis, the reward combination R4 successfully satisfied the safe, precise, and smooth landing conditions, as represented in the final figures.

R1

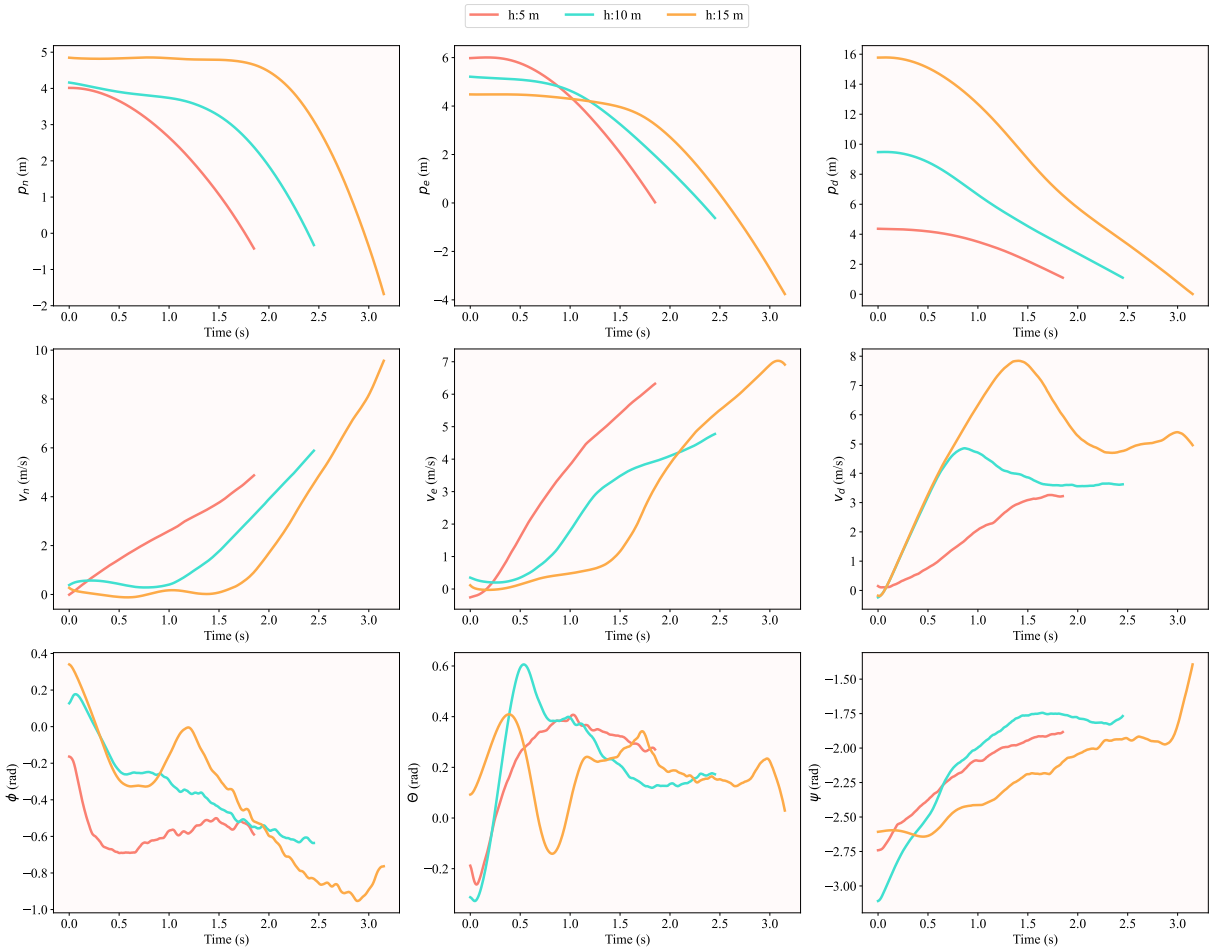


Figure A.1: State parameters for reward R1 under nominal conditions

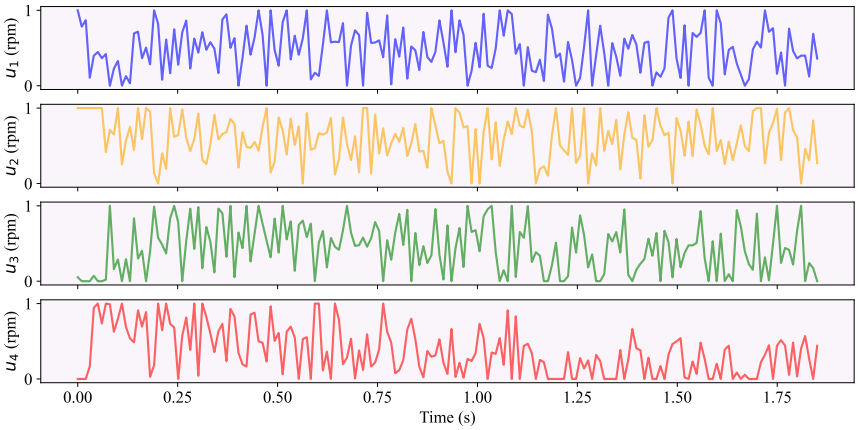


Figure A.2: Generated control inputs for reward R1 under nominal conditions

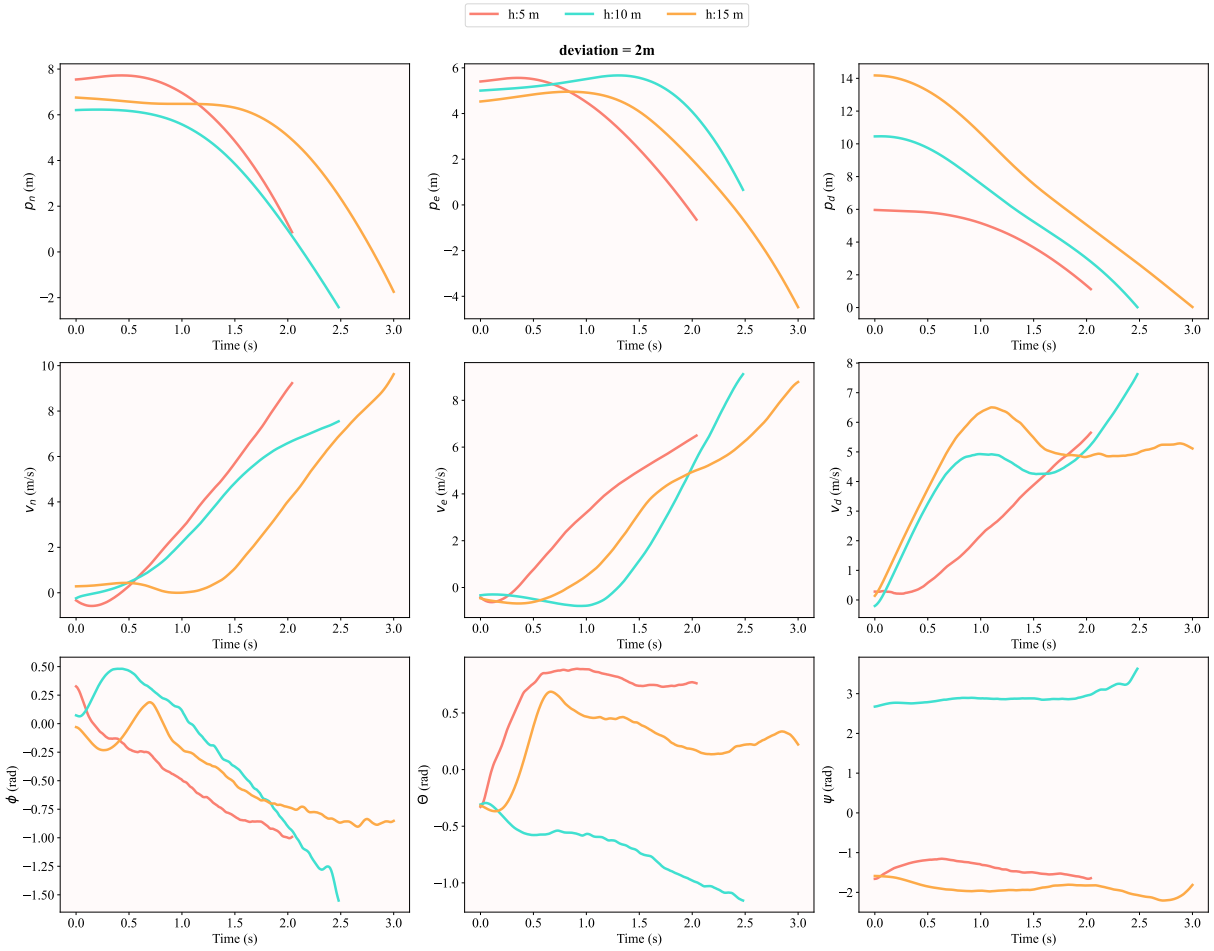


Figure A.3: State parameters for reward R3 with a deviation of 2m

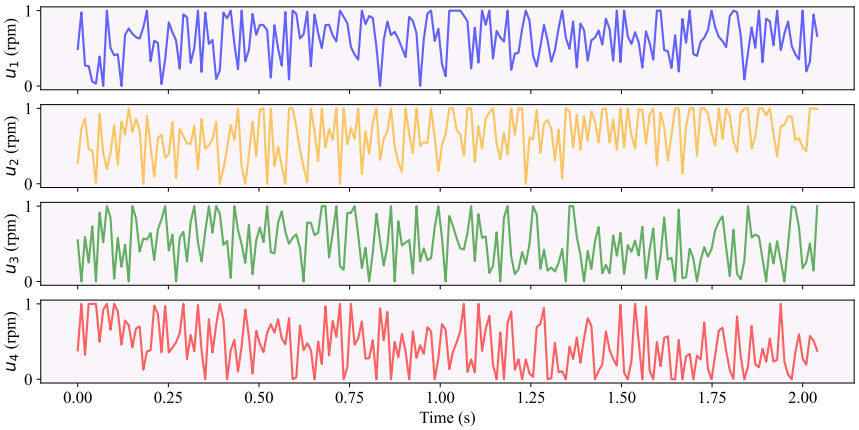


Figure A.4: Generated control inputs for reward R1 with a deviation of 2m

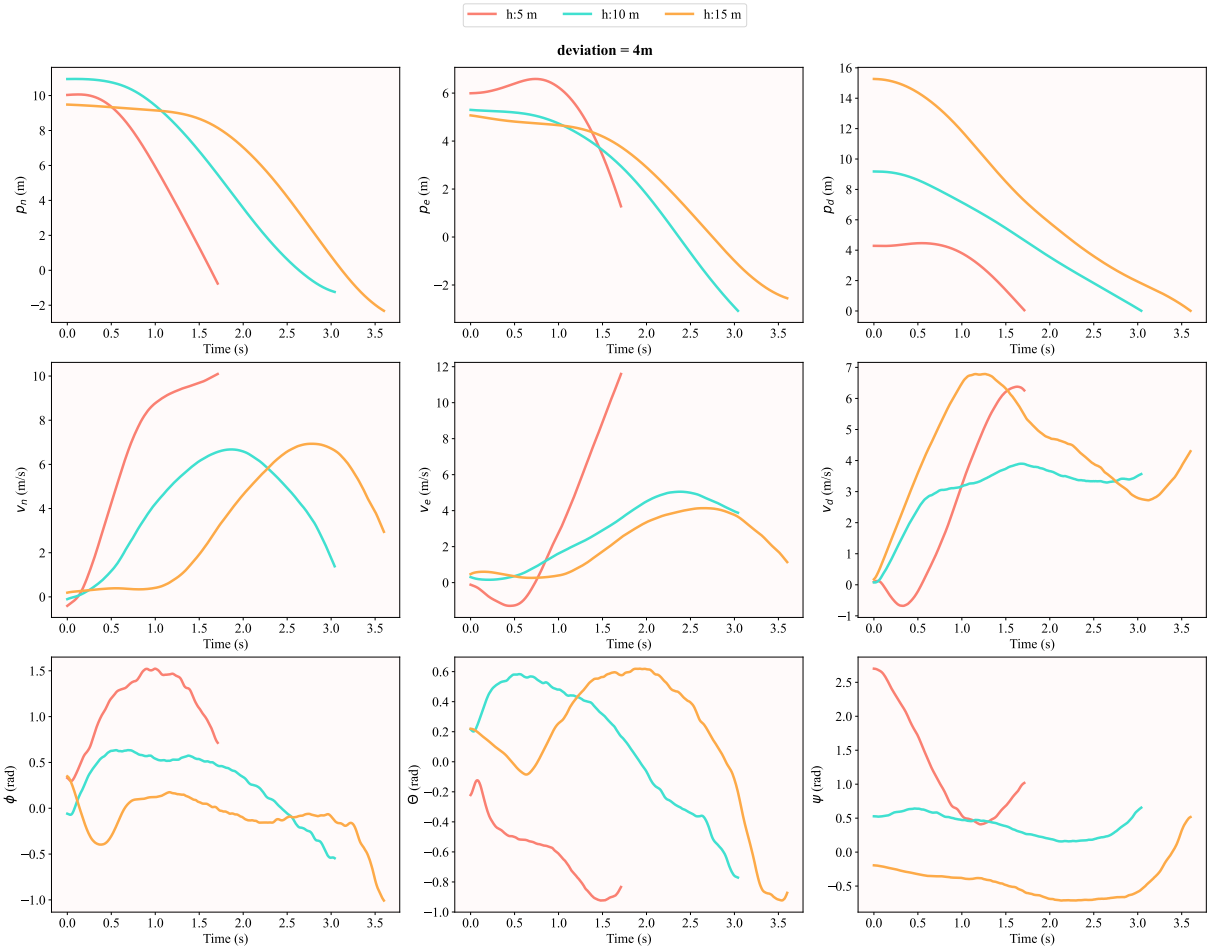


Figure A.5: State parameters for reward R3 with a deviation of 4m

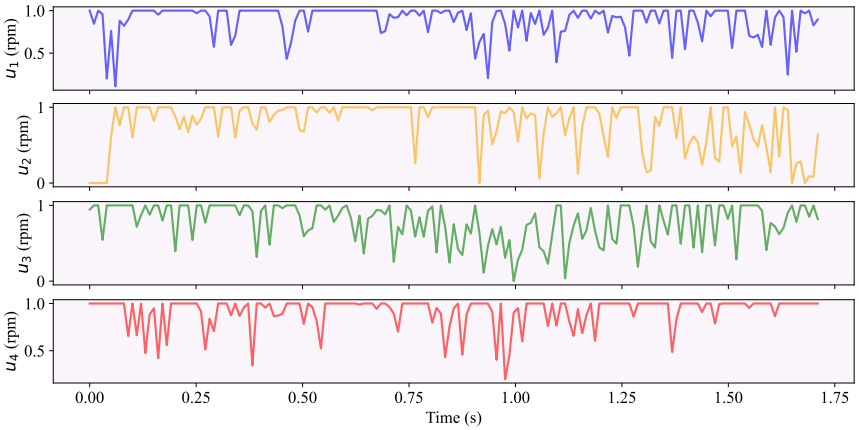


Figure A.6: Generated control inputs for reward R1 with a deviation of 4m

R2

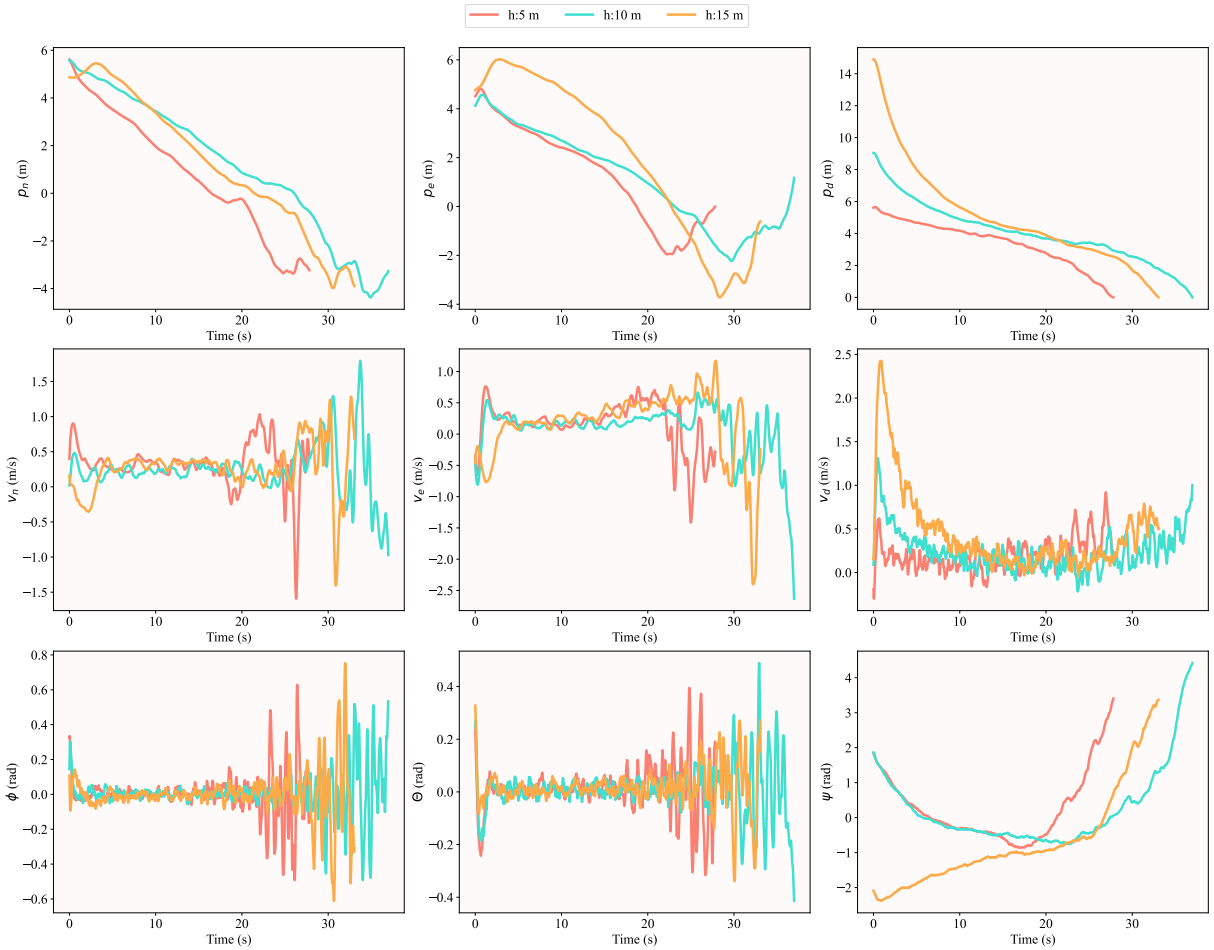


Figure A.7: State parameters for reward R2 under nominal conditions

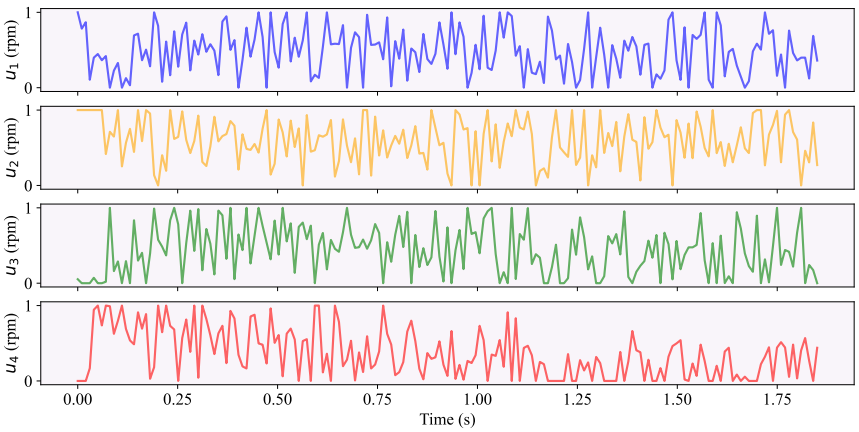


Figure A.8: Generated control inputs for reward R2 under nominal conditions

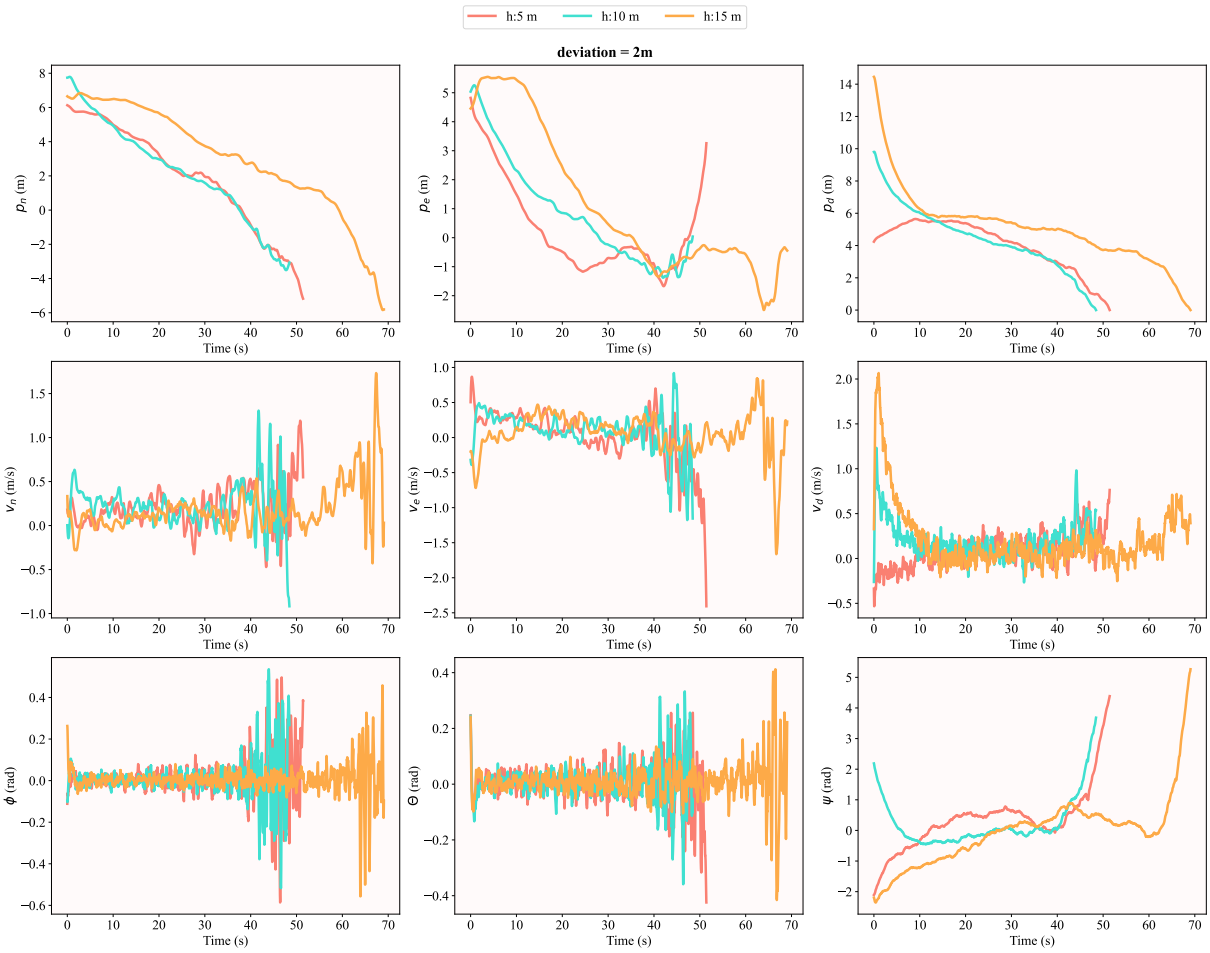


Figure A.9: State parameters for reward R2 with a deviation of 2m

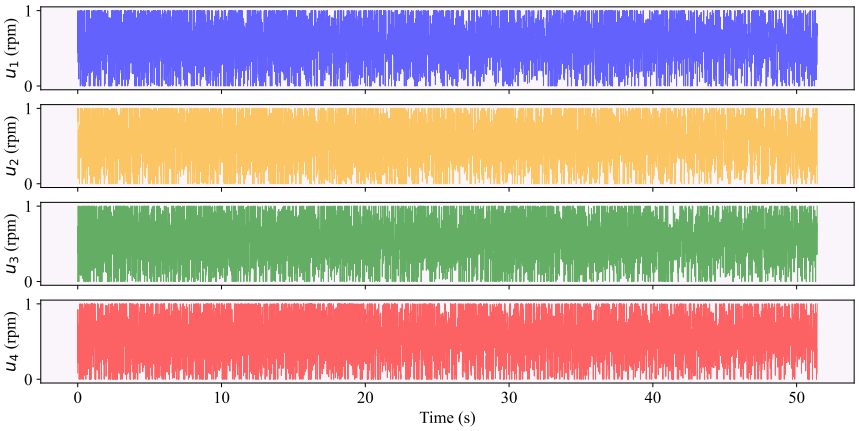


Figure A.10: Generated control inputs for reward R2 with a deviation of 2m

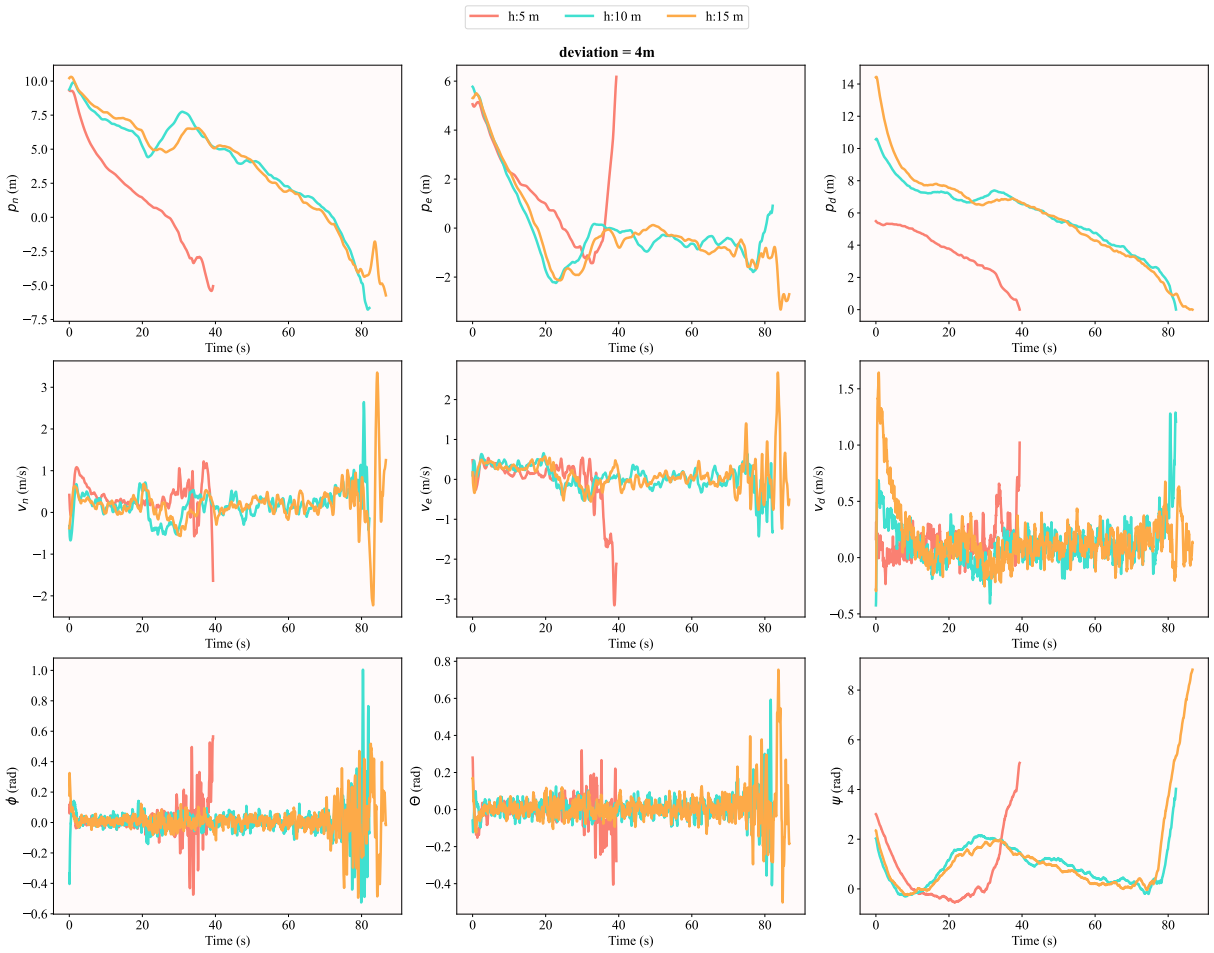


Figure A.11: State parameters for reward R2 with a deviation of 4m

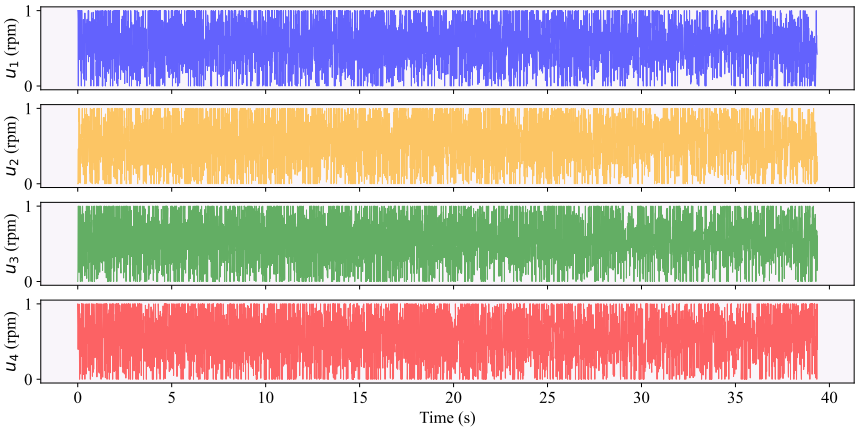


Figure A.12: Generated control inputs for reward R2 with a deviation of 4m

R3

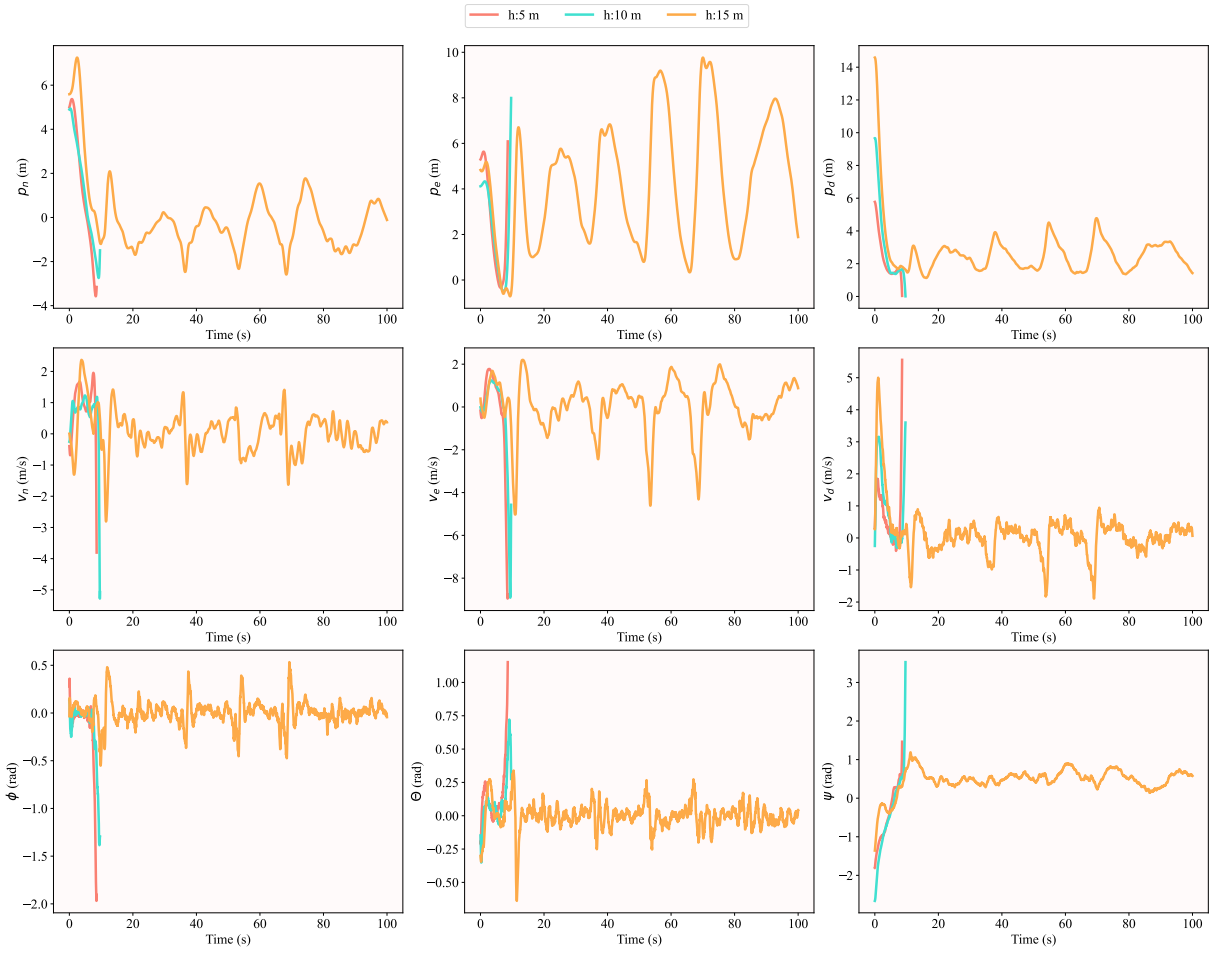


Figure A.13: State parameters for reward R3 under nominal conditions

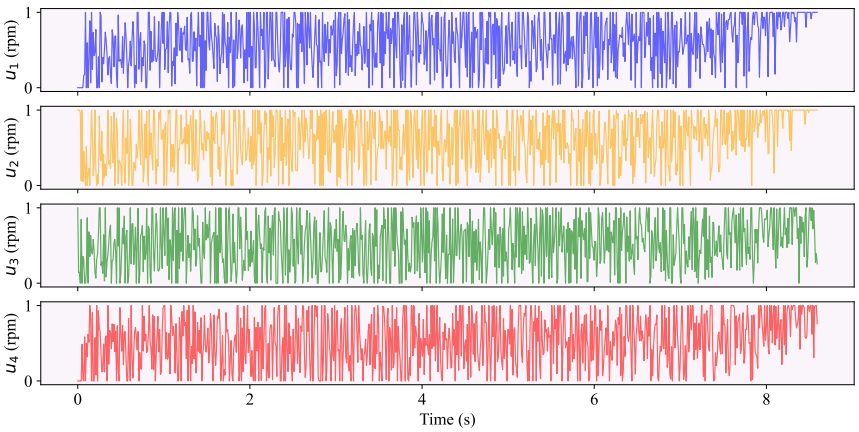


Figure A.14: Generated control inputs for reward R3 under nominal conditions

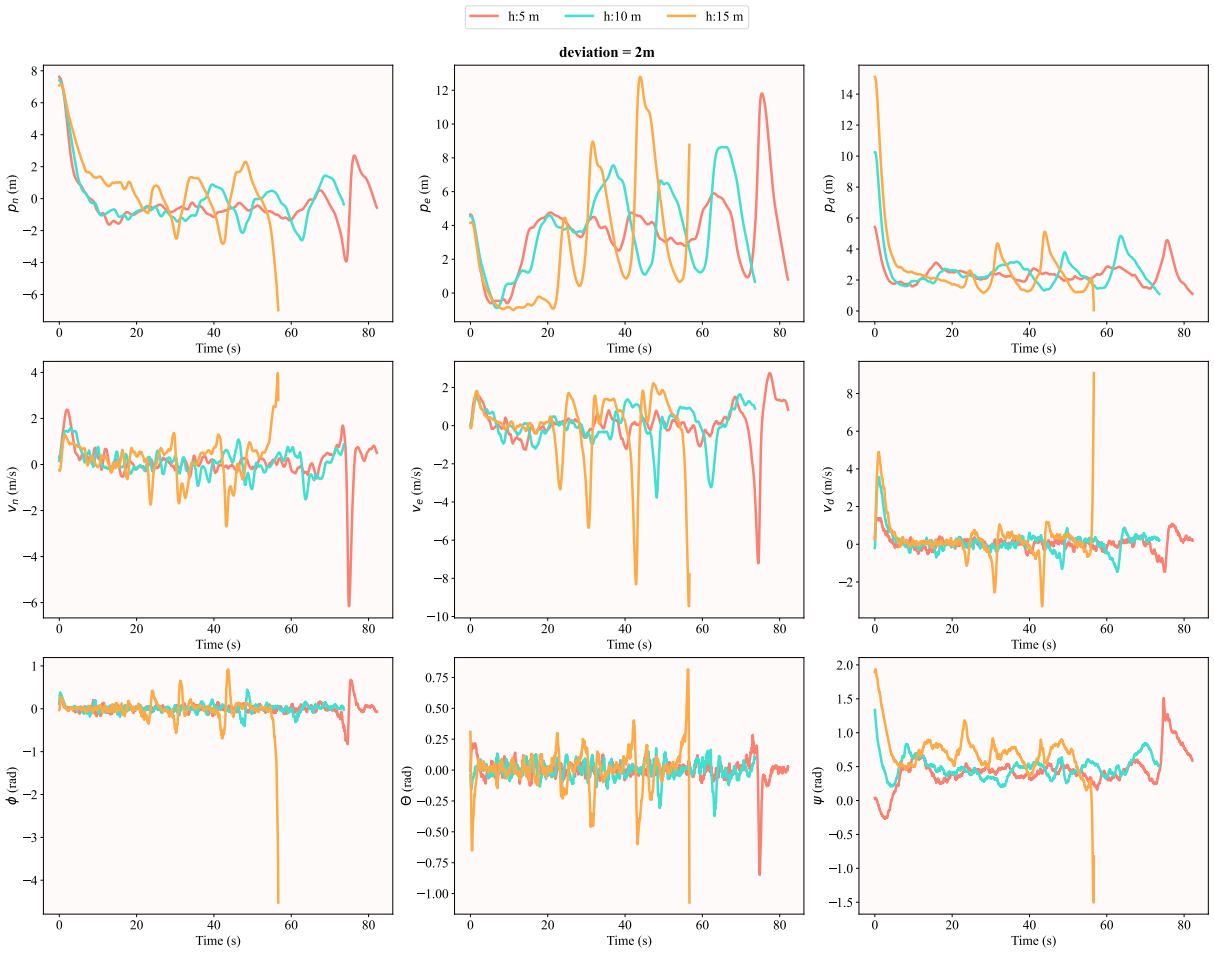


Figure A.15: State parameters for reward R3 with a deviation of 2m

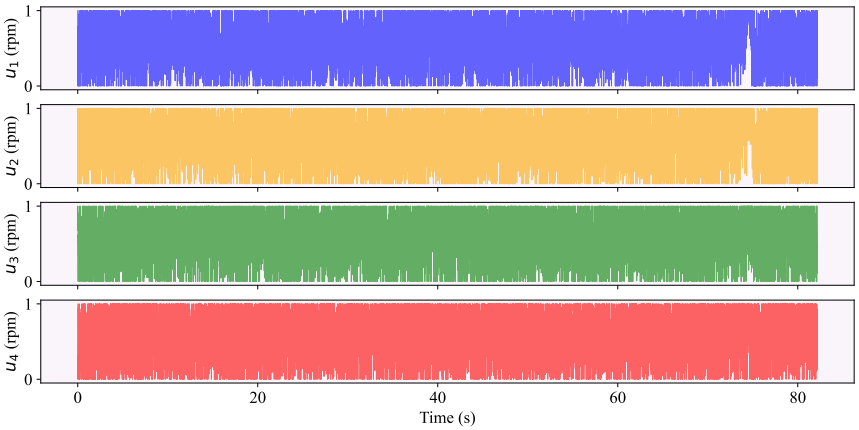


Figure A.16: Generated control inputs for reward R3 with a deviation of 2m

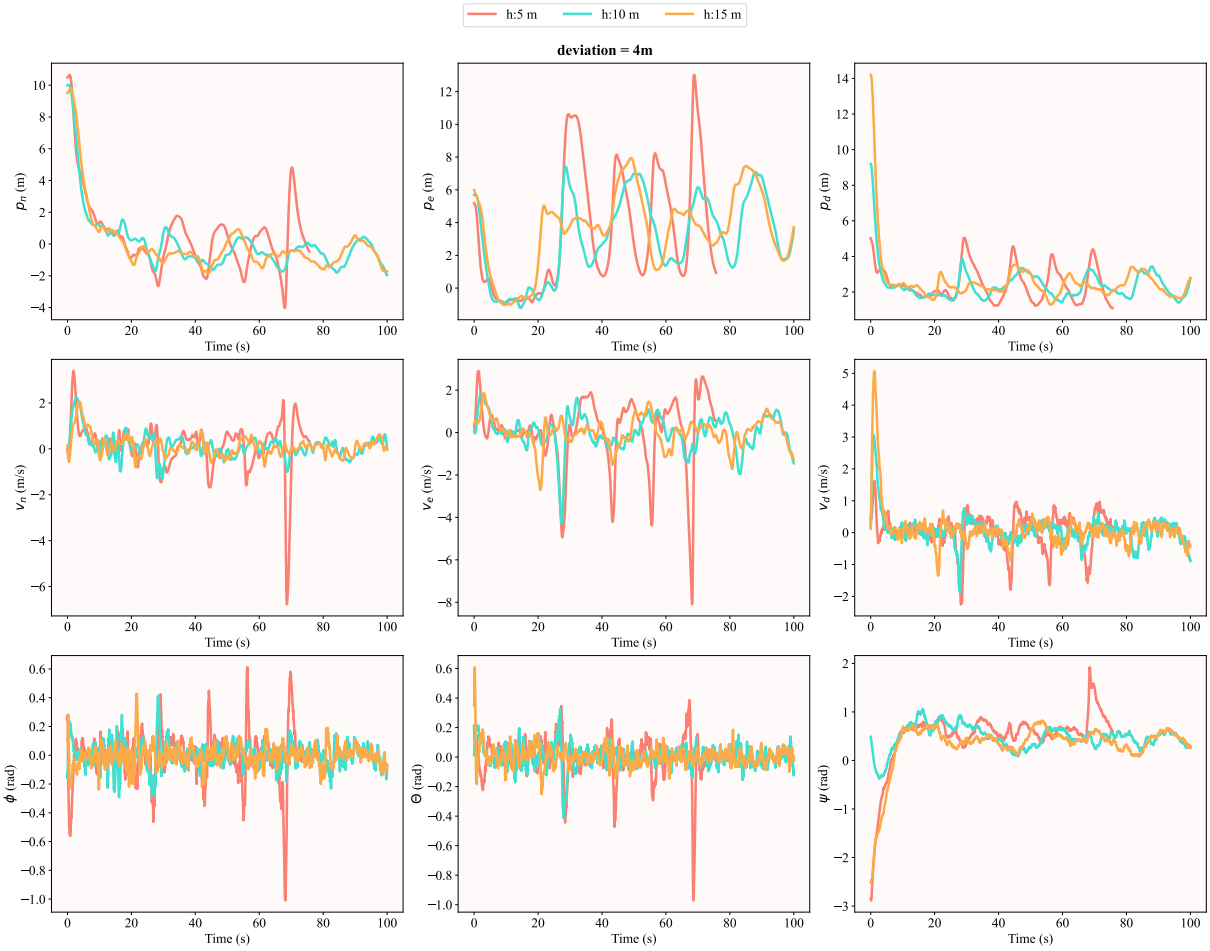


Figure A.17: State parameters for reward R3 with a deviation of 4m

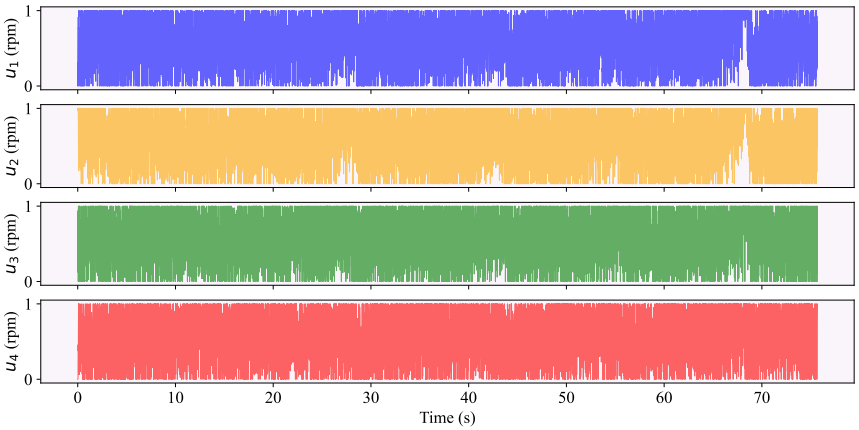


Figure A.18: Generated control inputs for reward R3 with a deviation of 4m

R4

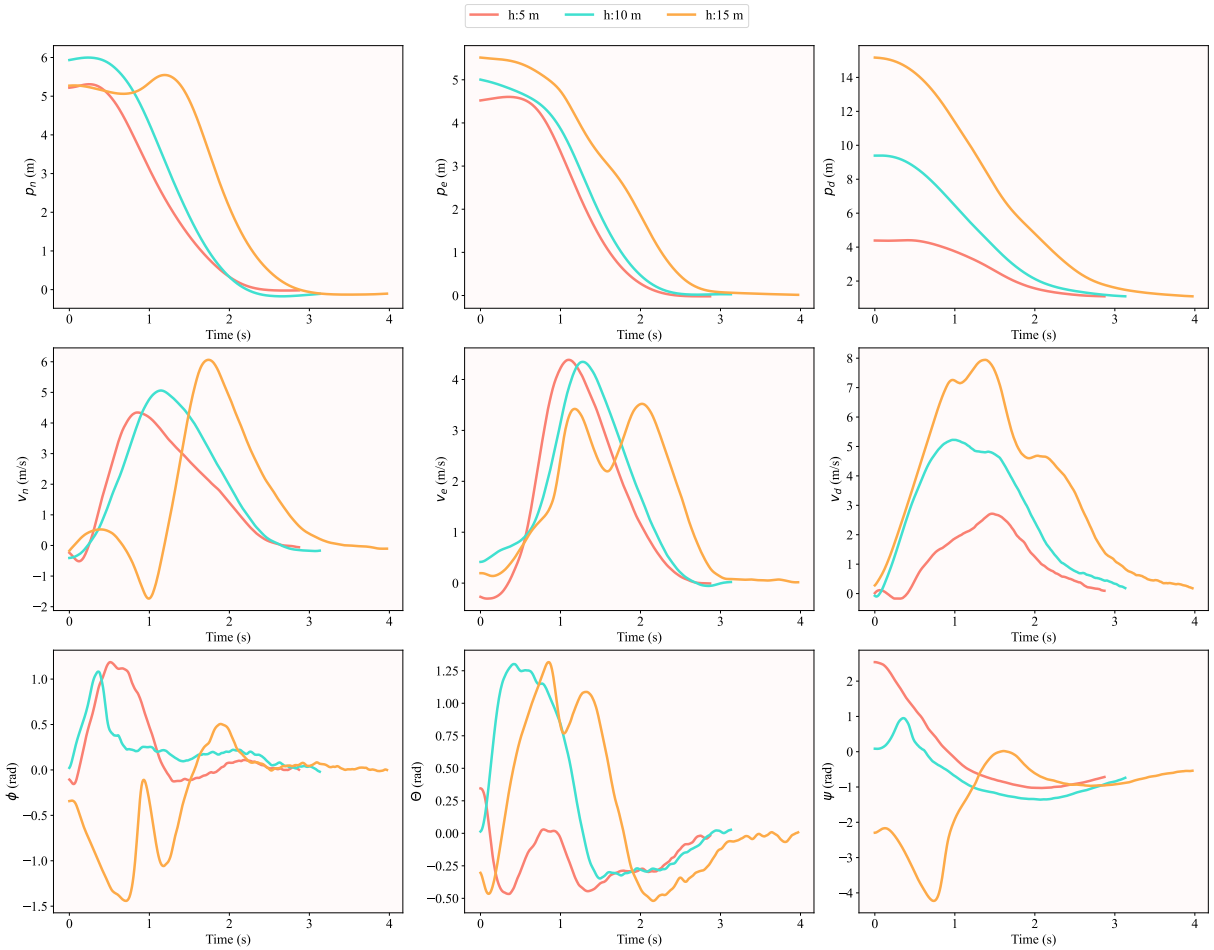


Figure A.19: State parameters for reward R4 under nominal conditions

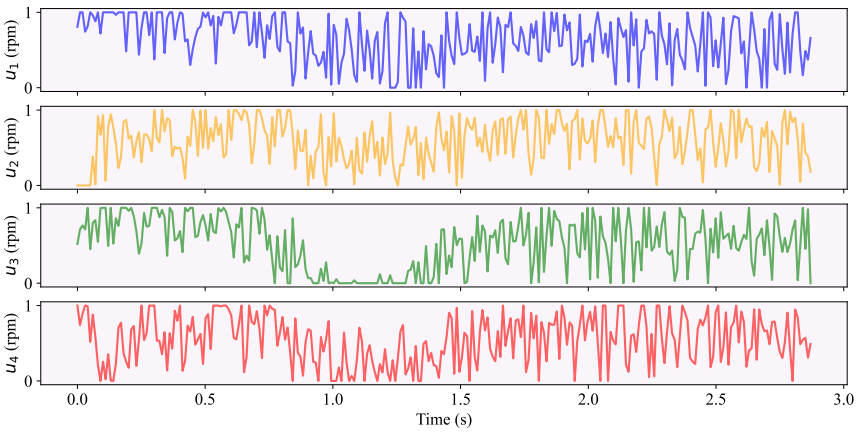


Figure A.20: Generated control inputs for reward R4 under nominal conditions

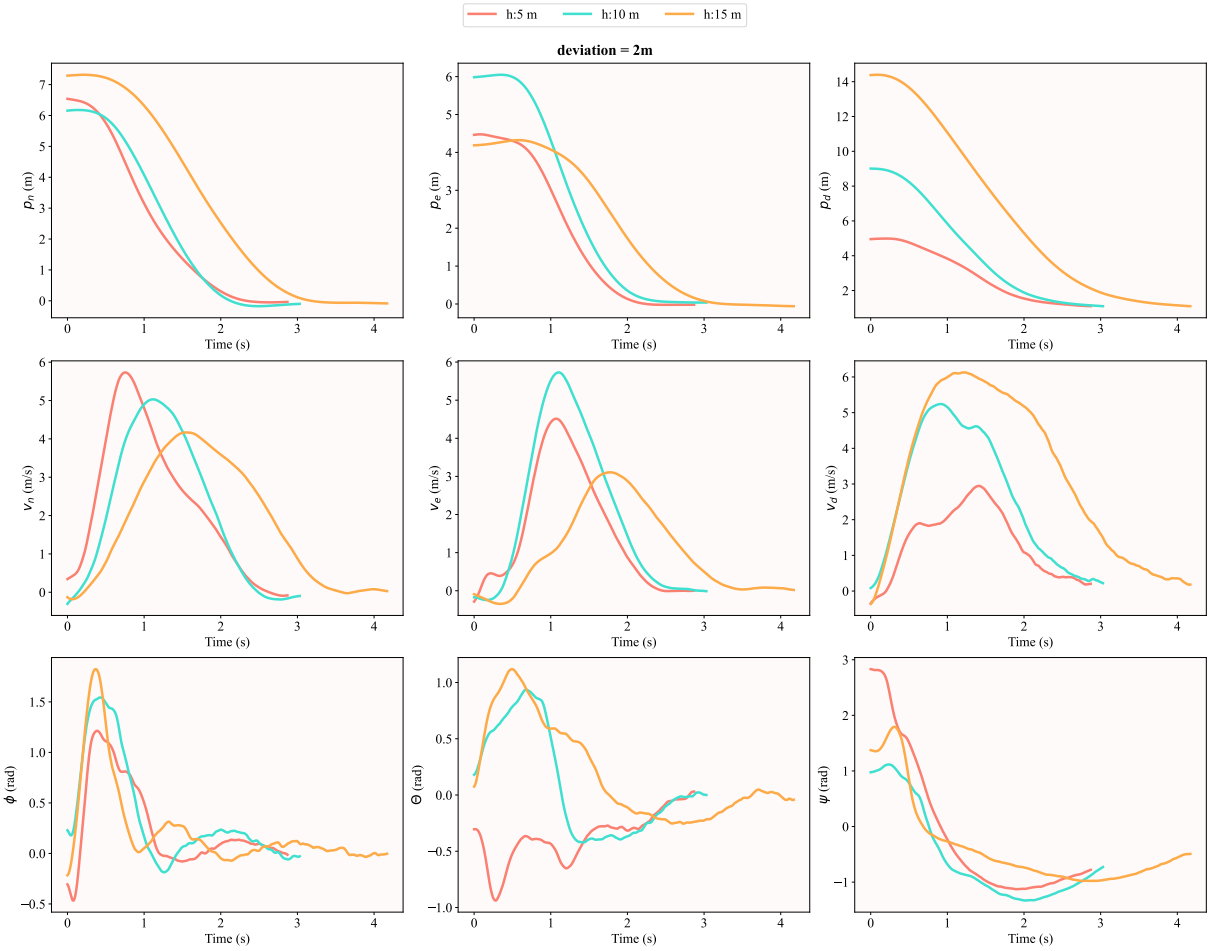


Figure A.21: State parameters for reward R4 with a deviation of 2m

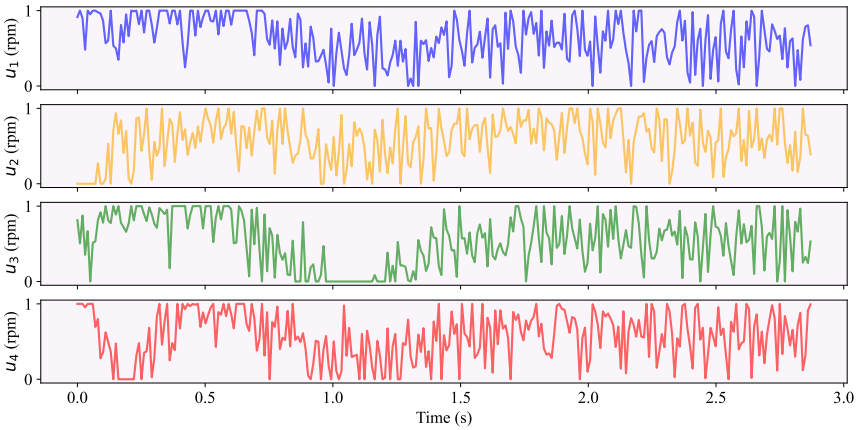


Figure A.22: Generated control inputs for reward R4 with a deviation of 2m

B

ANDI Input-Output Representation

In this chapter, all modifications made in reference models and error controller blocks to obtain the final transfer function for the simplified model are provided.

Reference Model

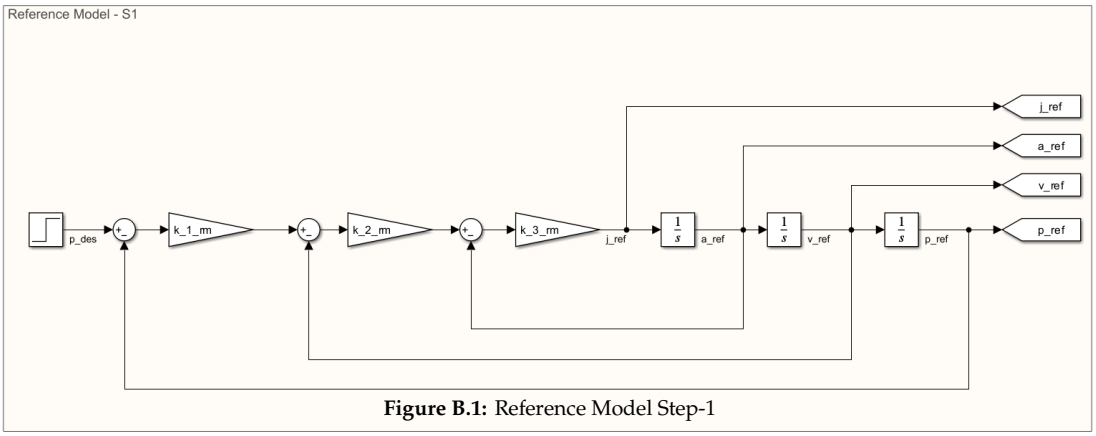


Figure B.1: Reference Model Step-1

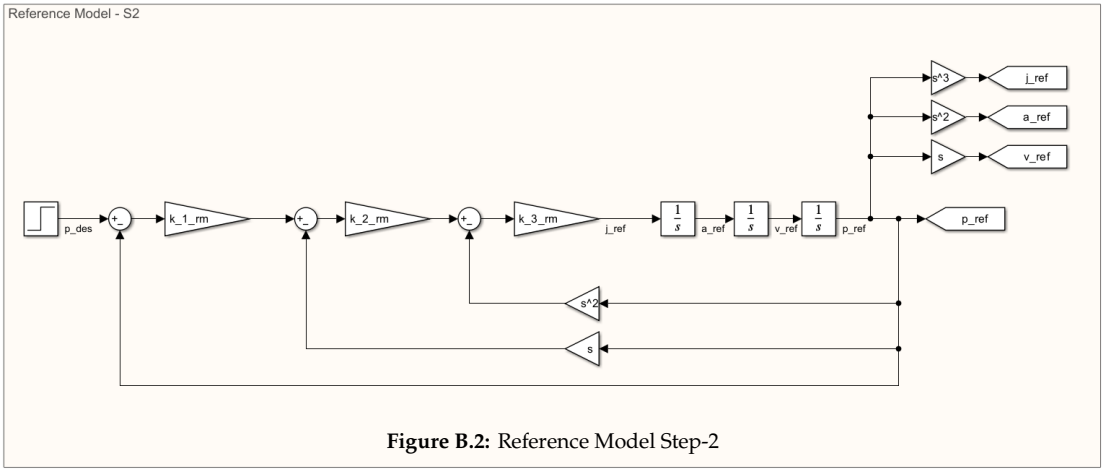


Figure B.2: Reference Model Step-2

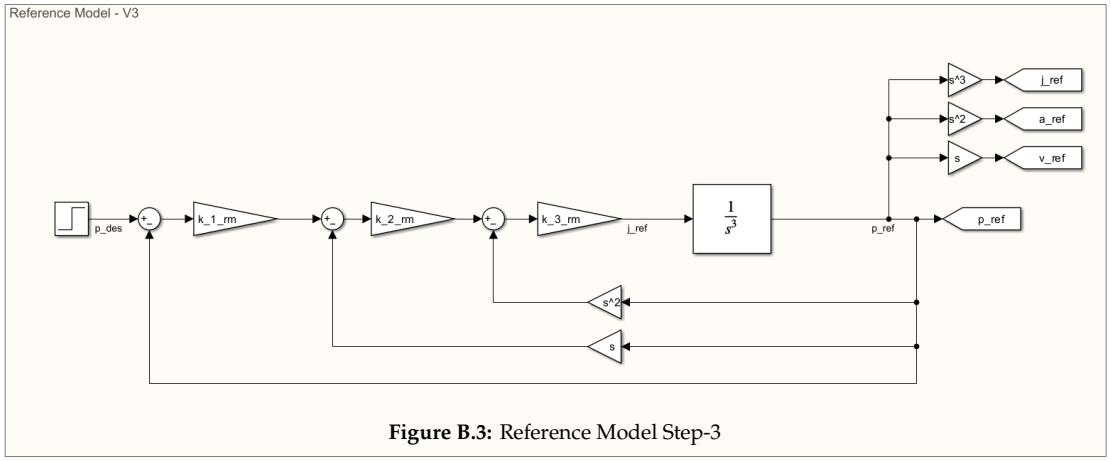


Figure B.3: Reference Model Step-3

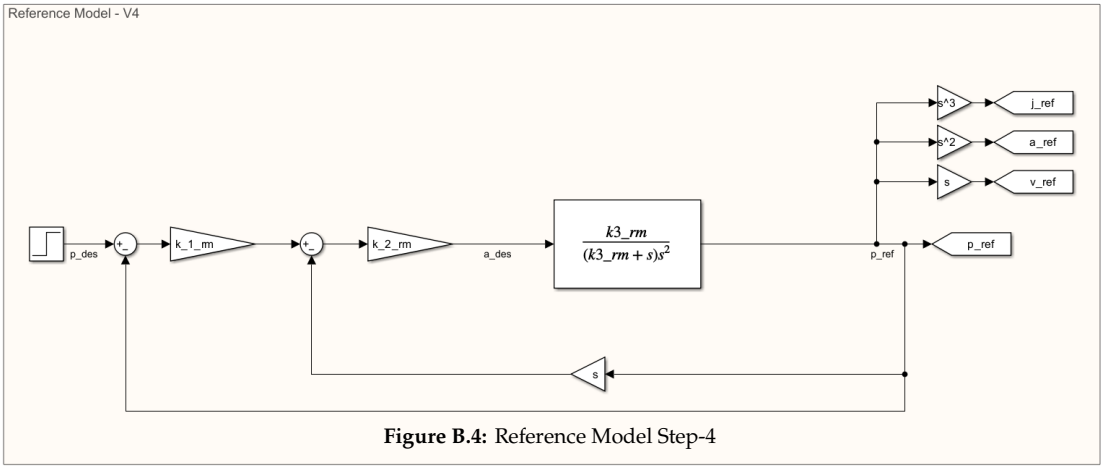


Figure B.4: Reference Model Step-4

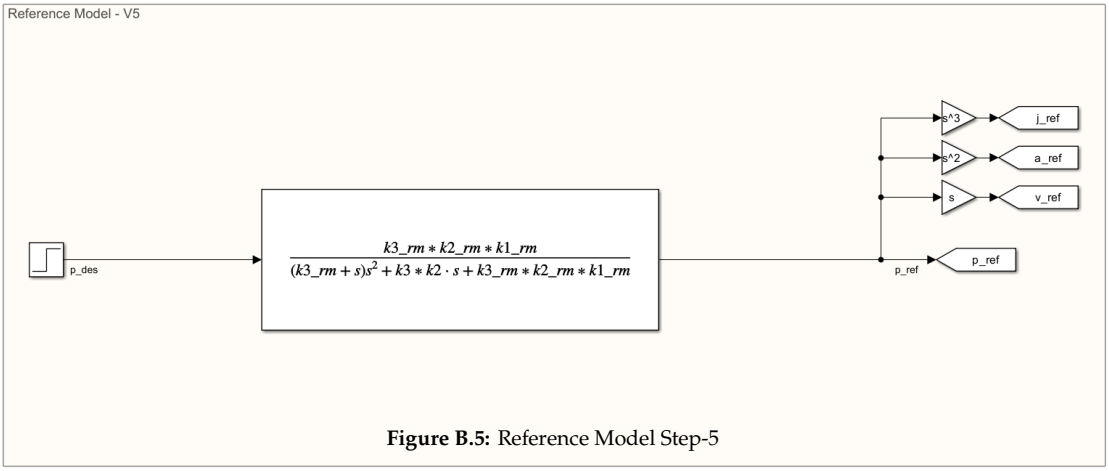


Figure B.5: Reference Model Step-5

Error Controllers and ANDI with attitude dynamics

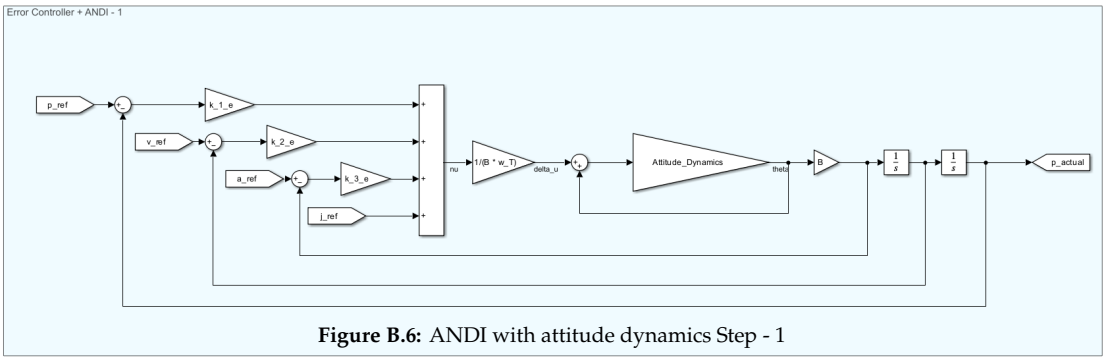


Figure B.6: ANDI with attitude dynamics Step - 1

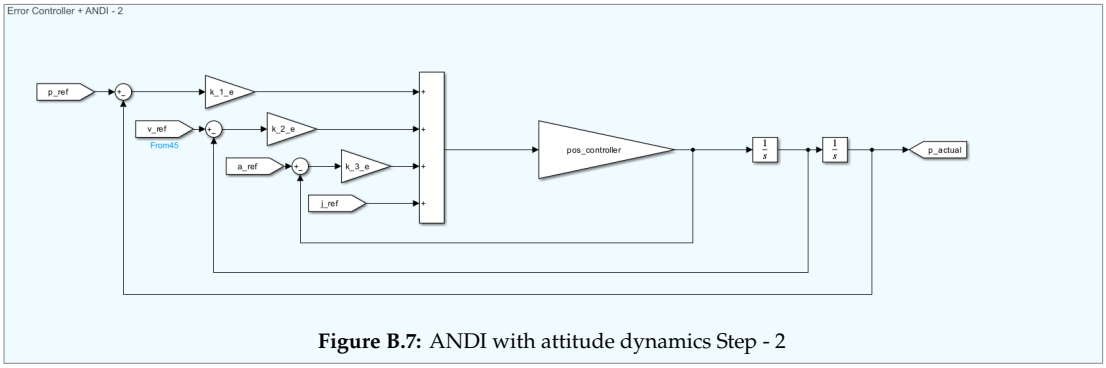


Figure B.7: ANDI with attitude dynamics Step - 2

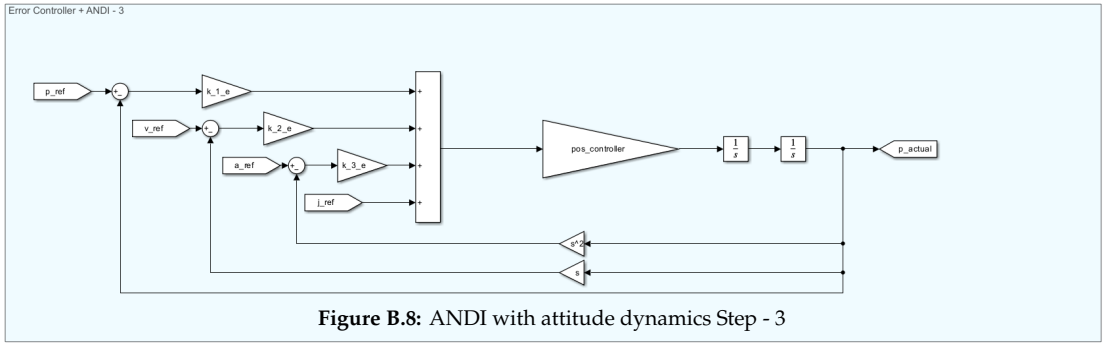


Figure B.8: ANDI with attitude dynamics Step - 3

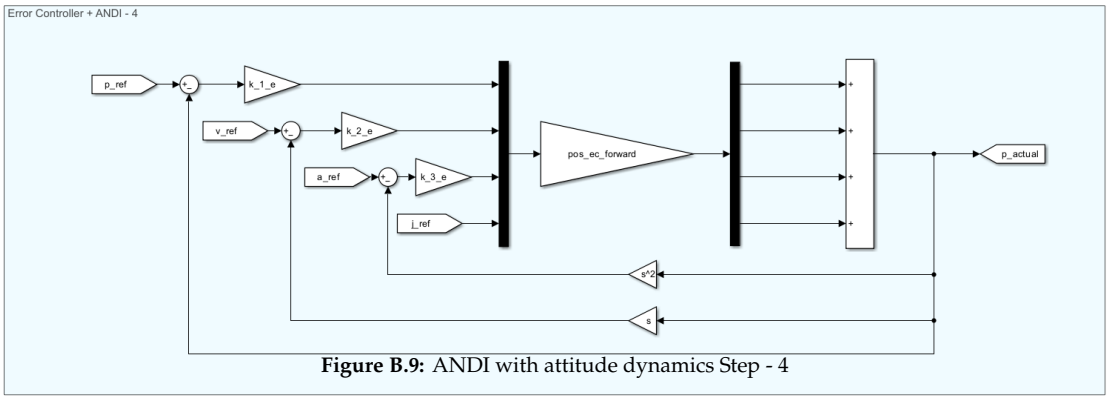


Figure B.9: ANDI with attitude dynamics Step - 4

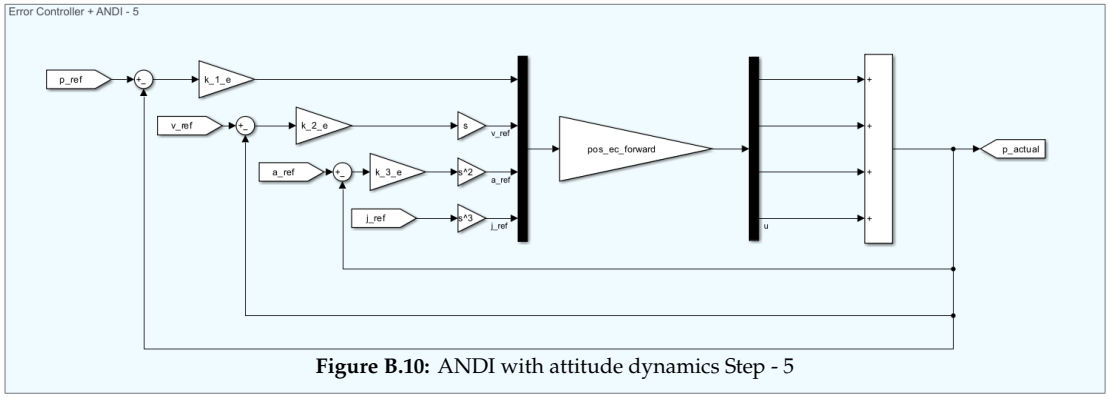


Figure B.10: ANDI with attitude dynamics Step - 5

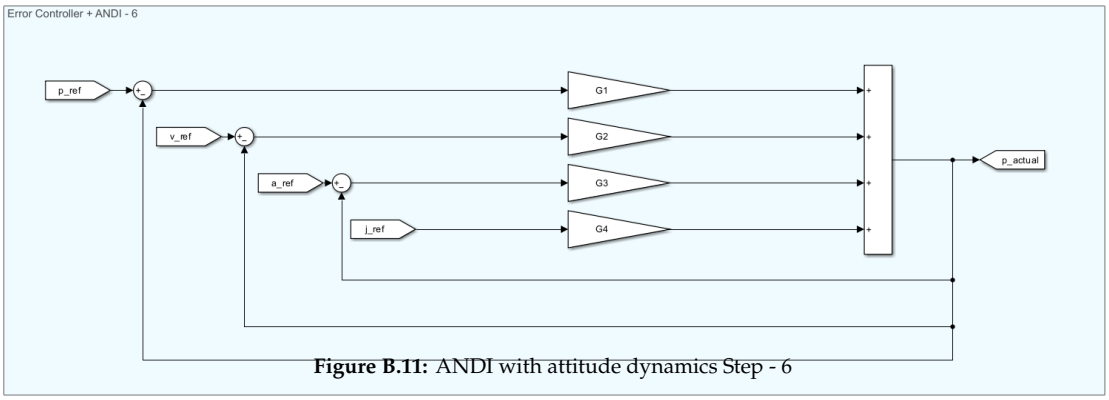


Figure B.11: ANDI with attitude dynamics Step - 6

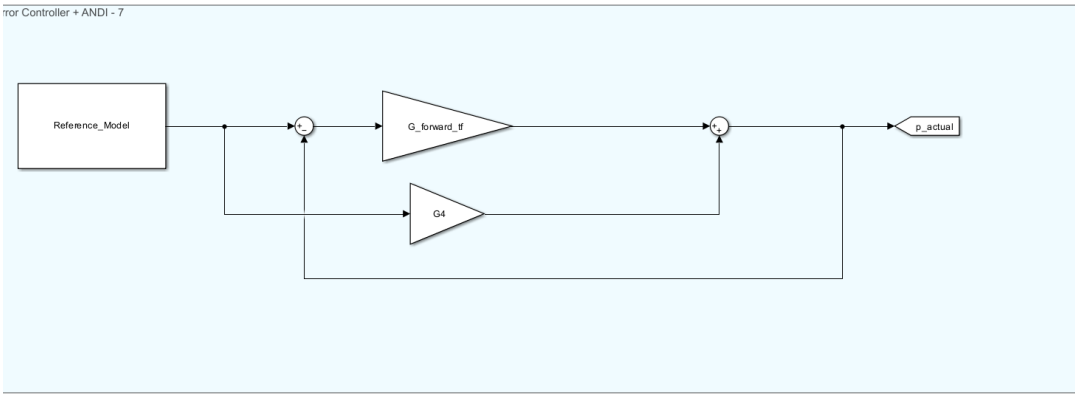


Figure B.12: ANDI with attitude dynamics Step - 7

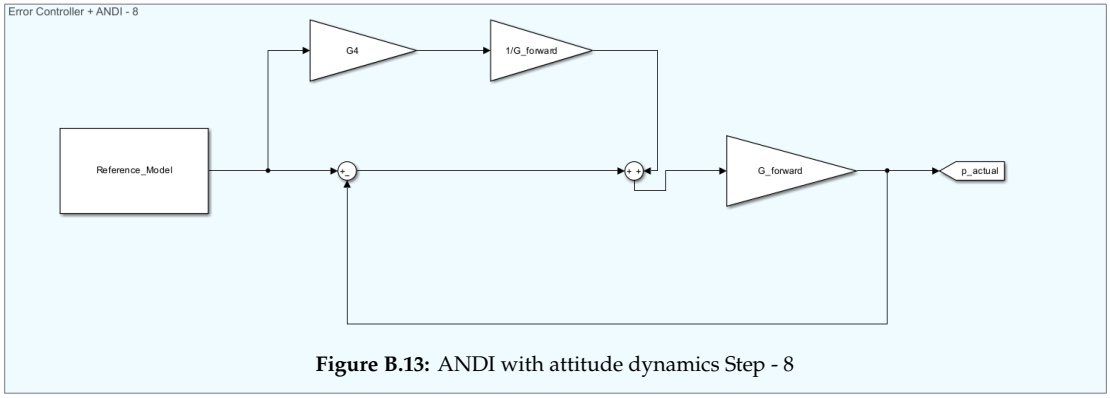


Figure B.13: ANDI with attitude dynamics Step - 8

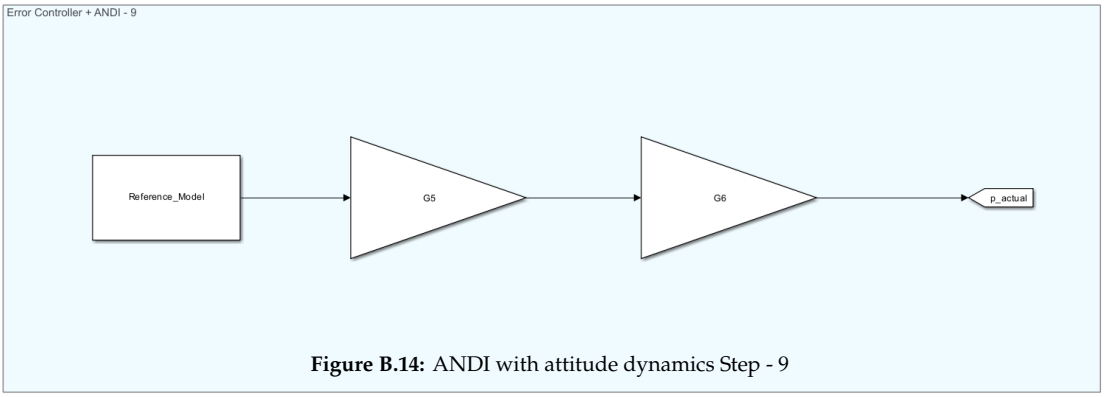


Figure B.14: ANDI with attitude dynamics Step - 9

Matlab Code

```
% define the symbolic variables
syms p1 p2 p3 p_att s k1_rm k2_rm k3_rm k1_e k2_e k3_e k1_att_rm k2_att_rm
k3_att_rm B omega_att omega_n zeta p_1 p_d omega_J_num omega_J_den
```

```
att_rm = (p1*p2*p3)/(s^3 + (p1+p2+p3)*s^2 + (p1*p2 + p1*p3 + p2*p3)*s + p1*p2*p3)
```

att_rm =

$$\frac{p_1 p_2 p_3}{s^3 + (p_1 + p_2 + p_3)\, s^2 + (p_1 p_2 + p_1 p_3 + p_2 p_3)\, s + p_1 p_2 p_3}$$

```
tao    = (p1*p2+p1*p3+p2*p3)/(p1*p2*p3)
```

tao =

$$\frac{p_1 p_2 + p_1 p_3 + p_2 p_3}{p_1 p_2 p_3}$$

```
omega_att = 1/tao
```

omega_att =

$$\frac{p_1 p_2 p_3}{p_1 p_2 + p_1 p_3 + p_2 p_3}$$

```
inv_andi = 1/(B*omega_att)
```

inv_andi =

$$\frac{p_1 p_2 + p_1 p_3 + p_2 p_3}{B\, p_1 p_2 p_3}$$

```
datt2att = collect(simplify(att_rm/(1-att_rm)),s)
```

datt2att =

$$\frac{p_1 p_2 p_3}{s^3 + (p_1 + p_2 + p_3)\, s^2 + (p_1 p_2 + p_1 p_3 + p_2 p_3)\, s}$$

```
pos_controller = collect(simplify(inv_andi * datt2att * B),s)
```

pos_controller =

$$\frac{p_1 p_2 + p_1 p_3 + p_2 p_3}{s^3 + (p_1 + p_2 + p_3)\, s^2 + (p_1 p_2 + p_1 p_3 + p_2 p_3)\, s}$$

```
pos_ec_forward = collect(simplify(pos_controller / s^2),s)
```

pos_ec_forward =

$$\frac{p_1 p_2 + p_1 p_3 + p_2 p_3}{s^5 + (p_1 + p_2 + p_3)\, s^4 + (p_1 p_2 + p_1 p_3 + p_2 p_3)\, s^3}$$

```
G1 = collect(simplify(k1_e * pos_ec_forward),s)
```

G1 =

$$\frac{k_{1,e} p_1 p_2 + k_{1,e} p_1 p_3 + k_{1,e} p_2 p_3}{s^5 + (p_1 + p_2 + p_3)\, s^4 + (p_1 p_2 + p_1 p_3 + p_2 p_3)\, s^3}$$

```
G2 = collect(simplify(k2_e * s * pos_ec_forward),s)
```

G2 =

$$\frac{k_{2,e} p_1 p_2 + k_{2,e} p_1 p_3 + k_{2,e} p_2 p_3}{s^4 + (p_1 + p_2 + p_3)\, s^3 + (p_1 p_2 + p_1 p_3 + p_2 p_3)\, s^2}$$

```
G3 = collect(simplify(k3_e * s^2 * pos_ec_forward),s)
```

G3 =

$$\frac{k_{3,e} p_1 p_2 + k_{3,e} p_1 p_3 + k_{3,e} p_2 p_3}{s^3 + (p_1 + p_2 + p_3)\, s^2 + (p_1 p_2 + p_1 p_3 + p_2 p_3)\, s}$$

```
G4 = collect(simplify(s^3 * pos_ec_forward),s)
```

G4 =

$$\frac{p_1 p_2 + p_1 p_3 + p_2 p_3}{s^2 + (p_1 + p_2 + p_3)\, s + p_1 p_2 + p_1 p_3 + p_2 p_3}$$

```
G_forward = collect(simplify(G1+G2+G3),s)
```

G_forward =

$$\frac{(k_{3,e} p_1 p_2 + k_{3,e} p_1 p_3 + k_{3,e} p_2 p_3)\, s^2 + (k_{2,e} p_1 p_2 + k_{2,e} p_1 p_3 + k_{2,e} p_2 p_3)\, s + k_{1,e} p_1 p_2 + k_{1,e} p_1 p_3 + k_{1,e} p_2 p_3}{s^5 + (p_1 + p_2 + p_3)\, s^4 + (p_1 p_2 + p_1 p_3 + p_2 p_3)\, s^3}$$

```
G5 = collect(simplify((1+G4/G_forward)),s)
```

G5 =

$$\frac{s^3 + k_{3,e}\, s^2 + k_{2,e}\, s + k_{1,e}}{k_{3,e}\, s^2 + k_{2,e}\, s + k_{1,e}}$$

```
G6 = collect(simplify(G_forward/(1+G_forward)),s)
```

G6 =

$$\frac{\sigma_1 + \sigma_2 + k_{1,e}\,p_1\,p_2 + k_{1,e}\,p_1\,p_3 + k_{1,e}\,p_2\,p_3}{s^5 + (p_1 + p_2 + p_3)\,s^4 + (p_1\,p_2 + p_1\,p_3 + p_2\,p_3)\,s^3 + \sigma_1 + \sigma_2 + k_{1,e}\,p_1\,p_2 + k_{1,e}\,p_1\,p_3 + k_{1,e}\,p_2\,p_3}$$

where

$$\sigma_1 = (k_{3,e}\,p_1\,p_2 + k_{3,e}\,p_1\,p_3 + k_{3,e}\,p_2\,p_3)\,s^2$$

$$\sigma_2 = (k_{2,e}\,p_1\,p_2 + k_{2,e}\,p_1\,p_3 + k_{2,e}\,p_2\,p_3)\,s$$

```
pos_ec_andi = collect(simplify(G5*G6),s)
```

pos_ec_andi =

$$\frac{(p_1\,p_2 + p_1\,p_3 + p_2\,p_3)\,s^3 + \sigma_1 + \sigma_2 + k_{1,e}\,p_1\,p_2 + k_{1,e}\,p_1\,p_3 + k_{1,e}\,p_2\,p_3}{s^5 + (p_1 + p_2 + p_3)\,s^4 + (p_1\,p_2 + p_1\,p_3 + p_2\,p_3)\,s^3 + \sigma_1 + \sigma_2 + k_{1,e}\,p_1\,p_2 + k_{1,e}\,p_1\,p_3 + k_{1,e}\,p_2\,p_3}$$

where

$$\sigma_1 = (k_{3,e}\,p_1\,p_2 + k_{3,e}\,p_1\,p_3 + k_{3,e}\,p_2\,p_3)\,s^2$$

$$\sigma_2 = (k_{2,e}\,p_1\,p_2 + k_{2,e}\,p_1\,p_3 + k_{2,e}\,p_2\,p_3)\,s$$

```
% aref to pos actual
a_ec_andi =  collect(simplify(pos_ec_andi/s^2),s)
```

a_ec_andi =

$$\frac{\sigma_3\,s^3 + \sigma_1\,s^2 + \sigma_2\,s + k_{1,e}\,p_1\,p_2 + k_{1,e}\,p_1\,p_3 + k_{1,e}\,p_2\,p_3}{s^7 + (p_1 + p_2 + p_3)\,s^6 + \sigma_3\,s^5 + \sigma_1\,s^4 + \sigma_2\,s^3 + (k_{1,e}\,p_1\,p_2 + k_{1,e}\,p_1\,p_3 + k_{1,e}\,p_2\,p_3)\,s^2}$$

where

$$\sigma_1 = k_{3,e}\,p_1\,p_2 + k_{3,e}\,p_1\,p_3 + k_{3,e}\,p_2\,p_3$$

$$\sigma_2 = k_{2,e}\,p_1\,p_2 + k_{2,e}\,p_1\,p_3 + k_{2,e}\,p_2\,p_3$$

$$\sigma_3 = p_1\,p_2 + p_1\,p_3 + p_2\,p_3$$

```
% ades to aref
a__rm = k3_rm/(s+k3_rm)
```

a__rm =

$$\frac{k_{3,\mathrm{rm}}}{k_{3,\mathrm{rm}}+s}$$

```
% ades to p actual
last_tf = collect(simplify(a_ec_andi*a__rm),s)
```

```
% aref to a actual
a_ec_andi = collect(simplify(pos_ec_andi),s)
```

```
a_ec_andi =
```

$$\frac{(p_1 p_2 + p_1 p_3 + p_2 p_3)\, s^3 + \sigma_1 + \sigma_2 + k_{1,e} p_1 p_2 + k_{1,e} p_1 p_3 + k_{1,e} p_2 p_3}{s^5 + (p_1 + p_2 + p_3)\, s^4 + (p_1 p_2 + p_1 p_3 + p_2 p_3)\, s^3 + \sigma_1 + \sigma_2 + k_{1,e} p_1 p_2 + k_{1,e} p_1 p_3 + k_{1,e} p_2 p_3}$$

where

$$\sigma_1 = (k_{3,e} p_1 p_2 + k_{3,e} p_1 p_3 + k_{3,e} p_2 p_3)\, s^2$$

$$\sigma_2 = (k_{2,e} p_1 p_2 + k_{2,e} p_1 p_3 + k_{2,e} p_2 p_3)\, s$$

C

Evaluation of the reward combinations for the VSQP

The evaluation of reward functions for the VSQP is given in Chapter 3 in terms of performance metrics. The analysis has showed that the form given in Eq. C.1 provides the most effective results for smooth landings. Reward combinations used in the study are labelled and given in Table C.1.

$$
\begin{aligned}
R_{final} = & \left|p_{r_{\text{old}_v}}\right| - \left|p_{r_{\text{new}_v}}\right| + 2\left\|\mathbf{p}_{r_{\text{old}_h}}\right\| - 2\left\|\mathbf{p}_{r_{\text{new}_h}}\right\| \\
& + \left(v_{r_{\text{old}_\text{v}}} - v_{r_{\text{new}_\text{v}}}\right)\left(\frac{h_t + h}{h_t}\right) \\
& + 2\left(v_{r_{\text{old}_\text{h}}} - v_{r_{\text{new}_\text{h}}}\right)\left(\frac{h_t + h}{h_t}\right) \\
& + k\left|v_{d_{\text{new}_z}}\right|^2 + k\left|v_{p_{\text{new}_z}}\right|^2
\end{aligned}
\tag{C.1}
$$

Table C.1: Reward Coefficients

k:10	**k:25**	**k:50**	**k:75**	**k:100**
R1	R2	R3	R4	R5

Later on, a further analysis was conducted to identify a set of suitable coefficients for the considered landing problem. The analysis has showed that coefficients in the range of 10 to 100 are able to led to successful landings in terms of performance metrics. In this chapter, these reward combinations are tested for three different ship motion characteristics for which the corresponding parameters are given in Table C.2. The purpose is to visualize how the drone responds to varying sets of motion. Additionally, the final form incorporates the collision penalty that penalizes the absolute velocities, meaning that landings are expected to occur when both the drone and the platform exhibit near-zero velocities.

Table C.2: Ship Motion Characteristics for the evaluation

Ship motion	**Mixed frequency** (f_{mix}, Hz)	**Nominal frequency** (f_{mix}, Hz)	**Amplitude** (m)
S1	0	0.05	1
S2	0.2	0	0.5
S3	0	0.2	2

For the given ship characteristics, the landing scenarios are illustrated in Figures C.1, C.3, C.6, C.7, and C.9.

The most significant result obtained from the figures is that the drone, when exposed to sufficient sinusoidal wave-like motions during flight, waits for the next wave to arrive as it happened for the second and third type of ship motions instead of landing at random points with a small velocity. This is significantly important as this behaviour is not fully driven by the reward function. Results also showed that the agent is controlled by the tracking elements of the reward, in cases where the ship motion has less variation. Therefore, for the first type of ship motion, the drone chooses to land with contributions from the position and velocity rewards. It should be noted that even though the agent does not exactly end up at the top, which was never a necessity for the landing, for all the scenarios, the touchdown velocities are lower than the critical value, ensuring a safe landing in the end.

R1

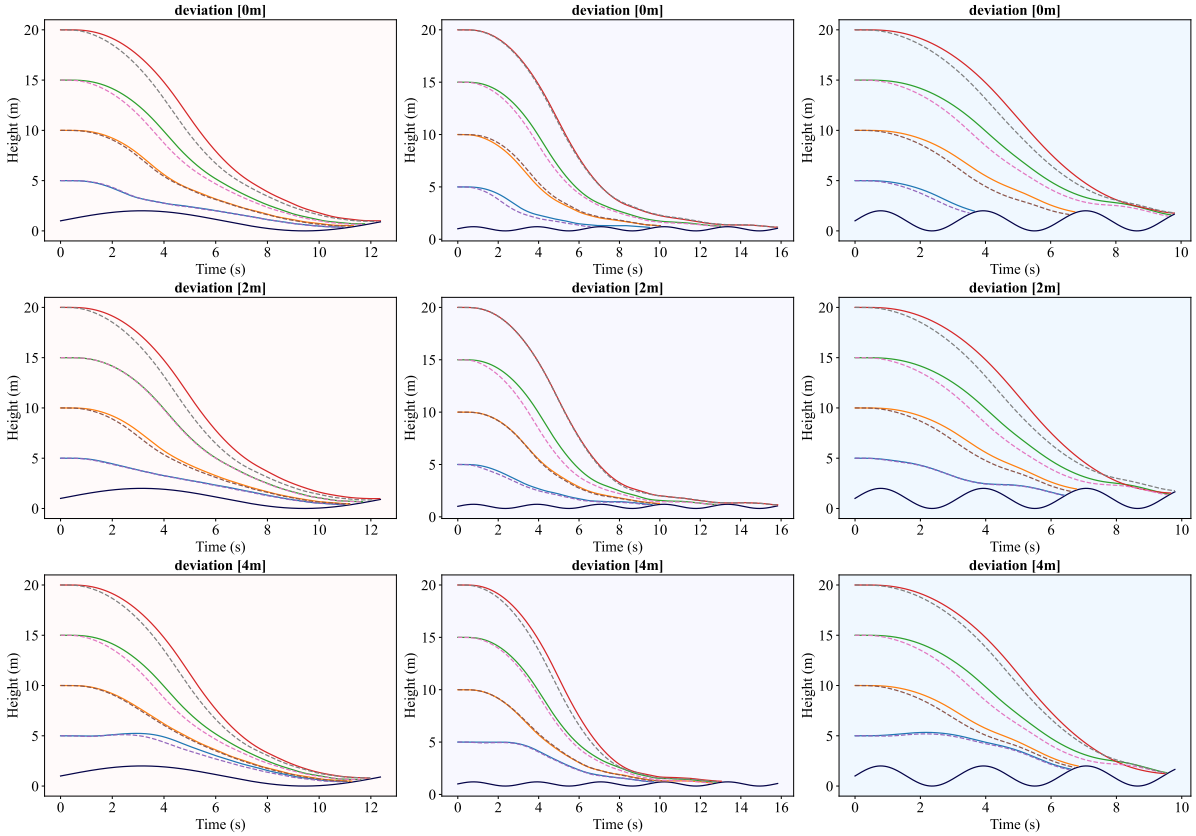


Figure C.1: Reward R1 Evaluation Scenarios

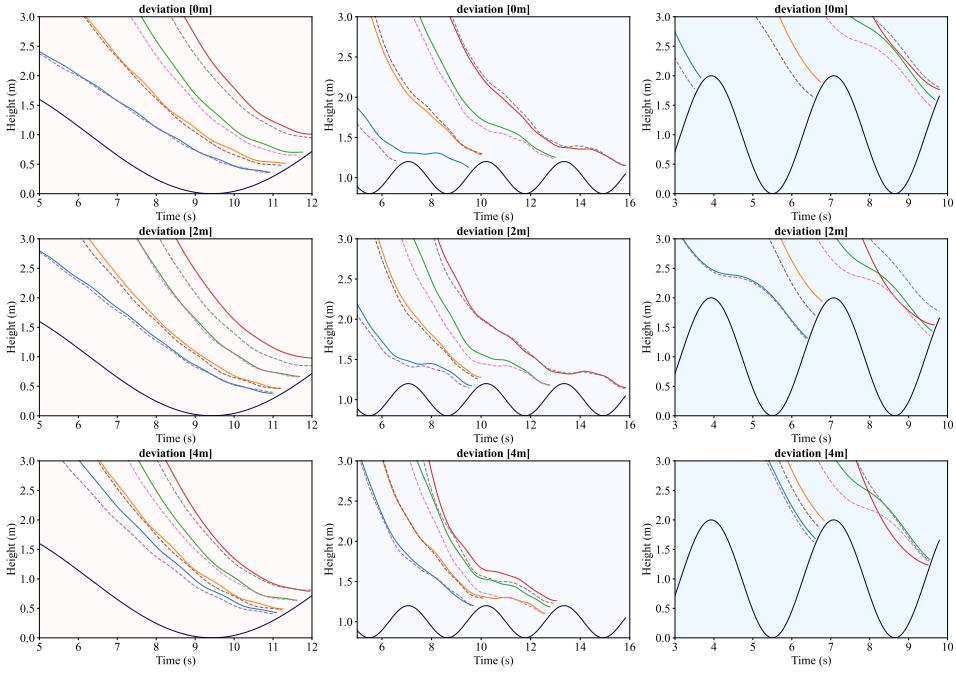


Figure C.2: Reward R1 Evaluation Scenarios - zoom in

R2

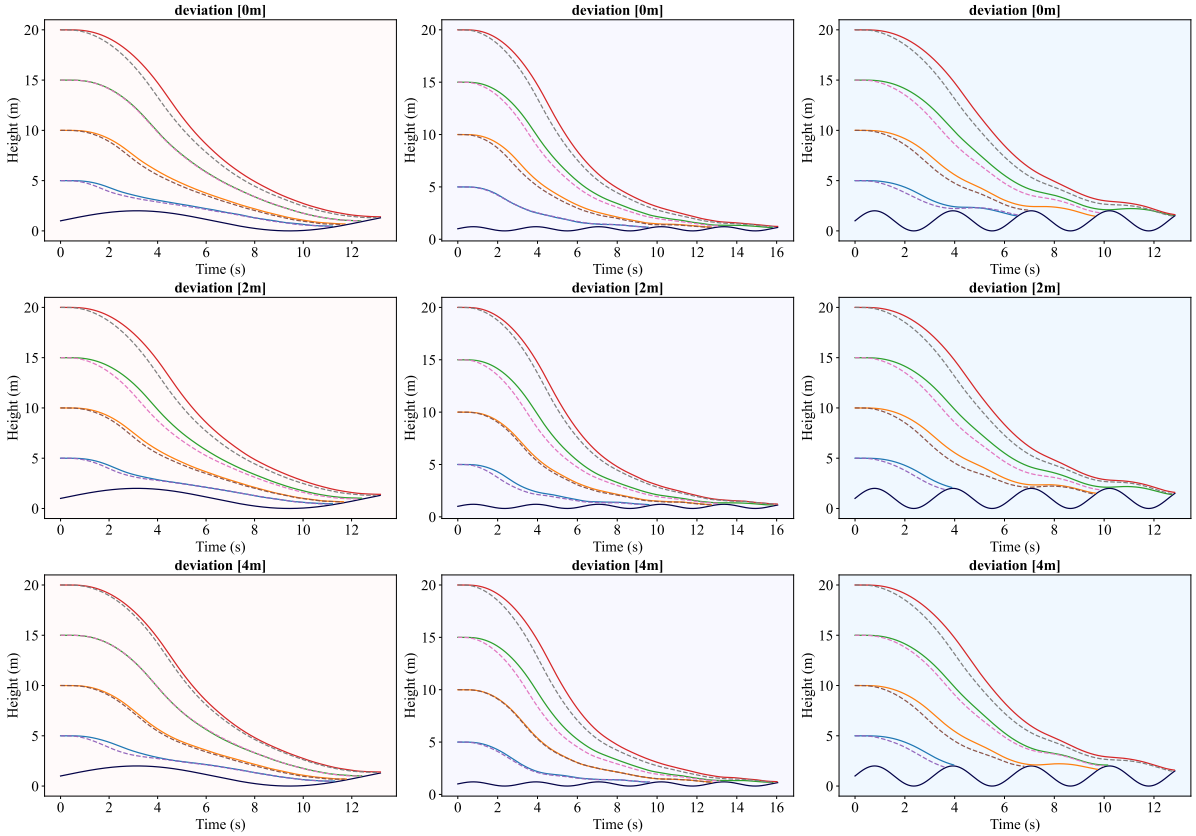


Figure C.3: Reward R2 Evaluation Scenarios

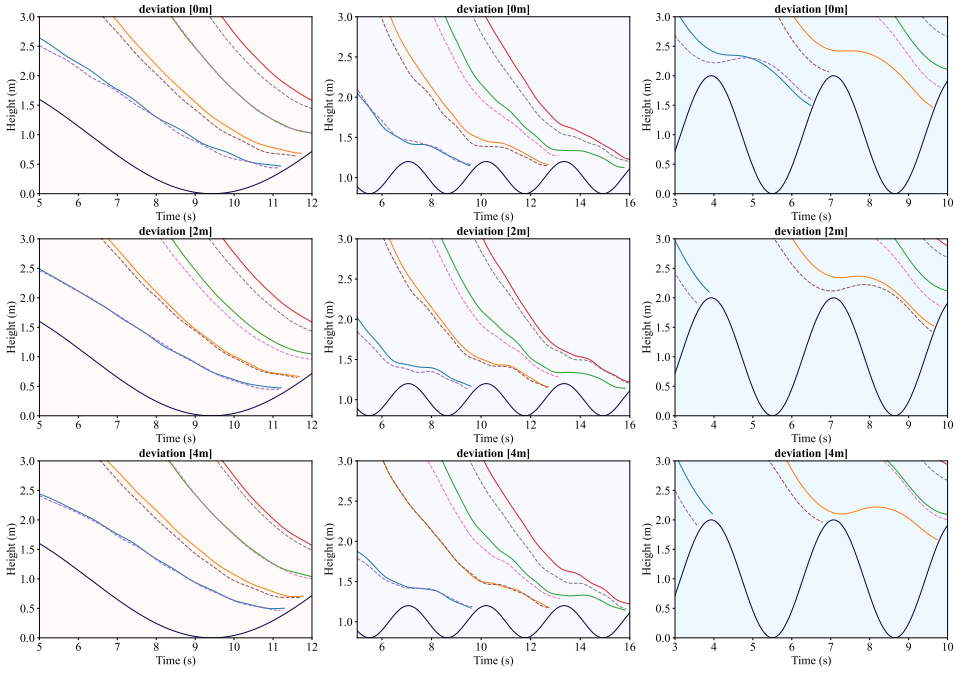


Figure C.4: Reward R2 Evaluation Scenarios - zoom in

R3

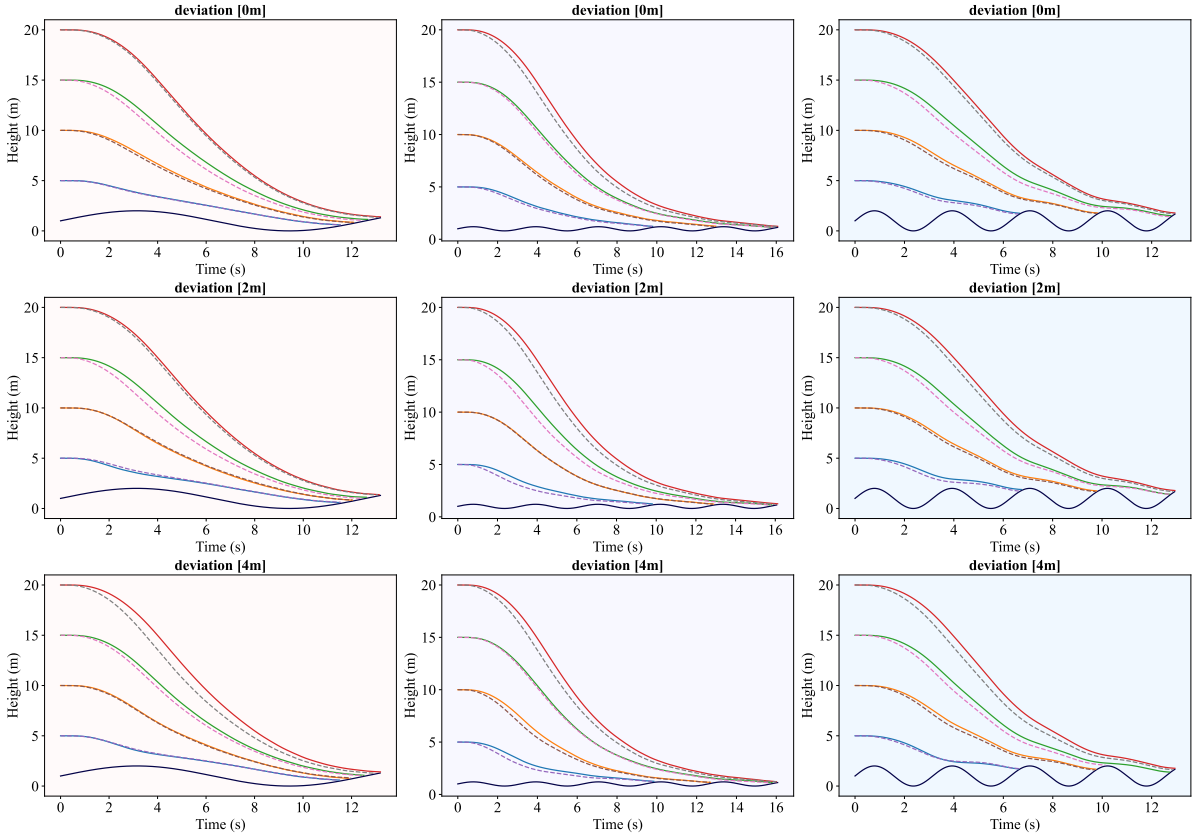


Figure C.5: Reward R3 Evaluation Scenarios

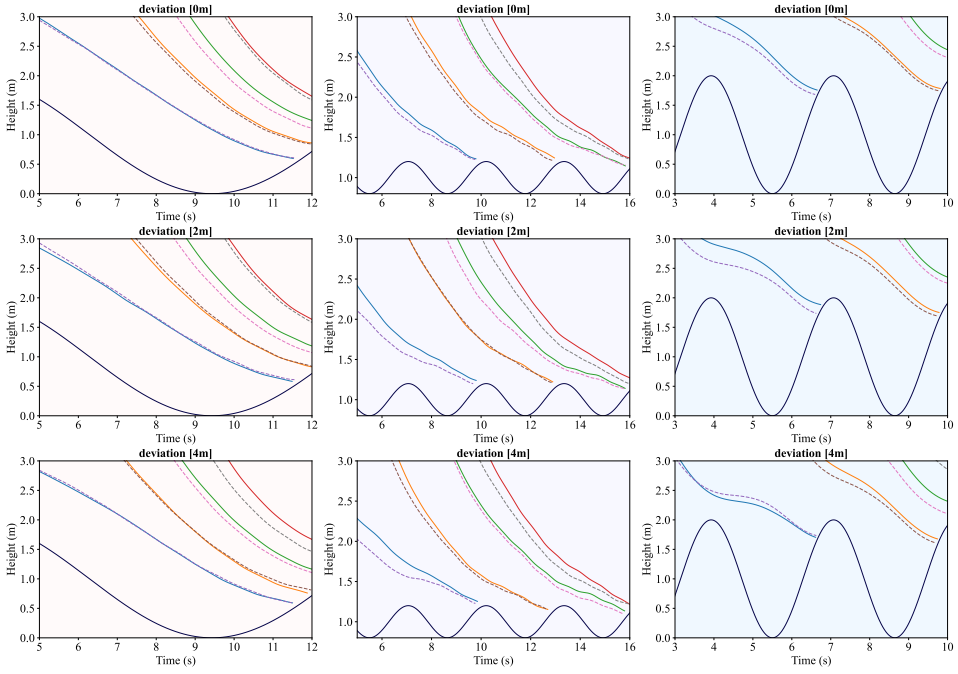


Figure C.6: Reward R3 Evaluation Scenarios - zoom in

R4

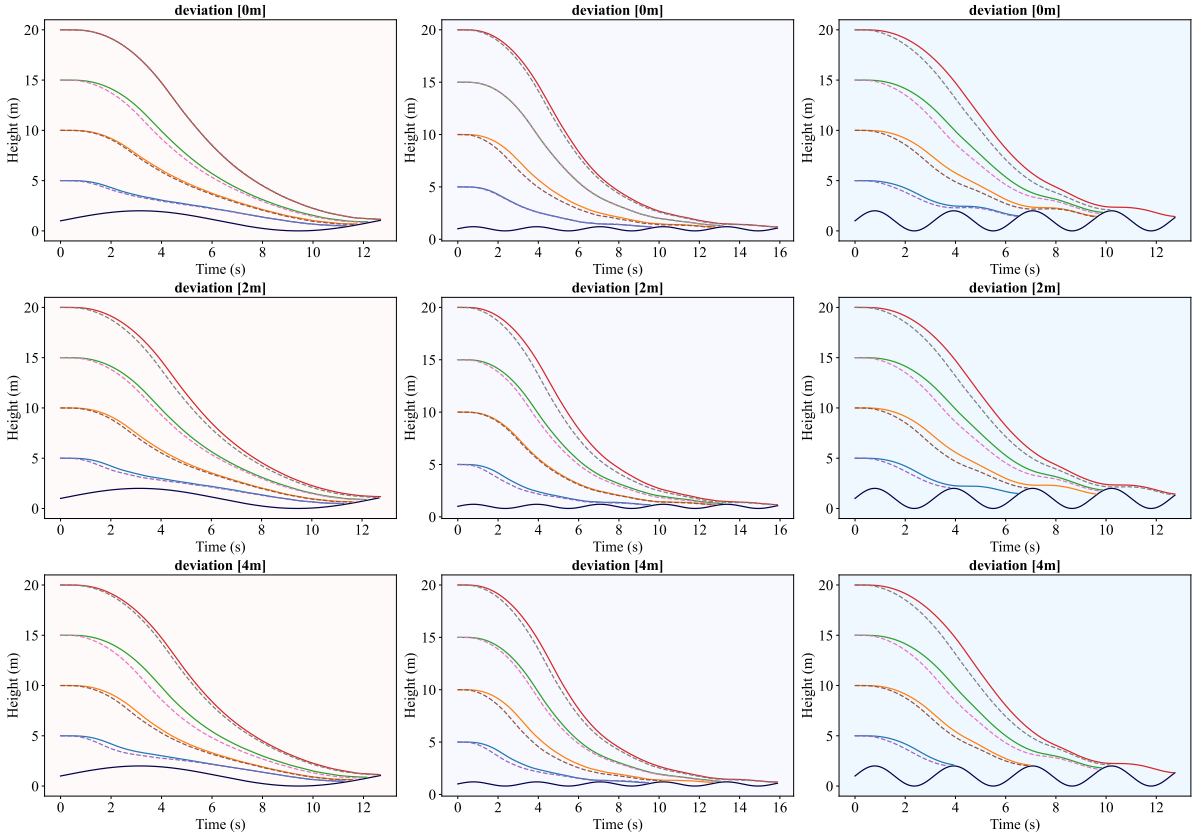


Figure C.7: Reward R4 Evaluation Scenarios

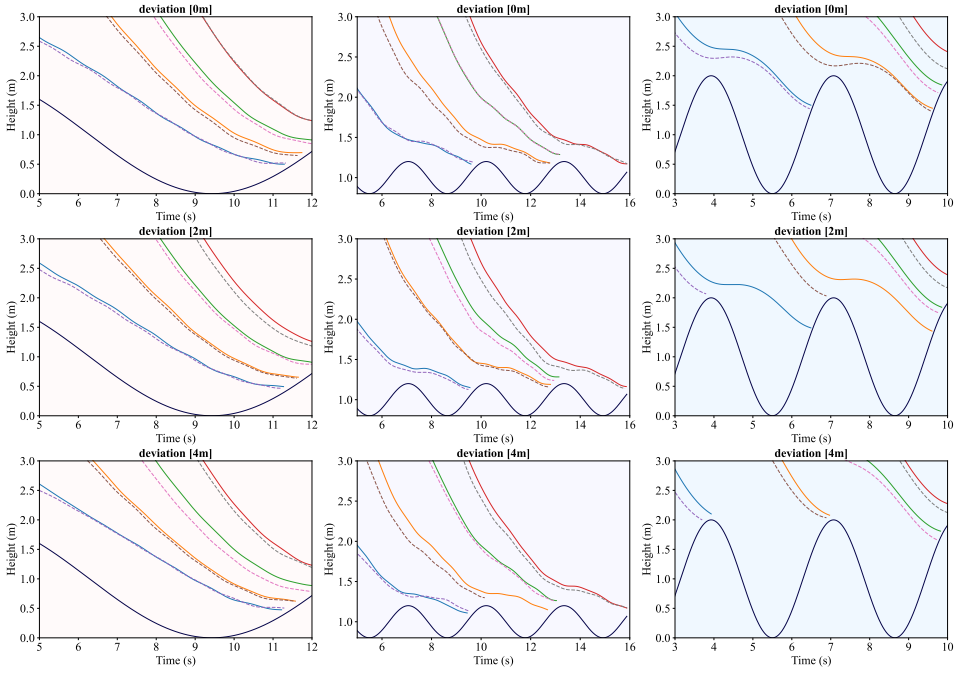


Figure C.8: Reward R4 Evaluation Scenarios - zoom in

R5

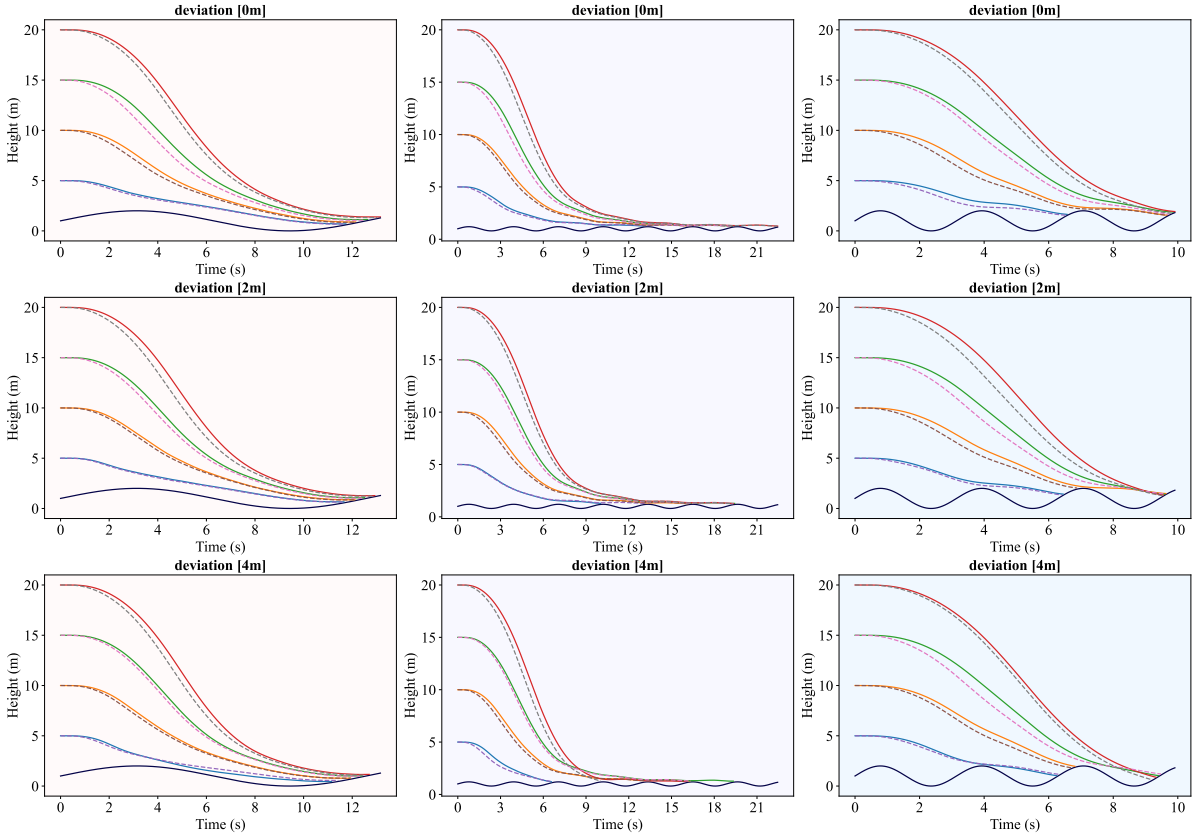


Figure C.9: Reward R5 Evaluation Scenarios

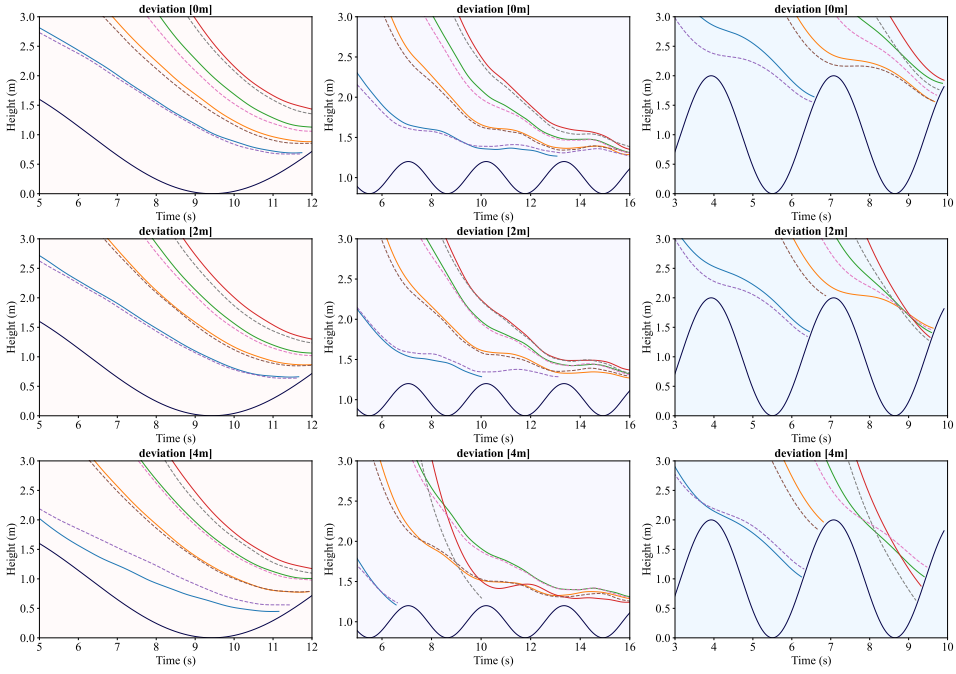


Figure C.10: Reward R5 Evaluation Scenarios - zoom in

D

Validation of the reward combinations for the VSQP

In this chapter, we follow the same validation process outlined in the Scientific Article for the reward combinations presented in the previous section. The trajectories are visualized in Figures D.1, D.3, D.5, D.7, D.9 and the corresponding error plots are provided in Figures D.2, D.4, D.6, D.8, D.10.

In the final reward, the varying coefficients significantly impacted the smoothness and duration of the landing. The higher the collision penalty, the longer the drone took to land, with a slight decrease in touchdown velocity. Results indicated that excessively penalizing the drone (beyond k:100 in this case) prevented landing altogether, scaring away the agent.

Overall, the validation study demonstrated that the given interval of coefficients resulted in different but successful landings. This not only showcased the efficiency of the reinforcement algorithm but also identified a stable region for autonomous landing, which is extremely valuable.

All the code can be accessed through the link `https://github.com/cansuyklmz/thesis_RL_cansu`.

R1

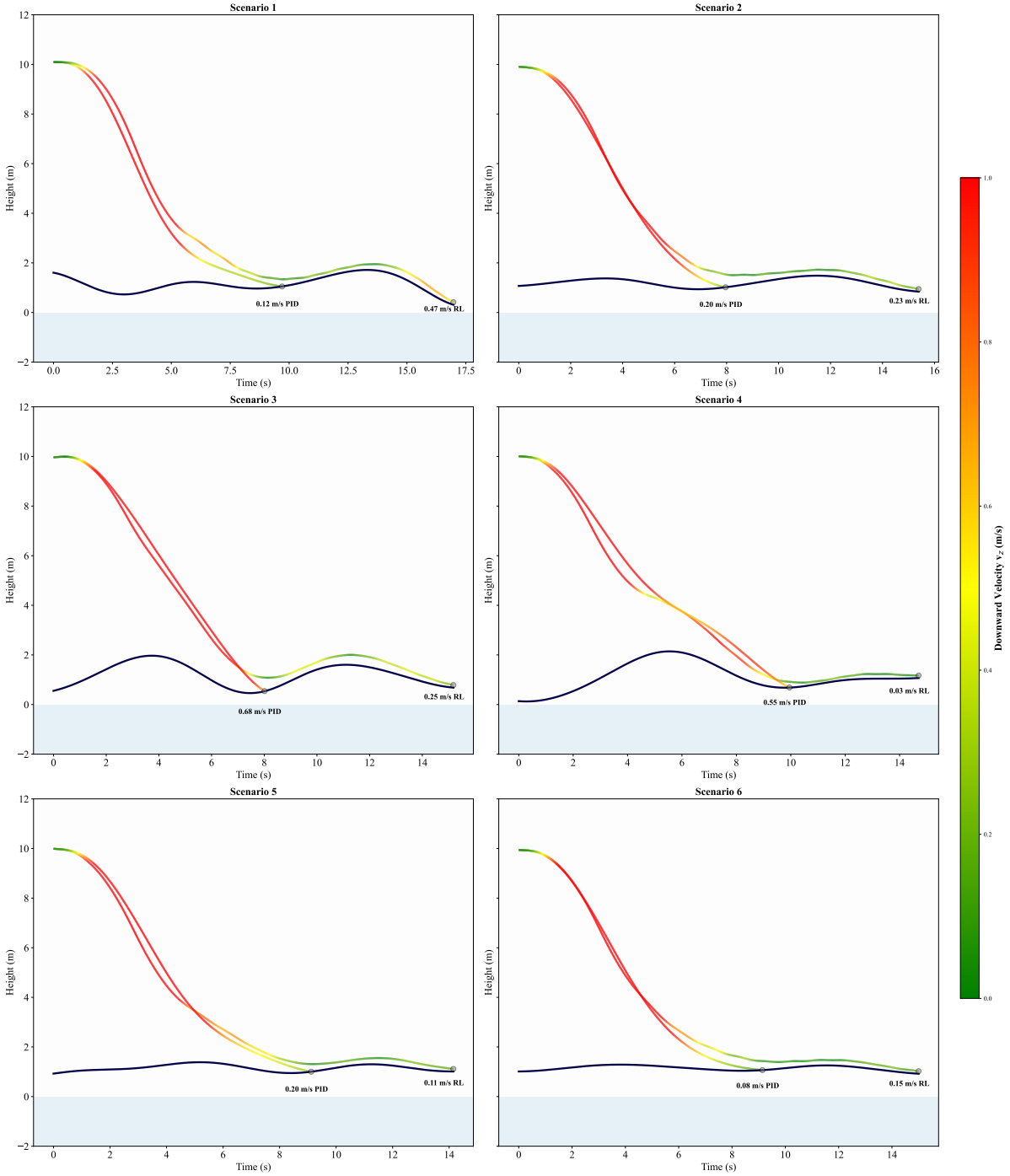


Figure D.1: Reward R1 Ship Validation Scenarios

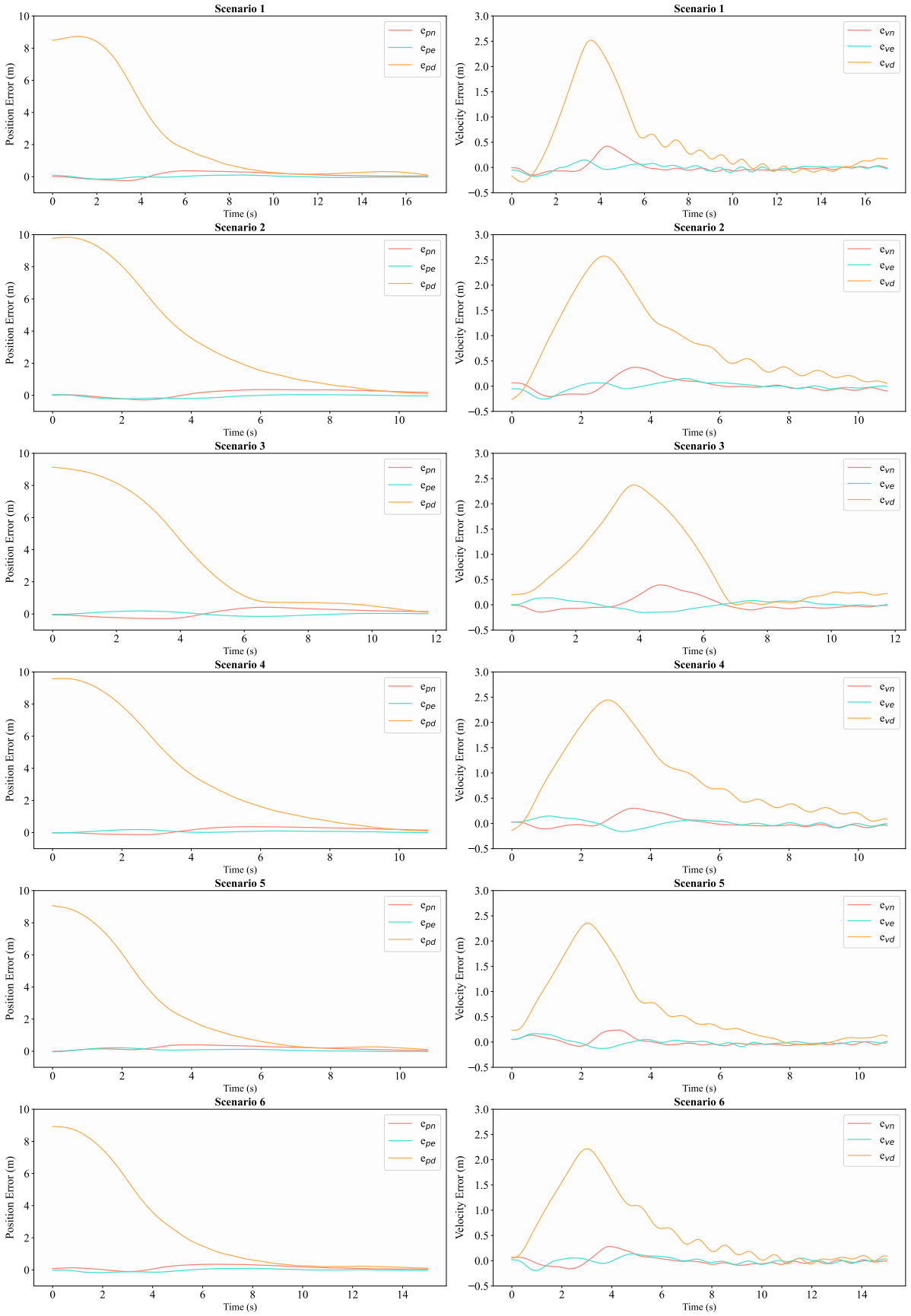


Figure D.2: Error Plots for Reward R1

R2

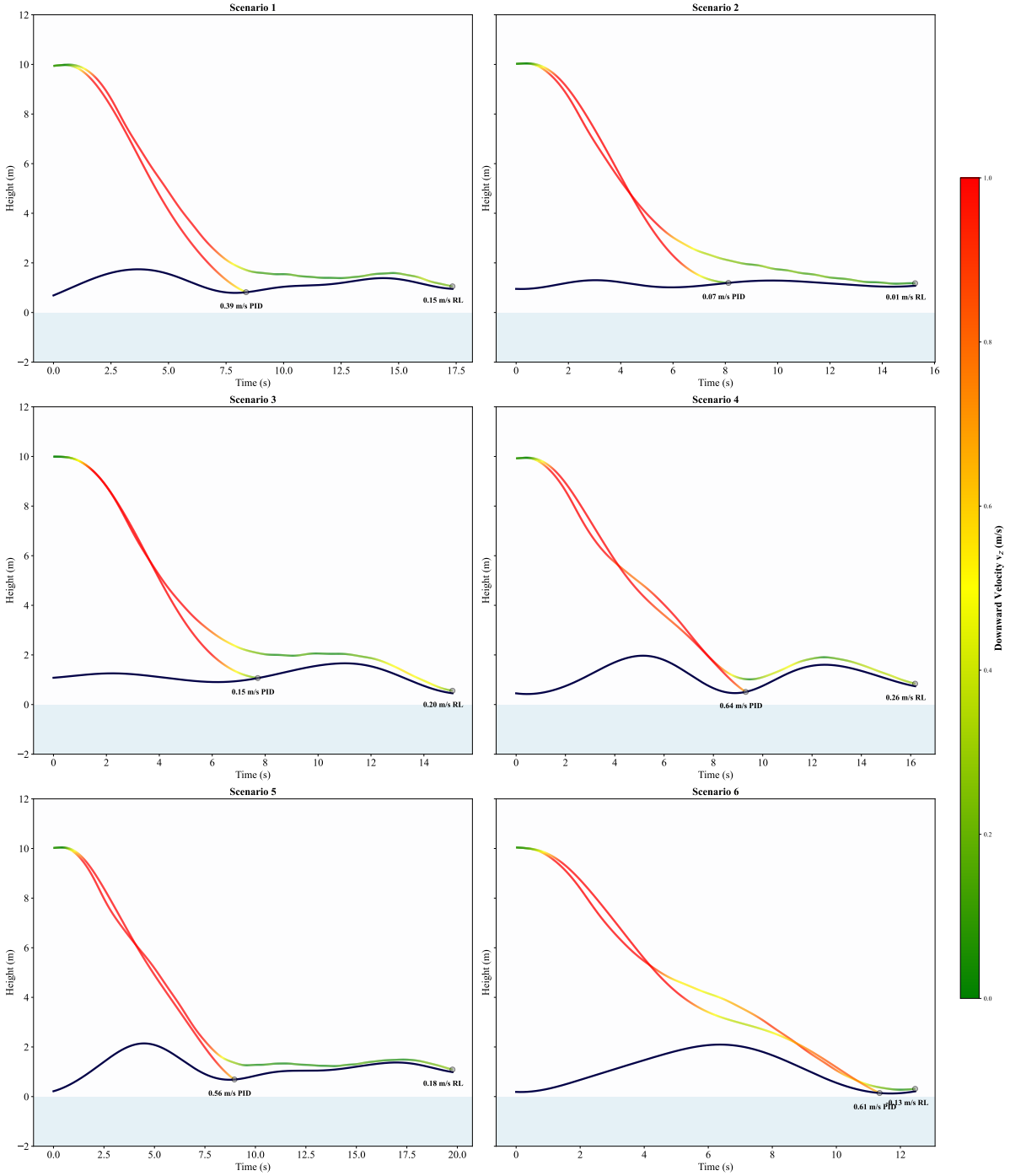


Figure D.3: Reward R2 Ship Validation Scenarios

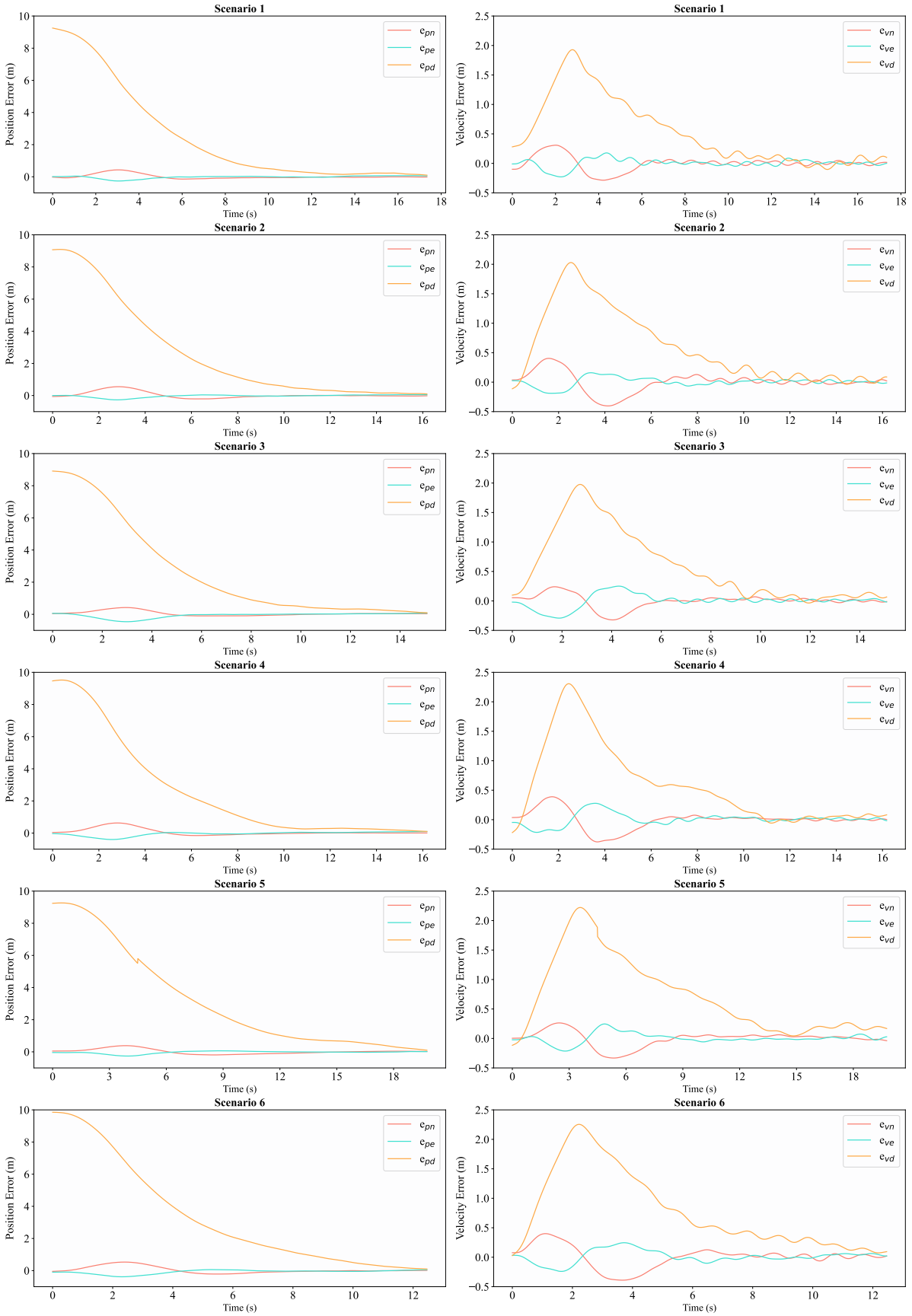


Figure D.4: Error Plots for Reward R2

R3

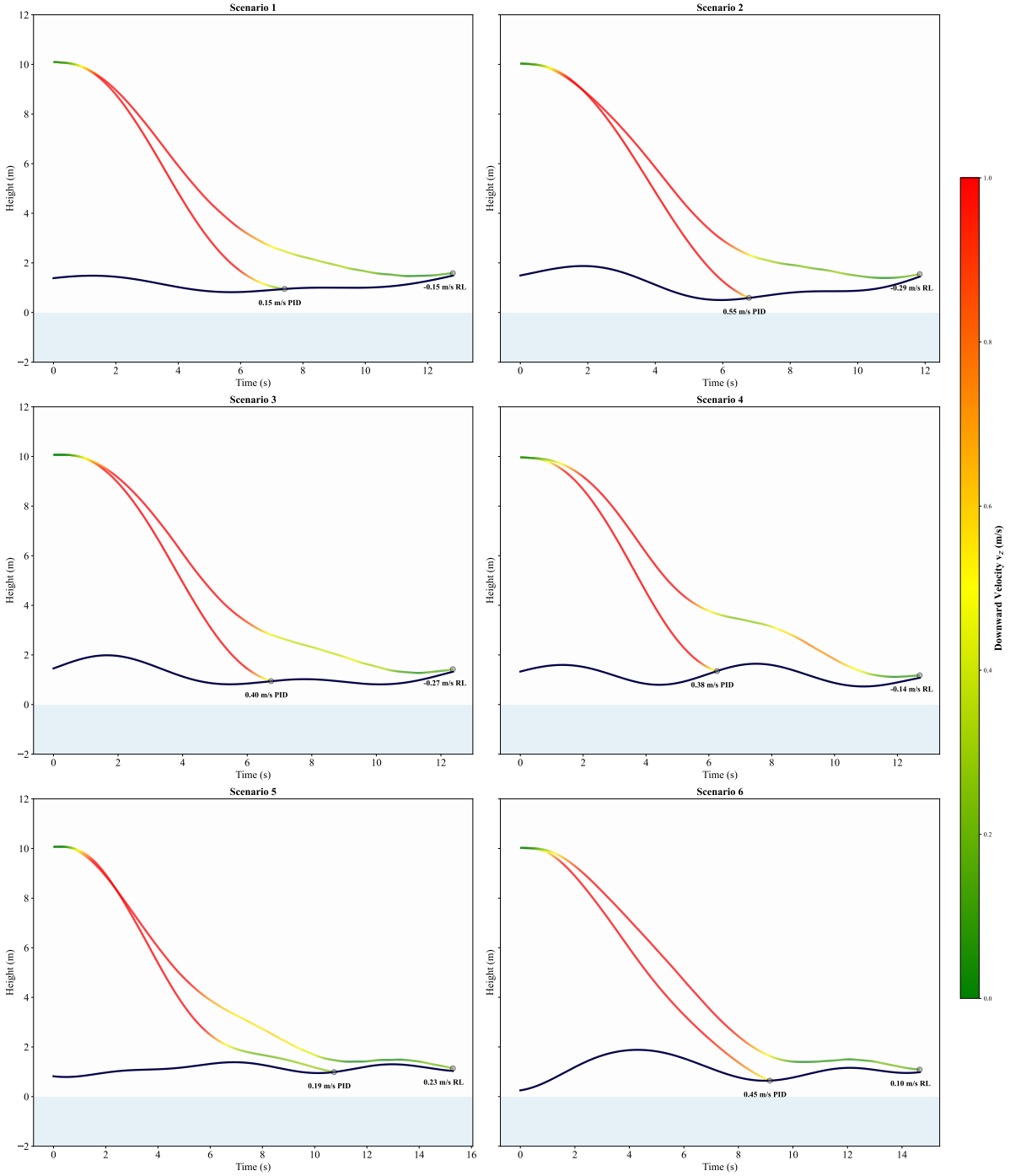


Figure D.5: Reward R3 Ship Validation Scenarios

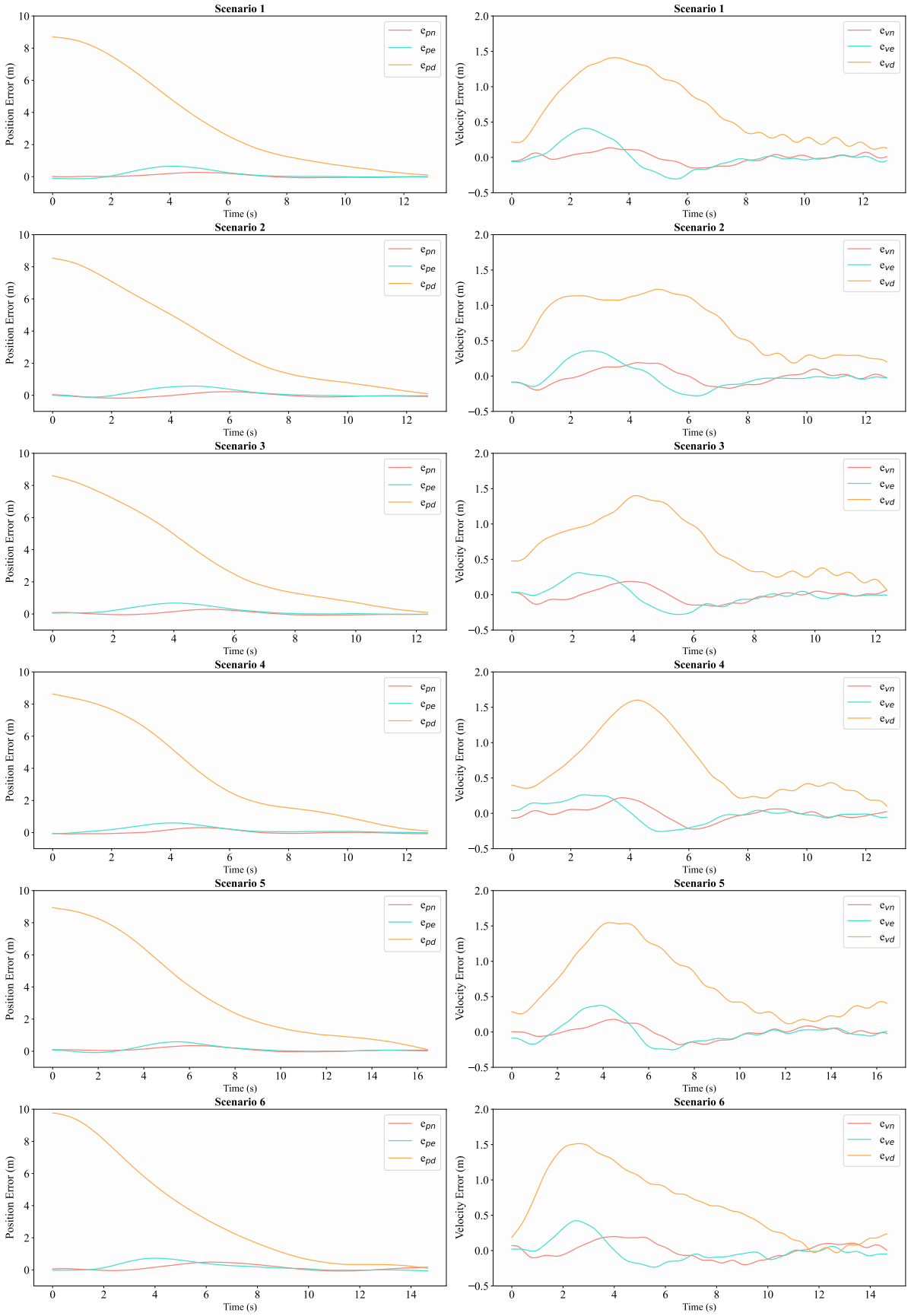


Figure D.6: Error Plots for Reward R3

R4

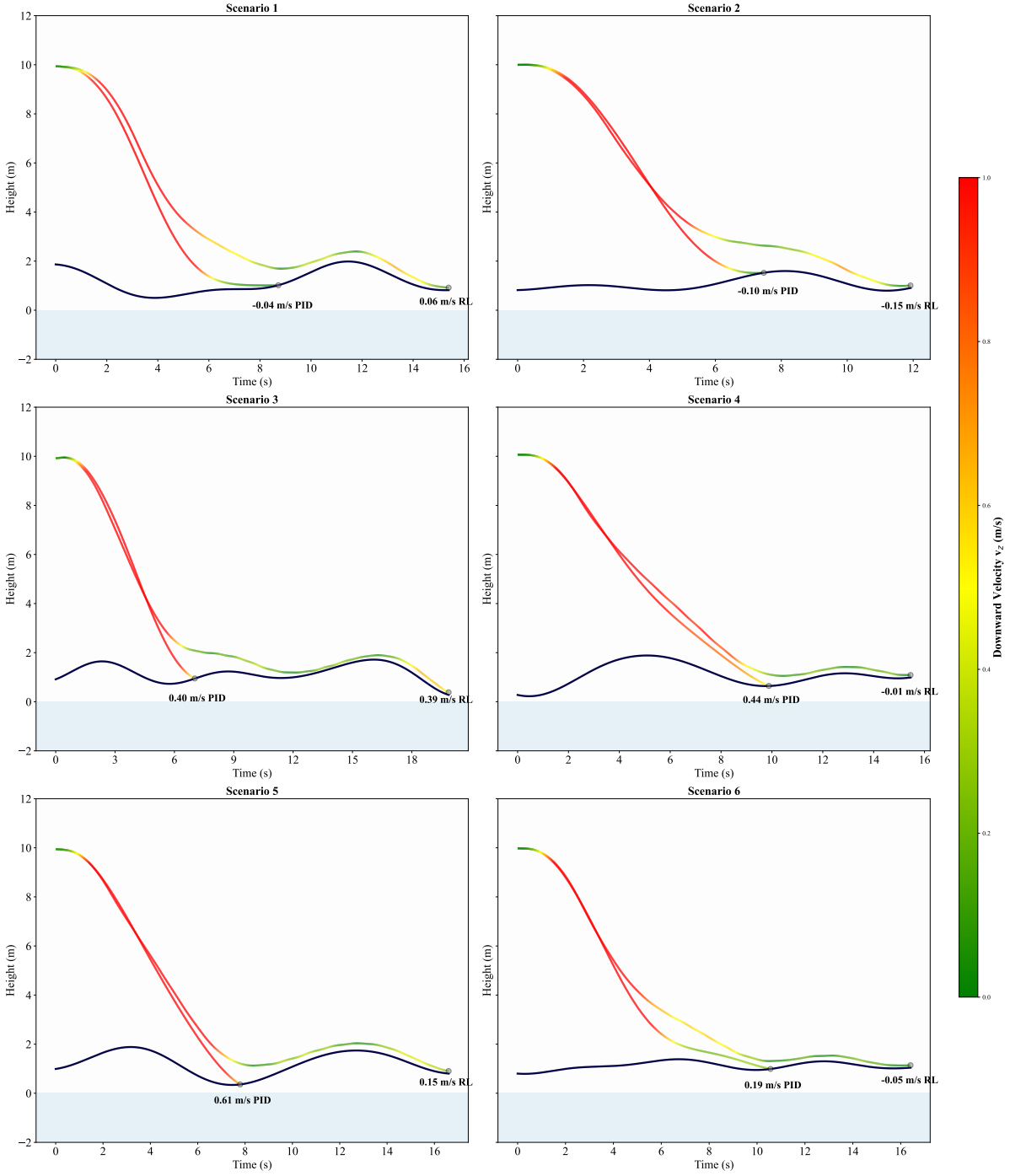


Figure D.7: Reward R4 Ship Validation Scenarios

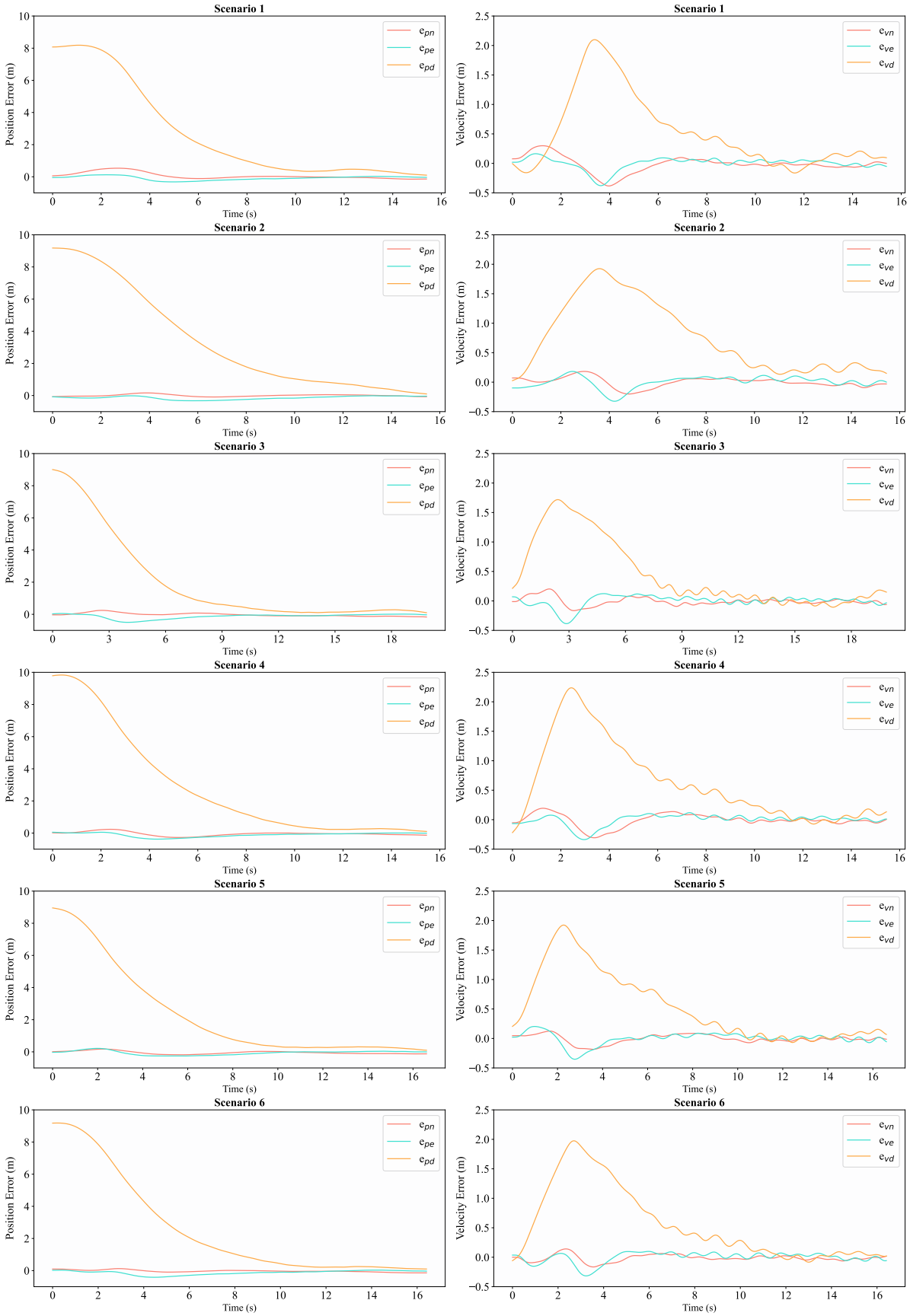


Figure D.8: Error Plots for Reward R4

R5

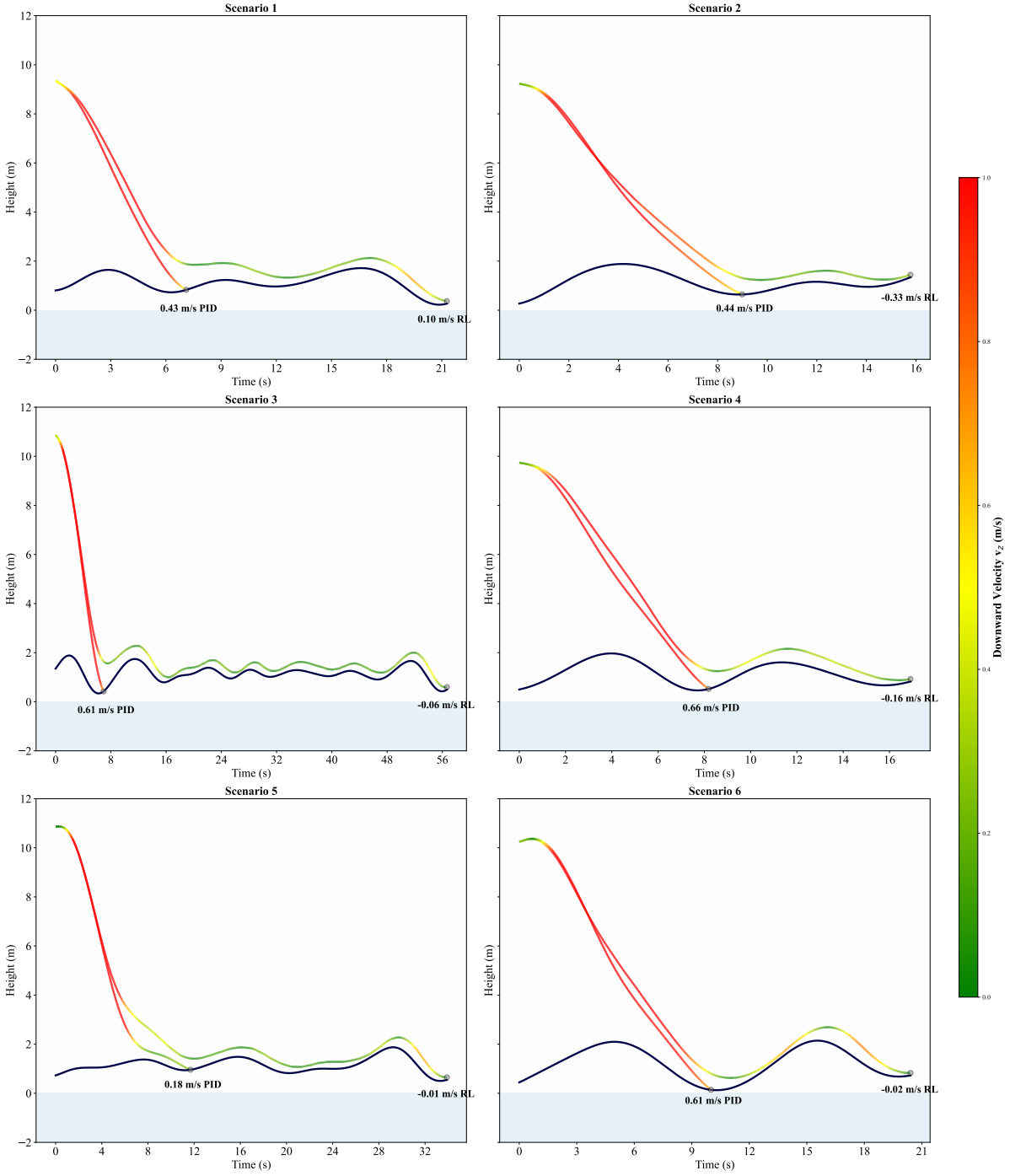


Figure D.9: Reward R5 Ship Validation Scenarios

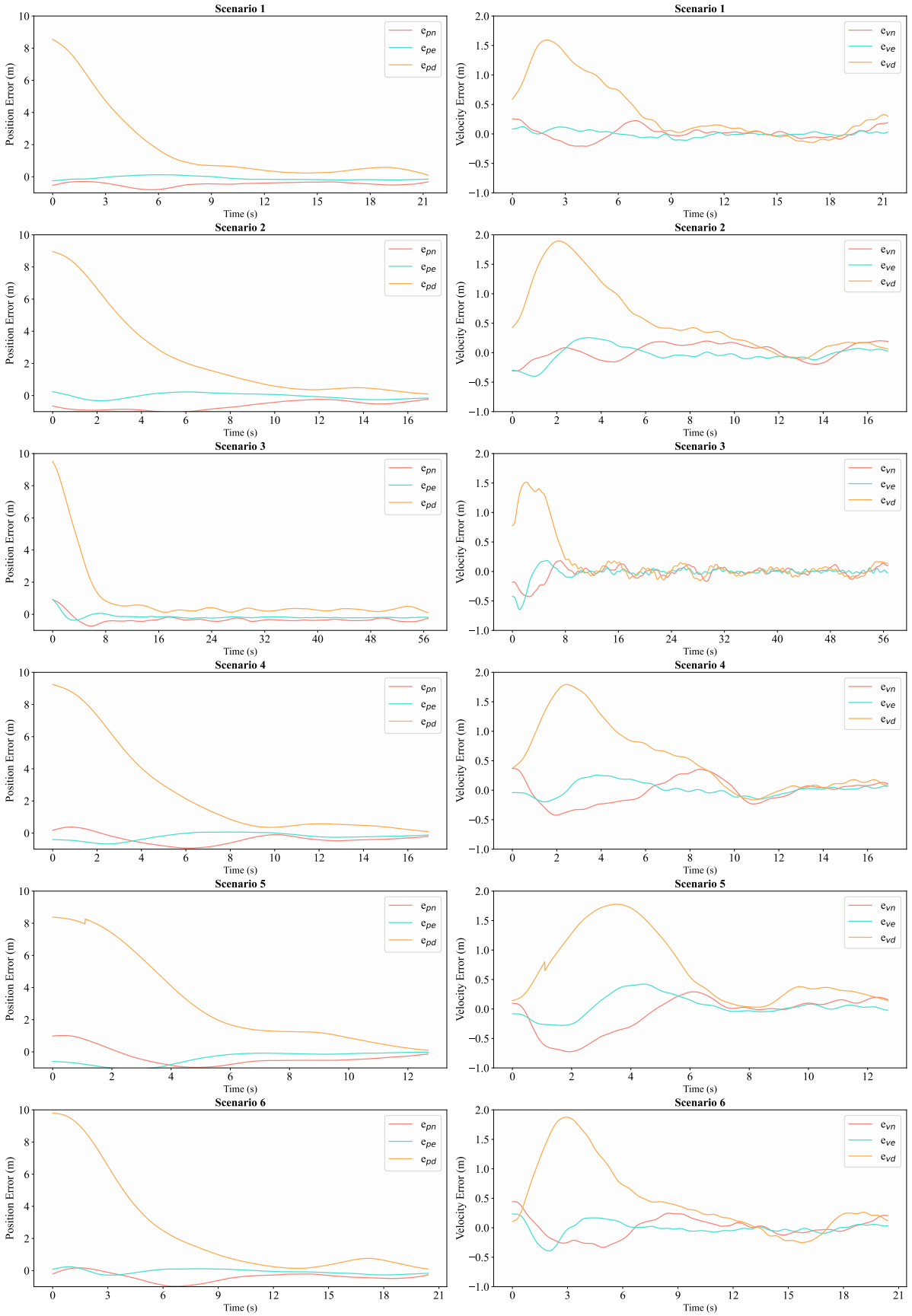


Figure D.10: Error Plots for Reward R5