3-2023

Estimation of Optimal Flight Trajectory in a Total Power Loss Scenario Using Proximal Policy Optimization

Eidahn Eliash

ESTIMATION OF OPTIMAL FLIGHT CONTROL FOR A HELICOPTER IN A TOTAL POWER LOSS SCENARIO USING PROXIMAL POLICY OPTIMIZATION

THESIS

Eidahn Eliash, Captain, Israeli Air Force

AFIT-ENY-MS-23-M-267

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENY-MS-23-M-267

ESTIMATION OF OPTIMAL FLIGHT CONTROL FOR A HELICOPTER IN A TOTAL POWER LOSS SCENARIO USING PROXIMAL POLICY OPTIMIZATION

THESIS

Presented to the Faculty

Department of Aeronautical Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Aeronautical Engineering

Eidahn Eliash, B.S.

Captain, Israeli Air Force

2023

AFIT-ENY-MS-23-M-267

ESTIMATION OF OPTIMAL FLIGHT CONTROL FOR A HELICOPTER IN A TOTAL POWER LOSS SCENARIO USING PROXIMAL POLICY OPTIMIZATION

Eidahn Eliash, B.S.
Captain, Israeli Air Force

Committee Membership:

Dr. Donald Kunz
Chair

Dr. Brett Borghetti, PhD
Member

Lt. Col. David King, PhD
Member

AFIT-ENY-MS-23-M-267

Abstract

In previous work conducted at AFIT, GPOPS-II (a pseudo-spectral optimal control solver) was successfully implemented for finding optimal controls and optimal trajectories of a helicopter in total power loss scenarios. GPOPS-II was deemed a useful tool for generating pre-flight data (such as an HV diagram a.k.a "the death curve), although due to the capabilities of the program, the solutions were provided in time history form only and no explicit control law (eg. u=f(x,t)) could be produced. Furthermore, GPOPS-II was not applicable for real time emergency use, as calculation times were on the order of minutes. Ideally, application in a real time emergency would consist of mapping the algorithms solutions to a visual aid inside the cockpit, possibly presentable on a HUD or MFD. Due to the limitations aforementioned, recent research has been conducted in using deep reinforcement learning algorithms to generate an optimal control law for a helicopter in such a scenario, which ultimately would allow for design of various cockpit pilot aids if proven successful. Such pilot aids are predicted to reduce pilot workload during execution of an emergency powerless landing, and to overall increase the probability of landing safely in such an event. The proposed topic discusses the use of the Proximal Policy Optimization (PPO) algorithm in order to find an optimal control policy for a helicopter in a total power loss emergency landing scenario. The task consists of designing and training an actor neural network and a critic neural network using the PPO algorithm, along with developing the dynamic equations of motion for the helicopter, defining a reward structure and tweaking relevant hyper-parameters. The PPO algorithm has been proven overall feasible for bringing a simplistically modeled helicopter to safe landings over a wide range of initial flight conditions.

Acknowledgments

Firstly, I would like to thank my thesis advisor, Dr. Kunz, for helping me consistently with my thesis research. Dr. Kunz was very patient and understood that progress is not a linear process with respect to time, and guided me when needed. Dr. Kunz also provided me with extensive teachings and education in the fields of dynamics, helicopter performance and helicopter stability and control.

I would also like to thank Dr. Borghetti, Lt. Col. King and Lt. Col Zollars. Dr. Borghetti provided me extensive and qualitative education in the field of neural networks and provided me with important feedback for my thesis research, along with Lt. Col. King. Lt. Col. Zollars provided me with all the necessary information towards understanding functional optimization and control, and more specifically GPOPS-II.

Furthermore, I would like to thank Maj. Yielding and Maj. Merrick from AFRL for their help with questions relating to deep reinforcement learning. I had very little background in the field prior to this thesis, and their tips many times allowed me to surpass obstacles.

I would like to thank my family back home and I would also like to thank the IAF for giving me the opportunity to study abroad at AFIT. Lastly, I would like to give a special thanks to Col. Dvorjetski and Michael J. Paprocki for coordinating the academic collaboration between the IAF and the USAF, and for their support and assistance in all personal matters.

Eidahn Eliash

Table of Contents

Page

Page

List of Figures

List of Tables

ESTIMATION OF OPTIMAL FLIGHT CONTROL FOR A HELICOPTER IN A TOTAL POWER LOSS SCENARIO USING PROXIMAL POLICY OPTIMIZATION

I. Introduction

1.1 Chapter Overview

When a helicopter loses engine power, the pilot is expected to fully reduce the main rotor collective pitch and lower the nose pitch angle if necessary as initial steps of an emergency landing procedure. Given a high enough altitude and velocity at the moment of engine power loss, the pilot must perform a maneuver called autorotation. Autorotation is a maneuver which minimizes the total energy loss from the helicopter during descent, thus preserving it for a soft landing. The ability for a helicopter to land safely depends on a number of factors including helicopter properties (e.g. weight, chassis shape and main rotor drag), environmental properties (such as air density), initial altitude, initial airspeed, flight-path and most importantly - pilot skill. The autorotation maneuver, combined with the psychological distress of an emergency scenario, can require intense pilot workloads. An example of intense pilot workload tasks from the fixed wing world is an F-18 landing on an aircraft carrier in rough sea and weather conditions. In order to mitigate the risks involved in jet carrier landings, engineers have developed Optical Landing (OL) systems - visual aid lighting systems planted on carriers which provide pilots with glide-path information on the terminal phase of the approach.

Previous work conducted has shown that optimal control algorithms, specifically

General-Purpose Optimal Control Software - II (GPOPS-II), were useful for simulating helicopter dynamics in a power loss scenario and could provide insight on flight safety margins and overall emergency landing techniques. On the contrary, optimal control methods lacked the ability to provide instant real-time data.

Due to the drawbacks of optimal control aforementioned, this thesis aims to test the feasibility of using a Deep Reinforcement Learning (DRL) method, specifically Proximal Policy Optimization (PPO), in order to produce an optimal controller for a helicopter in a total power loss scenario, and to be able to regenerate an Height-Velocity (HV) diagram. Achieving the goals above could possibly provide a means of visual aid to a pilot, thus reducing the workload in the cockpit and mitigating flight risks, and could even be implemented in helicopter simulators for training. The visual aid could be for instance a Heads Up Display (HUD) where the recommended flight parameters (such as torque and pitch angle) are presented to the pilot at each point in time. Both implementations could potentially decrease the rate of casualties in such emergency scenarios.

1.2 Autorotation

The term autorotation refers to the state of flight where the main rotor RPM is maintained or increased aerodynamically, without receiving power from the engines. This is typically performed by maintaining a controlled descent at a specific range of airspeeds defined by the helicopter manufacturer. Descending at these airspeeds generates an upward airflow through the rotor, which in turn preserves its RPM. This rotorcraft flight condition is analogous to the glide of a fixed-wing aircraft; lift is still produced but at a much lesser extent than during normal operations with full engine power. Autorotations are typically performed during emergencies after a helicopter experiences a mechanical failure of either the engine, driveshaft, or tail

rotor. Consequently, an autorotation is also commonly performed when there are other emergencies on board, such as a fire or control system malfunction, requiring the helicopter to land quickly [6]. It is important to mention that the autorotation maneuver can only be performed at a high enough initial airspeed and initial velocity due to the time required to reach the typical autorotation airspeeds. In the event of engine power loss at a low airspeed and a low altitude, the autorotation technique cannot be used although a successful landing is still attainable.

An autorotation maneuver can be described in three basic phases, which are discussed in more detail in rotorcraft flight handbooks such as [6] and [7]. The first phase is the entry into autorotation, which occurs immediately after the helicopter loses engine power. Initially, the rotor slows down, since it no longer has the power required to maintain its rotational momentum. The collective pitch is then lowered by the pilot in order to decrease the rotor's drag and lift. Furthermore, the nose pitch of the helicopter is adjusted such that the autorotation airspeed (defined by the manufacturer) is obtained. Performing the above actions causes air to flow upward through the main rotor, which in turn preserves the rotor angular velocity without the need for an external power source. The pilot's reaction time during this initial phase significantly affects the autorotation. The next phase is steady-state autorotation during which the helicopter continues to descend to maintain its rotor RPM, at the defined autorotation airspeed. As the helicopter approaches the ground, the third and final phase of the maneuver is performed. The pilot performs a flare before touchdown in order to bleed off airspeed and to minimize the rate of descent. The flare is performed by increasing the nose pitch of the helicopter which increases the airflow through the main rotor, thereby generating more lift. Finally, the pilot levels the helicopter in order to prevent tail-strike, and gradually increases the main rotor collective pitch in order to allow for a soft and safe touchdown. In turn, the rotor

speed decreases during the flare and the increase in main rotor collective pitch. The general autorotation scheme provides a relative measure for the analysis of artificially generated control inputs to a helicopter during an emergency descent. Figure 1.1 illustrates these phases of an autorotation.

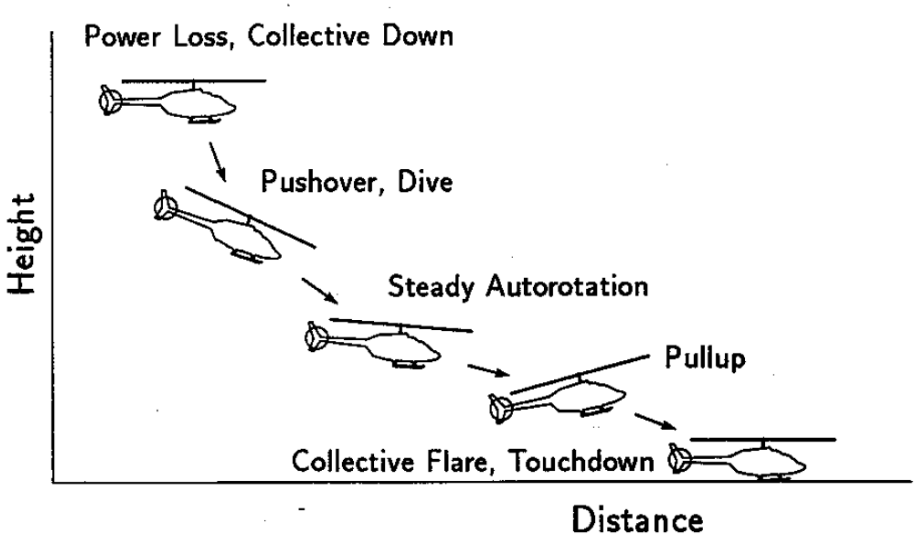


Figure 1.1. A Helicopter During Autorotation [1]

1.3 HV Diagram

The ability for a helicopter to land safely within its structural limits following an autorotation is contingent on the helicopter's initial forward airspeed and height above the ground when engine power is lost. HV diagrams show the combination of these two factors based on the particular helicopter in question, the gross weight, the density altitude and sometimes external configuration. Figure 1.2 shows an example HV diagram for low-speed flight. The shaded portion of the diagram depicts the unsafe flight region in which a safe landing could not occur if engine failure occurs. In this region, the rotor speed does not have enough time to recover after its initial drop to perform a successful flare. Without a successful flare, the helicopter impacts the ground at a damaging velocity. The dividing curve between the safe and unsafe

regions on the HV diagram is referred to as the "deadman's curve" and includes three key, defining points: the low hover point, high hover point, and knee point. The two hover points represent the lowest and highest points on the curve, respectively. Above the upper portion of the curve, the rotor speed can recover enough during the descent to flare and thus land safely. Below the lower portion of the curve, the height of the helicopter above the ground is not high enough for the helicopter to accelerate to unrecoverable speeds, since it will touch down on the ground very quickly after the power loss. Hence, the power loss in this region allows for a safe landing. As forward airspeed increases, the autorotative descent rate decreases [8]. This leads to the knee point where the upper and lower portions of the curve intersect. HV diagrams typically also have a high-speed unsafe region near the ground where the helicopter's high airspeed does not allow for a sufficient reduction in velocity without the tail striking the ground prior to touchdown. These two unsafe regions help create a flight path for pilots to follow during routine takeoffs and landings by displaying flight regions to avoid. Knowledge of the HV diagram for a specific helicopter provided by the manufacturer can be a useful measure for comparison when analyzing new computational methods for predicting helicopter trajectories in a power loss scenario, such as optimal control methods or DRL methods.

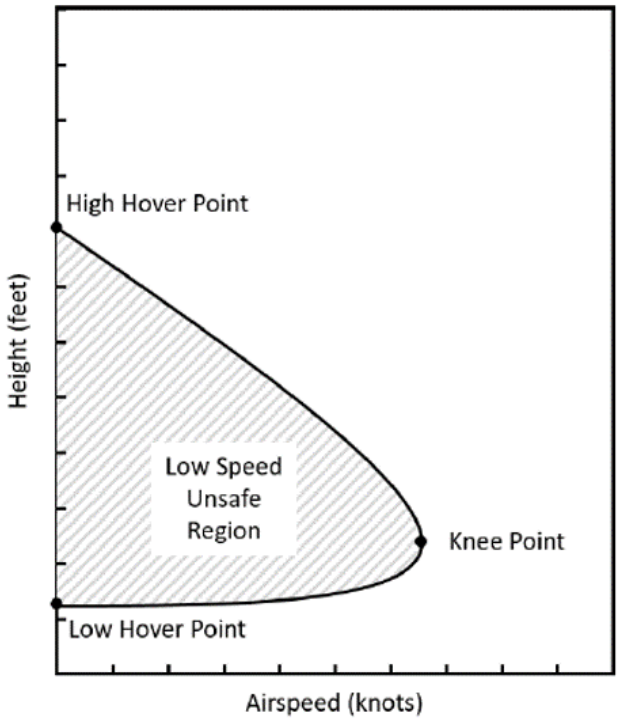


Figure 1.2. Example Height-Velocity Diagram [2]

As described above, HV diagrams inform pilots of which regions of flight would be hazardous if engine failure were to occur there. In addition to having HV diagrams for total engine power loss, multi-engine rotorcraft have additional HV diagrams for One Engine Inoperable (OEI) flight in which only one engine fails. Naturally, the unsafe region is smaller for OEI flight than for total power loss, since there is still some power supplied to the rotor from the remaining engine(s). With multiple engines, the chances of total power loss is significantly reduced. Although a majority of military helicopters are equipped with at least two engines, and the probability of multiple engines failing simultaneously are extremely low, this thesis aims to research the feasibility of using a DRL method for the worst case scenario being total power loss such that all areas of risk are covered.

1.4 Research Objectives

This thesis aims to contribute to research in the field of DRL combined with flight dynamics, specifically with regards to predicting optimal flight controls and trajectories for a helicopter in a total power loss scenario. The objectives of this research are as follows:

- **Primary Objective**: Research feasibility and effectiveness of using DRL methods, specifically the PPO algorithm, to generate an optimal control policy for a helicopter in a total power loss scenario.
- Develop a simplistic mathematical model for helicopter dynamics.
- Analyze the trajectories of a helicopter in a total power loss scenario given a generated optimal control policy.
- Generate an HV diagram for the helicopter.

1.5 Thesis Overview

This chapter has introduced the topics of autorotation and HV diagrams, and has briefly discussed visual pilot aids for workload reduction. Furthermore, this chapter has provided the main objectives of this research. Chapter 2 provides background on prior research into estimating optimal control paths for helicopters in a total power loss scenario, using both pseudo-spectral optimization methods and DRL methods. Furthermore, Chapter 2 provides a buildup to the PPO algorithm, from classic Reinforcement Learning (RL) and Markov Decision Processes to the derivation of policy gradient algorithms. In Chapter 3 the dynamic equations of motion for the helicopter model are derived, and the methodologies for tuning and training the PPO agent are explained. Chapter 4 provides the results of the hyper-parameter selection phase, the training phase and the test phase of the PPO algorithm with the helicopter model environment. Lastly, Chapter 5 states the conclusions from this research and recommendations for future research.

II. Background

2.1 Chapter Overview

This chapter discusses functional optimization and control methods and the previous research conducted using such methods for predicting optimal helicopter flight paths and for generating Height-Velocity (HV) diagrams. Moreover, the drawbacks of functional optimization are presented, along with the proposed solution of replacing functional optimization with Deep Reinforcement Learning (DRL) methods. This chapter also provides an overview of neural networks, which serve as universal function approximators and can be used to approximate optimal control laws over a wide range of mission tasks. In addition, this chapter discusses classic Reinforcement Learning (RL) as a background to more advanced DRL methods. The development and derivations of various continuous state-action DRL algorithms are elaborated, beginning with the classic vanilla policy gradient algorithm and ending at the Proximal Policy Optimization (PPO) algorithm. In addition, a background to reward shaping is discussed, which serves as the feedback element of a reinforcement learning environment. Lastly, this chapter presents similar works in the fields of DRL for helicopters in a total power loss scenario.

2.2 Functional Optimization and Control

Optimal control theory provides a means of determining the control inputs into a system that optimize a desired performance criterion while satisfying physical constraints on the system [9]. One of the first steps in formulating an optimal control problem is to develop a mathematical model of the system being analyzed. These models are commonly represented in state variable form, consisting of a set of state variables (states), which describe specific components of the system, and control vari-

ables (controls), which describe the inputs to the system that drive the states to optimize a performance index. The state and control variables are related by a set of first-order differential equations that describe the dynamics of the system. Finally, a scalar performance index (or objective/cost function) is selected to be either minimized or maximized. The set of controls along with the associated states that minimize the performance index while satisfying the constraints on the system constitute the solution of the optimal control problem.

Given the complex nature of most optimal control problems, optimal control solutions are typically determined using direct numerical methods rather than being solved indirectly. Indirect methods apply the calculus of variations to the optimal control problem, which for complex dynamic environments such as a helicopter in flight can result in many highly non-linear equations which are highly difficult to solve analytically. Furthermore, the equations derived from the calculus of variations contain Lagrange multipliers (namely co-states) which may not have any physical significance, and thus added difficulty can arise when solving for them. This is especially difficult for certain instances where an initial solution guess is required. In contrast, direct (collocation) methods discretize and parameterize the continuous optimal control problem into a Non-Linear Programming (NLP) problem and find the resulting NLP solution [10]. Solving an optimal control problem using direct methods is generally considered more practical due to the fact that they can handle very complex and non-linear state models, whereas indirect methods are generally impractical for complex physical problems.

One type of direct collocation methods are pseudospectral methods. In pseudospectral methods, the optimal control problem is transcribed into a NLP problem using polynomials to parameterize the states and controls, and a Gaussian quadrature to collocate the differential-algebraic equations [11]. Rao et al. developed a Matrix

Laboratory (MATLAB) software called General-Purpose Optimal Control Software - II (GPOPS-II), which utilizes pseudospectral methods and sparse nonlinear programming to solve optimal control problems [12]. GPOPS-II is a popular programming tool used globally and moreover, has been used at The Air Force Institute of Technology, WPAFB (AFIT) to solve various optimal control problems. Although direct method solvers such as GPOPS-II have proven to be powerful tools for calculating optimal trajectories, they cannot generate closed form analytical control laws which indirect methods can do. For instance, the Linear Quadratic Regulator (LQR), a special form of indirect optimal control, always provides a control law in the form of $u = -K * X(t)$.

Hence, if the LQR problem is analytically solvable (which may not be the case for a complex problem), a value for the control law can be calculated given any state vector. On the contrary, direct method solvers such at GPOPS-II can only provide the values of the controls in time history form from the initial time to the final time. Therefore, with regards to optimal flight trajectories of a helicopter during a power loss scenario, using GPOPS-II is a useful tool for simulating an optimal flight trajectory or for generating pre-flight data such as an HV diagram - but is not useful when the objective is to derive a control law as a function of state for real world implementation in real-time situations. Such an implementation could be expressed as a visual pilot aid inside the cockpit of a helicopter, for example a Heads Up Display (HUD) providing the necessary helicopter state and control info to the pilot in command in an engine power loss scenario.

In contrast to optimal control methods which aren't practical for real-time applications, DRL methods can be practical for such applications as they can be used to generate a global control law, defined by a neural network, relating a helicopter's states to the required controls necessary for a safe emergency landing. Hence, this

thesis researches the use of DRL methods as a means to calculate optimal control laws of a helicopter in an engine power loss scenario.

2.3 Neural Networks

The transition between classic RL and DRL generally consists of replacing the notion of a look-up table typically with an Artificial Neural Network (ANN) (synonymously referred to as a Deep Neural Network (DNN)). An ANN is a group of neurons which hold connections between one another, that's purpose is to understand and translate a data input of one form into a desired output, usually in another form. In general, an ANN can be considered a universal function approximator. The concept of the ANN was inspired by human biology and the way neurons of the human brain function together to understand inputs from human senses.

An artificial neuron (also referred to as a node) is essentially a scalar function in itself - the neuron can receive a number of inputs and can output only one scalar value. The neuron typically performs an activation on a linear operation in the form of:

$$y = f(b + w_1x_1 + w_2x_2 + w_3x_3 + ... + w_nx_n) \tag{2.1}$$

The node linearly transforms the multi-scalar input to a single scalar value, which is then sent into an activation function (such as "tanh", "sigmoid", "ReLu", etc.), which finally produces the output of the neuron. Figure 2.1 shows a visual representation of a neuron/node.

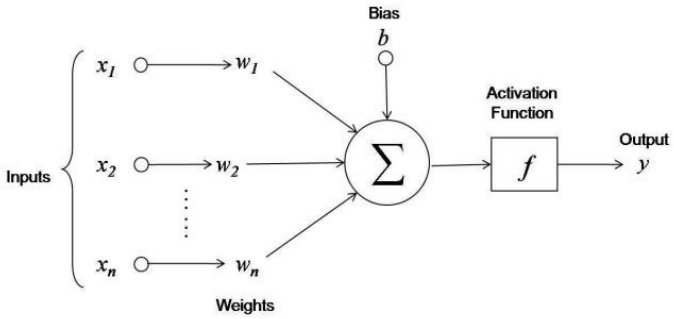


Figure 2.1. Artificial Neuron Visual Depiction

An ANN consists of an input layer, one or more hidden layers and an output layer. Each layer holds a set of neurons, ranging from as little as one neuron to as many as thousands. A fully connected (a.k.a dense) ANN is a network in which the inputs to a neuron in any layer are the outputs of all the neurons in the previous layer, and the output of that neuron is sent to every single neuron in the next layer. See Figure 2.2 for a visual representation of an ANN.

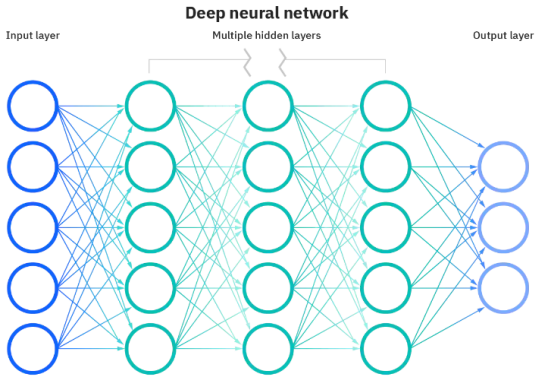


Figure 2.2. ANN Visual Depiction

The study of machine learning applications which utilize neural networks is called "deep learning". Due to the typically large neural network and data-set sizes involved with most practical deep learning applications, and specifically with deep reinforcement learning, the numerical weight and biases update steps are calculated using mini-batch stochastic gradient descent as opposed to regular gradient descent. The

neural networks weights and biases are updated using the back-propagation algorithm. Many times gradient descent optimizers are added to deep learning processes in order to decrease training times. Examples of such optimizers are AdaGrad and RMSprop, and the Adam optimizer which holds elements of the latter optimizers and is generally considered the most effective stochastic gradient descent optimizer.

Although majority of deep learning applications use regularization techniques as a method for reducing training variance and thus reducing the possibility of overfitting, it is less common to apply regularization methods to the neural networks used in DRL algorithms. Although being less common, research in the field [13] has shown that applying regularization techniques (commonly used for supervised learning) with policy based DRL methods can improve performance. Nonetheless, this thesis did not incorporate any regularization methods.

For DRL applications which utilize dynamic equations of motion, such as the research topic of this thesis, the general consensus is that 2 hidden layers are enough to reach the task goal, and only adjustment of the amount of nodes per layer is necessary.

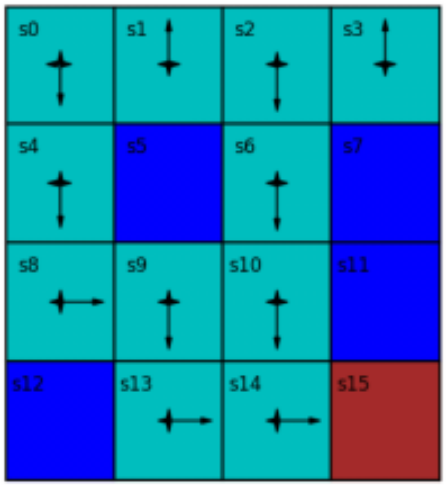
2.4 Reinforcement Learning

RL is an area of machine learning concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward. An agent is an intelligent body that makes decisions or takes actions (for example, a human being), an environment holds all the possible states an agent can be in or perceive (for instance, the universe is an environment for the human species), an agent receives a reward when it reaches certain states in the environment, and an action is something that an agent actively performs such that the environment is possibly altered (for instance, a human pressing the gas pedal in a vehicle will change the

vehicles position in the universe). Regarding further terminologies, an episode is a chronological series of actions and states, starting with an initial state and ending with a terminal state. In the case of this thesis, the initial state will be defined by the initial height and initial velocity of the helicopter at moment of power loss, and the terminal state by definition will be the state at the moment of touchdown. Furthermore, a policy is somewhat analogous to a control law - it is essentially a mathematical model relating the current state to a desired action, and can be iteratively improved through a reinforcement learning process. Lastly, during any reinforcement learning process, a "Q-Value" (also named "expected future rewards" and labeled "q") is predicted for every state-action pair. The Q-Value is a prediction of how good a certain state is given a certain action taken at that state.

In classical RL, all the combinations of possible actions and possible states are generally described by a "Q-Table", where each value in the Q-Table holds a Q-Value. For example, the "Frozen-Lake" [3] environment is a 4x4 grid where each space on the grid is either ice or a lake. The agent can attempt to move up, down, left or right on any space in the grid, giving a total of 4 possible actions and 16 possible states. See Figure 2.3 for a visual representation of the frozen lake environment and see Figure 2.4 for an example of a Q-Table corresponding to the frozen lake environment.

Figure 2.3. OpenAI-Gym Frozen Lake Environment [3]



	S	E	N	W
s0	-0.08	-0.13	-0.11	-0.10
s1	-0.83	-0.17	-0.13	-0.20
s2	-0.01	-0.12	-0.06	-0.11
s3	-0.82	-0.21	-0.13	-0.18
s4	-0.04	-0.83	-0.19	-0.06
s5	0.00	0.00	0.00	0.00
s6	0.07	-0.80	-0.28	-0.78
s7	0.00	0.00	0.00	0.00
s8	-0.76	0.14	-0.02	-0.00
s9	0.39	0.21	-0.75	-0.02
s10	0.45	-0.71	-0.06	0.28
s11	0.00	0.00	0.00	0.00
s12	0.00	0.00	0.00	0.00
s13	0.34	0.58	0.17	-0.70
s14	0.66	0.87	0.40	0.45
s15	0.00	0.00	0.00	0.00

Figure 2.4. Frozen Lake Example with Q-Table [4]

The agent in the environment always starts an episode on coordinate (1,1) on the grid, where its main objective is to reach the desired terminal coordinate on the grid, (4,4). Since (4,4) is the desired terminal coordinate, the agent shall receive a high reward if it manages to reach it. On the contrary, if the agent falls into one of the lakes in its numerous attempts to reach the desired terminal state, it shall receive a penalty (a.k.a a negative reward). The agent moves through the grid until any terminal state is reached, in an episodic manner. During this, rewards and penalties are accumulated and the policy and/or the Q-Table is updated in an iterative manner. In classic RL, the Bellman equation [14] is used to update the Q-Table, and the policy is many times defined as picking the action which provides the highest Q-value. See Figure 2.5 for a general flow diagram of a classic Q-Table based RL network.

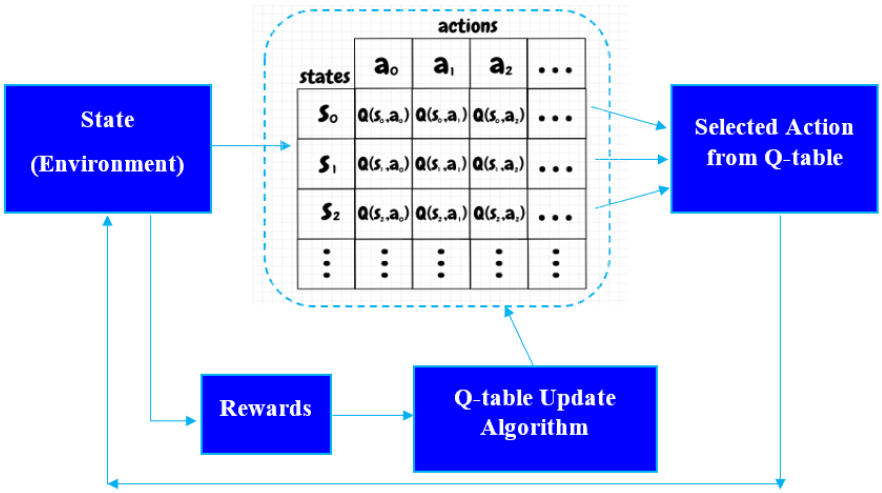


Figure 2.5. Reinforcement Learning General Flow Diagram

Classical RL is limited to discrete action spaces and also discrete state spaces, as observed in the frozen-lake example above. These limitations are problematic in many real-world applications where both the action and state spaces are continuous. Furthermore, classic RL utilizes a Q-Table which is essentially a look-up table, so even if the continuous environments could be transformed to discrete domains, the computational memory required to keep track of all Q-values would be vast for physically realistic environments, and hence this approach would be impractical.

A general solution to these problems is to replace the notion of Q-Tables with a DNN, which will be elaborated on in the next sections.

2.5 Markov Decision Processes in Reinforcement Learning

A Markov process (or Markov chain) is defined as a sequence of possible events in which the probability of a future event depends only on the state attained in the current event. Mathematically, this property can be described in Equation 2.2.

$$\mathbb{P}(S_{t+1}|S_t) = \mathbb{P}(S_{t+1}|S_t, S_{t-1}, S_{t-2}, ..., S_1) \tag{2.2}$$

In other words, a Markov process is a process in which each state holds all of the information of the preceding states. This is considered a first order Markov process, whereas for instance in a second order Markov process the probability of an event would depend on the past 2 preceding states. In most cases, any fully observable environment with a fully defined state-space could allow for the process occurring in it to be defined as Markovian, since all information is attainable and all history is encapsulated in the current state inside the state space. For example, consider a container with 5 red balls and 2 green balls, where each event consists of removing a ball at random without returns. If a state is defined only as the color of the ball removed in the most recent event, then one cannot accurately predict the probability of removing a ball of certain color in the next event since knowledge of past history is required, and the history itself is not defined and saved as a state. Whereas, if every current state consists of the colors of all balls removed in the sequence of all the events that have already occurred, then the process is by definition Markovian. Furthermore, blackjack is an example of a partially observable game where the only time it can be considered a Markov process is if both the dealer and the player have exceptional memory and can memorize all cards dealt between shuffles, thus allowing for a state to consist both of cards in hand and cards already dealt previously. Lastly, dice games like backgammon can be considered Markov processes as the next move is dictated only by the current state and a dice roll. The thesis research topic assumes that enough independent state variables are taken into account such that the process behind the dynamics is (first order) Markovian. An example where this assumption wouldn't hold is if the velocity of an object cannot be directly observed, and must be estimated by differentiation of the past 2 location observations - for which then the process would only be considered Markovian of second order.

The importance of the Markov property is that it enables theoretical proofs, such

as proofs of convergence to optimal policies and optimal value functions for most reinforcement learning algorithms. In fact, majority of the reinforcement algorithms available (PPO included) rely on the Bellman Equation [15], which stems itself from the assumption that the underlying process is Markovian (a.k.a Markov assumption).

A reinforcement learning problem is called a Markov Decision Process (MDP) if each state of the environment holds the Markov property. In the context of reinforcement learning, an MDP can be characterized by 5 elements:

1. $\mathcal{S}$ - The set of all possible states (a.k.a state space) obtainable in the environment under the Markov assumption

2. $\mathcal{A}$ - The set of all possible actions the agent can execute in the environment, for every state.

3. $\mathcal{P}$ - The probabilistic transition function between current state and possible future states given a current action. The transition function is defined as $\mathcal{P}(s, a, s') = \mathbb{P}(S_{t+1} = s' | S_t = s, A_t = a)$, where $\mathbb{P}$ is the probability operator. This function can be discarded in any formulations if the environment is completely deterministic, meaning any action $A_t = a$ taken from a state $S_t = s$ will always cause the agent to reach state $S_{t+1} = s'$. Furthermore, this function is not applicable when using a model-free RL algorithm.

4. $\mathcal{R}$ - Reward function which holds the expected rewards given to the agent when transitioning from one state to another. The reward is statistically expected if the transition between states is non-deterministic or if the reward function is not static. The reward function is defined as $\mathcal{R}(s, a) = \mathbb{E}(R_{t+1} | S_t = s, A_t = a)$, where $\mathbb{E}$ is the expected value operator.

5. γ - Discount factor necessary for convergence of key variables in the RL algorithms. For the necessity of convergence, the discount factor ranges from

$\gamma \in [0, 1]$

Since the notion of RL is to maximize a cumulative reward at any given time, one can define the term G_t as total discounted future rewards at time t:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... + \gamma^{n-1} R_{t_n} = \sum_{i=0}^{\infty} \gamma^i R_{t+1+i} \tag{2.3}$$

Past rewards are not taken into account because the past cannot be altered or affected, and hence cannot provide useful information for future decisions. Furthermore, the total rewards must be discounted in order to insure convergence to a finite number if for any reason the process is endless. The closer γ is to a value of 0, the less future rewards are accounted for and the more immediate rewards are accounted for in the RL process. The opposite is true for values of γ close to 1

In a RL task, the agent makes its decisions of actions to take through a policy π. A policy for a MDP represents a mapping from states to actions, which describes the overall behavior of an agent. In classic RL the policy is typically value-based, meaning the mapping from states to actions is not defined explicitly, rather derived from Q-values such that $\pi = argmax_a(Q(s, a))$. In continuous state-action DRL, the mapping is explicitly represented by a neural network which outputs actions given an input of a state. The definition of the state value $V(s)$ under policy π is described in Equation 2.4.

$$V_\pi(s) = \mathbb{E}_\pi(G_t|S_t = s) = \mathbb{E}_\pi(\Sigma_{i=0}^{\infty} \gamma^i (R_{t+1+i}|S_t = s)) \tag{2.4}$$

Furthermore, the Q-value $Q(s, a)$ under policy π is defined in Equation 2.5.

$$Q_\pi(s, a) = \mathbb{E}_\pi(G_t|S_t = s, A_t = a) = \mathbb{E}_\pi(\Sigma_{i=0}^{\infty} \gamma^i (R_{t+1+i}|S_t = s, A_t = a)) \tag{2.5}$$

Additionally, the state value is by definition a policy weighted average of the Q-value over all possible actions available given a state, as shown in Equation 2.6.

$$V_\pi(s) = \sum_a \pi(a|s)Q(s,a) = \mathbb{E}_\pi[Q(s,a)] \tag{2.6}$$

$$\sum_a \pi(a|s) = 1 \tag{2.7}$$

The derivations above, defined under the Markov assumption, allow for further derivations of RL and DRL methods. Policy gradient methods are a family of DRL methods which define the policy as a numeric function represented by a neural network. Defining the policy this way allows for solving RL problems with continuous state-action spaces efficiently. In order to optimize the policy with the goal to maximize rewards, an objective function containing the policy must be defined. The sections below contain the derivation to the classic policy gradient method, as well as further improvements to more sophisticated algorithms.

2.6 Deep Reinforcement Learning

2.6.1 Development Background

DRL is an improved approach to classic RL. Although the first accounts of the term RL date back to the 1960's, the first successful application of DRL was only in 1992 [16]. Gerry Tesauro developed an algorithm named "TD-Gammon", which was the first known successful account of a DNN being trained using reinforcement learning techniques rather than classical supervised learning techniques. As the name suggests, the algorithm was an autonomous backgammon player. The algorithm would initialize with little to no knowledge of the game, and would only contain information on what states of the game were considered favorable or unfavorable. The neural networks were trained by having the algorithm play against itself for numerous

episodes, where at that point it was capable of beating world class backgammon players. Although stated above is a successful account of DRL, the subject only started to expand heavily around the year 2012.

The first DRL algorithms would only work in discrete state and action spaces, such as the backgammon example stated above. Later on, newly developed algorithms could essentially handle continuous state spaces, although were still limited to discrete action spaces. In 2013, Volodymyr Mnih et al. from DeepMind, had published the first paper [17] on the Deep Q-Network (DQN) algorithm. The paper discussed the algorithm and showed its performance on various Atari games. This algorithm is considered one of the first to revolutionize the use of an ANN with RL. Although the success of this algorithm was apparent, it was limited to discrete actions.

In 2015, DeepMind developers released a paper [18] on the Deep Deterministic Policy Gradient (DDPG) algorithm, which was one of the first DRL algorithms to allow for continuous action spaces alongside continuous state spaces. The algorithm belongs to the "Actor-Critic" family - a group of DRL algorithms which relies on policy gradient methods and policy iteration such that continuous actions can be produced.

Finally, in 2017 John Schulman et al. from OpenAI released a paper [5] on the PPO algorithm, which outperformed many of the other online policy gradient methods at the time in various Atari games. Due to this, the PPO algorithm was selected as the primary algorithm of choice for this thesis.

2.6.2 Overview

As mentioned previously, a solution for solving RL problems in a continuous state-action space is to replace the notion of Q-Tables with a DNN. Since neural networks are universal function approximators, they can be used to predict expected future

rewards or policy functions as a function of any state. Hence, instead of iteratively updating a Q-Table, the weights of the DNN would typically be updated using back-propagation algorithms employing a gradient descent/ascent method as mentioned before. The PPO algorithm investigated in this thesis belongs to the Actor-Critic policy gradient family, which on the basic level holds an actor network that's purpose is to generate a policy providing an action given any state, and a critic network that's purpose is to estimate the expected future reward given any state. Using a DNN instead of look-up tables provides a means for training agents in continuous state and action space environments with practical requirements for computational memory. see 2.6 for a general flow diagram of a DRL network.

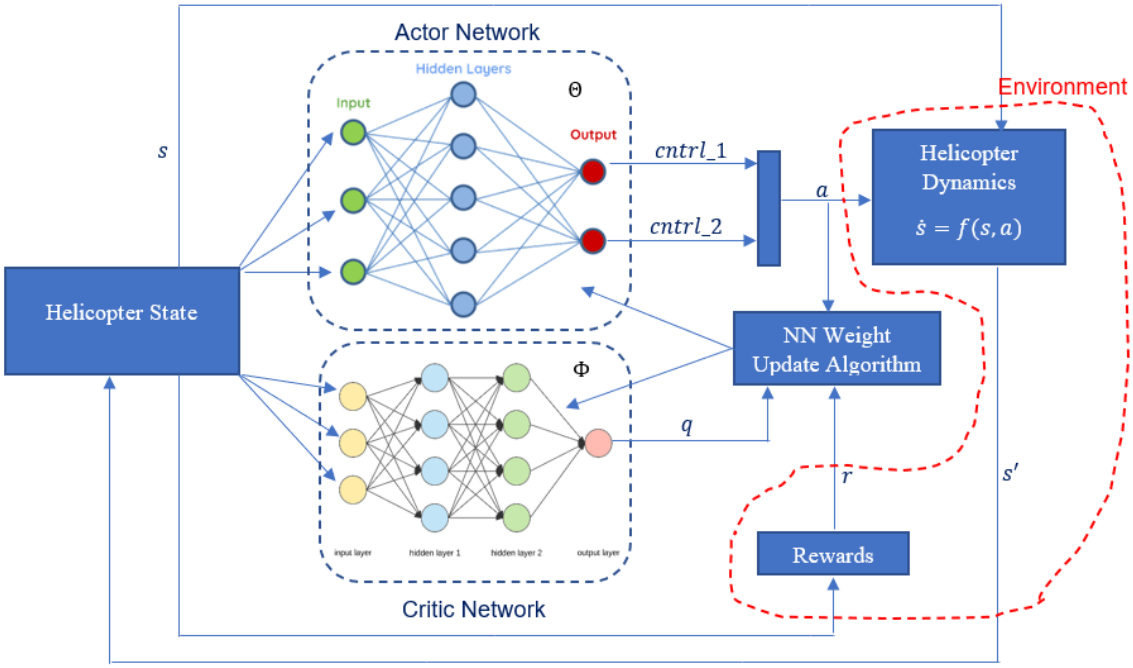


Figure 2.6. Actor-Critic General Flow Diagram

For each action type the actor network outputs a mean action value and optionally a variance of the corresponding action value. In some cases the variance can be defined manually and annealed over time for each action type. The mean and

variance values define the parameters of a Gaussian probability density function. The action values themselves are then sampled through the Gaussian density functions corresponding to each action type. Essentially, even though a fully observable deterministic environment (which for instance doesn't experience any random disturbances) doesn't hold any stochastic elements, and the succeeding state can always be fully determined from the current state and the action taken, a probability density function over the action space is still defined for the policy to allow for exploration of the agent in the environment. Hence, stochastic policies are necessary for any reinforcement learning task to ensure both exploration and exploitation. Thus, for N action types, the policy output is defined as a set of N Gaussian distributions. As mentioned previously, each Gaussian distribution $\mathbb{N}(\mu, \sigma)$ is defined by a varying mean action parameter and varying standard deviation (either output from the actor network of defined manually), such that:

$$\pi(S_t = s) = [\mathbb{N}_1(\mu_1(s), \sigma_1(s)), \mathbb{N}_2(\mu_2(s), \sigma_2(s)), ..., \mathbb{N}_N(\mu_N(s), \sigma_N(s))] \tag{2.8}$$

Furthermore, due to the fact that reward distributions can be sparse over the state-space, and some form of constant feedback is required for the agent to assess itself at any possible state. The critic's job is to estimate as best as possible the expected future rewards of any state-action pair. In classic RL, the critic is typically represented by the scalar values that appear inside a Q-Table as described in the previous section "Reinforcement Learning". The critic is described by an ANN where the input to the network is the state vector of the agent, and the output of the network is a singular scalar value representing the expected future rewards (a.k.a state value denoted $V(s)$) from that state. In PPO the critic network does not require an input of the action due to mathematical manipulations which allow for the unbiased estimation of the Q-value $Q(s, a)$ of a particular action taken at a particular state.

For this thesis, the actor network will serve as a flight controller, where the inputs to the actor network will be the dynamic states of the case study helicopter model (for example; altitude above ground level, velocity, rate of descent, etc.), and the outputs sampled from the Gaussian distributions will be defined control inputs to the helicopter model.

2.6.3 Vanilla Policy Gradient Algorithm

Since the objective is to maximize the discounted cumulative reward at all states, the cost function necessary for updating the actor and critic network weights (θ and w respectively) can be defined as the following:

$$Maximize\ J(\theta, w)\ s.t\ J(\theta, w) = V_{\pi}(s) = \sum_{a} \pi_{\theta}(a|s) Q_{w}(s, a) \tag{2.9}$$

The gradient of the cost function can then be derived as follows:

$$\nabla_{\theta} J(\theta, w) = \sum_{a} \nabla \pi_{\theta}(a|s) Q_{w}(s, a) = \sum_{a} \pi_{\theta}(a|s) \frac{\nabla \pi_{\theta}(a|s)}{\pi_{\theta}(a|s)} Q_{w}(s, a) \tag{2.10}$$

$$\nabla_{\theta} J(\theta, w) = \mathbb{E}[\frac{\nabla \pi_{\theta}(a|s)}{\pi_{\theta}(a|s)} Q_{w}(s, a)] = \mathbb{E}[\nabla ln \pi_{\theta}(a|s) Q_{w}(s, a)] \tag{2.11}$$

Generally, the vanilla policy gradient has low bias but high variance. In order to reduce the variance, a baseline is introduced to the original vanilla policy gradient term. The baselines purpose is to bring the mean gradient values to zero, such that the algorithm can easily differentiate between a good policy (gradient is positive) and a bad policy (gradient is negative). The typical baseline used is called the Advantage Function, which is defined in Equation 2.12.

$$A_{w}(s, a) = Q_{w}(s, a) - V_{w}(s) \tag{2.12}$$

Hence, the vanilla policy gradient term is modified to:

$$\nabla_\theta J(\theta, w) = \mathbb{E}[\nabla ln\pi_\theta(a|s) A_w(s, a)] \tag{2.13}$$

Furthermore, due to the linear properties of both the gradient operator and the expected value operator, the policy gradient operator can be transferred to be outside of the expected value operator, as shown in Equation 2.14.

$$\nabla_\theta J(\theta, w) = \nabla \mathbb{E}[ln\pi_\theta(a|s) A_w(s, a)] \tag{2.14}$$

Lastly, integration of both sides gives a new term for the objective function $J(\theta, w)$ as shown in Equation 2.15. The byproduct of integration function, dependent only on w, was omitted since the objective function captured all the information necessary to maximize itself in terms of both θ and w.

$$J(\theta, w) = \mathbb{E}[ln\pi_\theta(a|s) A_w(s, a)] \tag{2.15}$$

In terms of application, the critic network in the policy gradient algorithm family will only receive a state s as an input and not a state action pair (s, a), hence it cannot directly calculate $Q_w(s, a)$ and it can only calculate $V_w(s)$, and thus direct calculation of the advantage term cannot be executed. In order to bypass this limitation, the advantage is estimated based on the state values $V_w(s)$. John Schulman et al., creator of the PPO algorithm, presents a paper [19] on Generalized Advantage Estimation (GAE) which discusses the method for estimating the advantage term. The objective function can again be newly defined as an estimator:

$$\hat{J}(\theta, w) = \hat{\mathbb{E}}[ln\pi_\theta(a|s) \hat{A}_w(s)] \tag{2.16}$$

2.6.4 From Policy Gradient to PPO Algorithm

The base objective function presented in Equation 2.9 can also be manipulated such that it is a function of any arbitrary policy $\beta(a|s)$, synonymously named a behavior policy:

$$\hat{J}(\theta, w) = \sum_{a} \beta(a|s) \frac{\pi_\theta(a|s)}{\beta(a|s)} \hat{A}_w(s) = \hat{\mathbb{E}}[\frac{\pi_\theta(a|s)}{\beta(a|s)} \hat{A}_w(s)] \tag{2.17}$$

In 2015, John Schulman et al. released a paper [20] on the Trust Region Policy Optimization (TRPO) algorithm, which instead of utilizing the derived policy gradient objective function shown in Equation 2.16, proposed a "surrogate" objective function based on the derivation in Equation 2.17. The behavior policy $\beta(a|s)$ in the TRPO algorithm is defined as the previously updated policy $\pi_{\theta_{old}}(a|s)$:

$$\hat{J}(\theta, w) = \hat{\mathbb{E}}[\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} \hat{A}_w(s)] = \hat{\mathbb{E}}[r_\theta(a|s) \hat{A}_w(s)] \tag{2.18}$$

$$r_\theta(a|s) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} \tag{2.19}$$

As described in the paper, the surrogate objective was constrained outside of the objective itself, hence the term "trust region". This was done in order to eliminate the possibility of excessively large policy updates which could cause the algorithm to diverge or collapse. The algorithm proved to be more stable, efficient and quicker with respect to the regular policy gradient algorithms.

In 2017, John Schulman et al. released a paper [5] on the PPO algorithm which, although providing minor changes to the TRPO algorithm, proved to be simpler, faster and more sample efficient. Rather than constraining the objective function like in TRPO, the constraint was moved inside the objective function in the form of a

clipped surrogate objective:

$$\hat{J}(\theta, w) = \hat{\mathbb{E}}[min(r_\theta(a|s)\hat{A}_w(s), clip(r_\theta(a|s), 1-\epsilon, 1+\epsilon)\hat{A}_w(s))] \tag{2.20}$$

Figure 2.7 shows a visual depiction of the clipping function in the estimated objective term.

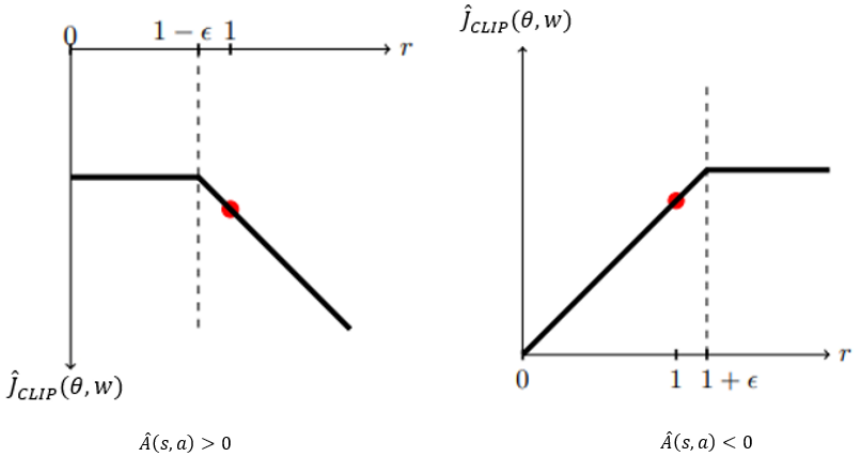


Figure 2.7. Visual Depiction of Objective Function Clipping [5]

The paper [5] on the PPO algorithm shows better performance and results than many other policy gradient algorithms such as A2C, Vanilla Policy Gradient and TRPO.

2.6.5 Reward Shaping

The environment of a DRL model consists of defining all possible states and all possible actions that can be taken from those states. In the continuous realm, the states and actions are defined in terms of bounds rather than specific states and actions which would correspond to the discrete case. For this thesis, the states and actions are defined by the dynamic equations of motion presented above. Other than the dynamics of the agent (or helicopter), a reward structure must be defined in the

environment in order to provide some form of feedback to the agent during training. A reward structure consists of two elements:

- Intermediate Rewards - These are rewards given to the agent between the initial state and the terminal state. Their main objectives are to guide the agent to the main goal sooner during the training process, and to encourage or discourage certain behaviors and/or certain states.

- Terminal Rewards - These are rewards given to the agent at a terminal state (if one exists at all). A terminal state can be directly related to the main objective, or can be defined as a state which has negative implications and still ends an episode. For instance, in the game of chess an objective terminal state could be attaining a check-mate against the opponent. Likewise, a undesirable terminal state could be losing to the opponent through check-mate.

Shaping the rewards holds many difficulties, such as properly defining the order of magnitudes of the various rewards, keeping the ratios between various rewards consistent with the objective, and properly defining the reward functions so that the agents rate of improvement is desirable. Furthermore, although distribution of intermediate rewards can reduce sparsity and drive the agent quicker to its objective, it can potentially create sub-optimal solutions by forcing the solution to a less desirable local minimum. According to engineers from Air Force Research Laboratory (AFRL) working in the field, it is desirable to not include intermediate rewards at all, and only add them if truly necessary.

For the sake of this research, it was recommended by AFRL engineers to immediately terminate an episode and give a negative reward if any pre-defined state limit was exceeded. This strategy was recommended in order to reduce running times whilst still encouraging the agent to reach the desired terminal state, and to also

reduce possibility of the agent reaching states which could cause a divergence of the dynamic model of the environment.

2.6.6 Ray RLlib

Ray RLlib is an open-source library for RL, offering support for production-level, highly distributed RL workloads while maintaining unified and simple Application Program Interfaces (API's) for a large variety of industry applications. It provides an organized structure for training and testing DRL networks, and offers various methods for algorithm hyper-parameter tuning such as classic grid-search, Bayesian Optimization and Distributed Hyper-Parameter Optimization. Ray RLlib was used with Python to create, train and test the PPO DRL network in this thesis.

2.7 Similar Work

2.7.1 Previous Work in Optimal Control

In 2018, Harris [2] applied optimal control using GPOPS-II to analytically derive an HV diagram and to also generate optimal flight trajectories for an AH-1Z helicopter in an One Engine Inoperable (OEI) or total power loss scenario. Harris developed a point-mass dynamic model that compared favorably to AH-1Z flight test data for low-speed flight. The point mass model was based on the models used by Johnson [21] and Lee [22] in their research regarding helicopter flight control using functional optimization. Furthermore, in 2020 Brown [23] continued the work performed by Harris which consisted of investigating different factors of the helicopter model and their impacts on the HV diagram, such as the sensitivity of the HV diagram solution to different ground effect models.

Although both theses provided favorable and plausible results towards generating HV diagrams, they were less practical for generating real-time solutions due to the fact

that the GPOPS-II simulations were not capable of generating closed form control law solutions, and running times were quite often on the order of minutes for one initial height-velocity pair.

Due to the limitations stated above, this thesis will research the use of DRL as a means to provide an optimal control law for a helicopter in a total power loss scenario, such that given any state, an optimal set of values for the helicopters controls can be generated instantly in a real-time setting.

2.7.2 Previous Work in Deep Reinforcement Learning

A similar thesis [24] was conducted at the Middle East Technical University involving training a DRL network to generate optimal policies for landing an OH-58A helicopter in a total power loss scenario. The helicopter was modeled as a point-mass and was restricted to motion in the vertical plane only. The model comprised of 6 state variables and the dynamic equations of motion and were based on Lee's [22] simplified helicopter model. Furthermore, the Asynchronous Advantage Actor Critic (A3C) policy gradient algorithm was used to update the actor and critic network weights. In addition, the author used ReLu activation functions for actor and critic hidden layers. The thesis mentions that the actor and critic networks consisted of 2 hidden layers, each with 32 neurons. The thesis showed reasonable and affirmative results, where the DRL network managed to bring the helicopter to landings at acceptable velocities even inside the "avoid" area of the official HV diagram. Also, the test and training performances seemed only to improve with number of episodes, where no signs of collapsing rewards or a collapsing policy were exhibited.

2.8 Summary

This chapter provided background in topics related to previous work such as functional optimization and control, as-well as the advantages and disadvantages that arose from the previous work which led to research conducted in this thesis case study. Furthermore, the chapter provided the necessary material for understanding DRL methods and their advantages in terms of replacing analytically derived control laws with trainable neural networks.

III. Research Methodology

3.1 Chapter Overview

This chapter describes the development of a helicopter dynamic model and the Deep Reinforcement Learning (DRL) problem formulation including the reward architectures used for this research. The chapter begins by developing a helicopter model based on previous research. The helicopter model is then incorporated into the environment of the DRL problem. Also part of the environment, the reward structure is defined in order to provide feedback to the agent. The hyper-parameter tuning methods are discussed and also general training methodologies, which provide further useful information, are presented below.

3.2 Basic Assumptions

A point-mass model has been selected for this research since point-mass models have been shown by several studies, including those by Johnson [21] and Lee [22], to adequately represent a helicopter during flight. Furthermore, in terms of generating an Height-Velocity (HV) diagram, the low height high speed portion will not be analyzed since it is related to tail strike and requires the modeled helicopter's exterior dimensions to be defined. Since a helicopter ideally experiences minimal lateral movement when performing an autorotation, the model confines helicopter motion within a vertical plane. In contrast to Harris's research [2], the rotor inflow was modeled using an empirical vortex ring state model. The helicopter mass is assumed to be constant throughout the autorotation, since any changes in mass due to factors such as fuel burn are negligible. The helicopter model reflects standard sea-level atmospheric conditions with no wind and constant air density.

3.3 8-State Dynamic Model

3.3.1 Helicopter Model

The helicopter dynamic models used by Lee [22] and Harris [2] formed the starting point for the development of the models used in this research. The force diagram shown in Figure 3.1 depicts a point-mass helicopter in descent and provides the basis for the dynamic model's equations of motion. Balancing the vertical and horizontal forces acting on the point-mass produces Equations 3.1 and 3.2, respectively.

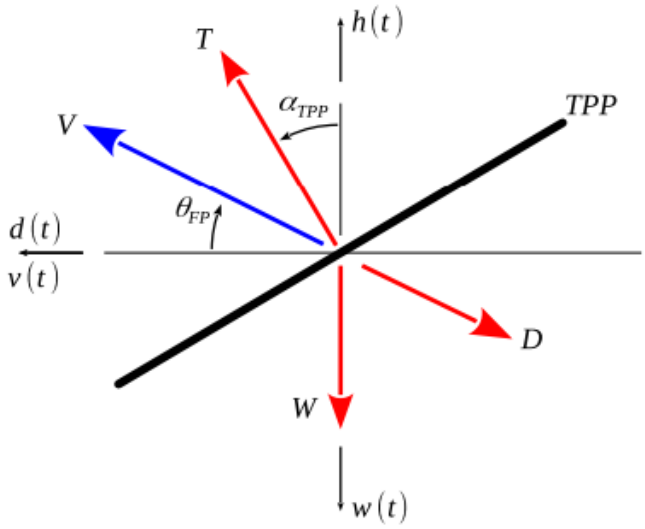


Figure 3.1. Point-Mass Helicopter Force Diagram

$$\dot{u} = \frac{1}{m} T \sin \alpha_{TPP} - \frac{1}{m} D \cos \theta_{FP} \tag{3.1}$$

$$\dot{w} = g - \frac{1}{m} T \cos \alpha_{TPP} - \frac{1}{m} D \sin \theta_{FP} \tag{3.2}$$

Figure 3.1 also presents many of the state and control variables for the dynamic model. The two controls (X_1 and X_2) are the rate of change of main rotor collective pitch

angle, $\dot{\theta}_{COLL}$ and the rate of change of angle of attack of the rotor disk plane, $\dot{\alpha}_{TPP}$ described in Equations 3.3 and 3.4. The thrust vector T is a function of θ_{COLL}. The integrals of the two control variables are defined as states, allowing to fully describe the state-space equations. Two of the states describe the position of the helicopter: h is the height of the helicopter above the ground, and d is the horizontal distance the helicopter travels starting at engine failure. The next two states are the vertical and horizontal velocities of the helicopter, w and u, respectively. By definition, the derivatives of the positions are the velocities as shown in Equations 3.5 and 3.6. It is important to note that a negative sign is included in Equation 3.6, since the height is defined as positive in the upward direction, and the vertical velocity is defined as positive in the downward direction.

$$X_1 = \dot{\theta}_{COLL} \tag{3.3}$$

$$X_2 = \dot{\alpha}_{TPP} \tag{3.4}$$

$$\dot{d} = u \tag{3.5}$$

$$\dot{h} = -w \tag{3.6}$$

Equations 3.7-3.9 show the derivation for the energy equation, which relates the potential, kinetic and main rotor rotational energies to the excess power available. The equations introduce the rotor rotational speed, Ω, as another state variable. Also, the excess power is defined as the difference between the current engine power available P_E and the required power P_{req}.

$$\frac{d}{dt}[\frac{1}{2}M(u^2 + w^2) + Mgh + \frac{1}{2}I_R\Omega^2] = P_E - P_{req} \tag{3.7}$$

$$Mu\dot{u} + Mw\dot{w} - Mgw + I_R\Omega\dot{\Omega} = P_E - P_{req} \tag{3.8}$$

$$\dot{\Omega} = \frac{1}{I_R\Omega}(P_E - P_{req} - Mu\dot{u} - Mw\dot{w} + Mgw) \tag{3.9}$$

Furthermore, an independent state equation defining the engine power decay after total power loss is defined with a time constant τ_E:

$$\tau_E \dot{P}_E = -P_E \tag{3.10}$$

In summary, the 8 state-space equations are described below:

$$\dot{d} = u \tag{3.5}$$

$$\dot{h} = -w \tag{3.6}$$

$$\dot{u} = \frac{1}{m} T \sin \alpha_{TPP} - \frac{1}{m} D \cos \theta_{FP} \tag{3.1}$$

$$\dot{w} = g - \frac{1}{m} T \cos \alpha_{TPP} - \frac{1}{m} D \sin \theta_{FP} \tag{3.2}$$

$$\dot{\Omega} = \frac{1}{I_R \Omega} (P_E - P_{req} - Mu\dot{u} - Mw\dot{w} + Mgw) \tag{3.9}$$

$$\dot{P}_E = -\frac{P_E}{\tau_E} \tag{3.11}$$

$$\dot{\theta}_{COLL} = X_1 \tag{3.3}$$

$$\dot{\alpha}_{TPP} = X_2 \tag{3.4}$$

where;

$$\cos \theta_{FP} = \frac{u}{\sqrt{u^2 + w^2}}$$

$$\sin \theta_{FP} = \frac{w}{\sqrt{u^2 + w^2}}$$

The auxiliary equations describing variables such as the thrust T, drag D and power required P_{req} are presented in the next sections. All auxiliary equations are provided in dimensionless form. Moreover, the state-space equations along with the states themselves are normalized such that they are dimensionless, in order to provide solutions in all relative scales. Furthermore, the time variable is also normalized such

that it is dimensionless:

$$\psi = t\Omega_0 \tag{3.12}$$

$$\frac{d}{d\psi} = \frac{1}{\Omega_0}\frac{d}{dt} = ()' \tag{3.13}$$

$$\bar{d}(\psi) = \frac{1}{R}d(\Omega_0 t) \tag{3.14}$$

$$\bar{h}(\psi) = \frac{1}{R}h(\Omega_0 t) \tag{3.15}$$

$$\bar{u}(\psi) = \frac{1}{\Omega_0 R}u(\Omega_0 t) \tag{3.16}$$

$$\bar{w}(\psi) = \frac{1}{\Omega_0 R}w(\Omega_0 t) \tag{3.17}$$

$$\bar{\Omega}(\psi) = \frac{1}{\Omega_0}\Omega(\Omega_0 t) \tag{3.18}$$

$$\bar{P}_E(\psi) = \frac{1}{P_{max}}P_E(\Omega_0 t) \tag{3.19}$$

$$\bar{\alpha}_{TPP}(\psi) = \alpha_{TPP}(\Omega_0 t) \tag{3.20}$$

$$\bar{\theta}_{COLL}(\psi) = \theta_{COLL}(\Omega_0 t) \tag{3.21}$$

3.3.2 Required Power

In theory, the total power required by the helicopter can be calculated as a combination of main rotor power, tail rotor power, and accessory power losses. After including the gearbox efficiencies, this summation becomes:

$$P_R = \frac{1}{\eta_C}\left(\frac{P_{MR}}{\eta_{MR}} + \frac{P_{TR}}{\eta_{TR}} + P_{Acc}\right) \tag{3.22}$$

For the research conducted in this thesis, it was assumed that the tail rotor power and accessory power were negligible and hence were not calculated nor added to the total power calculation. The power requirements for the main rotor are calculated using the methods discussed in Section 6.4 in Johnson's Rotorcraft Aeromechanics [8]. Johnson explains that the required rotor power for a helicopter in forward flight can

be estimated by aggregating the rotor induced power, profile power, parasite power, and climb power. Equation 3.24 shows this relationship in terms of power coefficients.

$$P_{MR} = \rho A(\Omega R)^3 C_{P_{MR}} \tag{3.23}$$

$$C_{P_{MR}} = C_{P_i} + C_{P_o} + C_{P_P} + C_{P_C} \tag{3.24}$$

The rotor induced power is the power that is associated with the production of thrust. The calculation for the induced power coefficient, C_{P_i}, includes two empirical correction factors; κ is the induced power correction factor which accounts for any general losses such as those from tip loss, nonuniform inflow, wake swirl and the fact that the helicopter has a finite number of blades, and B which is Prandtl's tip-loss factor. λ_i is the induced velocity ratio and λ_z is the rate of descent velocity ratio

$$C_{P_i} = C_T(\kappa\frac{\lambda_i}{B} + \lambda_z) \tag{3.25}$$

where;

$$\lambda_i = \frac{v_i}{V_{tip}}$$
$$\lambda_z = \frac{v_z}{V_{tip}}$$

The term for the coefficient of thrust C_T is given below. σ is the solidity ratio of the main rotor, which is the ratio between the total rotor blade area and the rotor disk area. Furthermore, μ is the advance ratio which represents the helicopter's airspeed in non-dimensional form. θ_{tw} is the twist angle of the rotor blades, c_{do} is the average

blade drag coefficient and a_0 is the rotor blade lift curve slope.

$$C_T = \frac{\sigma a_0}{2}[(\frac{1}{3} + \frac{1}{2}\mu^2)\theta_{COLL} - \frac{1}{8}\mu^2\theta_{tw} - \frac{1}{2}(\frac{\lambda_i}{B} + \lambda_z)(1 + \frac{cd_0}{a_0})] \tag{3.26}$$

$$\mu = \frac{1}{\bar{\Omega}_0(\psi)}[\bar{u}(\psi)cos\alpha_{TPP} + \bar{w}(\psi)sin\alpha_{TPP}] \tag{3.27}$$

The profile power is the power necessary for the rotor blades to rotate through the air. The following approximation of the profile power coefficient is accurate for low advance ratios, such as those that are encountered during this research.

$$C_{P_o} = \frac{\sigma c_{do}}{8}(\bar{\Omega}^2(\psi) + 4.65\mu^2) \tag{3.28}$$

Whereas the profile power accounts for blade drag, rotor parasite power encompasses the drag from the helicopter chassis. The drag that the helicopter experiences is estimated using an equivalent normalized longitudinal axis flat plate area, $\bar{f}_x$, when calculating the profile power coefficient, C_{P_p}.

$$C_{P_p} = |\frac{1}{2}\bar{f}_x\bar{u}^3(\psi)| \tag{3.29}$$

Where;

$$\bar{f}_x = \frac{f_x}{A}$$
$$\bar{f}_z = \frac{f_z}{A}$$

The climb power is the power necessary for the helicopter to change its altitude. The climb power coefficient, C_{P_c}, is derived from the helicopter's rate of descent

(ROD), w, and the aircraft weight coefficient, C_W.

$$C_{P_c} = -C_W \bar{w}(\psi) + |\frac{1}{2}\bar{f}_z \bar{w}^3(\psi)| \tag{3.30}$$

3.3.3 Rotor Inflow

Due to the descending flight path of helicopters during autorotation, the rotor will likely operate in Vortex Ring State (VRS). Johnson describes [25] the equations which utilize an empirical extension of the baseline momentum theory to determine rotor inflow during flight that encounters VRS, although the equations are also valid for flight out of VRS. The baseline momentum theory can be found in Johnson's book of Rotorcraft Aeromechanics [8].

3.3.4 Ground Effect

Whenever a helicopter is in close proximity to the ground, the induced velocity at the rotor decreases. This phenomenon, known as ground effect, causes a reduction in the amount of power required to produce a given thrust. Likewise, ground effect causes an increase in the amount of thrust produced by a rotor at a given power. Ground effect can be modeled by including a ground effect factor, k_g, which scales the thrust (or coefficient of thrust). 3.1 and 3.2 are then modified to be:

$$\dot{u} = \frac{1}{m} k_g T \sin \alpha_{TPP} - \frac{1}{m} D \cos \theta_{FP} \tag{3.31}$$

$$\dot{w} = g - \frac{1}{m} k_g T \cos \alpha_{TPP} - \frac{1}{m} D \sin \theta_{FP} \tag{3.32}$$

Some ground effect models are dependent only the height above ground level (AGL) of the helicopter (such as Cheeseman, Bennett and Hayden's models), and there exist some models which incorporate blade element theory variables (such as Law and Zbrozek's models). The ground effect model used for this thesis is Hayden's

(taken from section 4.8 of Johnson's book Rotorcraft Aeromechanics [8]), which is a function of height AGL only:

$$k_g(\bar{h}) = \left[0.9926 + \frac{0.03794}{(\frac{1}{2}\bar{h})^2}\right]^{2/3} \tag{3.33}$$

This thesis' objective was to research and analyze the use of DRL in order define optimal control laws for emergency landing scenarios in helicopters, with less importance on deriving a substantially accurate flight model with respect to the real world. Thus, Hayden's model was selected due to its simplicity.

3.3.5 Dimensionless Equations of Motion

The 8 dynamic equations described above were normalized such that they could be described in dimensionless form, and could incorporate the auxiliary equations defined above:

$$\bar{d}'(\psi) = \bar{u} \tag{3.34}$$

$$\bar{h}'(\psi) = -\bar{w} \tag{3.35}$$

$$\bar{u}'(\psi) = \frac{1}{\bar{M}}[k_g C_T \sin\alpha_{TPP}\bar{\Omega}^2 - \frac{1}{2}\bar{V}^2\bar{f}_x\cos\theta_{FP}] \tag{3.36}$$

$$\bar{w}'(\psi) = \bar{g} - \frac{1}{\bar{M}}[k_g C_T \cos\alpha_{TPP}\bar{\Omega}^2 - \frac{1}{2}\bar{V}^2\bar{f}_z\sin\theta_{FP}] \tag{3.37}$$

$$\bar{\Omega}'(\psi) = \frac{1}{\bar{\Omega}\bar{I}_R}[\bar{P}_{max}\bar{P}_E - \frac{C_{P_{req}}}{\bar{M}}\bar{\Omega}^3 + \bar{g}\bar{w} - \bar{v}'\bar{v} - \bar{w}'\bar{w}] \tag{3.38}$$

$$\bar{P}'_E(\psi) = -\frac{1}{\bar{\tau}_E}\bar{P}_E \tag{3.39}$$

$$\bar{\theta}'_{COLL}(\psi) = \bar{X}_1 \tag{3.40}$$

$$\bar{\alpha}'_{TPP}(\psi) = \bar{X}_2 \tag{3.41}$$

where;

$$\bar{M} = \frac{M}{\rho A R}$$
$$\bar{V} = \sqrt{\bar{u}^2 + \bar{w}^2}$$
$$\bar{g} = \frac{g}{\Omega_0^2 R}$$
$$\bar{I}_R = \frac{I_R}{M R^2}$$
$$\bar{P}_{max} = \frac{P_{max}}{M \Omega_0^3 R^2}$$
$$\bar{X}_1 = \frac{X_1}{\Omega_0}$$
$$\bar{X}_2 = \frac{X_2}{\Omega_0}$$

3.3.6 Scaling of Dimensionless States

The eight state variables selected for this model are listed in Equations 3.44-3.51. Each of these states have been scaled with respect to the normalized states 3.14-3.21 such that they're all on the order of $O(1)$, which is accepted practice when solving with Non-Linear Programming (NLP) algorithms and specifically with a Deep Neural Network (DNN). The scaling follows similar scaling techniques used by Lee [22] and Johnson [21].

$$\tau_s = \frac{\psi}{100} \tag{3.42}$$
$$\frac{d}{d\tau_s} = 100 \frac{d}{d\psi} \tag{3.43}$$
$$x_1(\tau_s) = \frac{1}{10} \bar{d}(\frac{\psi}{100}) \tag{3.44}$$
$$x_2(\tau_s) = \frac{1}{10} \bar{h}(\frac{\psi}{100}) \tag{3.45}$$
$$x_3(\tau_s) = 100 \bar{u}(\frac{\psi}{100}) \tag{3.46}$$
$$x_4(\tau_s) = 100 \bar{w}(\frac{\psi}{100}) \tag{3.47}$$

$$x_5(\tau_s) = \bar{\Omega}(\frac{\psi}{100}) \tag{3.48}$$

$$x_6(\tau_s) = \bar{P}_E(\frac{\psi}{100}) \tag{3.49}$$

$$x_7 = \bar{X}_1(\frac{\psi}{100}) \tag{3.50}$$

$$x_8 = \bar{X}_2(\frac{\psi}{100}) \tag{3.51}$$

Due to the fact that scaling the normalized states required simple arithmetic computations, the dynamic equations were solved in non-dimensional form (see Equations 3.34-3.41) and the states were scaled and un-scaled accordingly.

3.3.7 Controls

As presented above, the two control variables X_1 and X_2 are proportionally related to the; (1) rate of change of main rotor collective pitch angle and (2) the rate of change of rotor disk Tip Path Plane (TPP) angle with respect to the horizontal plane.

3.3.8 Summary of State Space Equations

Below are the state-space equations written in their initial dimensional form, without normalization and without scaling. These equations aren't directly implemented into the algorithms used for this research, but as shown above provide the basis for the normalized and scaled equations used in the DRL algorithms:

$$\dot{d} = u \tag{3.5}$$

$$\dot{h} = -w \tag{3.6}$$

$$\dot{u} = \frac{1}{m}k_g T \sin\alpha_{TPP} - \frac{1}{m}D\cos\theta_{FP} \tag{3.31}$$

$$\dot{w} = g - \frac{1}{m}k_g T \cos\alpha_{TPP} - \frac{1}{m}D\sin\theta_{FP} \tag{3.32}$$

$$\dot{\Omega} = \frac{1}{I_R\Omega}(P_E - P_{req} - Mu\dot{u} - Mw\dot{w} + Mgw) \tag{3.9}$$

$$\dot{P_E} = -\frac{P_E}{\tau_E} \tag{3.11}$$

$$\dot{\theta}_{COLL} = X_1 \tag{3.3}$$

$$\dot{\alpha}_{TPP} = X_2 \tag{3.4}$$

Below are the state-space equations written in non-dimensional form. Although the equations are dimensionless, they're not scaled and hence do not hold the same order of magnitudes. These first order ordinary differential equations are calculated in the algorithms of the helicopter dynamics, but are not used in the DRL algorithms until the states are scaled.

$$\bar{d}'(\psi) = \bar{u} \tag{3.34}$$

$$\bar{h}'(\psi) = -\bar{w} \tag{3.35}$$

$$\bar{u}'(\psi) = \frac{1}{\bar{M}}[k_g C_T \sin\alpha_{TPP}\bar{\Omega}^2 - \frac{1}{2}\bar{V}^2\bar{f}_x\cos\theta_{FP}] \tag{3.36}$$

$$\bar{w}'(\psi) = \bar{g} - \frac{1}{\bar{M}}[k_g C_T \cos\alpha_{TPP}\bar{\Omega}^2 - \frac{1}{2}\bar{V}^2\bar{f}_z\sin\theta_{FP}] \tag{3.37}$$

$$\bar{\Omega}'(\psi) = \frac{1}{\bar{\Omega}\bar{I}_R}[\bar{P}_{max}\bar{P}_E - \frac{C_{P_{req}}}{\bar{M}}\bar{\Omega}^3 + \bar{g}\bar{w} - \bar{v}'\bar{v} - \bar{w}'\bar{w}] \tag{3.38}$$

$$\bar{P}'_E(\psi) = -\frac{1}{\bar{\tau}_E}\bar{P}_E \tag{3.39}$$

$$\bar{\theta}'_{COLL}(\psi) = \bar{X}_1 \tag{3.40}$$

$$\bar{\alpha}'_{TPP}(\psi) = \bar{X}_2 \tag{3.41}$$

Lastly, below are the 8 scaled states which are provided to the actor DNN and critic DNN of the DRL algorithms. The scaling maintained an average order of $O(1)$ for all the states, thus allowing for more stable weight updates of the actor and critic networks:

$$\tau_s = \frac{\psi}{100} \tag{3.42}$$

$$\frac{d}{d\tau_s} = 100\frac{d}{d\psi} \tag{3.43}$$

$$x_1(\tau_s) = \frac{1}{10}\bar{d}(\frac{\psi}{100}) \tag{3.44}$$

$$x_2(\tau_s) = \frac{1}{10}\bar{h}(\frac{\psi}{100}) \tag{3.45}$$

$$x_3(\tau_s) = 100\bar{u}(\frac{\psi}{100}) \tag{3.46}$$

$$x_4(\tau_s) = 100\bar{w}(\frac{\psi}{100}) \tag{3.47}$$

$$x_5(\tau_s) = \bar{\Omega}(\frac{\psi}{100}) \tag{3.48}$$

$$x_6(\tau_s) = \bar{P}_E(\frac{\psi}{100}) \tag{3.49}$$

$$x_7(\tau_s) = \bar{X}_1(\frac{\psi}{100}) \tag{3.50}$$

$$x_8(\tau_s) = \bar{X}_2(\frac{\psi}{100}) \tag{3.51}$$

3.3.9 State and Action Limits

Below are the helicopter's structural and operational limits, and as well as limits which were defined such that the dynamic equations of motion wouldn't diverge, producing unfeasible values. The state limits are initially presented in dimensional form, followed by their presentation in scaled dimensionless form.

- $d > 0 \rightarrow x_1 > 0$ - This limit makes sure that the helicopter does not fly backwards at any instance.

- $h > 0 \rightarrow x_2 > 0$ - Any simulated episode will end once the helicopter has reached the ground. The helicopter is allowed to climb higher than its initial height if necessary.

- $0 < u[knts] < 150 \rightarrow 0 < x_3 < 35.13$ - A typical but arbitrary V_{NE} never exceed velocity of 150[knts] was defined. Surpassing this limit could potentially cause structural damage to the helicopter as well as possible main rotor stall.

- $-800 < w[\frac{ft}{s}] < 2000 \rightarrow -1.85 < x_4 < 4.63$ - The rate of descent has no direct affect on structural damage, but is known to be strongly correlated to main rotor rotational velocity (RPM) which does hold operational and structural limits.

- $70 < \Omega[\%] < 115 \rightarrow 0.7 < x_5 < 1.15$ - The main rotor RPM must be maintained within these limits in order to limit possibility of main rotor stall or excessive blade stress.

- $0 < P_E[\%] < 100 \rightarrow 0 < x_6 < 1$ - There is no need to define limits for engine power since it is designed and limited to its own performance capabilities and is independent of all helicopter controls. However, limits were defined anyways due to requirements of the Ray RLlib Proximal Policy Optimization (PPO) algorithm.

- $1 < \theta_{COLL}[deg] < 22 \rightarrow 0.017 < x_7 < 0.383$ - The main rotor collective angle is mechanically limited. The limits in dimensionless form are converted from degrees to radians.

- $-30 < \alpha_{TPP}[deg] < 30 \rightarrow -0.524 < x_8 < 0.524$ - The TPP angle is limited in order to discourage nose-dives or excessive flares.

The action limits were defined arbitrarily such that it would take 3 seconds to increase the main rotor collective pitch from the minimum setting to the maximum setting, and it would take 6 seconds to increase the TPP angle from the minimum setting to the maximum setting.

- $-7 < \dot{\theta}_{COLL}[\frac{deg}{sec}] < 7 \rightarrow -0.122 < X_1[rad] < 0.122$

- $-10 < \dot{\alpha}_{TPP}[\frac{deg}{sec}] < 10 \rightarrow -0.175 < X_2[rad] < 0.175$

3.3.10 Helicopter Model Parameters

Table 3.1 presents the physical and dimensionless helicopter model parameters.

Table 3.1. Helicopter Model Physical Parameters and Coefficients

Parameter Name	Symbol	Value
Gravity Constant	g	$32.172[\frac{ft}{s}]$
Air Density	ρ	$0.002378[\frac{slug}{ft^3}]$
Gross Weight	GW	$16200[lbf]$
100% RPM	Ω_0	$30[\frac{rad}{sec}]$
Main Rotor Radius	R	$24[ft]$
Solidity Ratio	σ	0.1
Average Lift Curve Slope	a_0	$5.73[\frac{1}{rad}]$
Average Blade Drag Coefficient	c_{d_0}	0.008
Induced Velocity Correction Factor	κ	1.08
Tip Loss Factor	B	0.97
Blade Twist $0 \rightarrow R$	θ_{tw}	$-10[deg]$
Longitudinal Effective Drag Area	f_x	$25[ft^2]$
Vertical Effective Drag Area	f_z	$168[ft^2]$
Main Rotor Moment of Inertia	I_R	$5440[slug * ft^2]$
Max Engine Power Available	P_{max}	$2500[hp]$
Failed Engine Time Constant	τ	$0.5[s]$
Rotor Hub Height	h_{hub}	$12[ft]$
Helicopter Mass	m	$\frac{GW}{g} = 503.5[slug]$
Rotor Disk Area	A	$\pi R^2 = 576\pi[ft^2]$
Operational Rotor Tip Speed	V_{tip_0}	$\Omega_0 R = 720[\frac{rad}{s}]$
Weight Coefficient	C_W	$\frac{GW}{\rho A {V_{tip_0}}^2} = 7.26 * 10^{-3}$
Normalized Gravity Constant	$\bar{g}$	$\frac{g}{\Omega_0^2 R} = 1.49 * 10^{-3}$
Normalized Mass Constant	$\bar{m}$	$\frac{m}{\rho A R} = 4.875$
Normalized Longitudinal Drag Area	$\bar{f_x}$	$\frac{f_x}{A} = 0.0138$
Normalized Vertical Drag Area	$\bar{f_z}$	$\frac{f_z}{A} = 0.0928$
Normalized Rotor Hub Height	$\bar{h}_{hub}$	$\frac{h_{hub}}{R} = 0.5$
Normalized Rotor Inertia	$\bar{I}_R$	$\frac{I_R}{mR^2} = 0.0188$
Normalized Max Power	$\bar{P}_{max}$	$\frac{P_{max}}{m\Omega_0^3 R^2} = 1.756 * 10^{-4}$
Normalized Engine Time Constant	$\bar{\tau}$	$\tau\Omega_0 = 15$

3.4 Deep Reinforcement Learning Problem Formulation

The DRL problem for this research implements the above helicopter model to estimate the optimal controls and optimal flight trajectories in a total power loss scenario such that landing velocities are minimized. The non-dimensionalized and scaled states above are input into the actor and critic neural networks of the PPO DRL model, for which after the appropriate controls input to the helicopter are generated through the policy functions defined by the parameters output from the actor network.

3.4.1 Reward Architecture

The reward architecture consisted only of terminal rewards. An episode terminated when the helicopter reached the ground ($h = 0$) although it could also terminate prematurely if any of the state limits were exceeded. The terminal reward was defined as follows:

if State Limit Exceeded **then**

$r_{terminal} = -\frac{x_2}{x_{2_{initial}}}$

Terminate Episode

else if Helicopter Reached Ground **then**

$r_{terminal} = \frac{5}{0.8+0.85x_4^2+0.15x_3^2}$

Terminate Episode

else

$r_{terminal} = 0$

end if

The premature terminal reward utilized the height of termination in order to encourage the agent to reach the ground as soon as possible. The height of termination was normalized by the initial episode height such that the values were always on the

order of $O(1)$, and so that all penalties would be defined with respect to the initial condition. Once the episodes terminated on ground impact, the terminal reward turned from a negative value to a positive value, thus giving extra feedback to the agent. Furthermore, the ground terminal reward increased even more if the ROD and the velocity were minimized. The values of the terminal reward were defined arbitrarily with the intention to keep the reward values on the order of $O(1)$ and to also give more weight to the ROD (x_4) as opposed to the velocity (x_3).

3.4.2 General Training Methodology

During hyper-parameter tuning or training, an episode was initiated by random selection of one out of 400 possible initial height-velocity combinations. The initial height was randomly selected out of 25 different values ranging between $24 - 600[ft]$ AGL, and the initial velocity was randomly selected out of 16 different values ranging between $0 - 50[KTAS]$. Another proposed method was to select an initial height-velocity pair based on a probability distribution defined by the initial total energy, with higher probability of selection favored towards higher initial energy states. This method was proposed based on the assumption that higher initial energy states already encapsulate information from lower energy states, thus possibly saving training times. This method was not implemented in the research discussed in this thesis, although it may be implemented in future work. The episode time-step was set to $\Delta t = 0.1[s]$ both for the training and test environments. The default Ray RLlib PPO configuration file was used as a baseline, where various hyper-parameters were tuned and configured after. The algorithm was set to use 8 roll-out workers as that was the maximum amount the computer could withstand without crashing when running the algorithm.

3.4.3 Algorithm Hyper-Parameter Selection

Trained DRL agents can provide drastically different results when different algorithm hyper-parameters are used. An undesirable combination of hyper-parameters used to train an agent can cause the total rewards to plateau at values much lower than the true optimum. Moreover, the rewards can many times even "collapse", meaning the agent is being trained negatively such that its performance is only deteriorating. In order to overcome this issue, the Ray Tune Distributed Asynchronous Hyper-Parameter Optimization (HyperOpt) algorithm was used to optimize various hyper-parameters. HyperOpt is a modified version of the Bayesian Optimization (BayesOpt) optimizer algorithm, and allows for optimization of parameters both in the discrete and continuous range. Both HyperOpt and BayesOpt are black-box optimizers, meaning the change in hyper-parameters cannot be directly inferred from the objective/cost.

The HyperOpt tuner was run for 2 experiments of 25 trials where each trial was run for $100,000$ episodes. The first set of trials consisted of tuning hyper-parameters on actor and critic networks smaller than the default Ray RLlib PPO network sizes. The default PPO network was defined as having 2 hidden layers holding 256 nodes each. The second set of trials consisted of tuning hyper-parameters on networks of larger size as compared to the default network size. The reason for splitting the hyper-parameter selection into two experiments was so that in addition to attaining optimal hyper-parameters, the effects of different sized networks could be examined. Before each trial, the HyperOpt algorithm picked a set of hyper-parameters based both on a uniform distribution of the possible values allowed, and also based on the performance information deduced from the previous trials. The tuner's objective was to maximize mean rewards over a number of episodes. The following hyper-parameters were tuned for the first set of trials:

- Learning Rate: $lr \in U([10^{-6}, 10^{-5}])$
- Discount Factor: $\gamma \in U([0.9, 0.99])$
- GAE Lambda: $\lambda \in U([0.9, 0.99])$
- Train Batch Size: $BatchSize \in U([2048, 8192])$
- Actor and Critic Hidden Layer Structure $\in U([[32, 32], [64, 64], [128, 128], [256, 256]])$

The following hyper-parameters were tuned for the second set of trials:

- Learning Rate: $lr \in U([10^{-6}, 10^{-5}])$
- Discount Factor: $\gamma \in U([0.9, 0.99])$
- GAE Lambda: $\lambda \in U([0.9, 0.99])$
- Train Batch Size: $BatchSize \in U([2048, 8192])$
- Actor and Critic Hidden Layer Structure: $NNsize \in U_{discrete}([[256, 256], [512, 512], [1024, 1024]])$

The Adam back propagation optimizer was chosen for updating the actor and critic networks as it is recommended to use with the PPO algorithm and is the Ray RLlib default optimizer. Furthermore, since the actor and critic networks were set to hold only 2 hidden layers, the expected difference in computational time between using ReLu activations and tanh was considered negligible and hence the Ray RLlib default tanh activation was used. Future work may include hyper-parameter tuning for various activation functions. In addition, 8 rollout workers were used for parallelized training. Furthermore all other hyper-parameters necessary for training were taken from Ray RLlib's default algorithm configuration dictionary for PPO - a number of the default settings are presented below.

- PPO Clip Parameter: $\epsilon = 0.3$

- PPO KL Coefficient: $KL = 0.2$
- PPO KL Target Coefficient: $KL_{tgt} = 0.01$
- Hidden Layers Activation: tanh
- Stochastic Gradient Descent (SGD) Iterations: 30
- SGD Minibatch Size: 128
- Back-Prop Optimizer: Adam
- Rollout Fragment Length: Auto (equals $\frac{TrainBatchSize}{N_{workers}}$)
- Exploration Type: Stochastic Sampling
- Framework: TensorFlow 2

3.5 Methodology Summary

This chapter describes the research methodology used for this thesis. It outlines the equations of motion for the dynamic model as well as the DRL problem formulation. The solution of the DRL problem incorporates the weights and biases of a trained actor network, which acts as a controller and represents a control law, and can instantaneously provide an optimal control solution given any helicopter state. The results using this methodology are presented in the next chapter.

IV. Results

4.1 Chapter Overview

This chapter discusses the results obtained in the hyper-parameter optimization phase, the training phase and the different test phases. In the hyper-parameter selection phase, the optimizing tuning algorithm ran two sets of 25 trials, where in each trial the tuner attempted to improve the mean reward obtained based on the performance of previous trials. In the training phase, the agent was trained using the best set of hyper-parameters from both trials of the previous phase. Lastly, the performance of the trained agent was tested, where examination of results included both analysis of the flight trajectories for specific initial flight conditions and also included analysis of a generated Height-Velocity (HV) diagram.

4.2 PPO Algorithm Hyper-Parameter Tuning

Two experiments consisting of 25 trials each were conducted in the hyper-parameter selection phase using the Distributed Asynchronous Hyper-Parameter Optimization (HyperOpt) black box optimizer. As explained in the previous chapter, the first experiment consisted of trials with neural networks smaller in size with respect to the default Ray RLlib Proximal Policy Optimization (PPO) actor and critic network size (holding 2 hidden layers with 256 nodes in each layer), and the second experiment consisted of trials with neural networks of larger size (holding 2 hidden layers with up to 1024 nodes in each layer). Figures 4.1 and 4.2 show the top 10 trials for each experiment and are labeled in order with respect to mean rewards accumulated, from most to least.

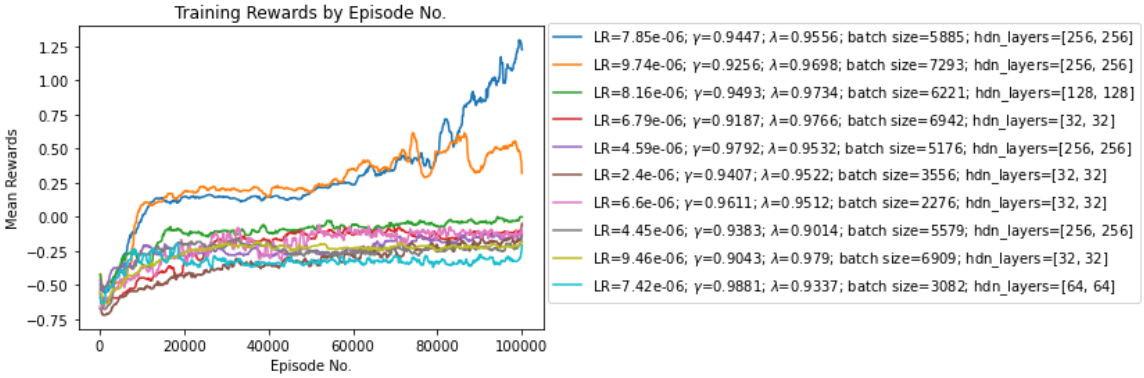


Figure 4.1. Ray Tune HyperOpt Results: Experiment 1

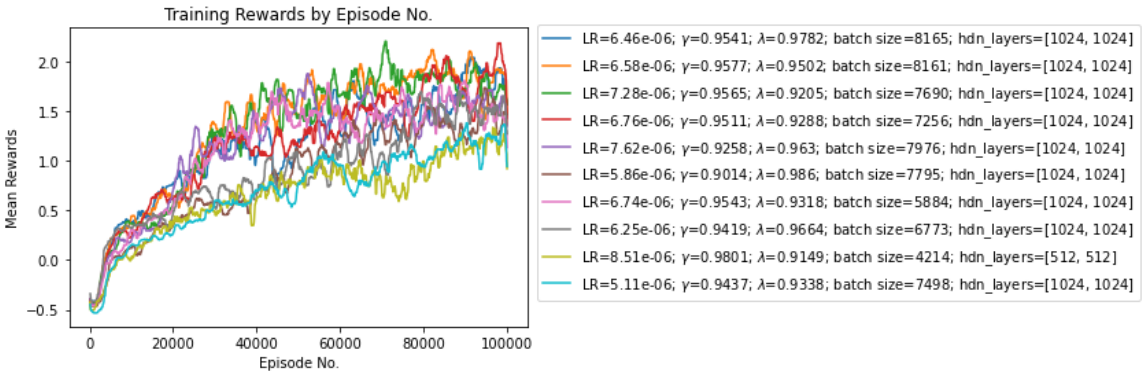


Figure 4.2. Ray Tune HyperOpt Results: Experiment 2

Figure 4.1 shows that the optimization in the first experiment generated mostly higher mean rewards for networks having closer to 256 nodes in each hidden layer, whereas majority of networks holding only 32 nodes in each hidden layer generated lower mean rewards. Moreover, all trials holding only 32 nodes in each hidden layer generated similar results, where the rewards plateaued after approximately 20,000 episodes of training. Lastly, the top 2 trials yielded similar reward values for the first 70,000 episodes, exhibiting a higher initial rate of increase of rewards with respect to the latter combinations and also plateauing after approximately 20,000 episodes. On the other hand, the rewards of the best trial increased again from 70,000 episodes whereas the second best trial exhibited instability of increasing and decreasing the

rewards. Overall, the best trial in experiment 1 generated a max reward of approximately 1.25 after 100,000 episodes.

Although the second experiment ran trials with networks of size [256,256] to [1024,1024], Figure 4.2 shows that 9 out of the 10 best trials were with 1024 nodes in each hidden layer, which was also the maximum hidden layer size defined for the experiment. Contrary to experiment 1, experiment 2 exhibited generally similar behavior between all trials, showing a steep increase in rewards for the first 10,000 episodes and then a more mild increase for the rest of the training episodes. Comparing to the best trial from experiment 1, the trials in experiment 2 show a monotonic increase in the rewards from a "zoomed out" perspective, although they also display signs of instability and noise from a "zoomed-in" perspective. In addition, none of the trials in experiment 2 plateaued to a max reward value. Lastly, the best trial in experiment 2 generated a max reward of approximately 2 after 100,000 episodes.

In summary, Figures 4.1 and 4.2 show in general that optimizing the hyperparameters for larger hidden layers gave in return larger mean rewards, but also exhibited more training noise and instabilities. This could possibly be compared and related to regular supervised learning where smaller neural networks provide higher bias but lower variance (thus exhibiting lower training performance but higher stability in the process), and larger neural networks yield a lower bias but higher variance and could possibly be over-trained if not careful.

In conclusion, based solely on the mean rewards accumulated, the neural networks of size [1024,1024] showed more favorable results. Furthermore, the best trial in experiment 1 which had a network size of [256,256] showed a smoother and more stable curve, and did not plateau. Hence, the best trials from both experiments were selected to be trained on a large number of episodes.

4.3 Training Results After Hyper-Parameter Selection

The following set of hyper-parameters were selected for long-term training based on the results presented above:

Table 4.1. Hyper-Parameters Selected After Tuning Phase

Parameter Name	Actor 1	Actor 2
Learning Rate (lr)	$7.85 * 10^{-6}$	$6.46 * 10^{-6}$
Discount Factor (γ)	0.9447	0.9541
GAE Lambda (λ)	0.9556	0.9782
Batch Size	5885	8165
Actor/Critic Hidden Layers Sizes	$[256, 256]$	$[1024, 1024]$

As mentioned previously, all other hyper-parameters relevant to the Ray RLlib PPO algorithm were kept at the default values defined in Ray. Below are the training results for each set of hyper-parameters.

4.3.1 Training Results: Actor 1

Figure 4.3 shows the PPO training results for the hyper-parameters of actor 1. The training session was run for approximately 1,100,000 episodes, which equated to approximately 19000 training batch updates. On average, a training batch contained approximately 61 full episodes.

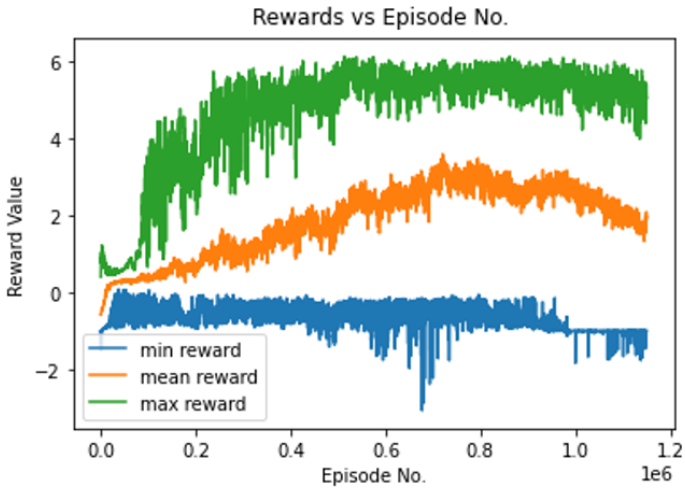


Figure 4.3. PPO Training Results for Actor 1

The max rewards represented the highest reward obtained in an episode within one training batch, the mean rewards represented the average reward of all episodes within one training batch and the minimum reward represented the minimum reward obtained in an episode within one training batch. Figure 4.3 shows a monotonic improvement of the mean rewards until approximately 700,000 episodes, for which after an "un-learning" phenomenon occurred. One of the main purposes of tuning hyper-parameters, aside from increasing training performance, was to eliminate the possibility of the un-learning phenomenon which in many cases PPO suffers from due to the nature of the surrogate objective. The hyper-parameters were selected based on performance obtained during the first 100,000 episodes only, and hence the late un-learning effects could not be predicted.

Due to this occurrence, a hypothesis was proposed which claimed that in the case of the helicopter environment, the optimal points of the objective function could possibly be surrounded by many "cliffs", the paths to the optimal points are very narrow, and the trust region generated by the PPO clipping factor is not sufficient for such circumstances. The hypothesis claimed that decreasing the learning rate at the peak reward value could reduce the chance of gradient stepping towards a cliff,

thus allowing the reward to increase even more rather than to decrease.

Figure 4.4 below shows a comparison between the training results before reducing the learning rate, and after reducing the learning rate to an arbitrary value of $lr = 4 * 10^{-6}$. Figure 4.4 shows that arbitrarily decreasing the learning rate allowed the mean reward to increase even more over a span of about 150,000 episodes, rather than to decrease, and thus the hypothesis was proven true.

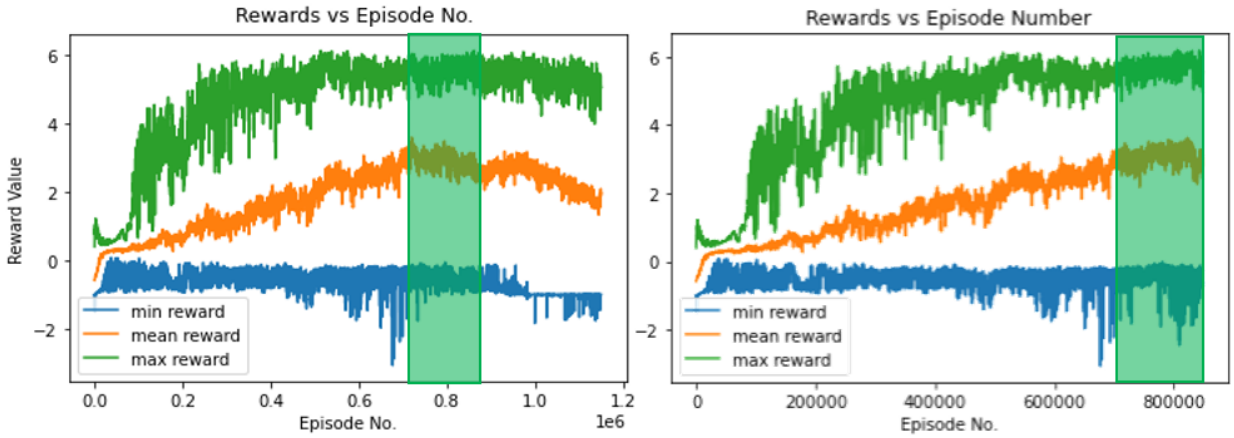


Figure 4.4. Left: Original Training Results. Right: Reduced Learning Rate Training Results

Following the conclusion drawn above, an arbitrary annealing of the learning rate was defined for the same combination of hyper-parameters. The initial learning rate was set at a higher value of $5 * 10^{-5}$ with respect to the original learning rate in order to account for the fact that the learning rate anneals to order of magnitudes significantly lower over time. The learning rate was arbitrarily decreased by 1% for every 10 training batches (approx. 610 episodes). Figure 4.5 shows the learning rate annealing schedule.

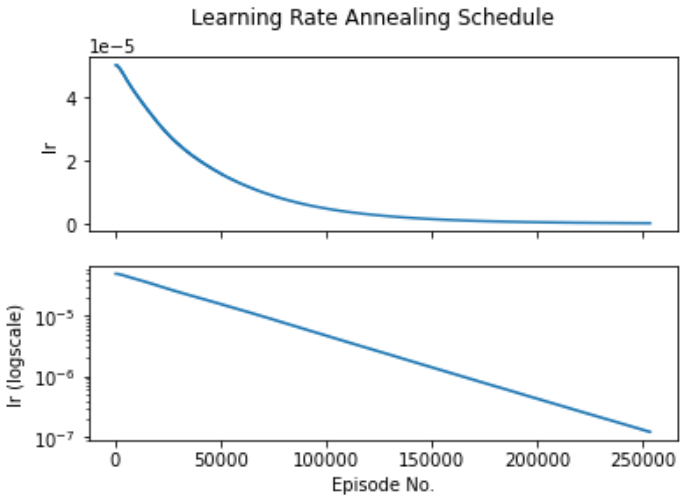


Figure 4.5. Learning Rate Annealing Schedule for Actor 1

Figure 4.6 shows the training results for the PPO agent with the newly defined annealing learning rate.

Figure 4.6. PPO Training Results for Actor 1 with Learning Rate Annealing

Figure 4.6 shows a non-monotonic but overall increase of both the mean and max rewards until their max values, where they remain at those values without decreasing for the duration of the training process. Training time was approximately 12 hours for approximately 250,000 episodes (equivalently 6000 training batches), on a computer

utilizing all 20 CPU's with 8 rollout workers - which equated to an average training rate of $20,833[\frac{episodes}{hr}]$.

4.3.2 Training Results: Actor 2

Actor 2 exhibited similar behavior compared to actor 1 without learning rate annealing, where initially the mean rewards increased in the training process and then decreased.

As executed previously, the learning rate was also annealed in order to overcome the un-learning phenomenon. The learning rate was arbitrarily decreased by 1% for every 10 training batches (approx. 610 episodes), and the initial learning rate was also slightly increased. Figure 4.7 below shows the learning rate annealing schedule for actor 2.

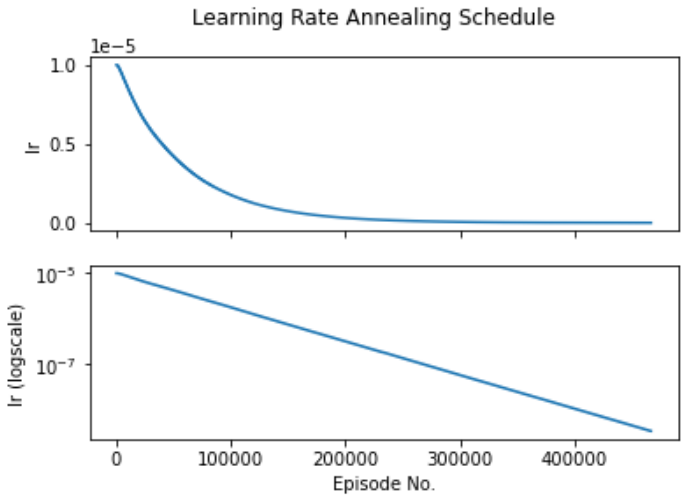


Figure 4.7. Learning Rate Annealing Schedule for Actor 2

Figure 4.8 below shows the training results for actor 2 after annealing the learning rate over the training episodes. Figure 4.8 shows similar mean and max rewards when compared to Figure 4.6, although instances of training instability are observable - such as the large dip at approximately 180,000 episodes.

Figure 4.8. PPO Training Results for Actor 2 with Learning Rate Annealing

Training time was approximately 44 hours for approximately 490,000 episodes (equivalently 8000 training batches), on a computer utilizing all 20 CPU's with 8 rollout workers - which equated to an average of training $11,136[\frac{episodes}{hr}]$. Actor 2 trained at a rate approximately 2 times slower than actor 1, which was was expected due to the larger actor and critic network size.

4.4 Testing of Trained Agents

The trained PPO agents for both actor 1 and actor 2, with added learning rate annealing, were tested in ideal conditions, meaning that no effects such as winds, gusts or any other disturbances were added to the testing environment. Furthermore, no action/control input delays were added to the testing environment. The non-ideal effects mentioned above may be researched and tested in future work.

4.4.1 Test Results: Actor 1

Figure 4.9 shows an HV diagram defined by discrete data points. The diagram depicts 400 terminal data points with initial heights between 0-600[ft] above ground

level (AGL) and initial velocities between 0-50[KTAS]. Each terminal data point can hold 3 different categories; non-lethal landing (green), lethal landing (red) and early termination of the episode due to exceeding of state limit (blue). As mentioned previously, a non-lethal landing is defined as a landing where the rate of descent (ROD) on touchdown is smaller than $5[\frac{ft}{s}]$ and the ground-speed of the helicopter on touchdown in smaller than $10[knots]$, which equates $10[KTAS]$ without winds. Figure 4.9 shows that the majority of simulated trajectories terminated with a non-lethal landing. Moreover, contrary to the generated results it was expected that initial conditions in the lower to upper hover region would terminate with lethal landings, as depicted for instance in the HV diagram for the single engine AH-1F [26]. Lastly, initial conditions of high energy, that is the sum of kinetic and potential energy, also terminated with either lethal landings or early termination due to exceeding of a state limit.

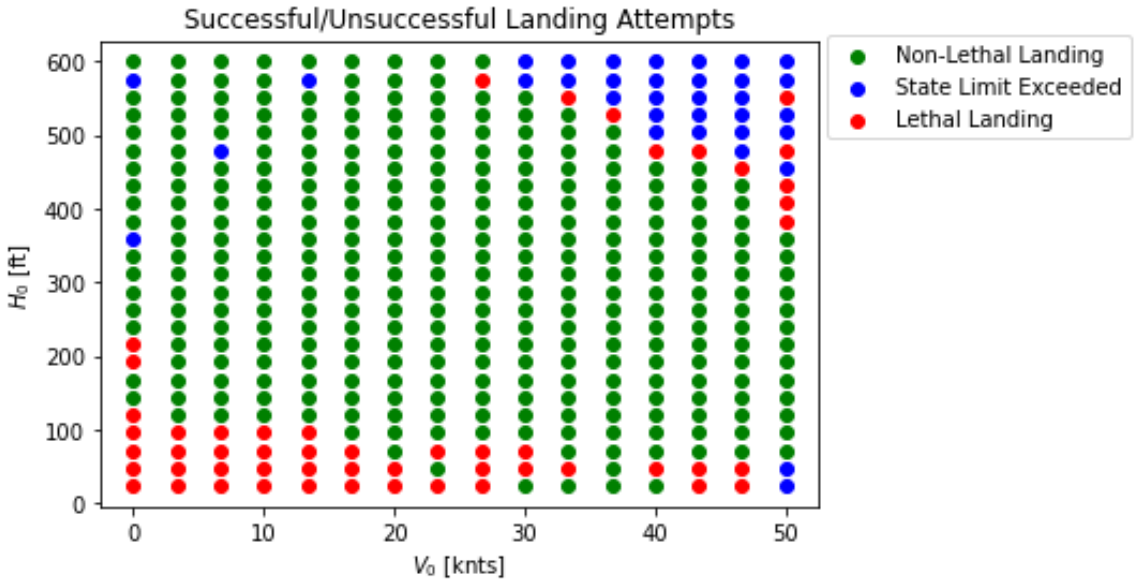


Figure 4.9. HV Diagram for Actor 1

In order to analyze the episodes that terminated early or the lethal landings which occurred when simulating high initial energies, the total energies were plotted at the

beginnings and ends of every episode. The energy was defined as the sum of the kinetic and potential energy heights as shown in Equation 4.1:

$$\bar{E} = \bar{E}_k + \bar{E}_p = h + \frac{1}{2g}(u^2 + w^2) \tag{4.1}$$

Figure 4.10 shows the initial energies and terminal energies of the helicopter for all initial conditions in the defined bounds.

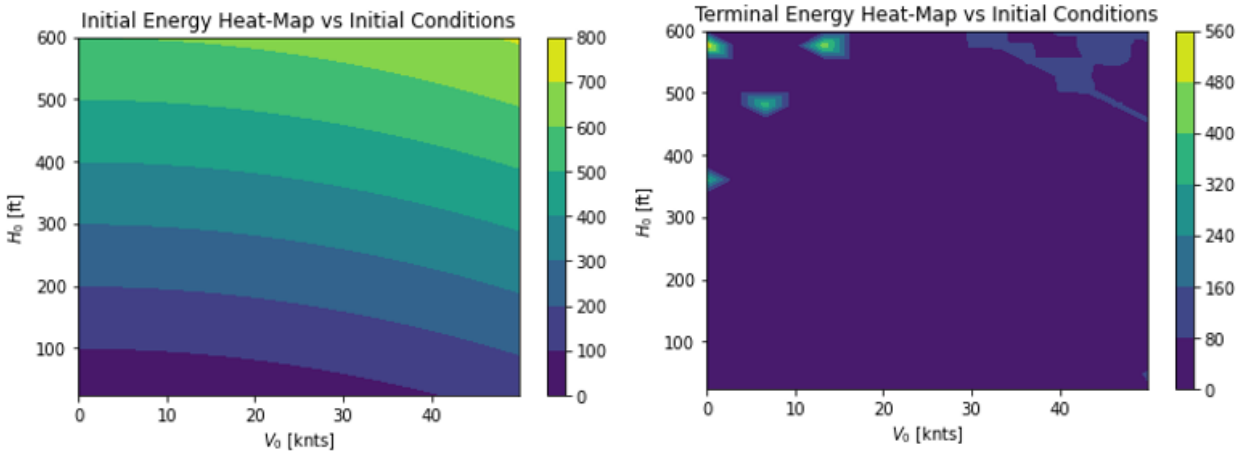


Figure 4.10. Initial (left) and Terminal (right) Energies vs Initial Conditions for Actor 1

Figure 4.10 shows small and randomly scattered areas of high terminal energies which lay close to the hover region of the HV diagram. This observation coincides with the initial conditions closer to the hover region that terminated early. From further analysis it appeared that those specific initial conditions had high variance in the results, where sometimes an episode would terminate early due to exceeding a state limit, and sometimes an episode would terminate with a non-lethal landing. In addition, when comparing Figures 4.9 and 4.10, lethal landing points and early termination points in the high initial energy region all terminated with very low energies. From this observation it is possible that the agent needed some more training

episodes. In addition, it may be necessary to favor higher initial energy conditions than lower initial energy conditions, due to the fact that higher initial energy conditions generate longer episodes and thus require more training, and solutions of lower initial energies are anyways encapsulated in solutions of higher initial energies.

4.4.2 Test Results: Actor 2

Figure 4.11 shows the generated HV diagram for actor 2. Compared to the HV diagram for actor 1 (Figure 4.9), there are significantly fewer non-lethal terminal landings and more lethal landings. Furthermore, the high initial energy region ends with lethal landings or early termination, similar to the HV diagram for actor 1.

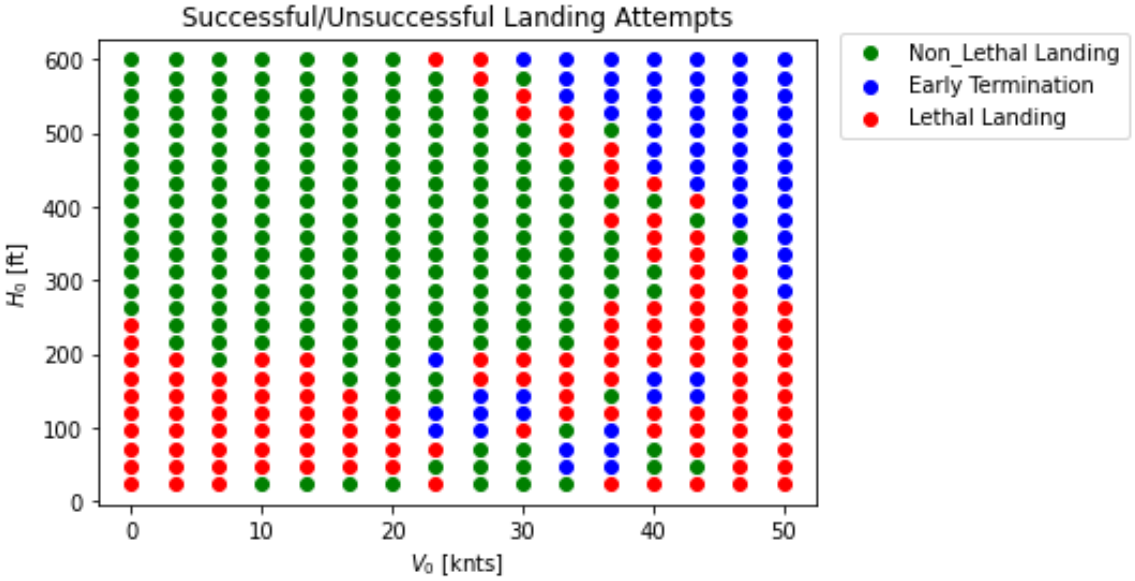


Figure 4.11. HV Diagram for Actor 2

Figure 4.12 shows smoother results with respect to the terminal energy diagram of actor 1, hinting that the test variance may be smaller. In addition, the majority of initial conditions terminate with low energy with the exception of the high initial energy states. When comparing between the results of actor 1 and actor 2, it may be possible that although the results initially look less promising, the larger actor and

critic networks have more potential to give better results. The author's hypothesis is that the larger networks (such as the one of actor 2) have a higher potential of generating desirable results due to higher network capacity, but the selection of hyper-parameters must be done carefully as the network is more sensitive to instabilities in training. Furthermore, smaller sized networks are quicker to train and encounter fewer instabilities during training, but can be limited by their capacity to cover all initial conditions without high test variance.

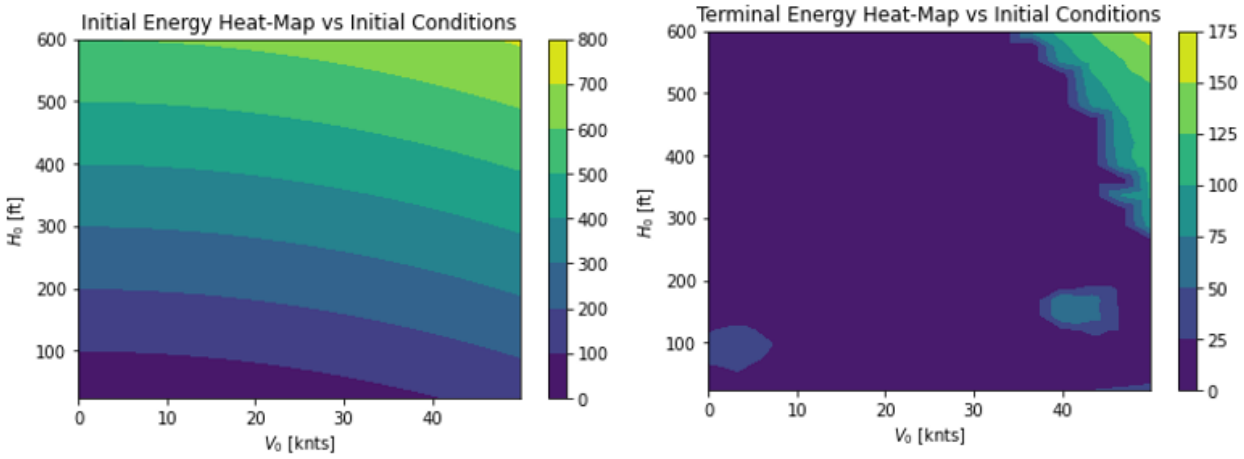


Figure 4.12. Initial (left) and Terminal (right) Energies vs Initial Conditions for Actor 2

4.4.3 Summary of Test Results

In summary, actor 1 generated over-all better results than actor 2. Again, this may not necessarily mean it is recommended to use a smaller network. The hyper-parameter tuning phase consisted of only 100,000 episodes with respect to the full training sessions which consisted of approximately 400,000 episodes. Ideally, the hyper-parameter phase would need to also run for at-least 400,000 episodes in order to truly distinguish between good and bad hyper-parameters. Furthermore, the annealing schedule was added to both combinations arbitrarily and ideally would also require tuning along with the other hyper-parameters. The author's hypothesis is

that by tuning **all** relevant hyper-parameters, and by increasing training times in the tuning phase, larger actor/critic networks will outperform smaller networks.

4.4.4 Flight Trajectory Analysis of Trained Agent

This section will analyze in short the generated flight trajectories of actor network 1. Figures 4.13-4.19 show the simulation results for an initial height of $H_0 = 300[ft]$ AGL and initial velocity of $V_0 = 30[KTAS]$, which is in the center of the HV diagram.

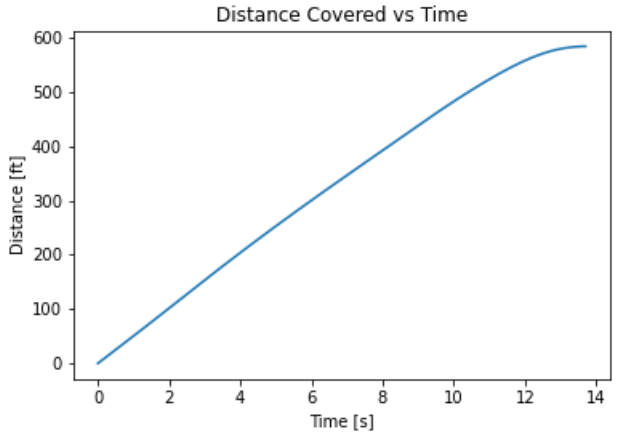


Figure 4.13. Actor 1: Distance vs Time - $H_0 = 300[ft]$, $V_0 = 30[KTAS]$

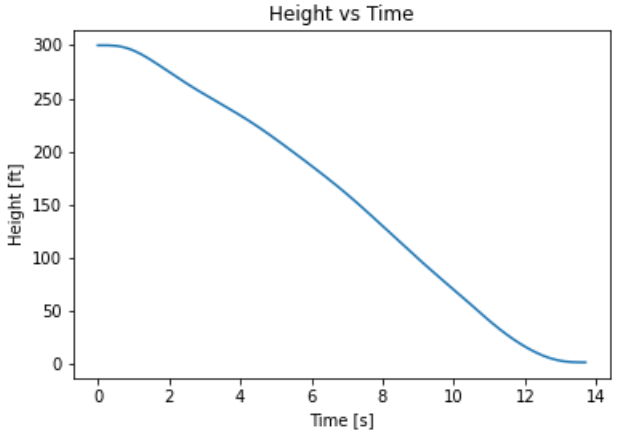


Figure 4.14. Actor 1: Height vs Time - $H_0 = 300[ft]$, $V_0 = 30[KTAS]$

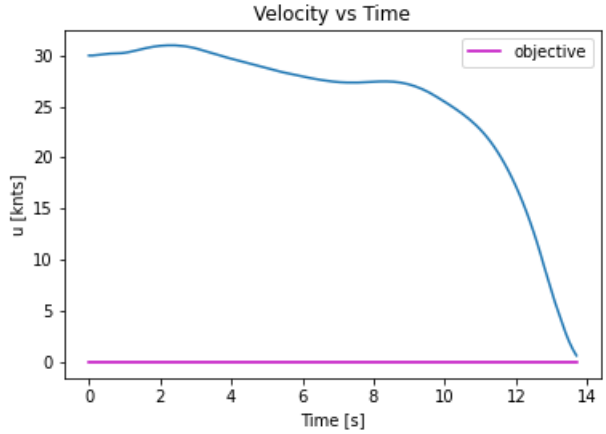


Figure 4.15. Actor 1: Velocity vs Time - $H_0 = 300[ft]$, $V_0 = 30[KTAS]$

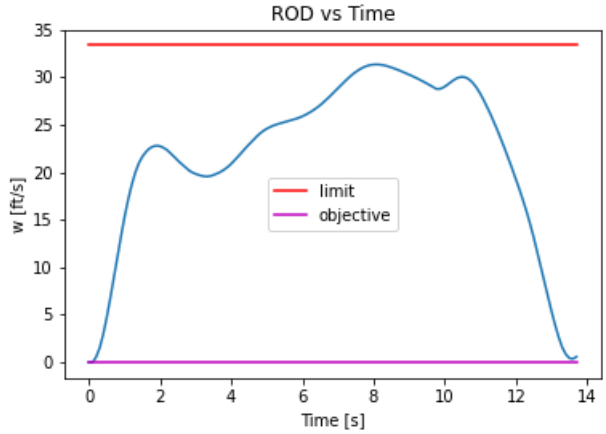


Figure 4.16. Actor 1: Rate of Descent vs Time - $H_0 = 300[ft]$, $V_0 = 30[KTAS]$

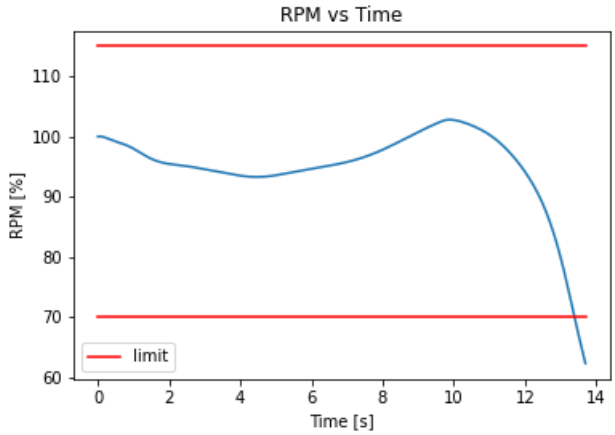


Figure 4.17. Actor 1: Main Rotor RPM vs Time - $H_0 = 300[ft]$, $V_0 = 30[KTAS]$

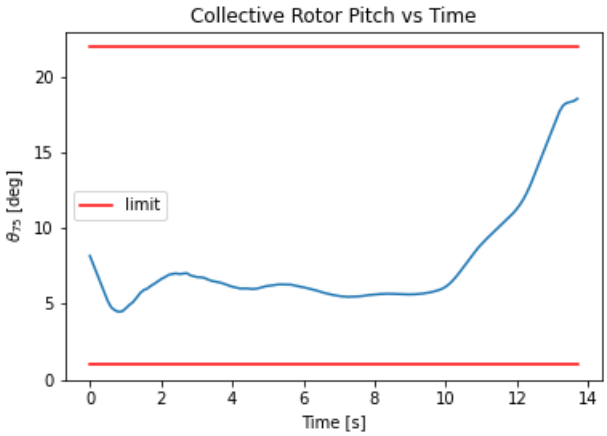


Figure 4.18. Actor 1: Main Rotor Collective Pitch vs Time - $H_0 = 300[ft]$, $V_0 = 30[KTAS]$

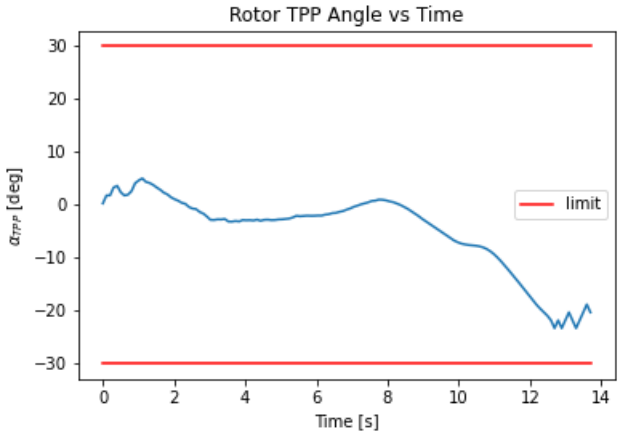


Figure 4.19. Actor 1: Tip Path Plane Angle vs Time - $H_0 = 300[ft]$, $V_0 = 30[KTAS]$

Figures 4.14-4.16 show that both the velocity and rate of descent were minimized close to 0. It is important to mention that although Figure 4.17 shows a surpassing of the minimum rotational velocity (RPM) limit, this was in-fact allowed for heights under $10[ft]$ in both the training and test phases since there is no real importance of maintaining the RPM in the operational bounds when close to the ground in an emergency scenario. Figure 4.18 shows that the main rotor collective pitch was maintained at a low value for majority of the trajectory and gradually increased towards the landing, which is the expected procedure when landing without power. Furthermore, the

trade-off between the main rotor collective pitch and RPM is evident, as expected in a real life procedure. Also, Figure 4.19 shows that the helicopter maintained a Tip Path Plane (TPP) angle parallel to the horizon for majority of the trajectory and gradually "decreased" into a flare towards the landing (note: positive TPP angle was defined as nose down and vice versa). On the other hand, the TPP angle was approximately $20[deg]$ nose up on landing which may be problematic for certain helicopters in a real life scenario. As opposed to a point mass model, defining the exterior dimensions of the modeled helicopter in future work will allow the agent to take into account possibilities of tail-strike. Simulating such occurrences may reduce the number of non-lethal landings generated by the PPO algorithm, and thus increase the size of the no-fly zone in the HV diagram.

Figures 4.20 and 4.21 compare the results of an initial condition with high test variance for actor 1, where one simulation terminated with a non-lethal landing and the other terminated early due to exceeding a state limit. The simulated initial condition was hover at a height of $H_0 = 600[ft]$ AGL:

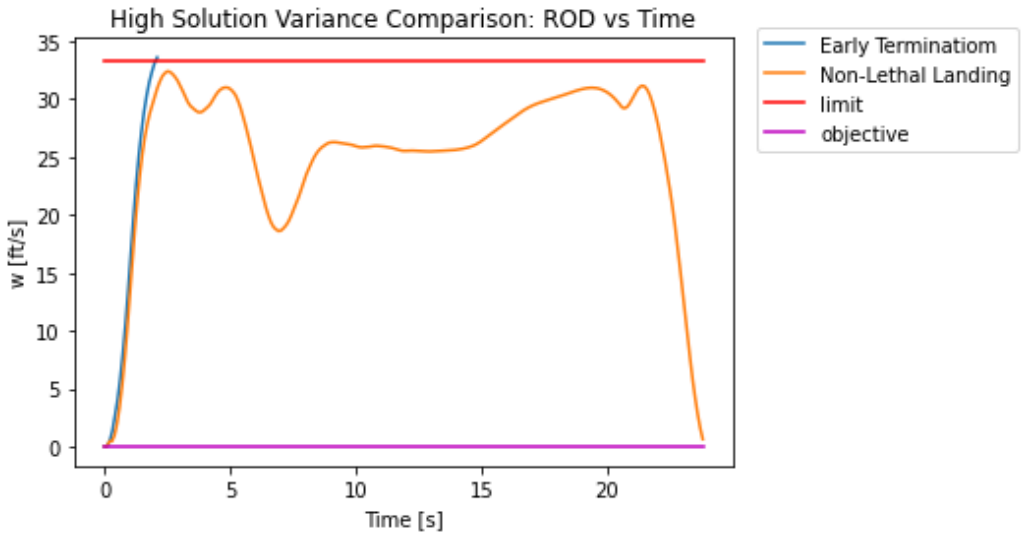


Figure 4.20. Actor 1: Early Termination (Left) vs Non-Lethal Landing (Right) - $H_0 = 600[ft]$, $V_0 = 0$

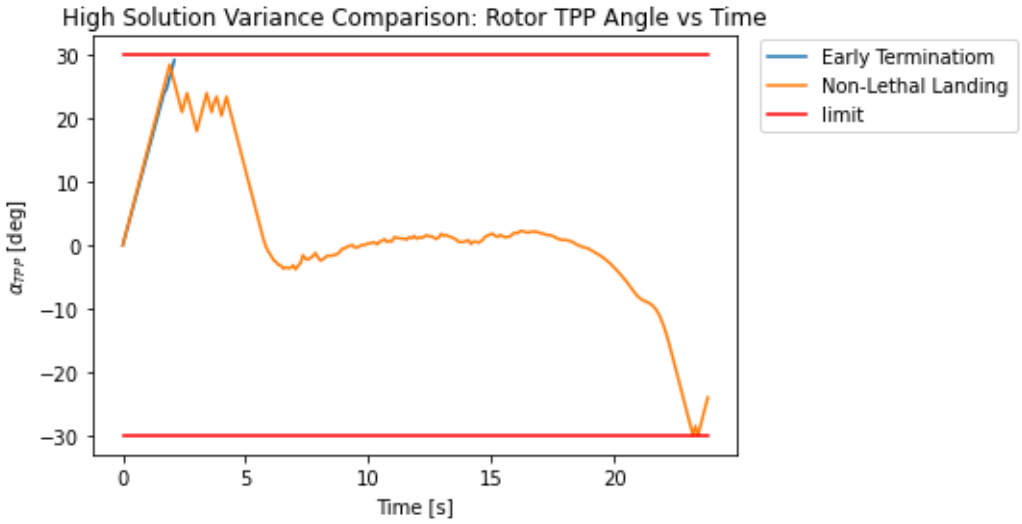


Figure 4.21. Actor 1: Early Termination (Left) vs Non-Lethal Landing (Right) - $H_0 = 600[ft]$, $V_0 = 0$

It is clear from Figures 4.20 and 4.21 that the most influential states varied greatly between two different runs, where in the failed run the helicopter pitched down to the max TPP angle and also surpassed the allowed ROD, and in the successful run the TPP angle and the ROD were maintained within the allowed bounds although were very close to surpassing their limits. It is possible that allowing a temporary deviation from the mission limits may decrease the number of early terminated episodes, especially those corresponding to initial conditions which exhibited high test variance. See Appendix A for trajectory figures for a variety of initial conditions for both actor 1 and actor 2.

4.5 Chapter Summary

This chapter presented the results of the hyper-parameter tuning phase, the full training phase and the test phase. The top combination of hyper-parameters was selected for each tuning experiment for further training. One experiment consisted of small hidden layers and the latter consisted of larger hidden layers. Arbitrary learning

rate schedules were then added to both hyper-parameter combinations for improved training performance. Finally, both actors (corresponding to both hyper-parameter combinations) were tested and compared to each other. In summary, feasibility for using the PPO algorithm to generate a control law for a helicopter in a total power loss was proven, and with further improvement could yield a powerful tool for real-time application.

V. Conclusions and Recommendations

5.1 Conclusions

All of the research objectives listed in Chapter 1 were achieved. A mathematical model for helicopter dynamics was developed and implemented into the Deep Reinforcement Learning (DRL) problem. Furthermore, the Proximal Policy Optimization (PPO) algorithm was successfully implemented in training an actor neural network to bring a helicopter to a safe landing in a total power loss scenario. Also, the PPO algorithm was able to generate trajectories over a variety of initial conditions inside the Height-Velocity (HV) diagram, using one actor network (as opposed to numerous actor networks for different initial conditions). Overall, the PPO algorithm was deemed a feasible candidate for generating a control law for a helicopter in a power loss scenario, which may allow for future development of cockpit pilot aid products. Below are the derived conclusions for various components of the research.

5.1.1 Comparison to Previous Work

As mentioned before, Kopsa [24] successfully implemented the Asynchronous Advantage Actor Critic (A3C) algorithm to generate optimal control trajectories for a helicopter in a total power scenario. Kopsa investigated a significantly lighter helicopter, weighing about 5 times lighter than the helicopter defined in this thesis study, which could be a reason for the difference in results. Also, Kopsa used a 6 state dynamic helicopter model whereas an 8 state dynamic model was developed and used for this thesis. Kopsa managed to train an actor network with a hidden layer size of $[32, 32]$ and with ReLu activation functions to successfully bring a helicopter to a safe landing for majority of initial conditions, with tighter touchdown velocity limits. On the other hand, the smallest optimal hidden layer size derived from Distributed

Asynchronous Hyper-Parameter Optimization (HyperOpt) in this thesis study was $[256, 256]$ and using a hidden layer size of $[32, 32]$ was inadequate with tanh activations. Also, Kopsa used the RMSprop optimizer for updating the actor/critic network weights whereas the Adam optimizer was used in this thesis. Both theses have proven the feasibility of using DRL algorithms for such objectives, although the difference in environments, strategies and results may indicate that the DRL algorithms aren't globally effective - meaning that different helicopter models for instance may perform better with different DRL algorithms, re-tuned hyper-parameters and variable training strategies.

5.1.2 Overall Training and Test Results

As mentioned in Chapter 4, actor 1 which has a hidden layer size of $[256, 256]$ produced better results than actor 2 which had a hidden layer size of $[1024, 1024]$. Actor 1 managed to bring the helicopter to a non-lethal landing over most initial conditions, especially those in the region expected to be lethal as observed for instance in the single engine AH-1F HV diagram [26]. As mentioned previously, it is expected that modeling the exterior dimensions of the helicopter would've allowed for simulation of tail-strike, thus possibly reducing the amount of non-lethal landings which don't correspond to the no-fly zones in HV diagrams such as that of the AH-1F. Also aforementioned, actor 2 did not generate high test variance in certain areas of the HV whereas actor 1 did. Hence, the data generated from the tuning, training and test phases is not sufficient to conclude which hidden layer size is preferable. An optimization tuning over a larger number of episodes, along with tuning hyper-parameters which weren't tuned in the thesis study (such as learning rate annealing parameters, different activation functions, rollout length, batch size horizon, mini-batch size and number of SGD iterations) may improve overall results and may allow for a definitive

conclusion for the recommended actor/critic network size.

As observed from both actors, simulated trajectories from the high initial energy regions terminated early probably due to lack of training. As mentioned before, a workaround may be to select initial conditions for training based on a non-uniform probability distribution, where the probability density is somewhat proportional to the initial energy, where higher initial energies are favored with higher probabilities of selection. This is expected to prioritize training on higher initial energy conditions, which encapsulate a lot of information from lower initial energy conditions.

Overall, the PPO algorithm generated successful trajectories without any intermediate rewards, which are expected to be more optimal than solutions with intermediate rewards since the total reward is related directly to the agent mission objective.

5.2 Recommendations for Future Work

The work accomplished during this research can be continued and improved upon in a number of ways. Below are some areas for future work and improvement.

5.2.1 Improvements to the Hyper-Parameter Tuning

The hyper-parameters were only optimized over 100,000 episodes for each of the 50 trials that were run, which took in total about 7 days. Future work may consist of using a dedicated machine learning computer with a strong GPU to tune the hyper-parameters over more episodes, and to also train the agent quicker. Again, tuning over a larger variety of hyper-parameters may also improve overall results.

5.2.2 Training Modifications and Experiments

As mentioned previously, a probability density function will be implemented for the initial condition selection, with higher probabilities favoring higher initial energy

conditions. Also, a simplistic helicopter model was used for the case study to prove feasibility. For future work, experimentation with higher fidelity 3 dimensional and 6 degrees of freedom helicopter models may be necessary in order to actually develop visual aid algorithms for real use. Such models will definitely have to take into account the exterior dimensions of the helicopter, blade bending, torsion, lag and rotor disk tilt in all degrees of freedom. Modeling the exterior dimensions of the helicopter may allow for investigation of the high velocity low height no fly zone in the HV diagram, as well as disqualify any solutions which would terminate with a tail-strike. Also, such models will have to take into account actuator lag and disturbances, and sensor errors and limitations.

In addition, an arbitrary reward structure was selected for this case study and hence further investigation of other reward structures will be considered in future work.

5.2.3 Further Test Scenarios

The agents trained in the case study were tested in fully deterministic and ideal environmental conditions. Future work will test the trained agents robustness to wind and gust effects, input lag, input bias and input noise, along with simulated sensor errors and coping with irregular sensor observations. If the test results were to show low robustness, the agent may need to be trained on such variable simulated conditions too.

Lastly, only one nominal aircraft weight was considered in this investigation. Future research will test the robustness of the algorithm on other various weights.

Appendix A.

I Test Results for Actor 1

a Initial Conditions: H0=50[ft], V0=30[KTAS]

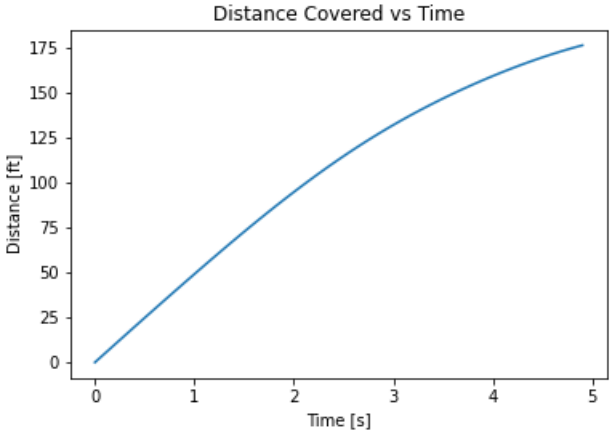


Figure A.1. Actor 1: Distance vs Time - $H_0 = 50[ft]$, $V_0 = 30[KTAS]$

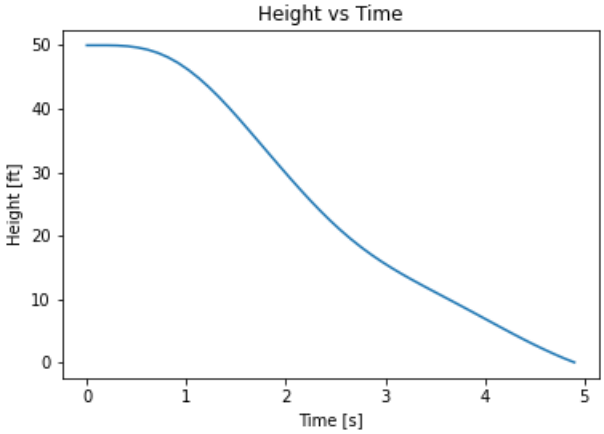


Figure A.2. Actor 1: Height vs Time - $H_0 = 50[ft]$, $V_0 = 30[KTAS]$

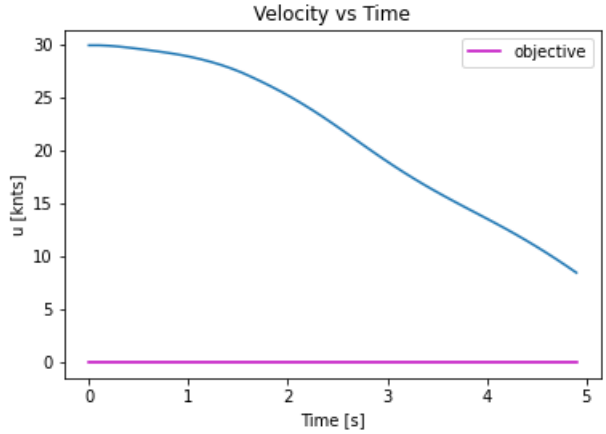


Figure A.3. Actor 1: Velocity vs Time - $H_0 = 50[ft]$, $V_0 = 30[KTAS]$

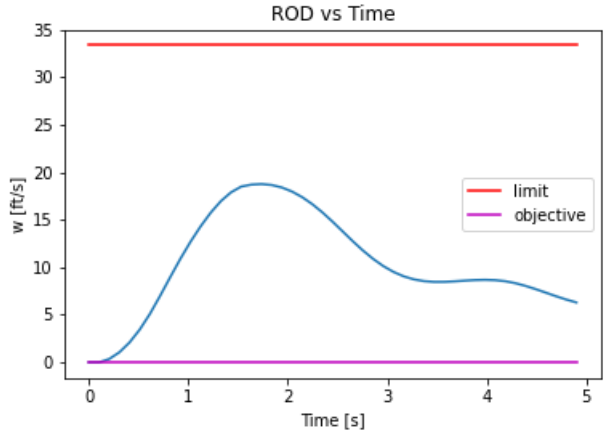


Figure A.4. Actor 1: Rate of Descent vs Time - $H_0 = 50[ft]$, $V_0 = 30[KTAS]$

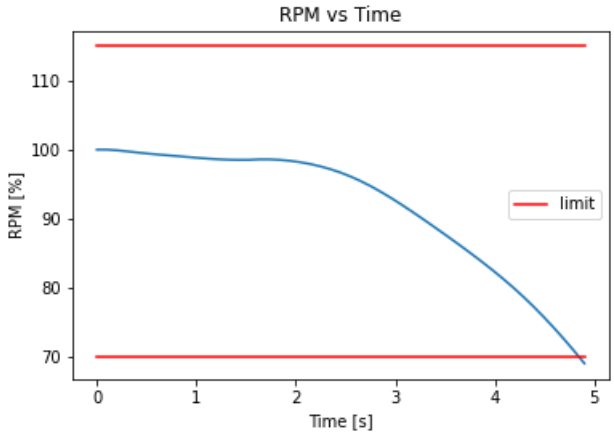


Figure A.5. Actor 1: Main Rotor RPM vs Time - $H_0 = 50[ft]$, $V_0 = 30[KTAS]$

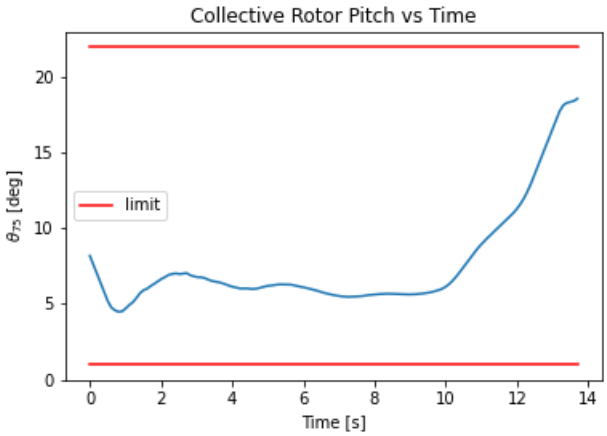


Figure A.6. Actor 1: Main Rotor Collective Pitch vs Time - $H_0 = 50[ft]$, $V_0 = 30[KTAS]$

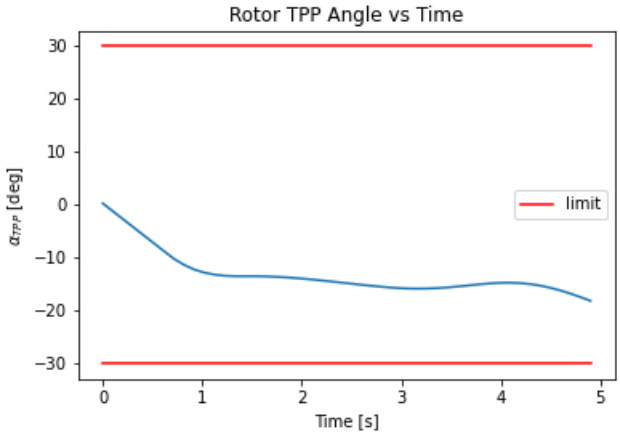


Figure A.7. Actor 1: Tip Path Plane Angle vs Time - $H_0 = 50[ft]$, $V_0 = 30[KTAS]$

b Initial Conditions: H0=100[ft], V0=0[KTAS]

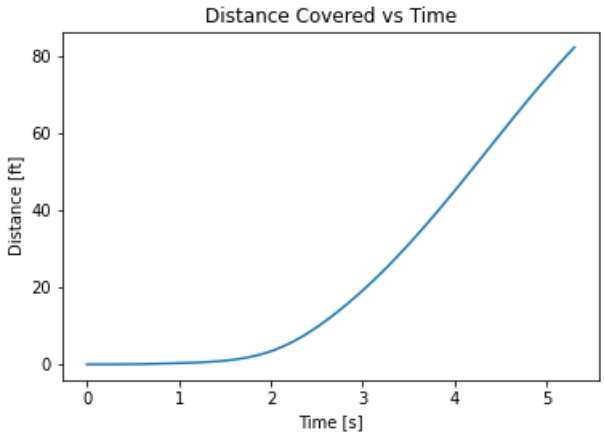


Figure A.8. Actor 1: Distance vs Time - $H_0 = 100[ft]$, $V_0 = 0[KTAS]$

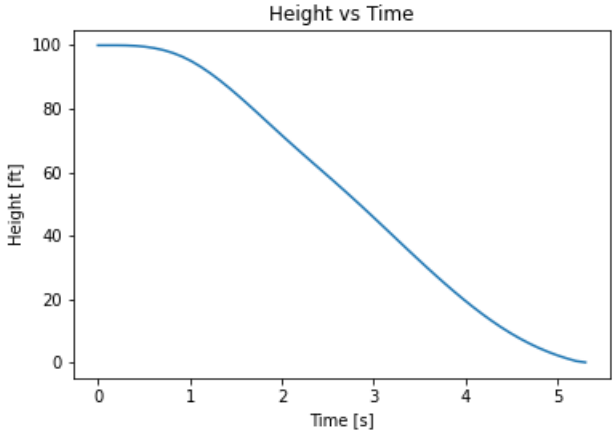


Figure A.9. Actor 1: Height vs Time - $H_0 = 100[ft]$, $V_0 = 0[KTAS]$

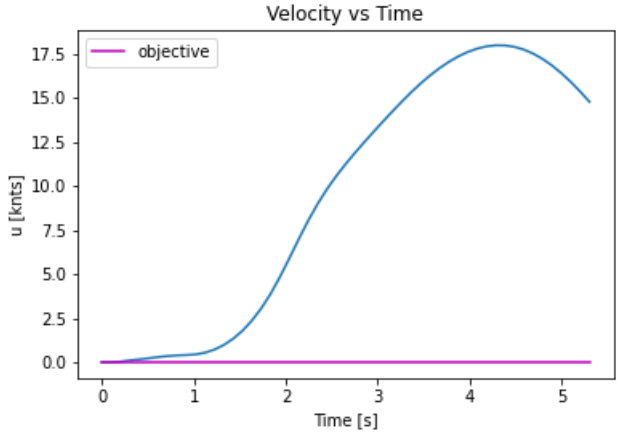


Figure A.10. Actor 1: Velocity vs Time - $H_0 = 100[ft]$, $V_0 = 0[KTAS]$

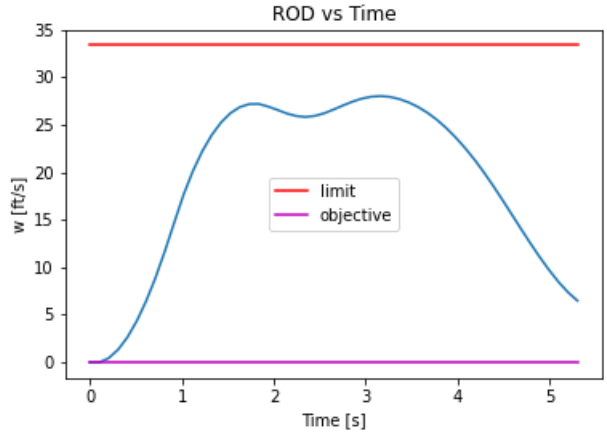


Figure A.11. Actor 1: Rate of Descent vs Time - $H_0 = 100[ft]$, $V_0 = 0[KTAS]$

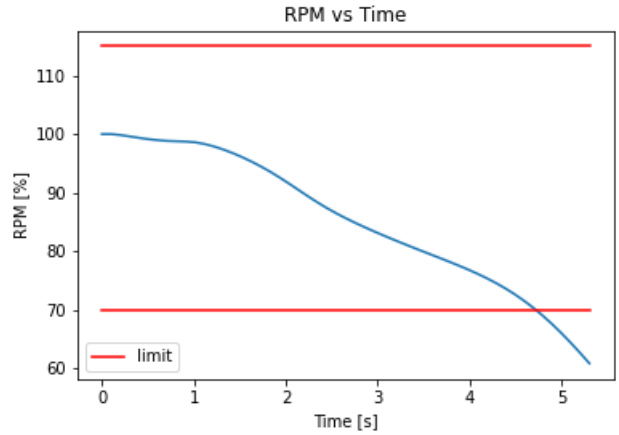


Figure A.12. Actor 1: Main Rotor RPM vs Time - $H_0 = 100[ft]$, $V_0 = 0[KTAS]$

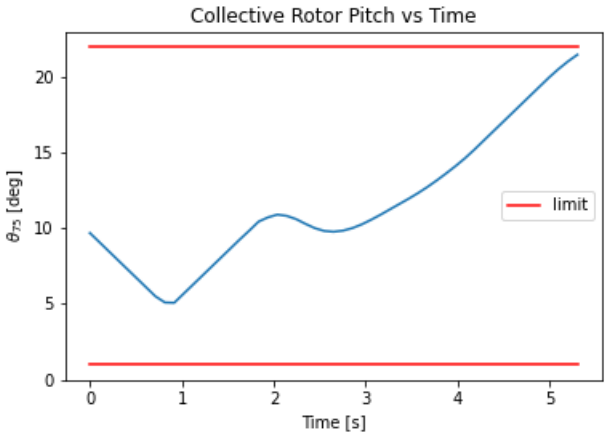


Figure A.13. Actor 1: Main Rotor Collective Pitch vs Time - $H_0 = 100[ft]$, $V_0 = 0[KTAS]$

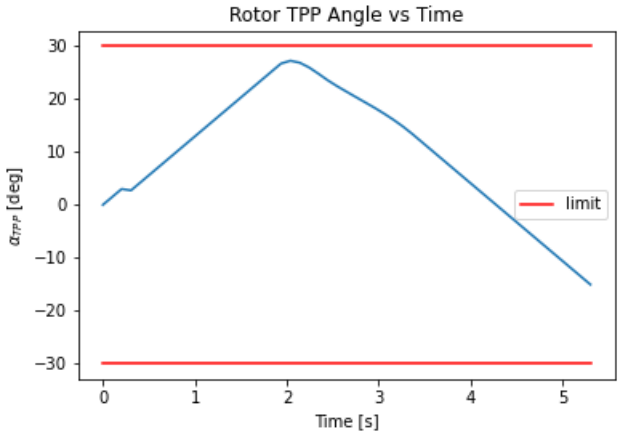


Figure A.14. Actor 1: Tip Path Plane Angle vs Time - $H_0 = 100[ft]$, $V_0 = 0[KTAS]$

c Initial Conditions: H0=300[ft], V0=0[KTAS]

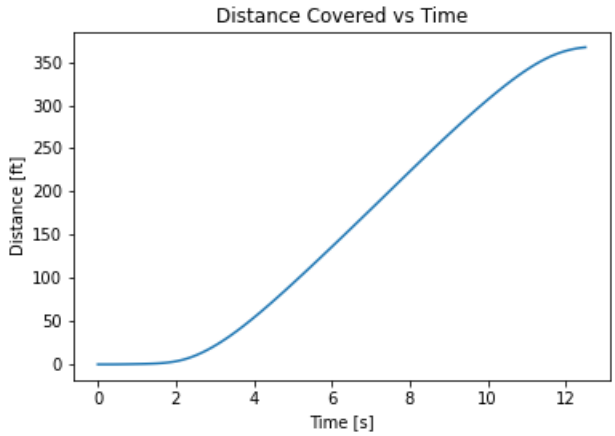


Figure A.15. Actor 1: Distance vs Time - $H_0 = 300[ft]$, $V_0 = 0[KTAS]$

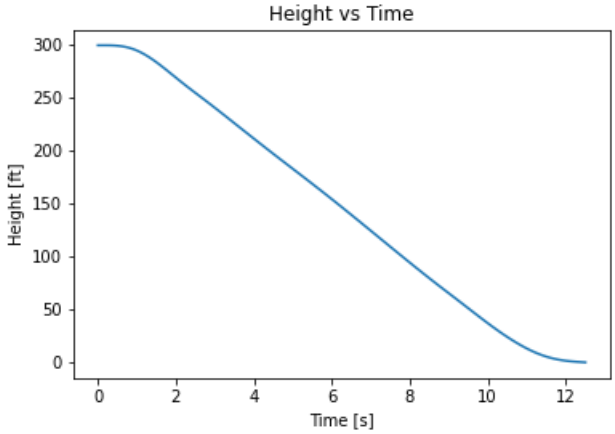


Figure A.16. Actor 1: Height vs Time - $H_0 = 300[ft]$, $V_0 = 0[KTAS]$

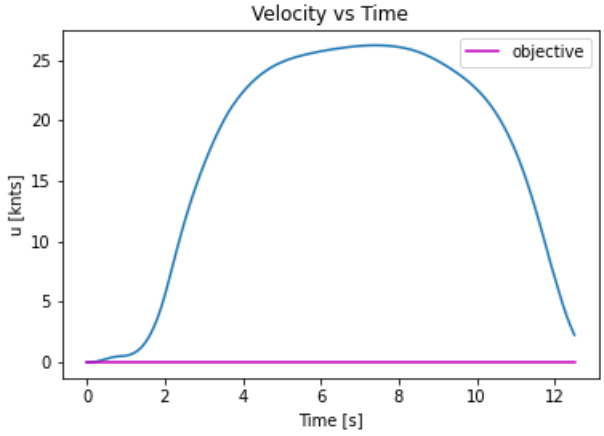


Figure A.17. Actor 1: Velocity vs Time - $H_0 = 300[ft]$, $V_0 = 0[KTAS]$

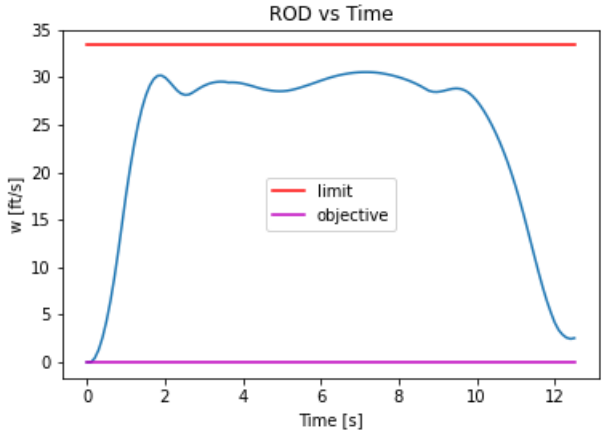


Figure A.18. Actor 1: Rate of Descent vs Time - $H_0 = 300[ft]$, $V_0 = 0[KTAS]$

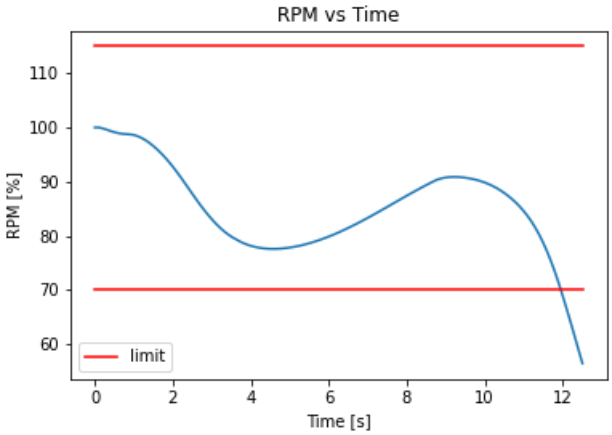


Figure A.19. Actor 1: Main Rotor RPM vs Time - $H_0 = 300[ft]$, $V_0 = 0[KTAS]$

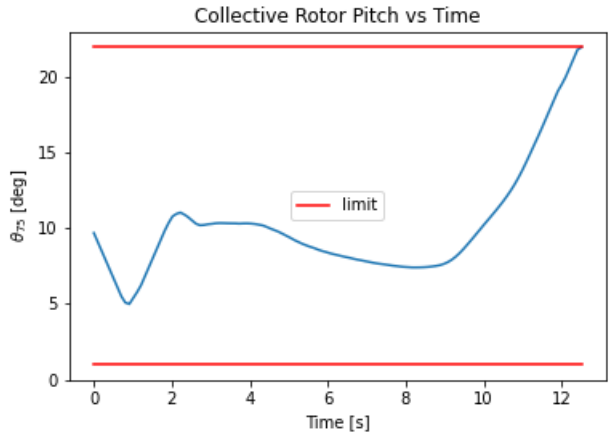


Figure A.20. Actor 1: Main Rotor Collective Pitch vs Time - $H_0 = 300[ft]$, $V_0 = 0[KTAS]$

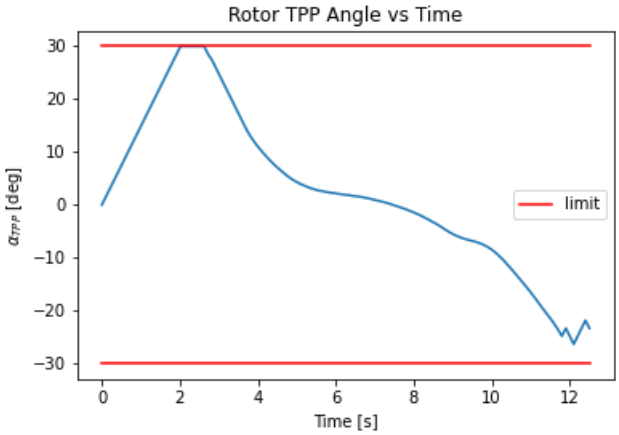


Figure A.21. Actor 1: Tip Path Plane Angle vs Time - $H_0 = 300[ft]$, $V_0 = 0[KTAS]$

d Initial Conditions: H0=300[ft], V0=50[KTAS]

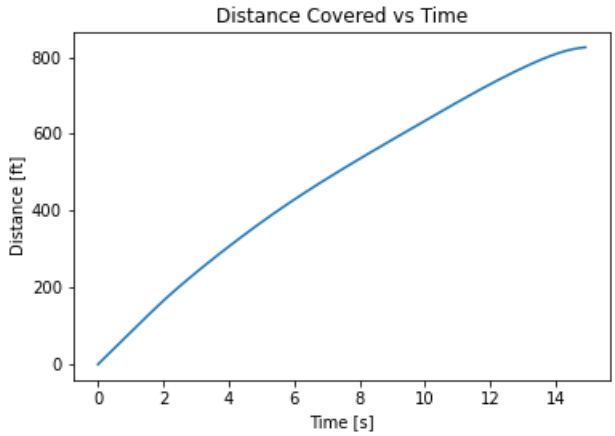


Figure A.22. Actor 1: Distance vs Time - $H_0 = 300[ft]$, $V_0 = 50[KTAS]$

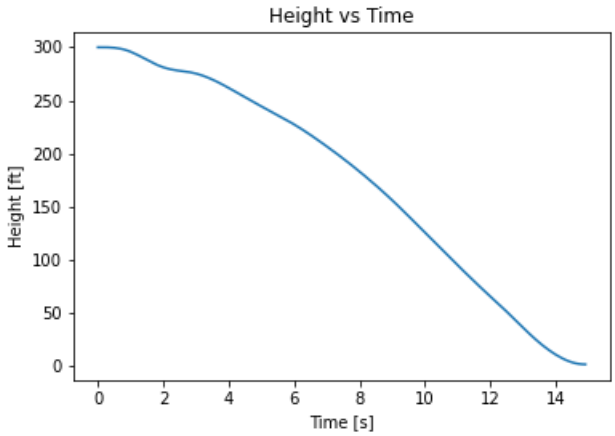


Figure A.23. Actor 1: Height vs Time - $H_0 = 300[ft]$, $V_0 = 50[KTAS]$

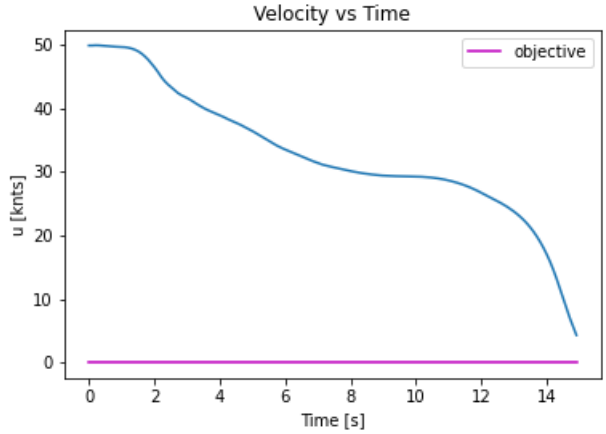


Figure A.24. Actor 1: Velocity vs Time - $H_0 = 300[ft]$, $V_0 = 50[KTAS]$

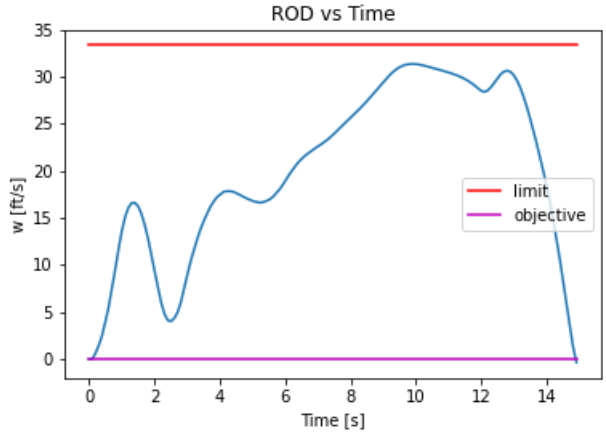


Figure A.25. Actor 1: Rate of Descent vs Time - $H_0 = 300[ft]$, $V_0 = 50[KTAS]$

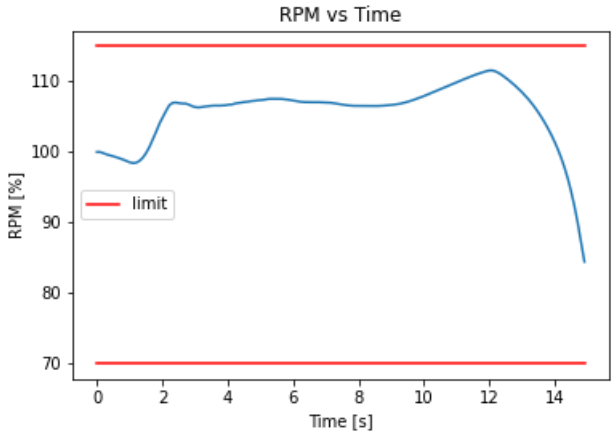


Figure A.26. Actor 1: Main Rotor RPM vs Time - $H_0 = 300[ft]$, $V_0 = 50[KTAS]$

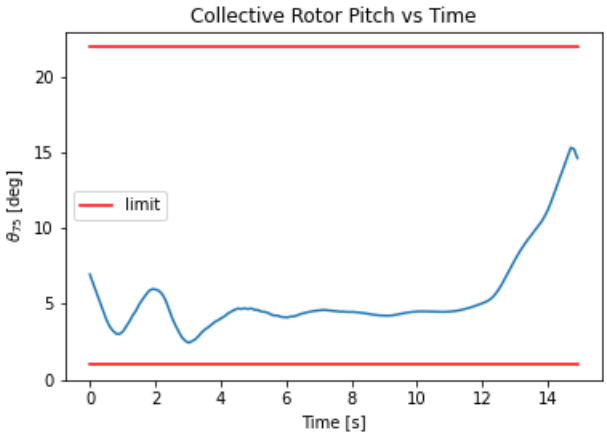


Figure A.27. Actor 1: Main Rotor Collective Pitch vs Time - $H_0 = 300[ft]$, $V_0 = 50[KTAS]$

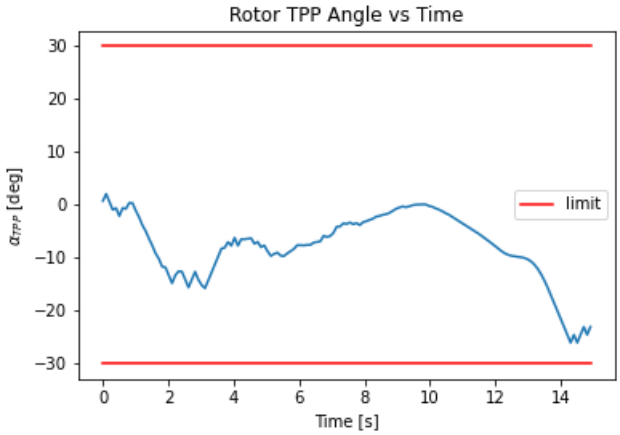


Figure A.28. Actor 1: Tip Path Plane Angle vs Time - $H_0 = 300[ft]$, $V_0 = 50[KTAS]$

e Initial Conditions: H0=500[ft], V0=10[KTAS]

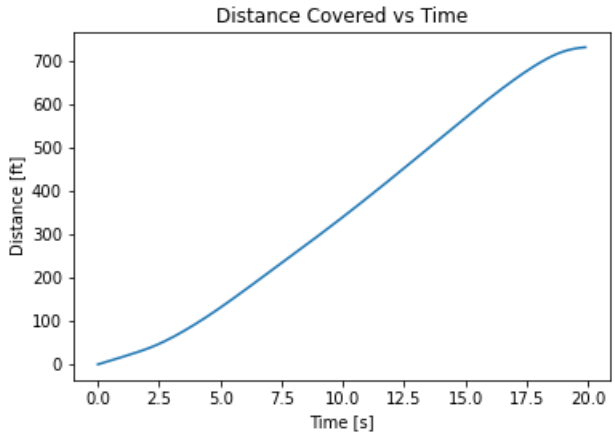


Figure A.29. Actor 1: Distance vs Time - $H_0 = 500[ft]$, $V_0 = 10[KTAS]$

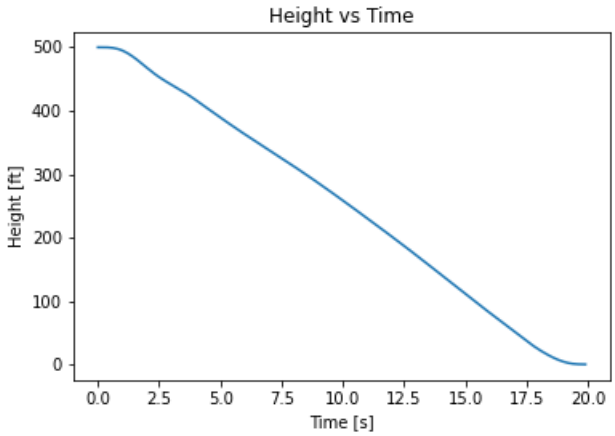


Figure A.30. Actor 1: Height vs Time - $H_0 = 500[ft]$, $V_0 = 10[KTAS]$

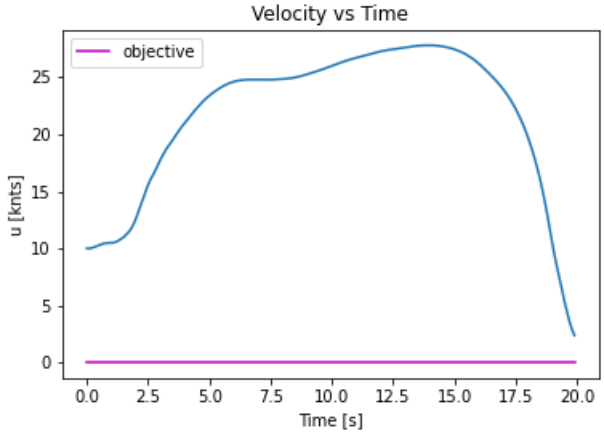


Figure A.31. Actor 1: Velocity vs Time - $H_0 = 500[ft]$, $V_0 = 10[KTAS]$

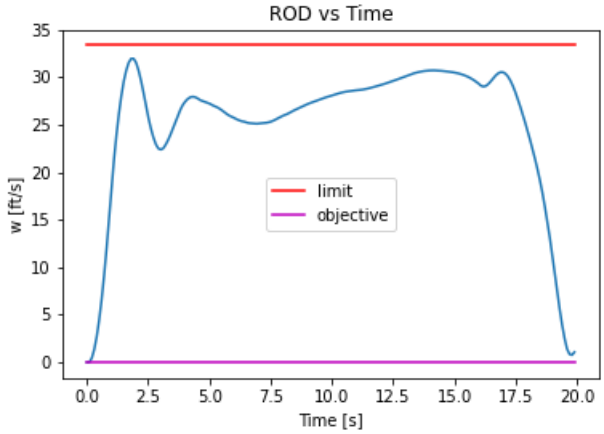


Figure A.32. Actor 1: Rate of Descent vs Time - $H_0 = 500[ft]$, $V_0 = 10[KTAS]$

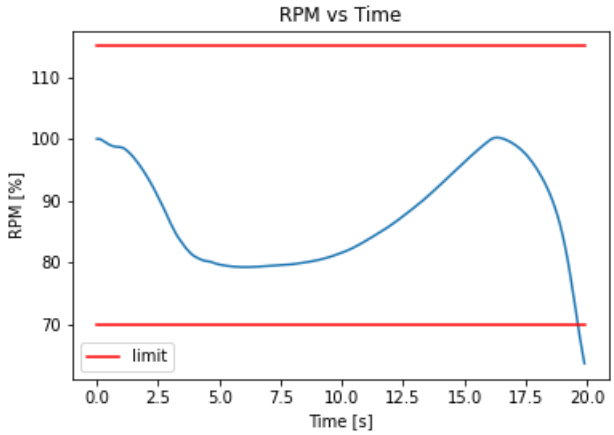


Figure A.33. Actor 1: Main Rotor RPM vs Time - $H_0 = 500[ft]$, $V_0 = 10[KTAS]$

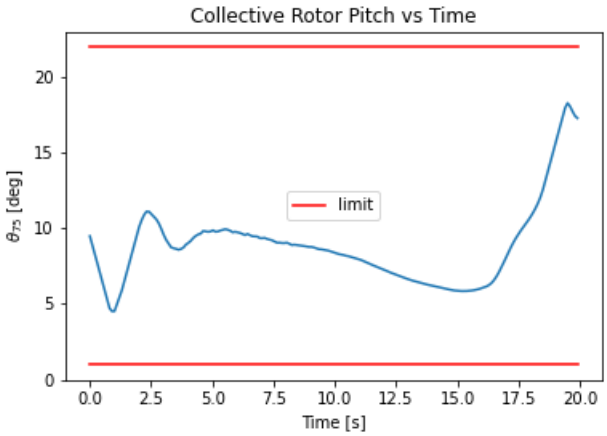


Figure A.34. Actor 1: Main Rotor Collective Pitch vs Time - $H_0 = 500[ft]$, $V_0 = 10[KTAS]$

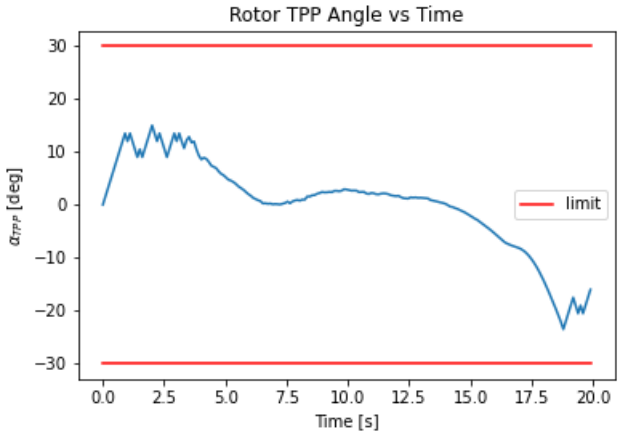


Figure A.35. Actor 1: Tip Path Plane Angle vs Time - $H_0 = 500[ft]$, $V_0 = 10[KTAS]$

f Initial Conditions: H0=500[ft], V0=40[KTAS]

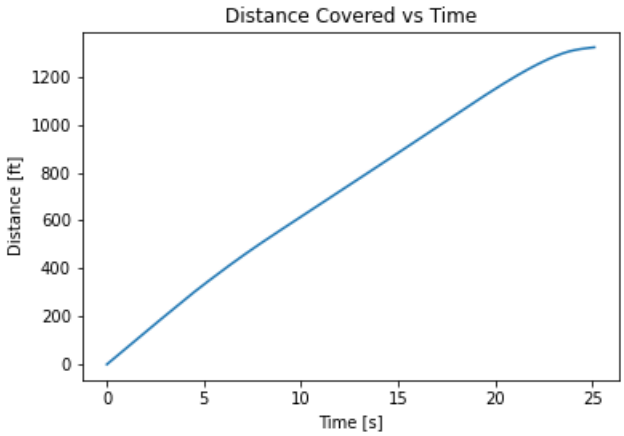


Figure A.36. Actor 1: Distance vs Time - $H_0 = 500[ft]$, $V_0 = 40[KTAS]$

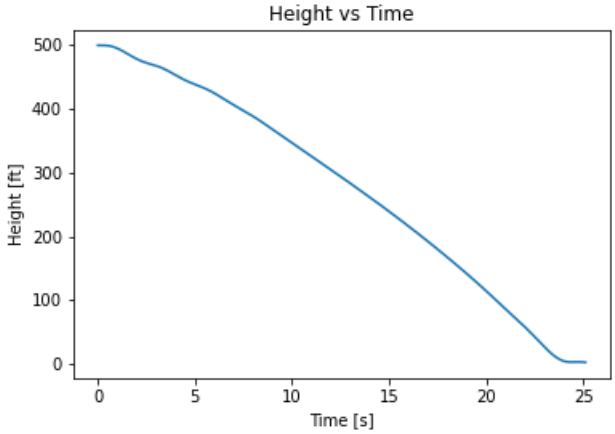


Figure A.37. Actor 1: Height vs Time - $H_0 = 500[ft]$, $V_0 = 40[KTAS]$

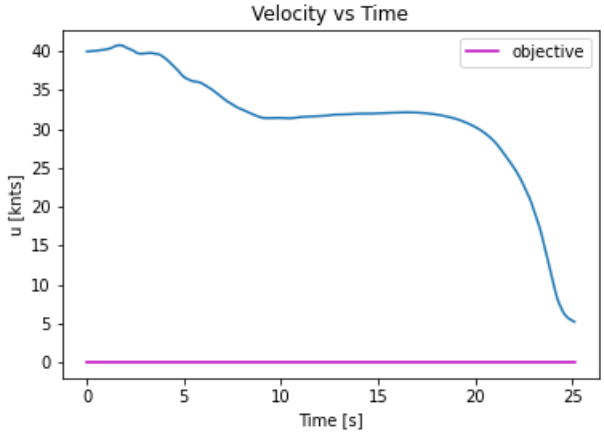


Figure A.38. Actor 1: Velocity vs Time - $H_0 = 500[ft]$, $V_0 = 40[KTAS]$

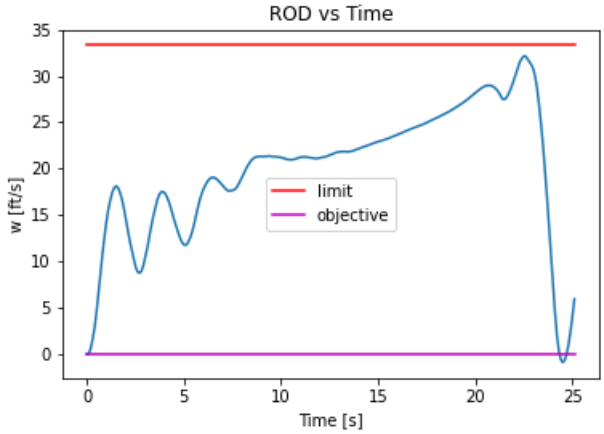


Figure A.39. Actor 1: Rate of Descent vs Time - $H_0 = 500[ft]$, $V_0 = 40[KTAS]$

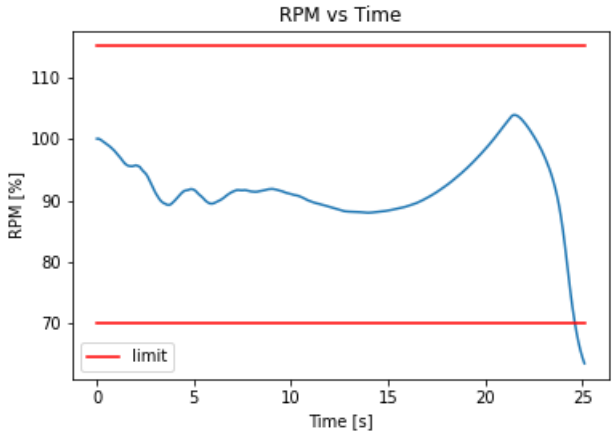


Figure A.40. Actor 1: Main Rotor RPM vs Time - $H_0 = 500[ft]$, $V_0 = 40[KTAS]$

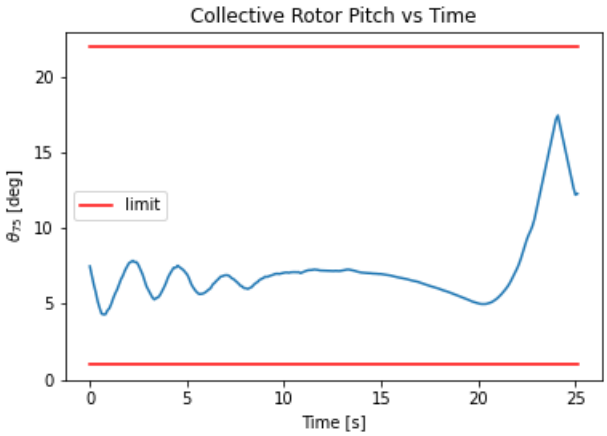


Figure A.41. Actor 1: Main Rotor Collective Pitch vs Time - $H_0 = 500[ft]$, $V_0 = 40[KTAS]$

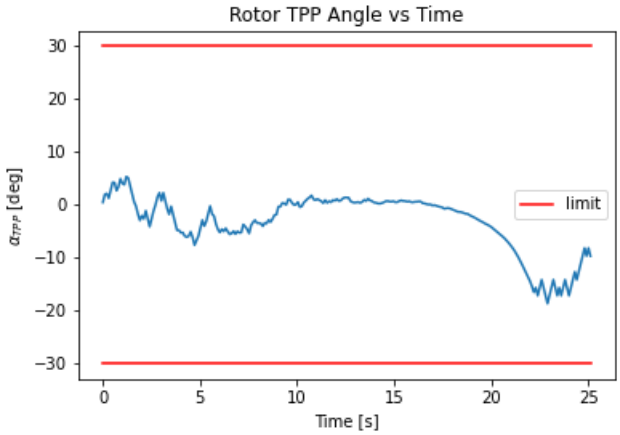


Figure A.42. Actor 1: Tip Path Plane Angle vs Time - $H_0 = 500[ft]$, $V_0 = 40[KTAS]$

g Initial Conditions: H0=600[ft], V0=40[KTAS]

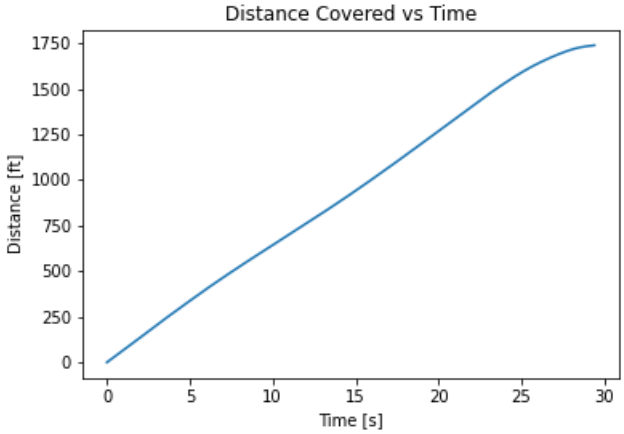


Figure A.43. Actor 1: Distance vs Time - $H_0 = 600[ft]$, $V_0 = 40[KTAS]$

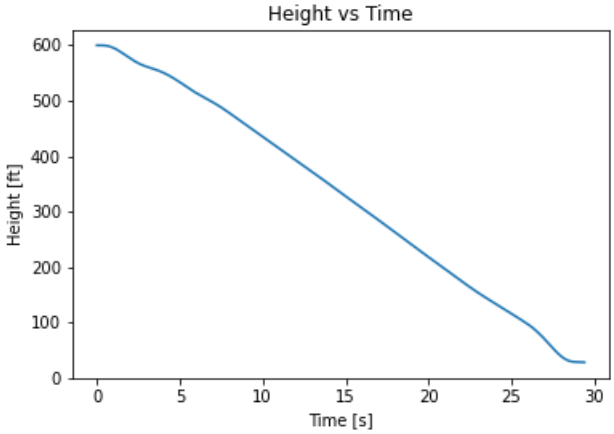


Figure A.44. Actor 1: Height vs Time - $H_0 = 600[ft]$, $V_0 = 40[KTAS]$

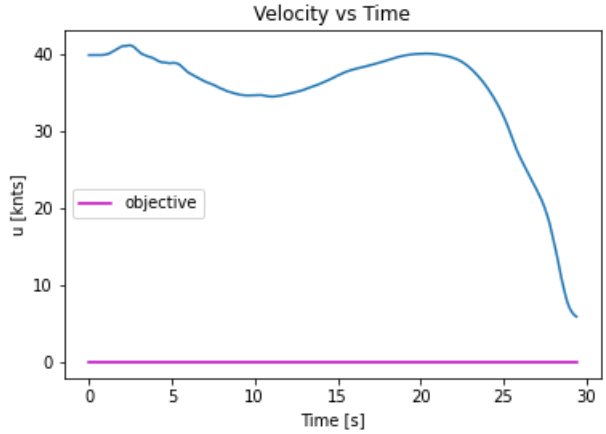


Figure A.45. Actor 1: Velocity vs Time - $H_0 = 600[ft]$, $V_0 = 40[KTAS]$

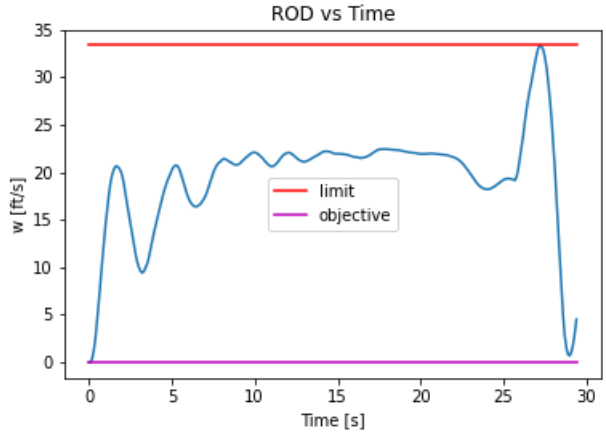


Figure A.46. Actor 1: Rate of Descent vs Time - $H_0 = 600[ft]$, $V_0 = 40[KTAS]$

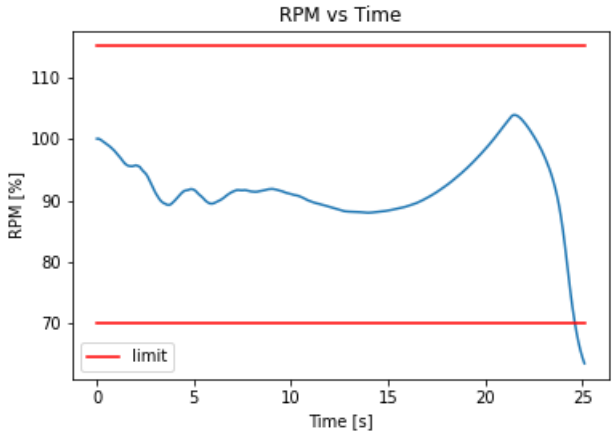


Figure A.47. Actor 1: Main Rotor RPM vs Time - $H_0 = 600[ft]$, $V_0 = 40[KTAS]$

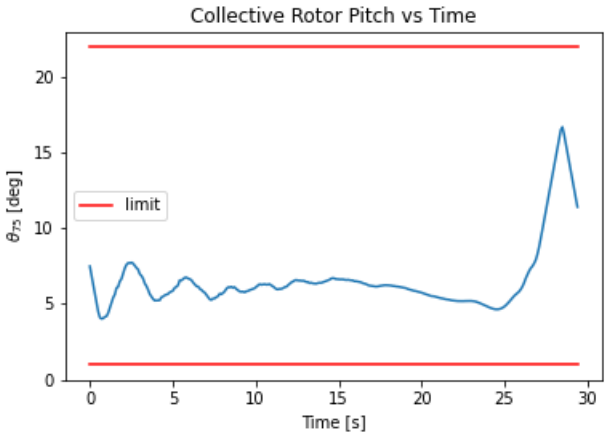


Figure A.48. Actor 1: Main Rotor Collective Pitch vs Time - $H_0 = 600[ft]$, $V_0 = 40[KTAS]$

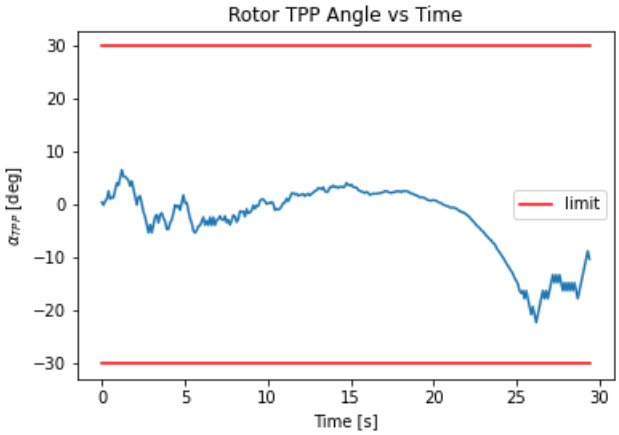


Figure A.49. Actor 1: Tip Path Plane Angle vs Time - $H_0 = 600[ft]$, $V_0 = 40[KTAS]$

h Initial Conditions: H0=600[ft], V0=50[KTAS]

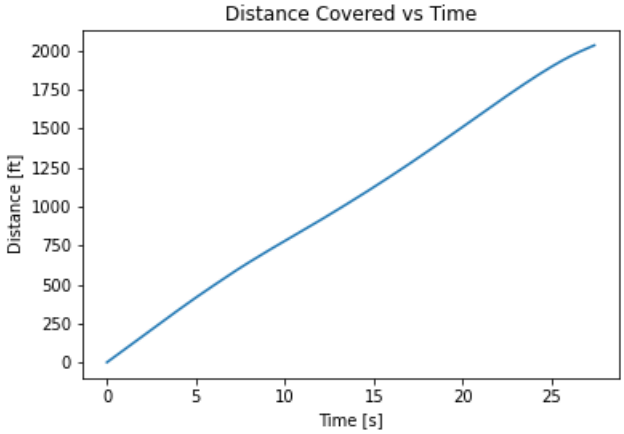


Figure A.50. Actor 1: Distance vs Time - $H_0 = 600[ft]$, $V_0 = 50[KTAS]$

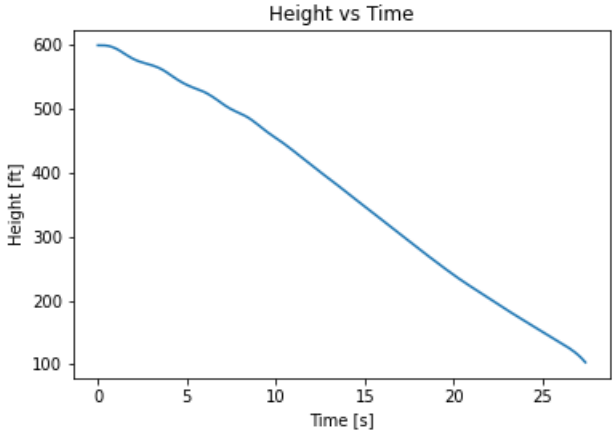


Figure A.51. Actor 1: Height vs Time - $H_0 = 600[ft]$, $V_0 = 50[KTAS]$

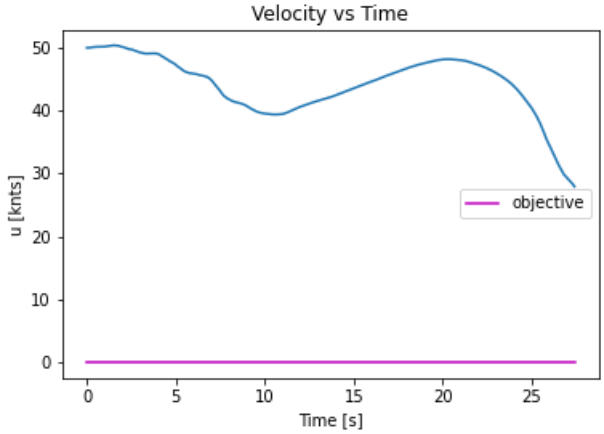


Figure A.52. Actor 1: Velocity vs Time - $H_0 = 600[ft]$, $V_0 = 50[KTAS]$

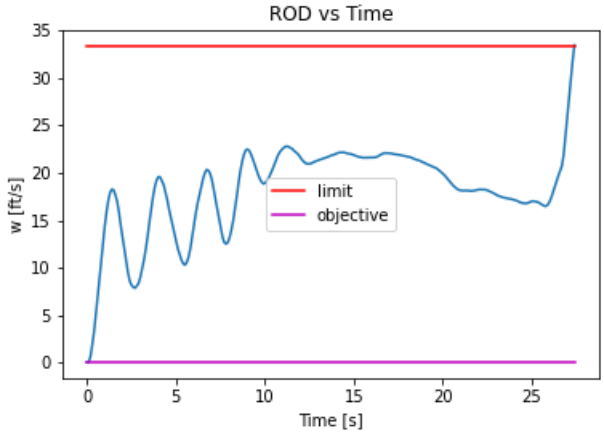


Figure A.53. Actor 1: Rate of Descent vs Time - $H_0 = 600[ft]$, $V_0 = 50[KTAS]$

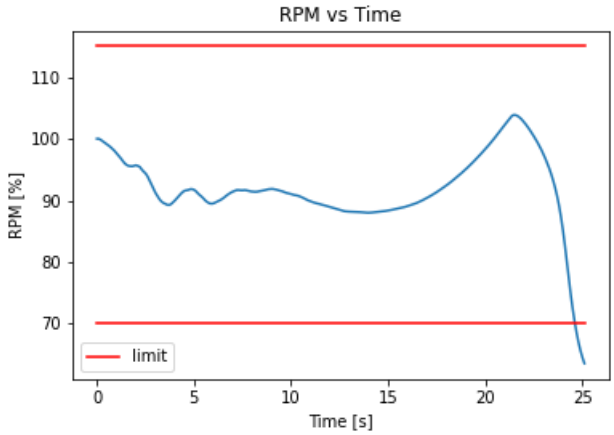


Figure A.54. Actor 1: Main Rotor RPM vs Time - $H_0 = 600[ft]$, $V_0 = 50[KTAS]$

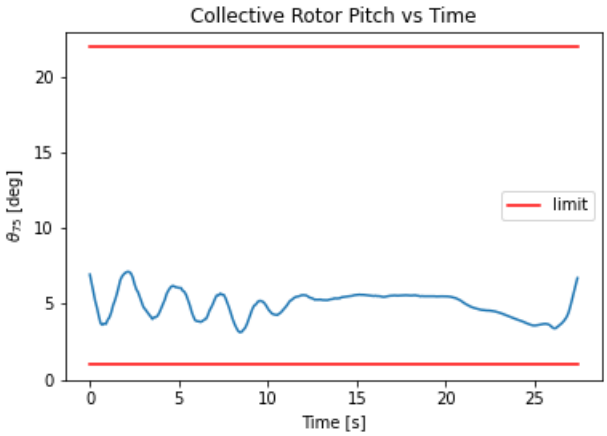


Figure A.55. Actor 1: Main Rotor Collective Pitch vs Time - $H_0 = 600[ft]$, $V_0 = 50[KTAS]$

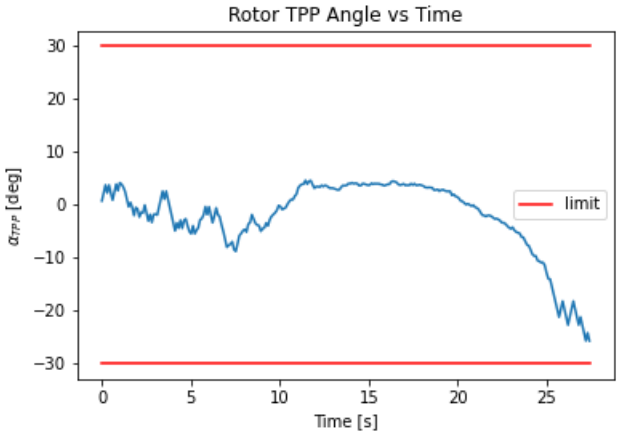


Figure A.56. Actor 1: Tip Path Plane Angle vs Time - $H_0 = 600[ft]$, $V_0 = 50[KTAS]$

II Test Results for Actor 2

a Initial Conditions: H0=50[ft], V0=30[KTAS]

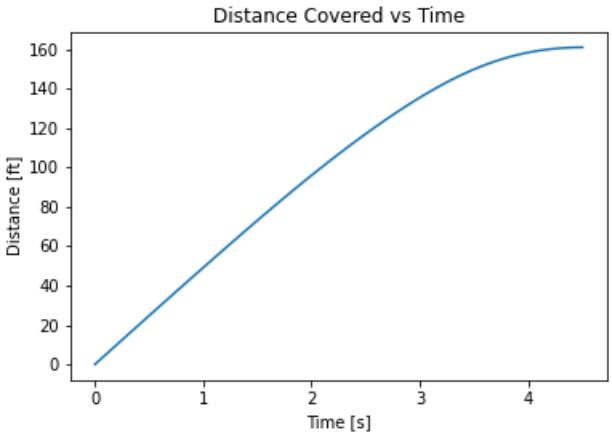


Figure A.57. Actor 2: Distance vs Time - $H_0 = 50[ft]$, $V_0 = 30[KTAS]$

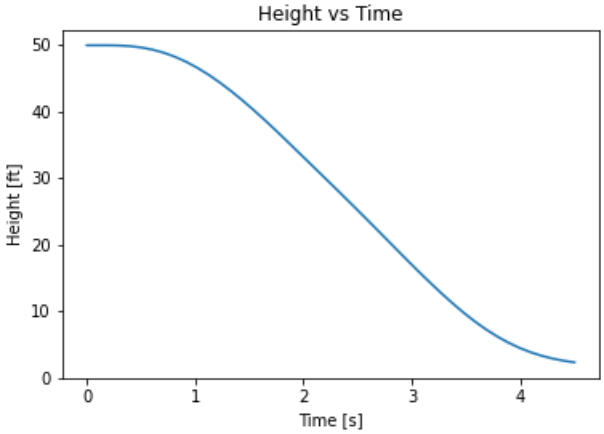


Figure A.58. Actor 2: Height vs Time - $H_0 = 50[ft]$, $V_0 = 30[KTAS]$

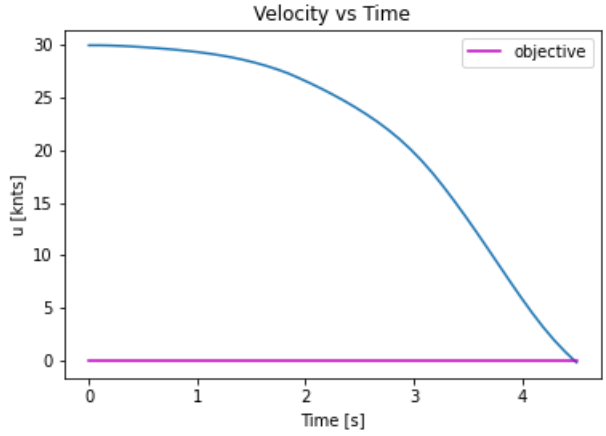


Figure A.59. Actor 2: Velocity vs Time - $H_0 = 50[ft]$, $V_0 = 30[KTAS]$

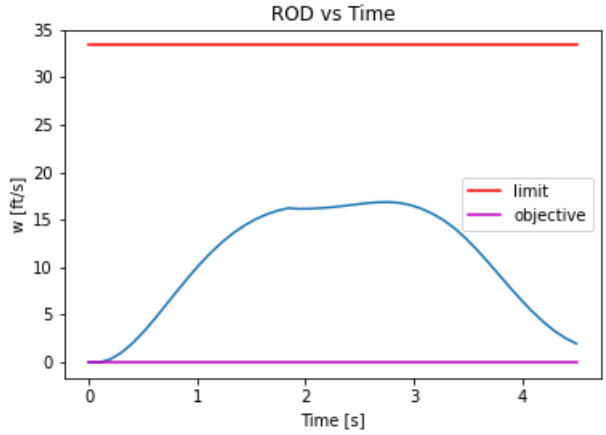


Figure A.60. Actor 2: Rate of Descent vs Time - $H_0 = 50[ft]$, $V_0 = 30[KTAS]$

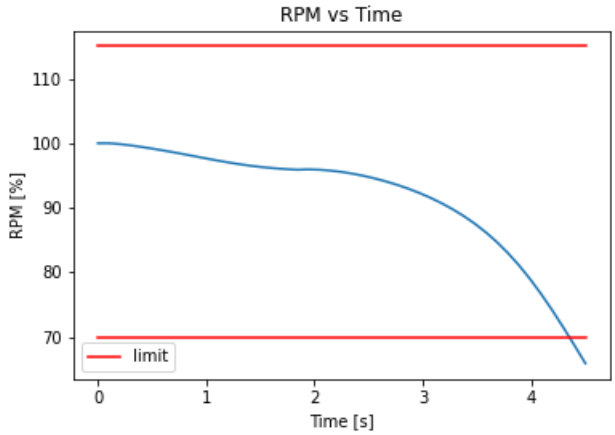


Figure A.61. Actor 2: Main Rotor RPM vs Time - $H_0 = 50[ft]$, $V_0 = 30[KTAS]$

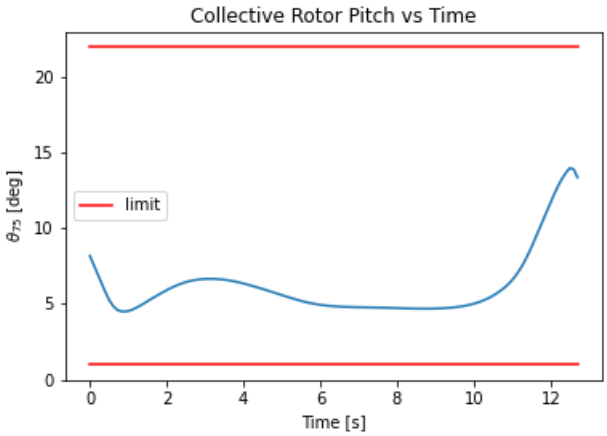


Figure A.62. Actor 2: Main Rotor Collective Pitch vs Time - $H_0 = 50[ft]$, $V_0 = 30[KTAS]$

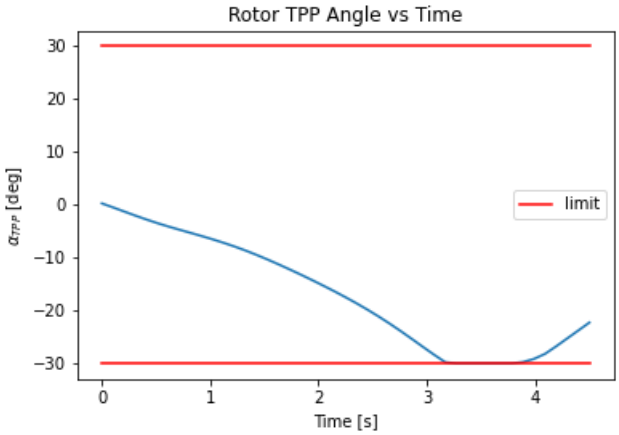


Figure A.63. Actor 2: Tip Path Plane Angle vs Time - $H_0 = 50[ft]$, $V_0 = 30[KTAS]$

b Initial Conditions: H0=100[ft], V0=0[KTAS]

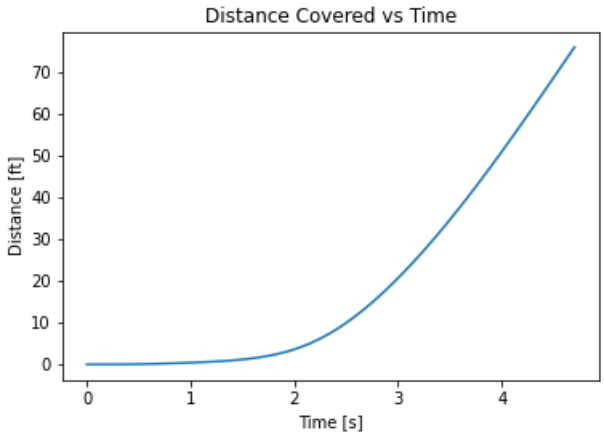


Figure A.64. Actor 2: Distance vs Time - $H_0 = 100[ft]$, $V_0 = 0[KTAS]$

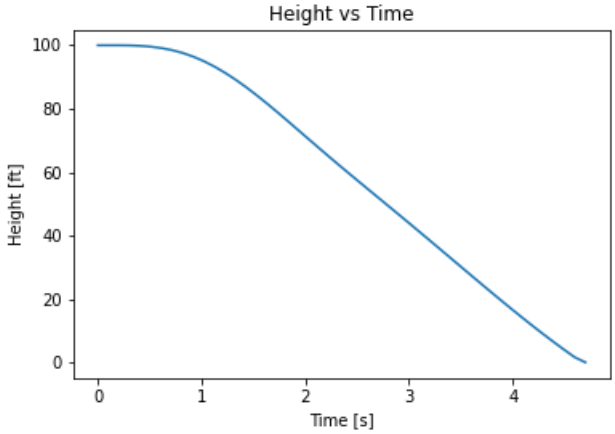


Figure A.65. Actor 2: Height vs Time - $H_0 = 100[ft]$, $V_0 = 0[KTAS]$

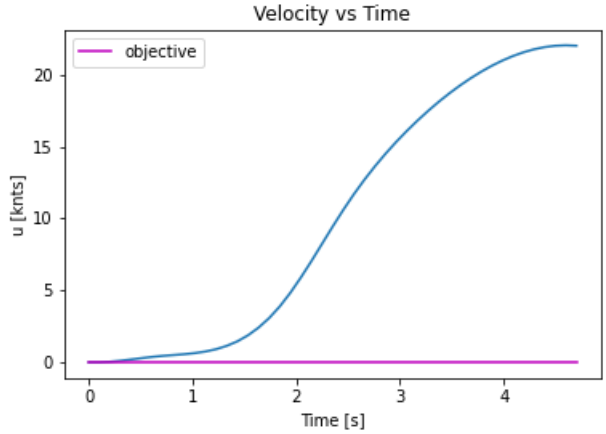


Figure A.66. Actor 2: Velocity vs Time - $H_0 = 100[ft]$, $V_0 = 0[KTAS]$

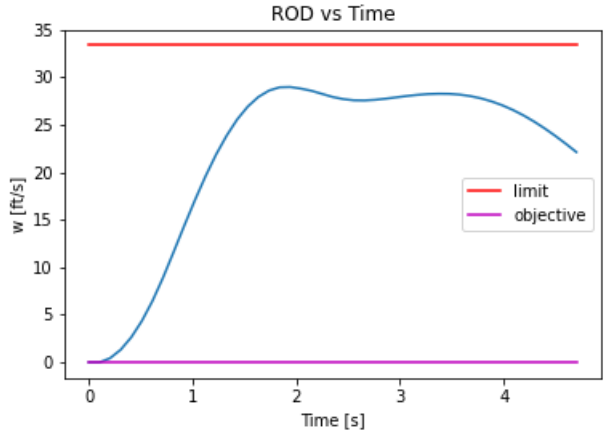


Figure A.67. Actor 2: Rate of Descent vs Time - $H_0 = 100[ft]$, $V_0 = 0[KTAS]$

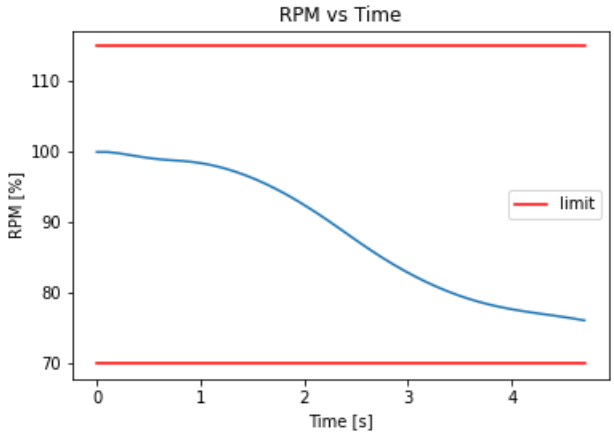


Figure A.68. Actor 2: Main Rotor RPM vs Time - $H_0 = 100[ft]$, $V_0 = 0[KTAS]$

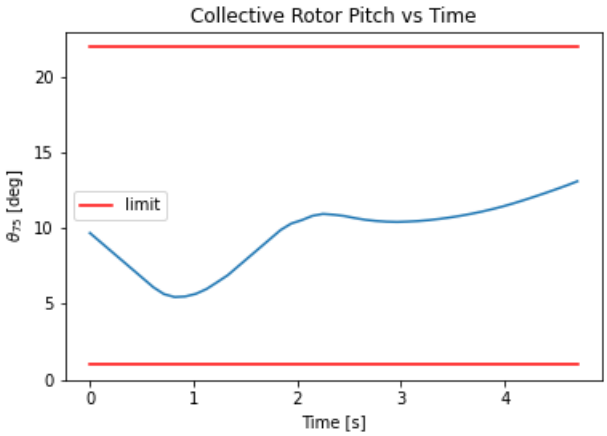


Figure A.69. Actor 2: Main Rotor Collective Pitch vs Time - $H_0 = 100[ft]$, $V_0 = 0[KTAS]$

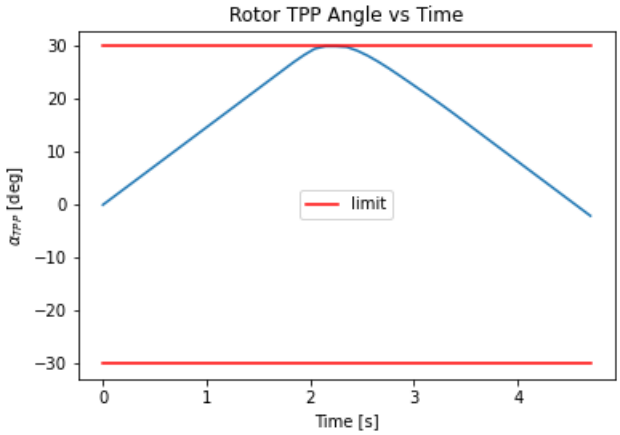


Figure A.70. Actor 2: Tip Path Plane Angle vs Time - $H_0 = 100[ft]$, $V_0 = 0[KTAS]$

c Initial Conditions: H0=300[ft], V0=0[KTAS]

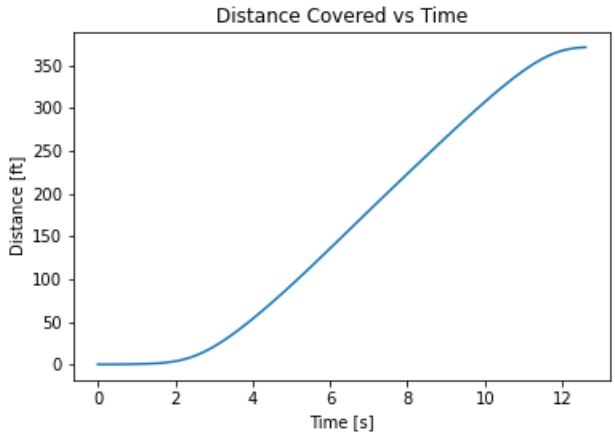


Figure A.71. Actor 2: Distance vs Time - $H_0 = 300[ft]$, $V_0 = 0[KTAS]$

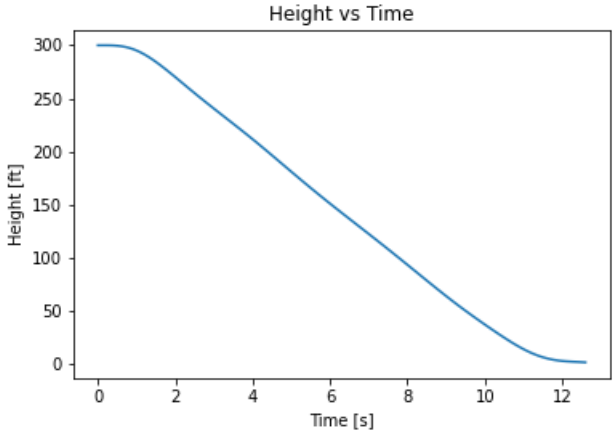


Figure A.72. Actor 2: Height vs Time - $H_0 = 300[ft]$, $V_0 = 0[KTAS]$

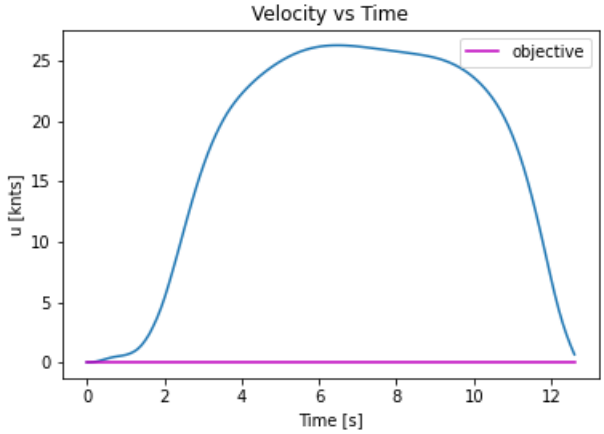


Figure A.73. Actor 2: Velocity vs Time - $H_0 = 300[ft]$, $V_0 = 0[KTAS]$

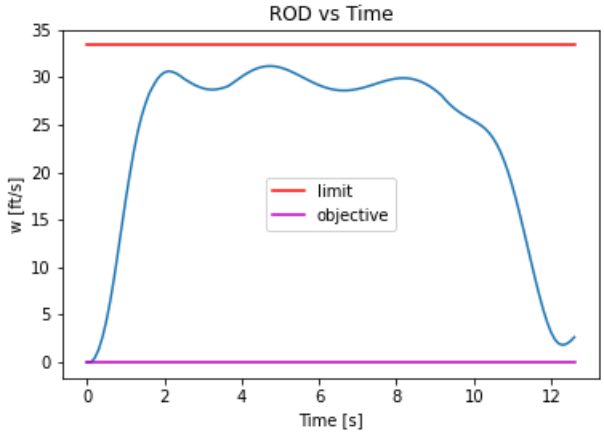


Figure A.74. Actor 2: Rate of Descent vs Time - $H_0 = 300[ft]$, $V_0 = 0[KTAS]$

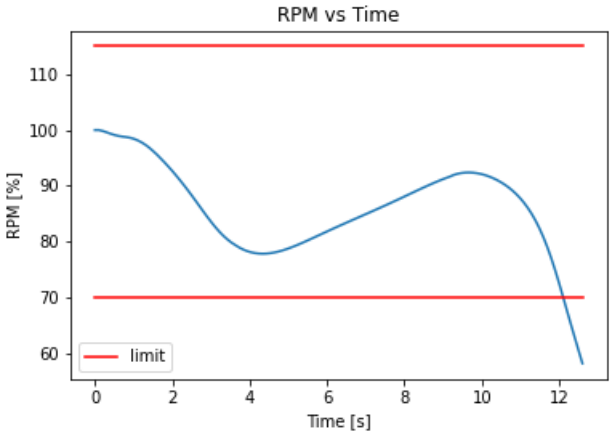


Figure A.75. Actor 2: Main Rotor RPM vs Time - $H_0 = 300[ft]$, $V_0 = 0[KTAS]$

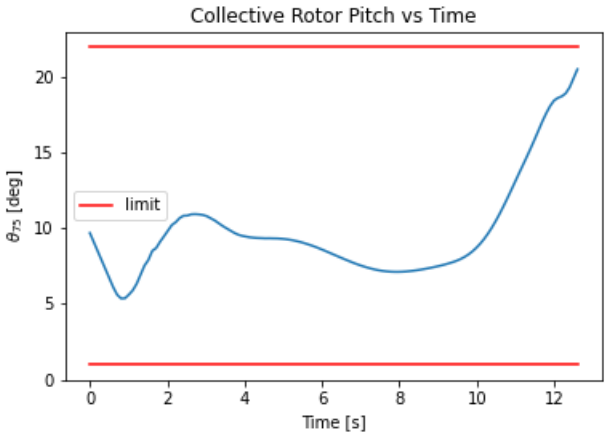


Figure A.76. Actor 2: Main Rotor Collective Pitch vs Time - $H_0 = 300[ft]$, $V_0 = 0[KTAS]$

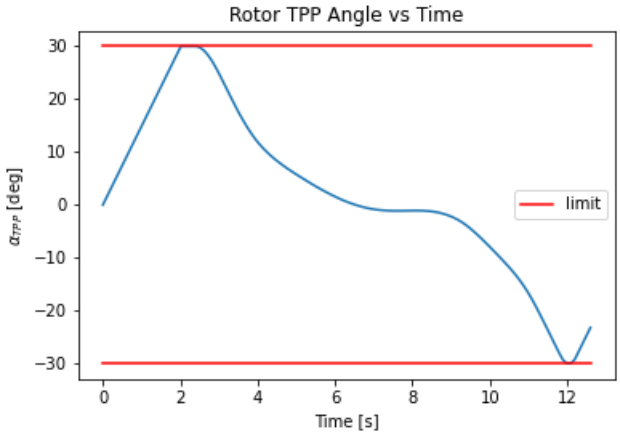


Figure A.77. Actor 2: Tip Path Plane Angle vs Time - $H_0 = 300[ft]$, $V_0 = 0[KTAS]$

d Initial Conditions: H0=300[ft], V0=50[KTAS]

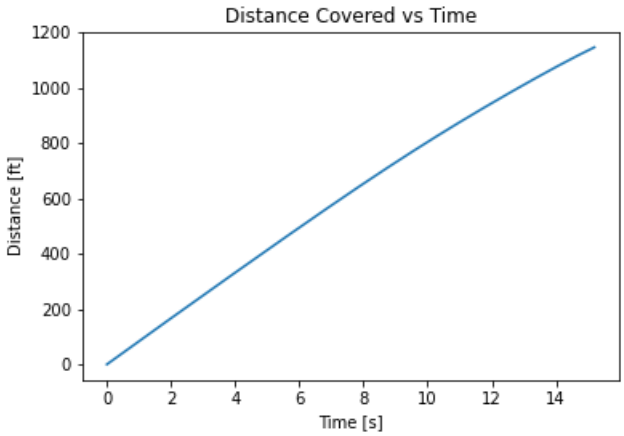


Figure A.78. Actor 2: Distance vs Time - $H_0 = 300[ft]$, $V_0 = 50[KTAS]$

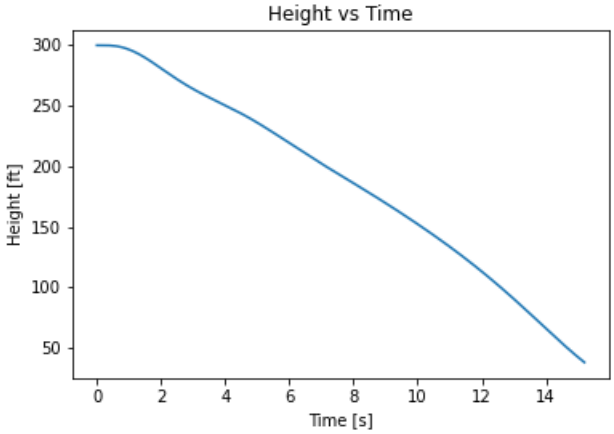


Figure A.79. Actor 2: Height vs Time - $H_0 = 300[ft]$, $V_0 = 50[KTAS]$

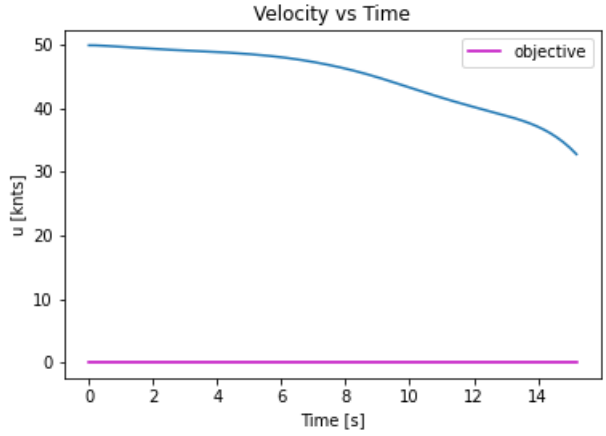


Figure A.80. Actor 2: Velocity vs Time - $H_0 = 300[ft]$, $V_0 = 50[KTAS]$

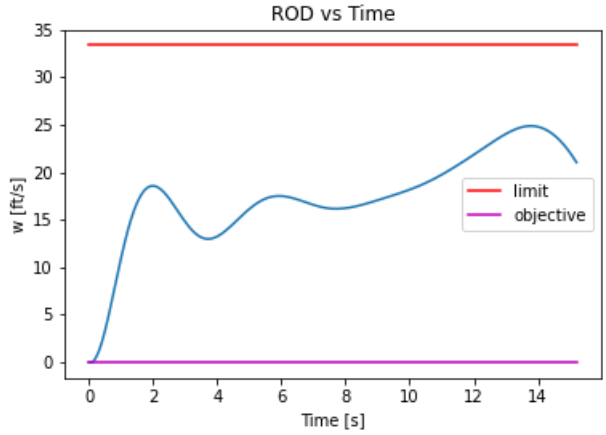


Figure A.81. Actor 2: Rate of Descent vs Time - $H_0 = 300[ft]$, $V_0 = 50[KTAS]$

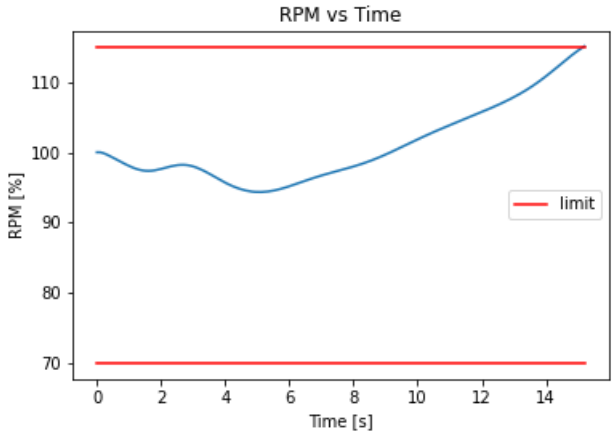


Figure A.82. Actor 2: Main Rotor RPM vs Time - $H_0 = 300[ft]$, $V_0 = 50[KTAS]$

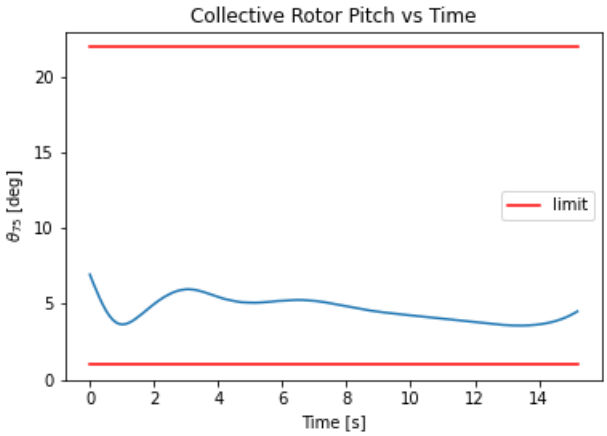


Figure A.83. Actor 2: Main Rotor Collective Pitch vs Time - $H_0 = 300[ft]$, $V_0 = 50[KTAS]$

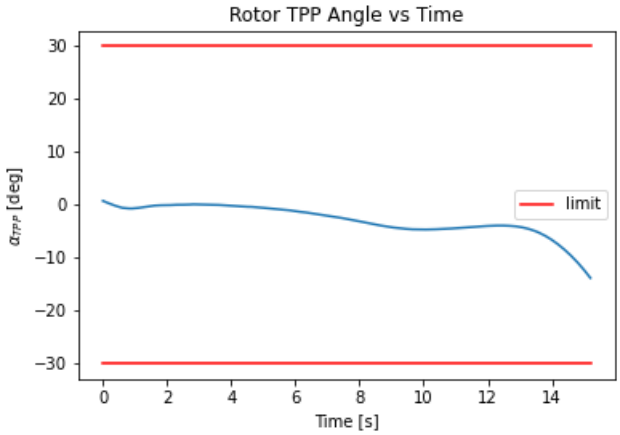


Figure A.84. Actor 2: Tip Path Plane Angle vs Time - $H_0 = 300[ft]$, $V_0 = 50[KTAS]$

e Initial Conditions: H0=500[ft], V0=10[KTAS]

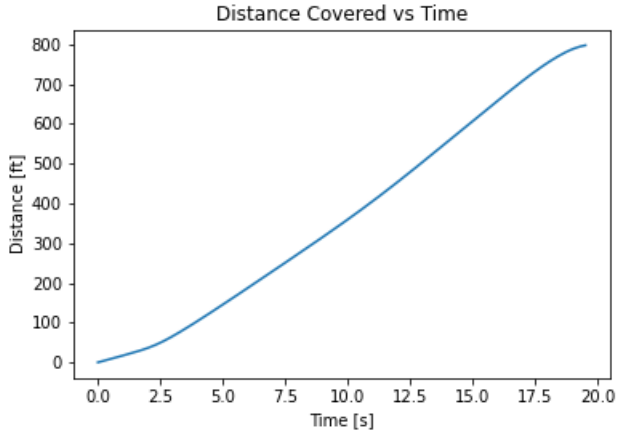


Figure A.85. Actor 2: Distance vs Time - $H_0 = 500[ft]$, $V_0 = 10[KTAS]$

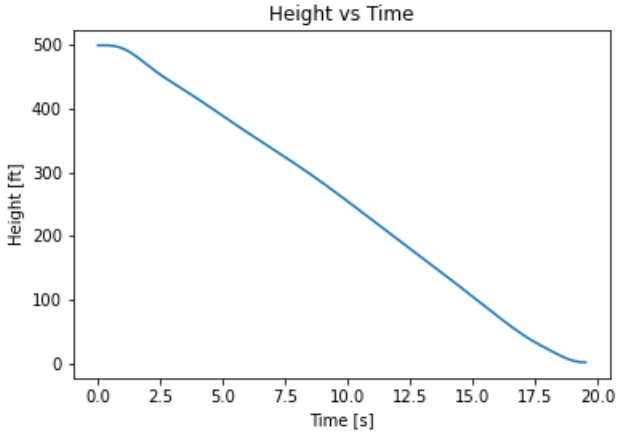


Figure A.86. Actor 2: Height vs Time - $H_0 = 500[ft]$, $V_0 = 10[KTAS]$

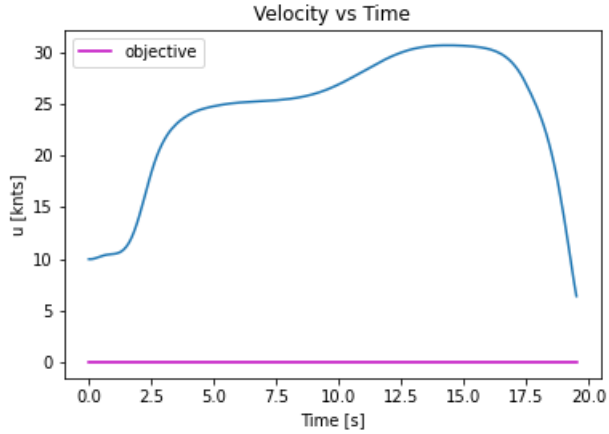


Figure A.87. Actor 2: Velocity vs Time - $H_0 = 500[ft]$, $V_0 = 10[KTAS]$

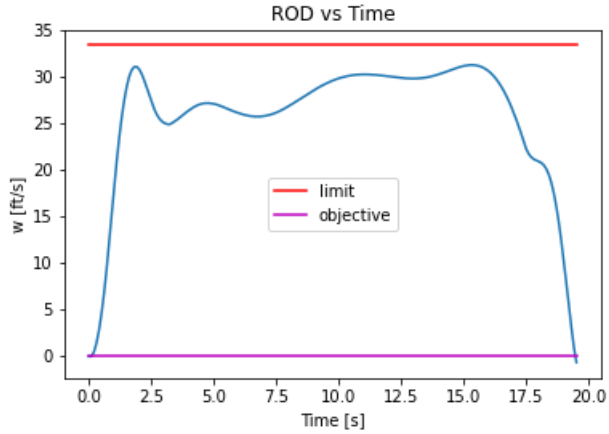


Figure A.88. Actor 2: Rate of Descent vs Time - $H_0 = 500[ft]$, $V_0 = 10[KTAS]$

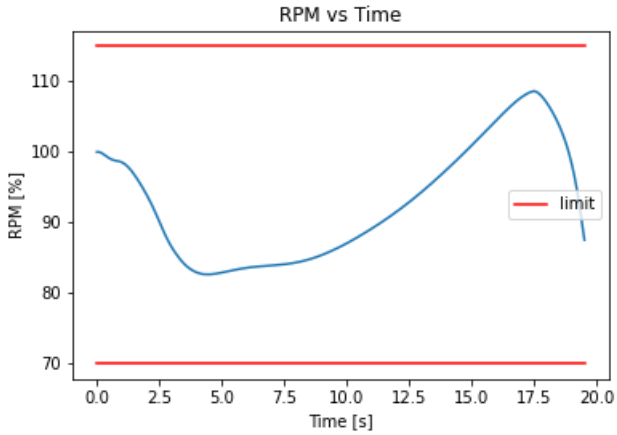


Figure A.89. Actor 2: Main Rotor RPM vs Time - $H_0 = 500[ft]$, $V_0 = 10[KTAS]$

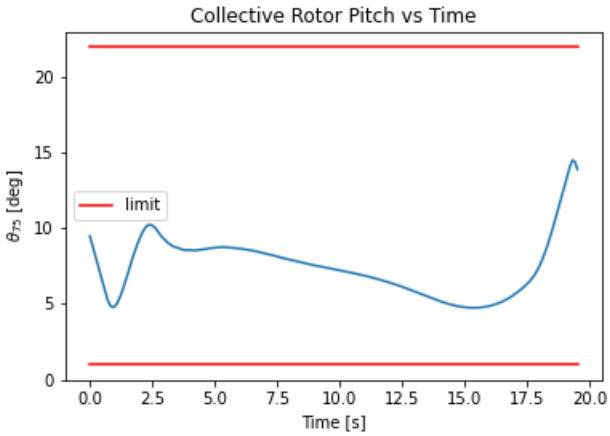


Figure A.90. Actor 2: Main Rotor Collective Pitch vs Time - $H_0 = 500[ft]$, $V_0 = 10[KTAS]$

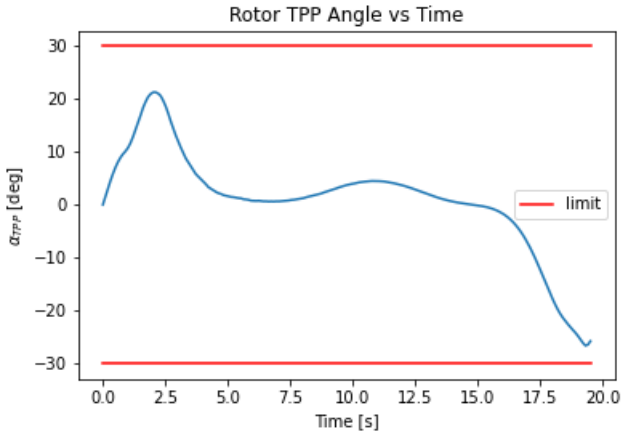


Figure A.91. Actor 2: Tip Path Plane Angle vs Time - $H_0 = 500[ft]$, $V_0 = 10[KTAS]$

f Initial Conditions: H0=500[ft], V0=40[KTAS]

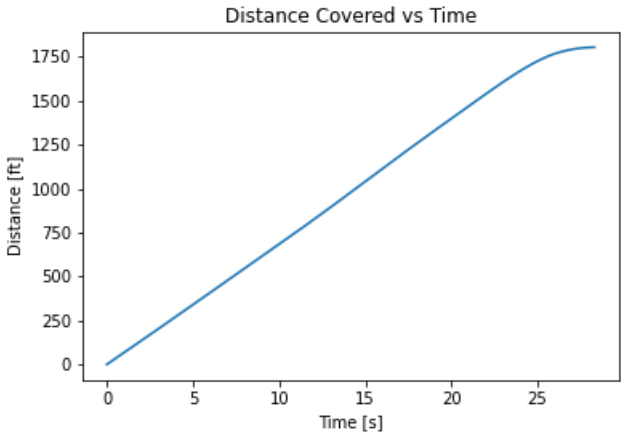


Figure A.92. Actor 2: Distance vs Time - $H_0 = 500[ft]$, $V_0 = 40[KTAS]$

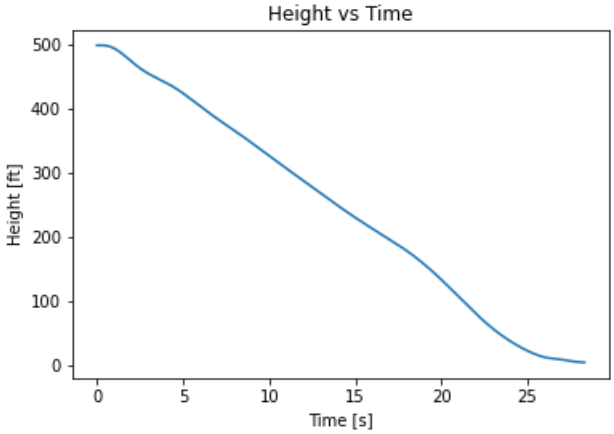


Figure A.93. Actor 2: Height vs Time - $H_0 = 500[ft]$, $V_0 = 40[KTAS]$

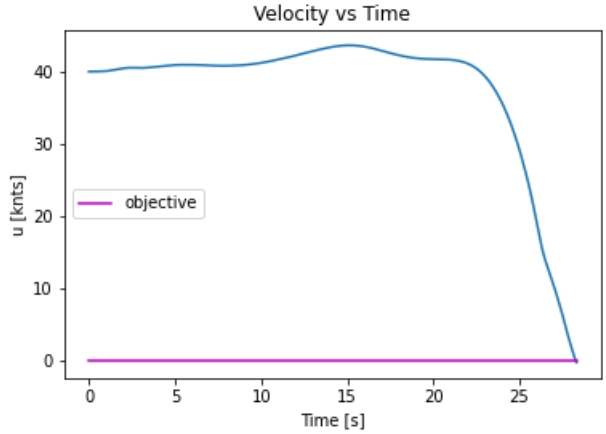


Figure A.94. Actor 2: Velocity vs Time - $H_0 = 500[ft]$, $V_0 = 40[KTAS]$

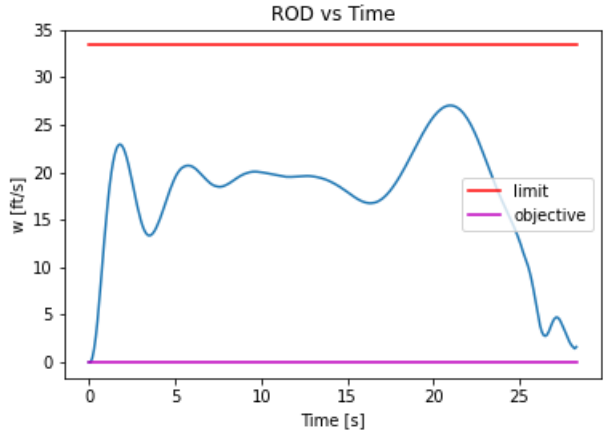


Figure A.95. Actor 2: Rate of Descent vs Time - $H_0 = 500[ft]$, $V_0 = 40[KTAS]$

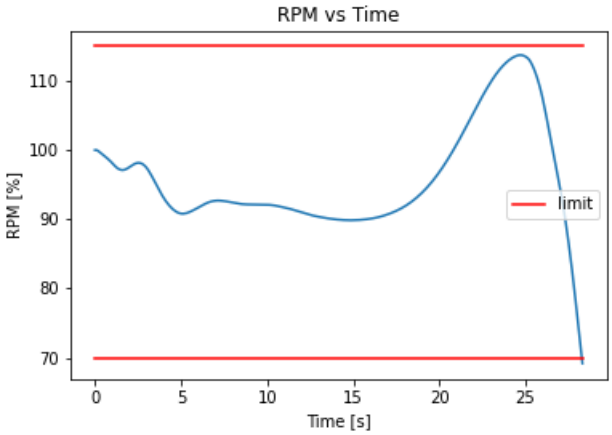


Figure A.96. Actor 2: Main Rotor RPM vs Time - $H_0 = 500[ft]$, $V_0 = 40[KTAS]$

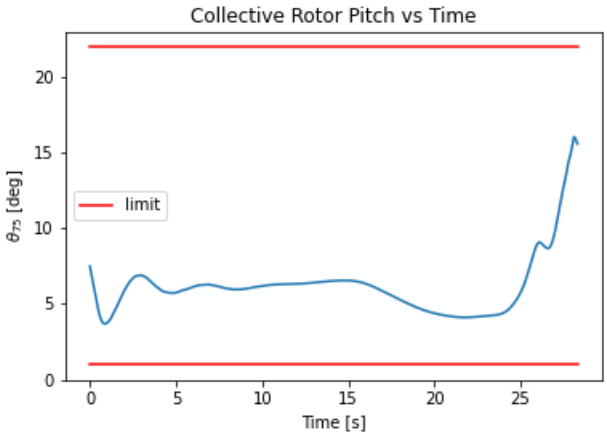


Figure A.97. Actor 2: Main Rotor Collective Pitch vs Time - $H_0 = 500[ft]$, $V_0 = 40[KTAS]$

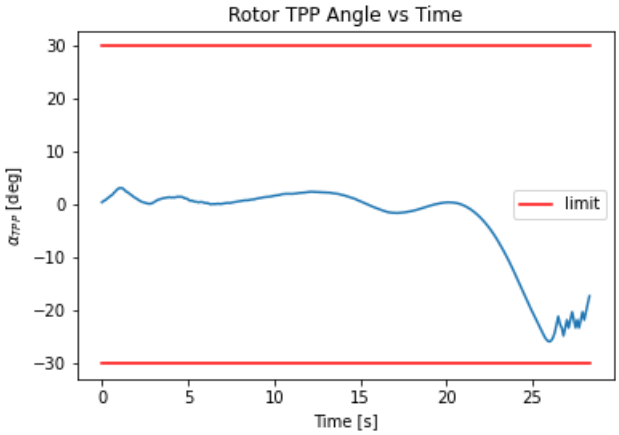


Figure A.98. Actor 2: Tip Path Plane Angle vs Time - $H_0 = 500[ft]$, $V_0 = 40[KTAS]$

g Initial Conditions: H0=600[ft], V0=40[KTAS]

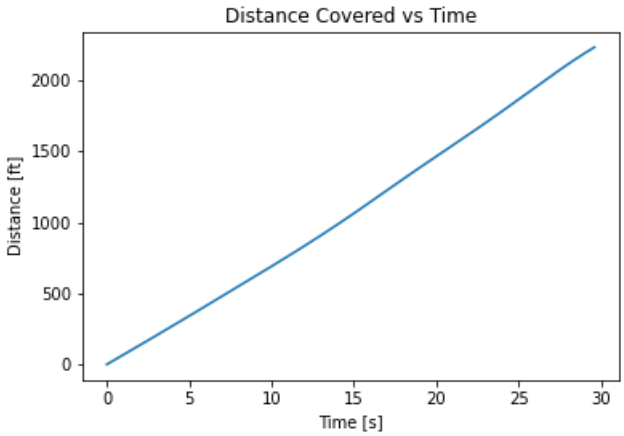


Figure A.99. Actor 2: Distance vs Time - $H_0 = 600[ft]$, $V_0 = 40[KTAS]$

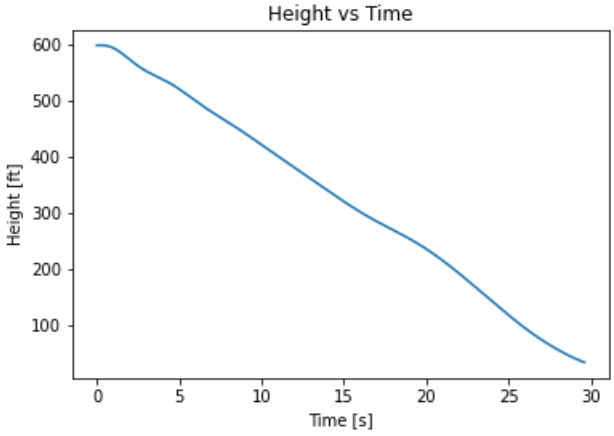


Figure A.100. Actor 2: Height vs Time - $H_0 = 600[ft]$, $V_0 = 40[KTAS]$

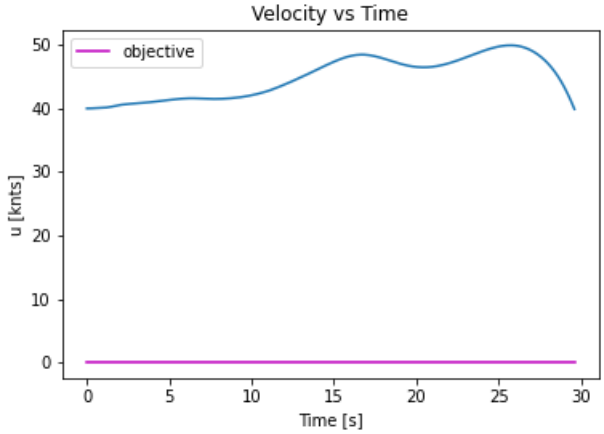


Figure A.101. Actor 2: Velocity vs Time - $H_0 = 600[ft]$, $V_0 = 40[KTAS]$

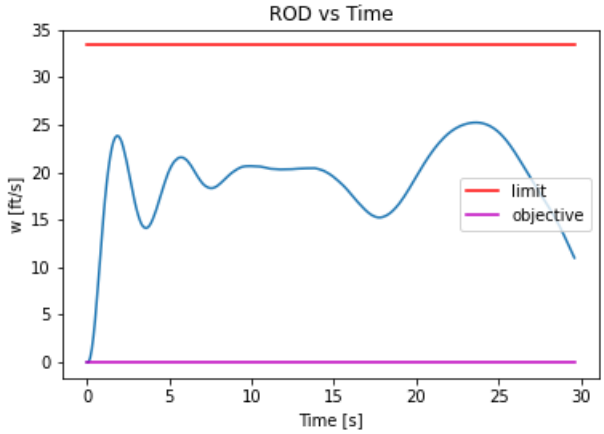


Figure A.102. Actor 2: Rate of Descent vs Time - $H_0 = 600[ft]$, $V_0 = 40[KTAS]$

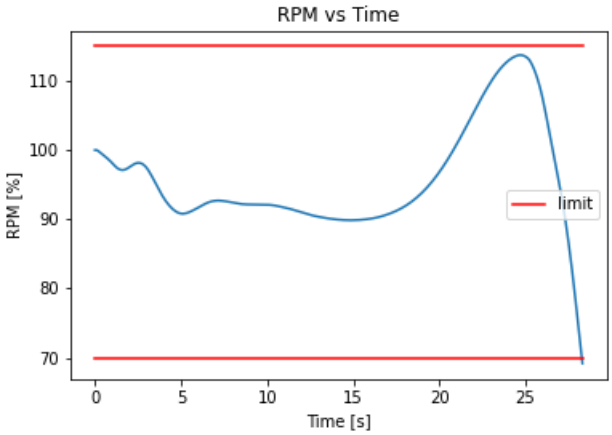


Figure A.103. Actor 2: Main Rotor RPM vs Time - $H_0 = 600[ft]$, $V_0 = 40[KTAS]$

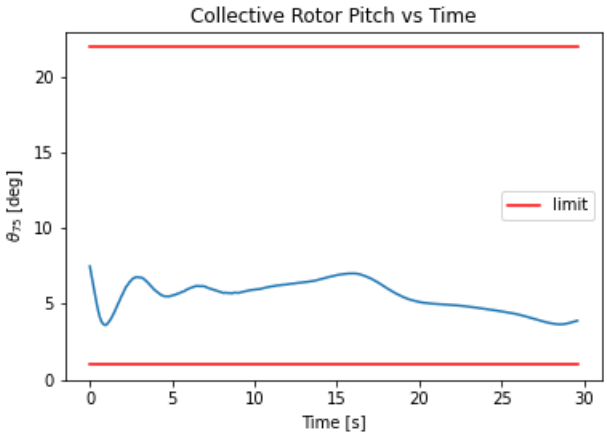


Figure A.104. Actor 2: Main Rotor Collective Pitch vs Time - $H_0 = 600[ft]$, $V_0 = 40[KTAS]$

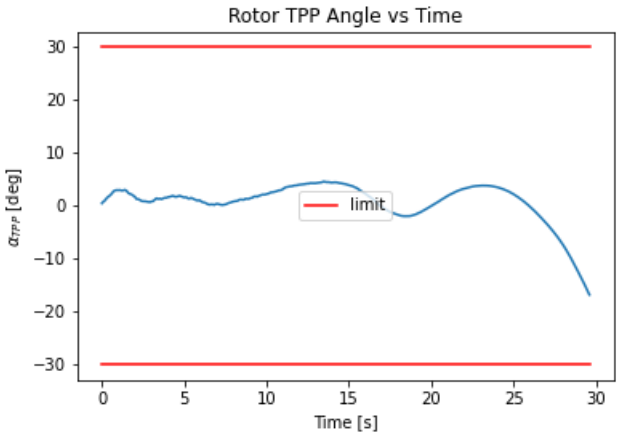


Figure A.105. Actor 2: Tip Path Plane Angle vs Time - $H_0 = 600[ft]$, $V_0 = 40[KTAS]$

h Initial Conditions: H0=600[ft], V0=50[KTAS]

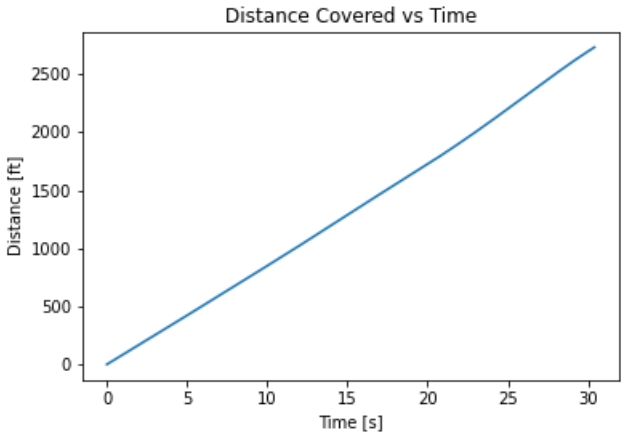


Figure A.106. Actor 2: Distance vs Time - $H_0 = 600[ft]$, $V_0 = 50[KTAS]$

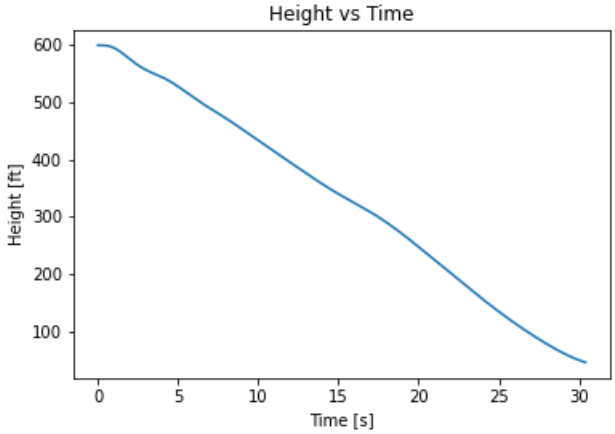


Figure A.107. Actor 2: Height vs Time - $H_0 = 600[ft]$, $V_0 = 50[KTAS]$

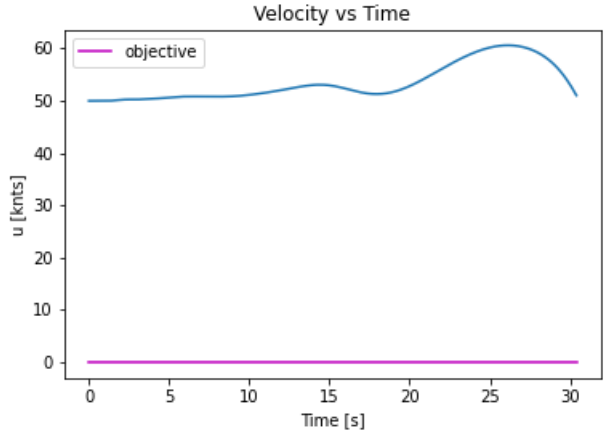


Figure A.108. Actor 2: Velocity vs Time - $H_0 = 600[ft]$, $V_0 = 50[KTAS]$

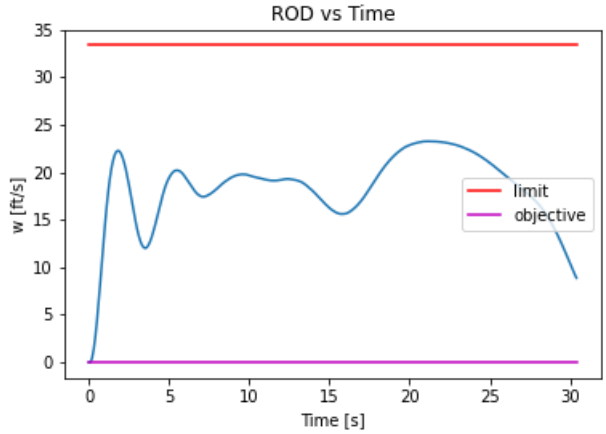


Figure A.109. Actor 2: Rate of Descent vs Time - $H_0 = 600[ft]$, $V_0 = 50[KTAS]$

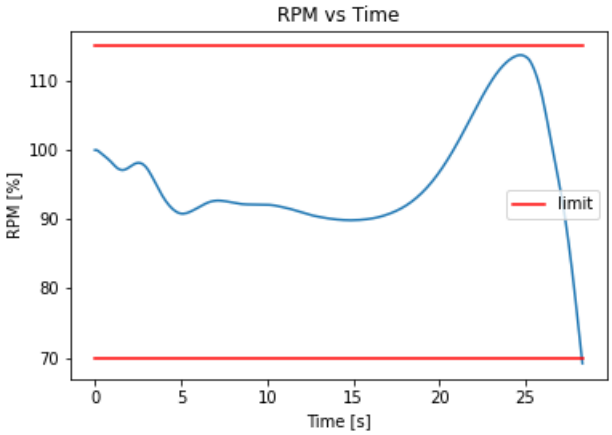


Figure A.110. Actor 2: Main Rotor RPM vs Time - $H_0 = 600[ft]$, $V_0 = 50[KTAS]$

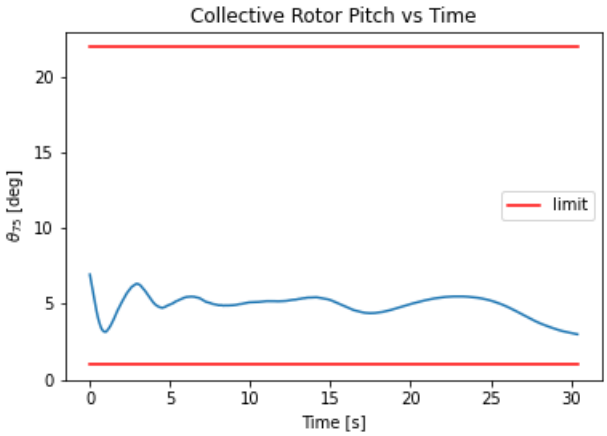


Figure A.111. Actor 2: Main Rotor Collective Pitch vs Time - $H_0 = 600[ft]$, $V_0 = 50[KTAS]$

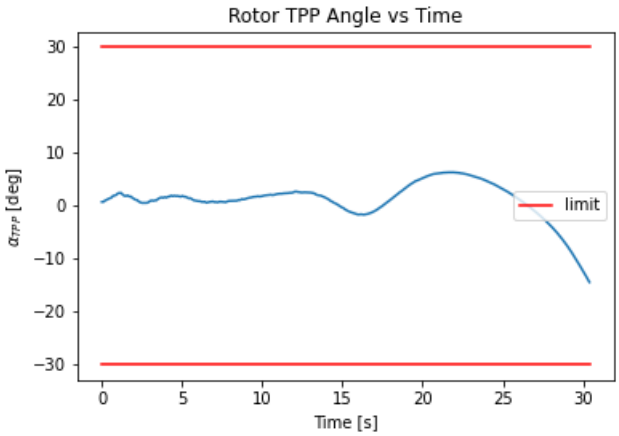


Figure A.112. Actor 2: Tip Path Plane Angle vs Time - $H_0 = 600[ft]$, $V_0 = 50[KTAS]$

Bibliography

[1] Y. Okuno *et al.*, "Analytical Prediction of Height-Velocity Diagram of a Helicopter Using Optimal Control Theory," *Journal of Guidance, Control, and Dynamics*, vol. 14, no. 2, pp. 453–459, 1991.

[2] M. J. Harris, "Analytical Determination of a Helicopter Height Velocity Diagram," Master's thesis, Air Force Institute of Technology, Wright Patterson Air Force Base, 2018.

[3] G. Brockman *et al.*, "Openai gym," *CoRR*, vol. abs/1606.01540, 2016. [Online]. Available: http://arxiv.org/abs/1606.01540

[4] L. Szymanski, "Lecture 7: Deep reinforcement learning," http://www.cs.otago.ac.nz/cosc470/09-deep-reinforcement-learning.pdf, 2018, accessed: 2022–10-22.

[5] J. Schulman *et al.*, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[6] U.S.A.M.C, "Engineering Design Handbook: Helicopter Engineering, Part One, Preliminary Design," Alexandria, VA, 1974.

[7] "Rotorcraft Flying Handbook," U.S. Department of Transportation, Federal Aviation Administration, 2000.

[8] W. Johnson, *Rotorcraft Aeromechanics*. New York, NY: Cambridge University Press, 2013.

[9] D. E. Kirk, *Optimal Control Theory: An Introduction*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1970.

[10] A. V. Rao, "A Survey of Numerical Methods for Optimal Control," in *2009 AAS/AIAA Astrodynamics Specialist Conference, AAS Paper 09-334*, Pittsburgh, PA, 2009, pp. 533–541.

[11] D. Garg *et al.*, "Direct Trajectory Optimization and Costate Estimation of Finite-Horizon and Infinite-Horizon Optimal Control Problems Using a Radau Psudospectral Method," *Computational Optimization and Applications*, vol. 49, pp. 335–358, 2011.

[12] M. A. Patterson and A. V. Rao, "GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hp-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming," *ACM Transactions on Mathematical Software*, vol. 41, no. 1, 2014.

[13] Z. Liu *et al.*, "Regularization matters in policy optimization," *CoRR*, vol. abs/1910.09191, 2019. [Online]. Available: http://arxiv.org/abs/1910.09191

[14] R. S. Sutton and A. G. Barto, *Reinforcement Learning, an Introduction*. Cambridge, MA: The MIT Press, 1998.

[15] S. J. Russell and P. Norvig, *Artificial Intelligence; A Modern Approach*. Pearson Education, 2009.

[16] M. Lee, "Td-gammon," http://www.incompleteideas.net/book/ebook/node108.html, 2005, accessed: 2022–20-22.

[17] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[18] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015. [Online]. Available: https://arxiv.org/abs/1509.02971

[19] J. Schulman *et al.*, "High-dimensional continuous control using generalized advantage estimation," 2015. [Online]. Available: https://arxiv.org/abs/1506.02438

[20] ——, "Trust region policy optimization," *CoRR*, vol. abs/1502.05477, 2015. [Online]. Available: http://arxiv.org/abs/1502.05477

[21] W. Johnson, "Helicopter Optimal Descent and Landing After Power Loss," NASA Ames Research Center, Tech. Rep. NASA-TM-73244, 1977.

[22] A. Y. Lee, "Optimal Landing of a Helicopter in Autorotation," Ph.D. dissertation, Stanford University, 1985.

[23] T. A. Brown, "Investigation of Factors Impacting a Helicopter Height-Velocity Diagram," Master's thesis, Air Force Institute of Technology, Wright Patterson Air Force Base, 2020.

[24] K. Kopsa, "Reinforcement Learning Control For Autorotation of a Simple Point-Mass Helicopter Model," Master's thesis, Middle East Technical University, Ankara, Turkey, 2018.

[25] W. Johnson, "Model for Vortex Ring State Influence on Rotorcraft Flight Dynamics," NASA Ames Research Center, Tech. Rep. NAS AfTP-2005-213477, 2005.

[26] Bell, *Operators Manual For Army Model AH-1F Attack Helicopter*, 2001.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From – To)*
02-03-2023	Master's Thesis	Oct 2021 – Mar 2023

TITLE AND SUBTITLE

ESTIMATION OF OPTIMAL FLIGHT CONTROL FOR A HELICOPTER IN A TOTAL POWER LOSS SCENARIO USING PROXIMAL POLICY OPTIMIZATION

5a. CONTRACT NUMBER N/A

5b. GRANT NUMBER N/A

5c. PROGRAM ELEMENT NUMBER N/A

6. AUTHOR(S)

Eliash, Eidahn, Capt, IAF

5d. PROJECT NUMBER N/A

5e. TASK NUMBER N/A

5f. WORK UNIT NUMBER N/A

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way, Building 640
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENY-MS-23-M-267

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Intentionally left blank

10. SPONSOR/MONITOR'S ACRONYM(S)

N/A

11. SPONSOR/MONITOR'S REPORT NUMBER(S) N/A

12. DISTRIBUTION/AVAILABILITY STATEMENT

DISTRUBTION STATEMENT A. APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

This material is declared a work of the U.S. Government or the Government of Israel and is not subject to copyright protection in the United States and Israel.

14. ABSTRACT

Recent research has been conducted in using deep reinforcement learning algorithms to generate closed form optimal control laws for helicopters in power loss scenarios. The control laws could allow for design of various cockpit visual pilot aids if proven successful. Such pilot aids are predicted to reduce pilot workload during execution of an emergency powerless landing, and to overall increase the probability of landing safely in such an event. The proposed topic discusses the use of the PPO algorithm in order to find an optimal control policy for a helicopter in a total power loss emergency landing scenario. The task consists of designing and training an actor neural network and a critic neural network using the PPO algorithm, along with developing the dynamic equations of motion for the helicopter, defining a reward structure and tweaking relevant hyper-parameters. The PPO algorithm has been proven overall feasible for bringing a simplistically modeled helicopter to safe landings over a wide range of initial flight conditions.

15. SUBJECT TERMS

Height-Velocity Diagram, Optimal Control, Autorotation, Helicopter, Deep Reinforcement Learning, Neural Networks, Machine Learning, Artificial Intelligence, Proximal Polcy Optimization

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Dr. Donald L. Kunz, AFIT/ENY
U	**U**	**U**	**UU**	125	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-6565, ext 3636 (donald.kunz@afit.edu)

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18