ECE498: Senior Capstone Project I

Project Proposal

Project Title: Experiments on 2-DOF Helicopter Using Approximate Dynamic Programming

Anthony Birge and Andrew Fandel
Advisor: Dr. Suruz Miah

November 30, 2017

Electrical and Computer Engineering Department
Caterpillar College of Engineering and Technology
Bradley University

Table of Contents

1 Introduction

Unmanned Aerial Vehicles (UAVs) offer a small-form solution to many aerial tasks and provide many benefits such as cost effectiveness and the lack of a pilot. Consequently, they are rapidly performing many more tasks in the civilian and military domains. UAVs can be controlled automatically by algorithms that serve to perform a given task with minimal control input to the UAV from its control system. Control algorithms that minimize the effort needed to control the UAV by actuation have been advanced in recent literature. 2-Degree of Freedom (2-DOF) helicopters exist as a low-budget solution to implement control algorithms in a testing lab setting. This project aims to advance motion control of a 2-DOF helicopter under various operating conditions using machine learning algorithms. Motion control strategies will be employed for a possible minimization of energy consumed, navigation time, or even the risk for an UAV to complete its pre-defined tasks in an environment.

One of the most challenging tasks in control applications is to model a system, such as the helicopter considered in this project, using a set of ordinary nonlinear differential equations. Under certain operating conditions, the complex model of a helicopter can be simplified to provide appropriate actuator commands to a helicopter. This work leverages the mechanical aspect of a 2-DOF helicopter, the Quanser AERO, manufactured by Quanser Inc. Figure 1 depicts the Quanser AERO configured as a 2-DOF helicopter. The AERO can be configured as a half-quadcopter by rotating one of the rotors a quarter of a turn as well. Note that the Quanser AERO can be modeled as a set of linear differential equations; therefore, a set of well-defined motion control strategies can be applied to control its pitch and yaw motion when its base is fixed. Here, we focus on implementing model-based reinforcement learning strategy to determine the sub-optimal actuator commands to the helicopter's actuators for it to track its pre-defined motion trajectories.

Figure 1: Quanser AERO configured as 2-DOF helicopter.

To demonstrate the effectiveness of the proposed control technique, we will implement the proposed control technique to both configurations of the AERO in various operating conditions. The proposed control technique will be simulated in both configurations in MATLAB, and the half-quadcopter configuration will be simulated in Virtual Robot Experimentation Platform (V-REP) [1]. Both configurations will then be implemented to the Quanser AERO. A user is expected to control the helicopter using a desktop or laptop computer in conjunction with a single-board microcomputer, such as a Raspberry Pi or BeagleBone. The ultimate goal of implementation will have a personal computer/laptop communicating to the single-board computer via Wi-Fi from which the single-board microcomputer will control the Quanser AERO. This implementation should result in a cost-effective and convenient control technique.

2 Background Study

With the increase in popularity and applications of UAVs, the control techniques implemented are being investigated more so. Due to the complex models of UAVs, traditional control techniques prove irrelevant. New nonlinear controller design techniques or learning controllers must be implemented to effectively control a desired

[1] See http://www.coppeliarobotics.com/ for details.

UAV. These types of control techniques were researched in order to obtain a background knowledge in quadcopter and helicopter control.

2.1 Helicopter

For the control of the 2-DOF helicopter, many techniques have provided satisfactory results. These control techniques include LQR, fuzzy control, adaptive programming, sliding control [1, 2, 3, 4, 5, 6, 7]. Due to the complexity of most of these techniques, real-time implementation of these techniques can be difficult or impossible. [1] provides a torque diagram for the 2-DOF helicopter which can be used to further understand the state-space model of the 2-DOF helicopter.

In addition to performing as a helicopter, the Quanser AERO also works as a half-quadcopter. Note that the half-quadcopter model does not exist in the commercial robot simulator, VREP, that will be used for conducting simulations. Therefore, we will be using a full-quadcopter model that exists in the robotic simulator to emulate the half-quadcopter. A brief description of a quadcopter model is provided below.

2.2 Quadcopter

In addition to the derivation of the quadcopter model, implementation of various control methods were researched for a quadcopter. [8] derived a model for a quadcopter and implemented a PI controller, but [8] is still working on more advanced controllers such as LQ optimization and H∞. [9] was able to successfully implement a derivation based on Newton-Euler and Euler-Lagrange equations with a PD controller. The effect of air dynamics is explained in [10]. Further, [11] developed a fuzzy logic controller for the platform.

2.3 Approximate Dynamic Programming

[12] uses approximate dynamic programming for the tracking problem of a mobile wheeled robot. Approximate dynamic programming is the strategic method of dividing a complex problem into simpler problems and then solving those simpler problems with use of applicable algorithms.

In the case of [12], the total time of control is divided into sampling periods of time T. During the time period T, the error of the states of the system are sampled every τ seconds and stored as data points. During the period T, an actor-critic neural network uses the data collected to find the optimal inputs to minimize the error of mobile wheeled robot. The new optimal inputs calculated by the actor-critic neural network will be used for the next period of time T.

The proposed technique in [12] is ideal for real-time implementation because of the recursive actor-critic neural network and data acquisition approach. The system model does not need to be known in order to use the technique as hinted by the author. After much research, we found that not many control techniques have been

proposed for the 2-DOF helicopter using approximate dynamic programming and an actor-critic neural network. The only trouble will be adapting the proposed technique from a wheeled mobile robot to a 2-DOF helicopter.

The Quanser AERO is a common laboratory 2-DOF helicopter used for research and teaching purposes. Through literature provided by Quanser, the Quanser AERO can be controlled using Simulink or a single-board microcomputer. With references on interfacing a single-board microcomputer, we feel a cost-effective method to implement the proposed approximate dynamic programming controller is achievable.

3 Functional Requirements

3.1 System Architecture

The ultimate goal of the project is to control a 2-DOF helicopter in the form of the Quanser AERO with the proposed approximate dynamic programming algorithm. To deliver a cost-effective solution to control the 2-DOF helicopter, implementation of the algorithm will consist of using a single-board microcomputer such as a BeagleBone or Raspberry Pi.

3.2 High-Level System Block Diagram

A high-level system block diagram of the targeted implementation is shown in Figure 2.

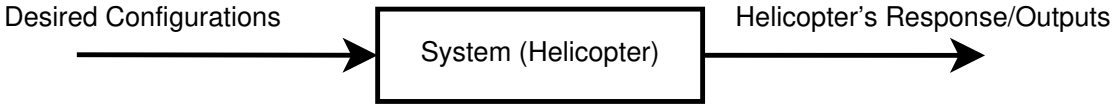

Figure 2: High-level system block diagram.

3.2.1 Inputs

- *Desired Configurations* - The pitch and yaw of the 2-DOF helicopter are the two desired configurations to be inputted into the helicopter setup. The pitch is the vertical adjustment of the helicopter, while the yaw is the horizontal adjustment.

3.2.2 System

- *Helicopter* - The helicopter is the physical Quanser AERO. The approximate dynamic programming controller will be implemented to the helicopter through the computer and single-board microcomputer as seen in Figure 3.

3.2.3 Outputs

- *Helicopter's Response/Outputs* - The measured pitch and measured yaw of the helicopter constitute the response of the system. If the controller functions properly and is implemented correctly, the response of the helicopter will closely track that of the desired configurations.

3.3 Helicopter Subsystem Block Diagram

The helicopter configuration to be used in the project is the Quanser AERO 2-DOF helicopter. The Quanser AERO can be configured both as a half-quadcopter and helicopter.

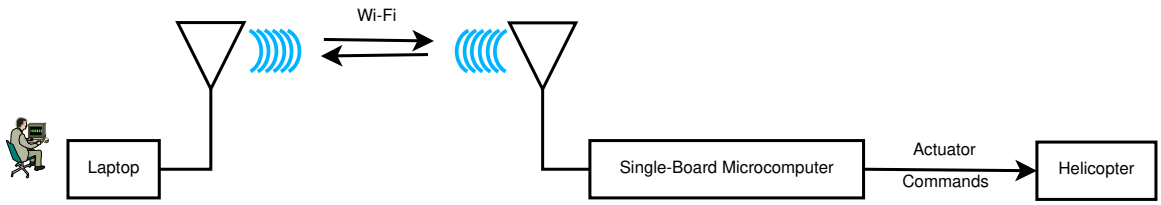

Figure 3: Helicopter subsystem block diagram.

3.3.1 System/Signals

- *Laptop* - The laptop controls all operations of the system. All commands are initialized by the computer to be processed by the other blocks of the system.
- *Wi-Fi* - The computer will send a Wi-Fi signal to either a Raspberry Pi or BeagleBone single-board microcomputer. This signal will contain the necessary instructions for operation of the single-board computer.
- *Single-Board Microcomputer* - The Raspberry Pi or BeagleBone single-board microcomputer will receive instructions from the computer through a Wi-Fi signal. These instructions will in turn be used to operate the helicopter block.
- *Helicopter* - The helicopter is the Quanser AERO 2-DOF helicopter. The subsystem block diagram for the helicopter can be seen in Figure 2.

3.4 Specifications

The only specification for 2-DOF helicopter controller is that the proposed algorithm be implemented on a single-board microcomputer. This specification can derive many other constraints especially those pertaining to the particular single-board microcomputer such as sampling time and computational overhead.

4 Preliminary Work

The preliminary work has been divided into two sections thus far. Preliminary work has been in progress concerning the half-quadcopter configuration. This work includes mathematical modeling of both the quadcopter and the half-quadcopter. MATLAB and VREP simulations are currently being developed on both of these configurations as well. We are specifically researching the quadcopter because of its similarity to the half-quadcopter. The half-quadcopter model is not available in VREP, so if we study the quadcopter model which is available, the goal is to constrain the quadcopter model to act like a half-quadcopter model in VREP. We will then be able to see the effectiveness of the approximate dynamic programming algorithm in a virtual-sense before implementation.

The other half of the preliminary work is concerned with the helicopter configuration. This work includes mathematical modeling as well as MATLAB simulations which are currently in progress. VREP simulations will not be considered for the helicopter configuration.

4.1 Half-Quadcopter

To use the simulation software, a mathematical model for the quadcopter must be derived. The PI control mentioned in the literature review will be tested to verify the model, then model will be specialized to the instance of the half-quadcopter by removing a pair of motors/propellers inputs into the system.

4.1.1 Quadcopter Modeling

The quadcopter model is shown in Figure 4. Thus far, a model has been presented that relates the linear and angular velocities and accelerations.

The quadcopter free body diagram is shown below. The state-space model of the quadcopter can be seen on the following slides.

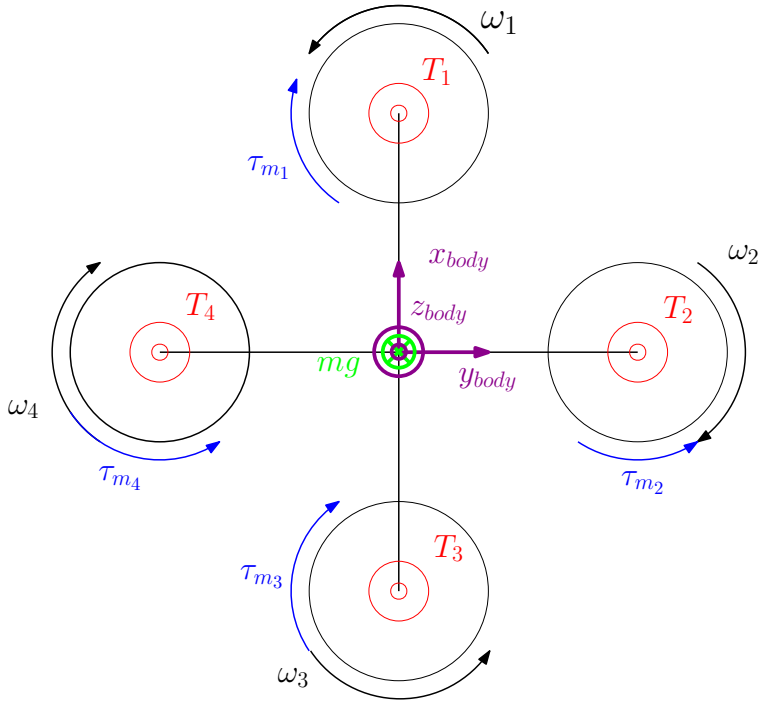

Figure 4: Free Body Diagram

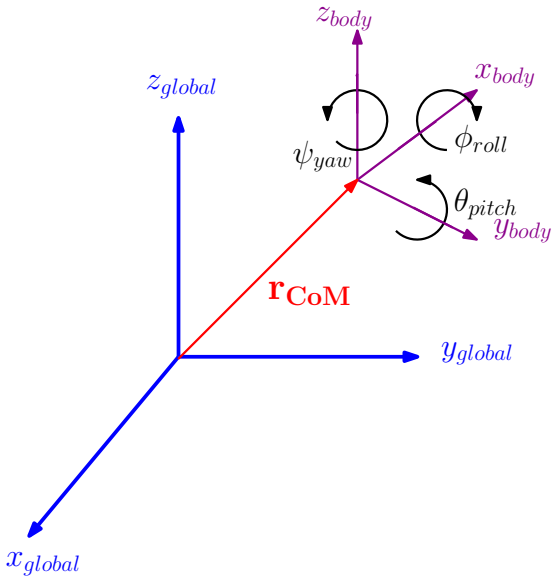

Figure 5: Global Frame, Body Frame, and Position Vector.

4.1.2 Modeling

The state-space model of the quadcopter is seen in the equations below. Research was done to better understand the model in order to make implementation easier. The model is highly non-linear but can be modeled through code.

First we'll need to define some terms:

- l is the length of a quadcopter arm; half of this distance is the distance from the center of mass to the motor shaft.
- m is the mass of the quadcopter.
- g is the gravitational constant.
- b is the drag coefficient for the propellers.
- F_d is the drag force coefficient relative to the quadcopter in X, Y, and Z.
- c_t is the thrust coefficient, derived from motor and propeller parameters.
- ω_i is the propeller (1-4) angular speed.
- $T_B = \sum_1^4 c_t \omega_i^2$
- x quadcopter's center of mass X coordinate in global frame.
- $\dot{x}$ quadcopter's center of mass X velocity in global frame.
- $\ddot{x}$ quadcopter's center of mass X acceleration in global frame.
- y quadcopter's center of mass Y coordinate in global frame.
- $\dot{y}$ quadcopter's center of mass Y velocity in global frame.
- $\ddot{y}$ quadcopter's center of mass Y acceleration in global frame.
- z quadcopter's center of mass Z coordinate in global frame.
- $\dot{z}$ quadcopter's center of mass Z velocity in global frame.
- $\ddot{z}$ quadcopter's center of mass Z accleration in global frame.
- ϕ is the roll (about X).
- $\dot{\phi}$ is velocity about body frame X.
- $\ddot{\phi}$ is angular acceleration about body frame X.
- θ is the pitch (about Y).
- $\dot{\theta}$ is velocity about body frame Y.
- $\ddot{\theta}$ is angular acceleration about body frame Y.
- ψ is the roll (about Z).
- $\dot{\psi}$ is velocity about body frame Z.

- $\ddot{\psi}$ is angular acceleration about body frame Z.
- I_{xx} is the moment of inertia of quadcopter about body x axis.
- I_{yy} is the moment of inertia of quadcopter about body y axis.
- Note that $I_{xx} = I_{yy}$ by symmetry.
- I_{zz} is the moment of inertia of quadcopter about body z axis.
- R is the matrix that transforms the body frame to the global frame

$$R = \begin{bmatrix} \cos(\psi)\cos(\theta) & \cos(\psi)\sin(\theta)\sin(\phi) - \sin(\psi)\cos(\phi) & \cos(\psi)\sin(\theta)\cos(\phi) + \sin(\psi)\sin(\phi) \\ \sin(\psi)\cos(\theta) & \sin(\psi)\sin(\theta)\sin(\phi) + \cos(\psi)\cos(\phi) & \sin(\psi)\sin(\theta)\cos(\phi) - \cos(\psi)\sin(\phi) \\ -\sin(\theta) & \cos(\theta)\sin(\psi) & \cos(\theta)\cos(\phi) \end{bmatrix} \tag{1}$$

where

$$x_1 = \begin{bmatrix} x & y & z \end{bmatrix}^T \tag{2a}$$
$$x_2 = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T \tag{2b}$$
$$x_3 = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T \tag{2c}$$
$$x_4 = \begin{bmatrix} \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T \tag{2d}$$

$$\dot{x_1} = x_2 \tag{3}$$
$$\dot{x_2} = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \frac{1}{m} R T_B + \frac{1}{m} F_d \tag{4}$$
$$\dot{x_3} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \cos\theta\sin\phi \\ 0 & -\sin\phi & \cos\theta\sin\phi \end{bmatrix}^{-1} x_4 \tag{5}$$
$$\dot{x_4} = \begin{bmatrix} \frac{1}{I_{XX}} & 0 & 0 \\ 0 & \frac{1}{I_{YY}} & 0 \\ 0 & 0 & \frac{1}{I_{ZZ}} \end{bmatrix} \left(\begin{bmatrix} lc_t(-\omega_2^2 + \omega_4^2) + \dot{\theta}\dot{\psi}(I_{yy} - I_{zz}) \\ lc_t(-\omega_1^2 + \omega_3^2) + \dot{\phi}\dot{\psi}(I_{zz} - I_{xx}) \\ b\sum_{i=1}^{4} \omega_i^2 \end{bmatrix} \right) \tag{6}$$
$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T \tag{7}$$

Our inputs to the system are the thrust values generated by each motor. The thrust is linked by the relationship shown in Equation 8. Since we are able to change the angular velocity of the propeller, we are able to change the square of the angular

velocity of the propeller as well. We need to derive an equation that relates the Quanser Aero's motor characteristics to Equation 8.

$$T_B = \sum_{1}^{4} c_t \omega_i^2 \tag{8}$$

The proposed non-linear state space representation is of the form Equation 9 and is shown in Equation 10.

$$\dot{\mathbf{q}} = \mathbf{f}(\mathbf{q}) + \mathbf{G}(\mathbf{q})\mathbf{u} \tag{9}$$

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{y} \\ \ddot{y} \\ \dot{z} \\ \ddot{z} \\ \dot{\phi} \\ \ddot{\phi} \\ \dot{\theta} \\ \ddot{\theta} \\ \dot{\psi} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ -\frac{F_d}{m} \\ \dot{y} \\ -\frac{F_d}{m} \\ \dot{z} \\ -\frac{F_d}{m} \\ \dot{\phi} \\ \dot{\theta}\dot{\psi}(I_{yy} - I_{zz}) \\ \dot{\theta} \\ \dot{\phi}\dot{\psi}(I_{zz} - I_{xx}) \\ \dot{\psi} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ \frac{c_t T_x}{m} & \frac{c_t T_x}{m} & \frac{c_t T_x}{m} & \frac{c_t T_x}{m} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{c_t T_y}{m} & \frac{c_t T_y}{m} & \frac{c_t T_y}{m} & \frac{c_t T_y}{m} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{c_t T_z}{m} & \frac{c_t T_z}{m} & \frac{c_t T_z}{m} & \frac{c_t T_z}{m} & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{-lc_t}{I_{xx}} & 0 & \frac{lc_t}{I_{xx}} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{-lc_t}{I_{yy}} & 0 & \frac{lc_t}{I_{yy}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ b & -b & b & -b & 0 \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \\ g \end{bmatrix} \tag{10}$$

Where:

$$T_x = (\sin\phi \sin\psi + \cos\psi \sin\theta \cos\phi) \tag{11a}$$

$$T_y = (-\sin\phi \cos\psi + \sin\psi \sin\theta \cos\phi) \tag{11b}$$

$$T_z = (\cos\psi \cos\phi) \tag{11c}$$

The proposed linearized state space representation is generated by assuming small $\dot{\phi}, \dot{\theta}, \dot{\psi}$ and deviations in ϕ, θ such that the quadcopter hovers at the destination point and is shown in Equation 12.

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{y} \\ \ddot{y} \\ \dot{z} \\ \ddot{z} \\ \dot{\phi} \\ \ddot{\phi} \\ \dot{\theta} \\ \ddot{\theta} \\ \dot{\psi} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{F_d}{m} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{F_d}{m} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{F_d}{m} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ z \\ \dot{z} \\ \phi \\ \dot{\phi} \\ \theta \\ \dot{\theta} \\ \psi \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{c_t}{m} & \frac{c_t}{m} & \frac{c_t}{m} & \frac{c_t}{m} & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{-lc_t}{I_{xx}} & 0 & \frac{lc_t}{I_{xx}} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{-lc_t}{I_{yy}} & 0 & \frac{lc_t}{I_{yy}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ b & -b & b & -b & 0 \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \\ g \end{bmatrix} \tag{12}$$

The error vector will be expressed as:

$$\mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \\ e_7 \\ e_8 \\ e_9 \\ e_{10} \\ e_{11} \\ e_{12} \end{bmatrix} = \begin{bmatrix} x^d - x \\ y^d - y \\ z^d - z \\ \dot{x}^d - \dot{x} \\ \dot{y}^d - \dot{y} \\ \dot{z}^d - \dot{z} \\ \phi^d - \phi \\ \theta^d - \theta \\ \psi^d - \psi \\ \dot{\phi}^d - \dot{\phi} \\ \dot{\theta}^d - \dot{\theta} \\ \dot{\psi}^d - \dot{\psi} \end{bmatrix} = \mathbf{x}^d - \mathbf{x} \tag{13}$$

Because we want the quadcopter to hover, we can assume that the desired roll pitch and yaw are 0 as well as the angular velocities. Further, we can assume the x, y, and z velocities to be zero as well. Thus the error vector is, more specifically:

$$\mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \\ e_7 \\ e_8 \\ e_9 \\ e_{10} \\ e_{11} \\ e_{12} \end{bmatrix} = \begin{bmatrix} x^d - x \\ y^d - y \\ z^d - z \\ -\dot{x} \\ -\dot{y} \\ -\dot{z} \\ -\phi \\ -\theta \\ \psi^d - \psi \\ -\dot{\phi} \\ -\dot{\theta} \\ -\dot{\psi} \end{bmatrix} = \mathbf{x}^d - \mathbf{x} \tag{14}$$

4.1.3 MATLAB Simulation Results

After implementing the quadcopter model in MATLAB code, we will verify the accuracy of the model by implementing a simple PI controller that will navigate the quadcopter from one point to another. Once this implementation has been verified, the approximate dynamic programming algorithm will be tested on the model. Currently, we are still working on implementing the PI controller.

4.1.4 V-REP Simulation Results

Thus far, interfacing MATLAB and V-REP has been difficult for simulating a quadcopter. Minimal literature on V-REP scenes exist, and much of the code must be written in LUA runtime code without the use of a compiler. Further, the simulations require attention to air dynamics. All of these considerations have made simulating a quadcopter in V-REP difficult, but work is continuing. Current results can be seen in Figure 6 and Figure 7.

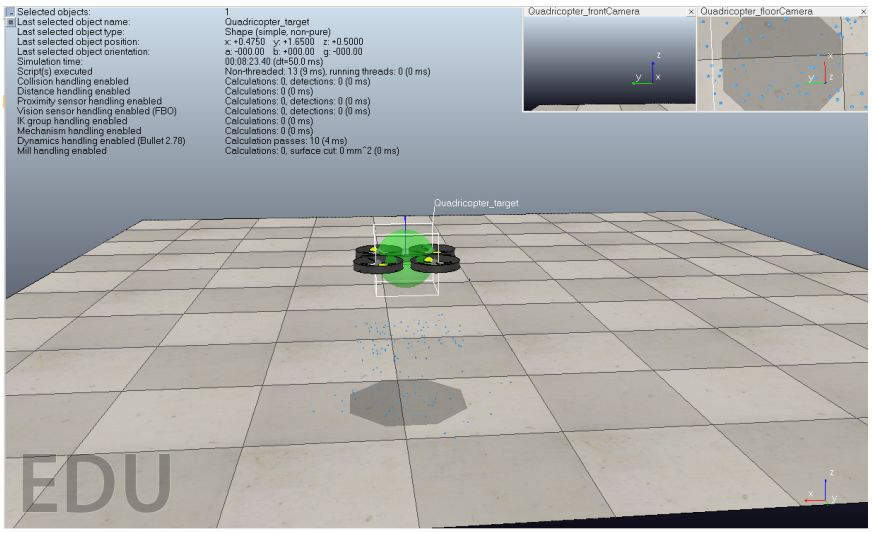

Figure 6: Quadcopter hovering in V-REP.

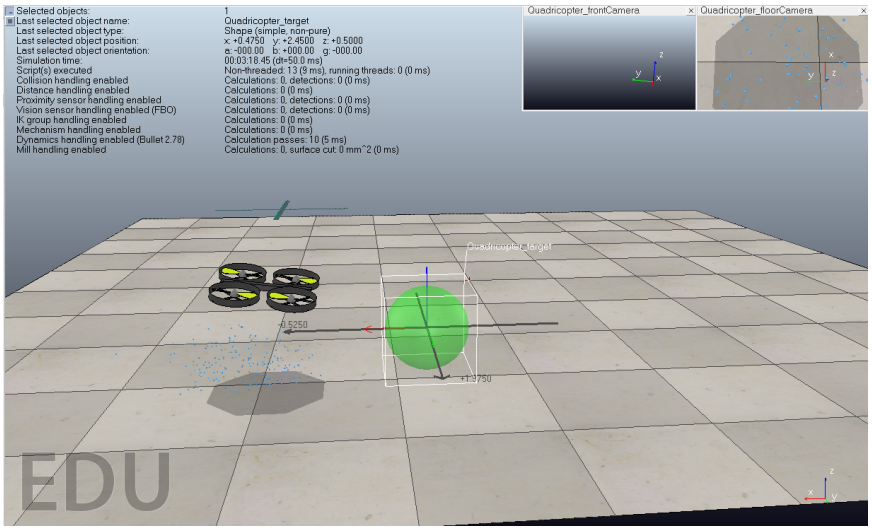

Figure 7: Quadcopter approaching a point in V-REP.

4.2 Helicopter

The 2-DOF helicopter configuration is the same as that seen in Figure 1. The preliminary work thus far has pertained to the mathematical modeling of the 2-DOF helicopter configuration and using that mathematical model as a platform for the proposed control algorithm in MATLAB.

4.2.1 Modeling

In order to derive the mathematical model of the 2-DOF helicopter configuration, we need to define the state variables and the inputs of the system. For our purposes, the state variables are $\mathbf{x}^{'} = [\theta, \psi, \dot{\theta}, \dot{\psi}]$ where θ is the pitch and ψ is the yaw of the 2-DOF

helicopter. The inputs are $\mathbf{u}^{'} = [V_p, V_y]$ where V_p is the applied voltage to the main rotor which changes the pitch, and V_y is the applied voltage to the tail rotor which changes the yaw.

The general free body diagram of the configuration can be seen in Figure 8.

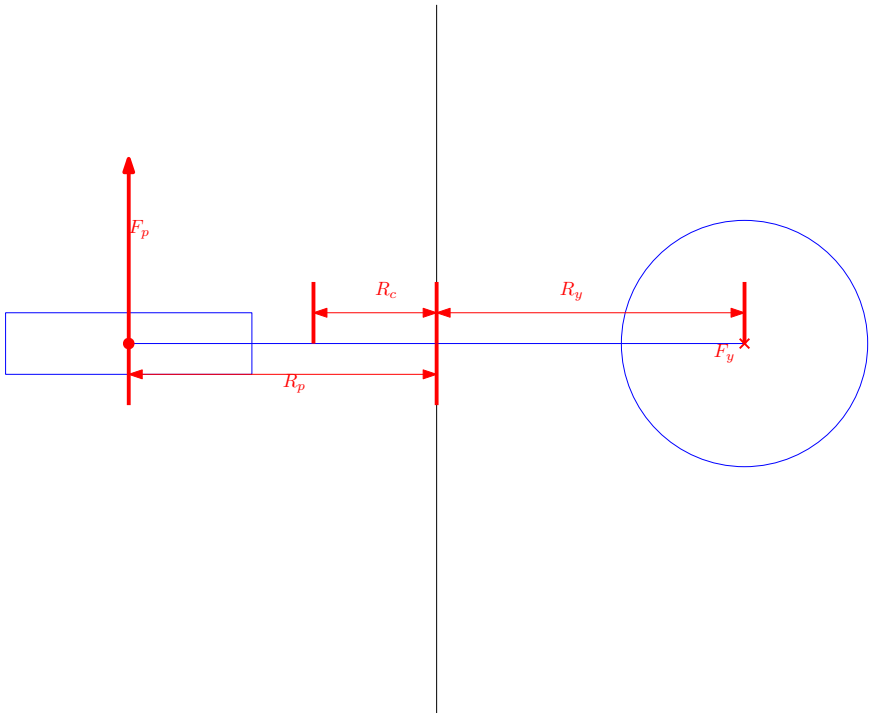

Figure 8: Free body diagram of the 2-DOF helicopter. Only the propeller forces are represented. Measurements are also shown.

The free body diagram of the 2-DOF helicopter can be divided into the vertical and horizontal planes. These free body diagrams can be seen in Figure 9 and Figure 10 respectively. Notice that all of the forces have been illustrated using [1] as a reference.

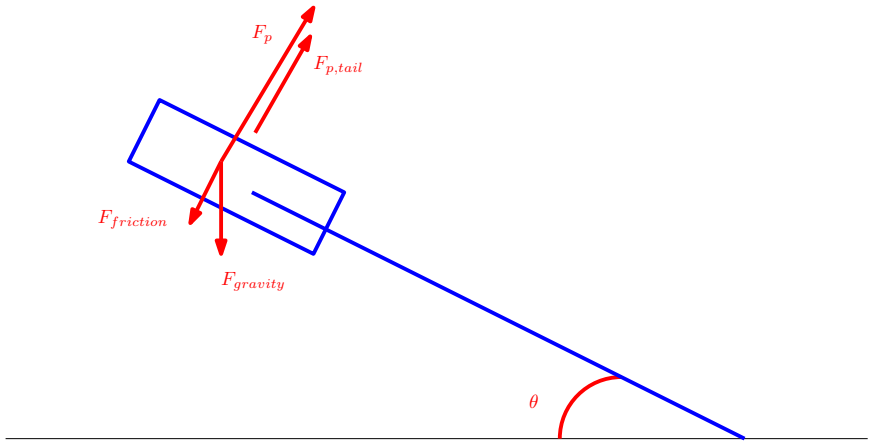

Figure 9: Free body diagram of the 2-DOF helicopter. Only the propeller forces are represented. Measurements are also shown.

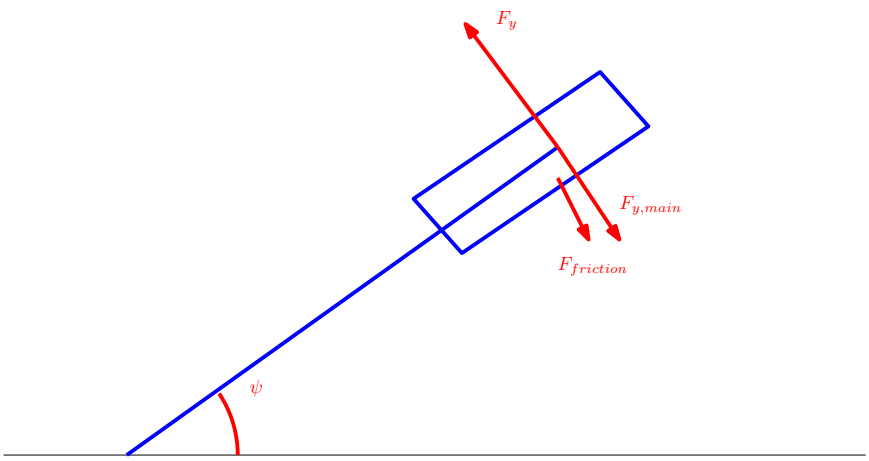

Figure 10: Free body diagram of the 2-DOF helicopter. Only the propeller forces are represented. Measurements are also shown.

Using the free body diagrams and standard force and torque equations, the state-space model of the 2-DOF helicopter configuration can be derived. The nonlinear state-space model can be seen in Equation 15.

$$\begin{bmatrix} \dot{\theta} \\ \dot{\psi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{R_p \beta_p}{I_p} & 0 \\ 0 & 0 & 0 & -\frac{R_y \beta_y}{I_y} \end{bmatrix} \begin{bmatrix} \theta \\ \psi \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{K_{p,main}}{I_p} & 0 \\ 0 & \frac{K_{y,tail}}{I_y} \end{bmatrix} \begin{bmatrix} V_p \\ V_y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{G_p}{I_p} - \frac{R_c m_{body} g sin(\theta)}{I_p} \\ -\frac{G_y}{I_y} \end{bmatrix} \tag{15}$$

The variables used here are as follows:

- R_p is the distance from the pitch rotor to the fork.
- R_y is the distance from the tail rotor to the fork.
- R_c is the distance from the fork to the center of mass of the body.
- β_p is the damping coefficient associated with the main rotor.
- β_y is the damping coefficient associated with the tail rotor.
- I_p is the rotational inertia of the main rotor.
- I_y is the rotational inertia of the tail rotor.
- $K_{p,main}$ is the thrust constant associated with the main rotor.
- $K_{y,tail}$ is the thrust constant associated with the tail rotor.
- G_p is the nonlinear coupling on the main rotor due to the tail rotor.
- G_y is the nonlinear coupling on the tail rotor due to the main rotor.

- m_{body} is the mass of the body.
- g is the acceleration due gravity.

The state-space model of the 2-DOF helicopter is highly nonlinear. The model can be linearized with a few assumptions ($sin(\theta) \approx \theta$, $G_p = K_{p,tail}V_y$, and $G_p = K_{y,main}V_p$).

With the assumptions listed above and the nonlinear state-space model of the 2-DOF helicopter, we found the linear model of the 2-DOF helicopter.

$$\begin{bmatrix} \dot{\theta} \\ \dot{\psi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{R_c m_{body} g}{I_p} & 0 & -\frac{R_p \beta_p}{I_p} & 0 \\ 0 & 0 & 0 & -\frac{R_y \beta_y}{I_y} \end{bmatrix} \begin{bmatrix} \theta \\ \psi \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{K_{p,main}}{I_p} & \frac{K_{p,tail}}{I_p} \\ -\frac{K_{y,main}}{I_y} & \frac{K_{y,tail}}{I_y} \end{bmatrix} \begin{bmatrix} V_p \\ V_y \end{bmatrix} \tag{16}$$

The model parameters were measured by Quanser, so it was only a matter of using them with the correct variables. In the end, the linearized state-space model of the Quanser AERO can be seen in Equation 17. This state-space model matches the one provided with the Quanser AERO software within rounding.

$$\begin{bmatrix} \dot{\theta} \\ \dot{\psi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1.7442 & 0 & -0.3307 & 0 \\ 0 & 0 & 0 & -0.9283 \end{bmatrix} \begin{bmatrix} \theta \\ \psi \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0.0512 & 0.0977 \\ -0.1139 & 0.0928 \end{bmatrix} \begin{bmatrix} V_p \\ V_y \end{bmatrix} \tag{17}$$

In order to use the proposed algorithm, we also needed to derive the error model of the 2-DOF helicopter. For the error model, we want the error to approach zero asymptotically, that is as $t \to \infty$, $\mathbf{e} \to 0$. Let us also assume that the 2-DOF helicopter will approach a desired pitch and yaw as $t \to \infty$ such that $\mathbf{x}^{d'} = [\theta^d, \psi^d, 0, 0]$. We can then define the error vector as the following.

$$\mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix} = \begin{bmatrix} \theta^d - \theta \\ \psi^d - \psi \\ -\dot{\theta} \\ -\dot{\psi} \end{bmatrix} = \mathbf{x}^d - \mathbf{x} \tag{18}$$

The error model of the 2-DOF helicopter can then be formulated as Equation 18.

$$\dot{\mathbf{e}} = \mathbf{A}\mathbf{e} - \mathbf{B}\mathbf{u} - \mathbf{A}\mathbf{x}^d \tag{19}$$

We can then use this error model to find the optimal inputs to make $\mathbf{e} \to 0$ as $t \to \infty$. Other error models may be used during the research process.

4.2.2 MATLAB Simulation Results

To simulate the proposed approximate dynamic programming controller, we can implement the error model derived and the proposed algorithm into MATLAB. These

simulations are currently being researched.

The proposed algorithm uses a actor-critic neural network to find the optimal inputs to minimize the error model. Because we linearized the state-space model of the 2-DOF helicopter, we must rely on the actor-critic neural network to find the optimal inputs to minimize the error model. The actor-neural network structure can be seen in Figure 11.

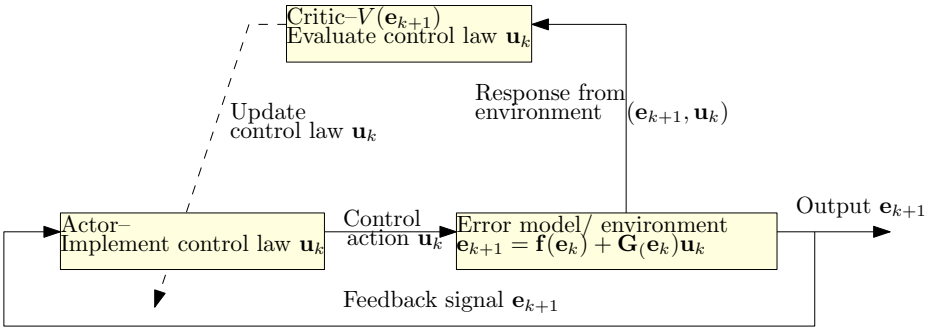

Figure 11: Actor-Critic neural network description.

For the MATLAB simulation, the initial inputs are applied to the linearized state-space model for the first sampling period of the simulation. This should create some type of error in the states due to the linearized model. After each period of using the calculated inputs, the actor-critic neural network weight approximation algorithm will determine the weights needed to determine the next set of optimal inputs. The process will then be repeated for the next period.

These MATLAB simulations are currently being researched. Due to difficulties with actor-critic neural network weight approximation algorithm, we are currently researching the regulator control problem. The regulator control problem is specifying the desired states to be all zero. We start from arbitrary initial conditions and wish to have the response converge to zero.

Because we are trying to develop a new control algorithm, we need another control method to compare the results with. One of the most popular control methods is the linear quadratic regulator (LQR). LQR utilizes a cost function to find the best state-feedback gain values for a system. The cost function can be seen in Equation 20.

$$J = \int_0^\infty [\mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{u}^T \mathbf{R} \mathbf{u}] dt \tag{20}$$

LQR has been used in many applications of the 2-DOF helicopter, so this control method will provide a baseline controller to compare our results to. The simulations developed so far can be seen in Figure 12, Figure 13, and Figure 14.

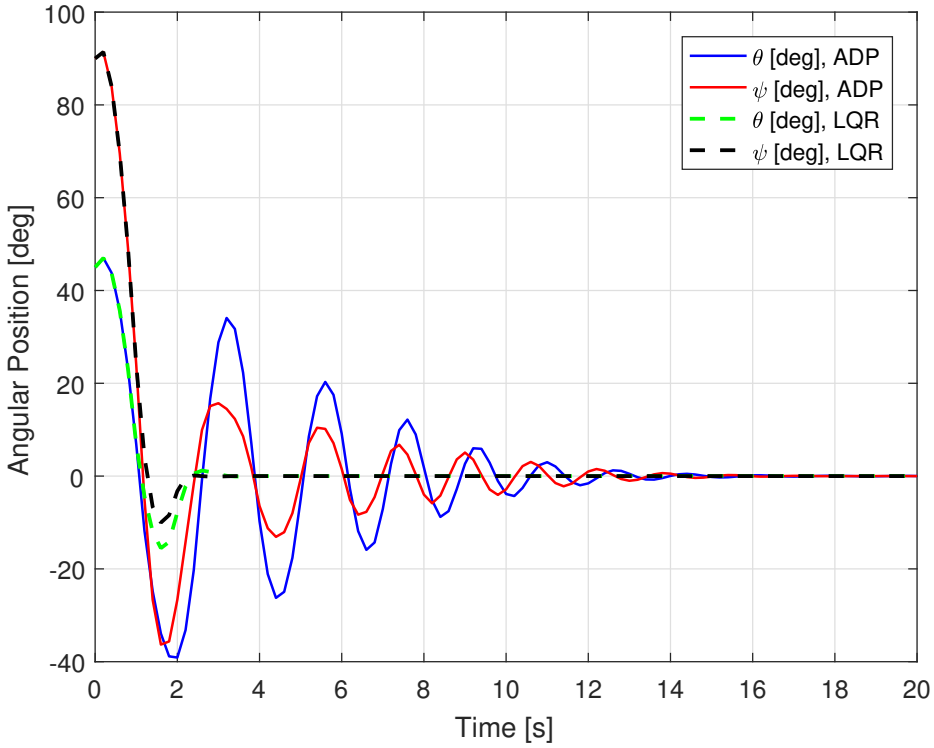

Figure 12: Angular position using the approximate dynamic programming algorithm and LQR.

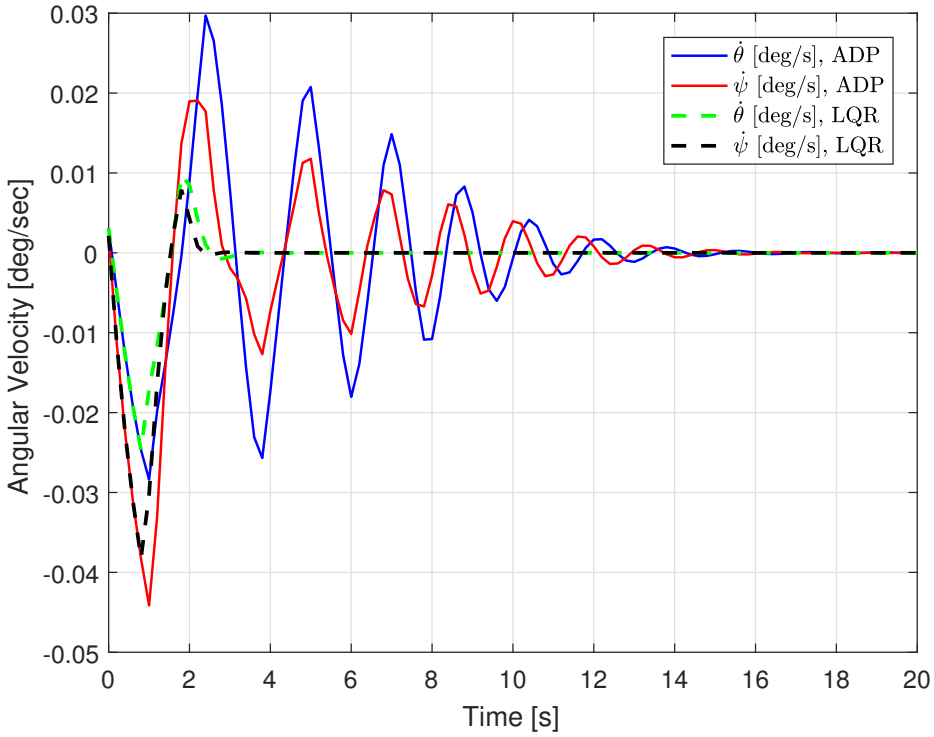

Figure 13: Angular velocity using the approximate dynamic programming algorithm and LQR.

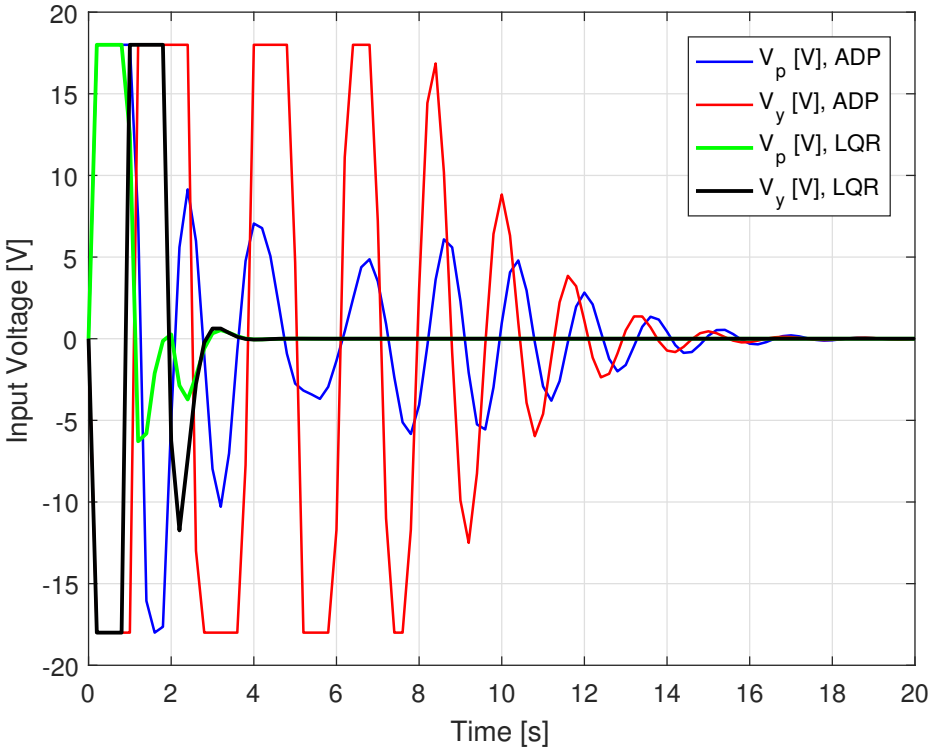

Figure 14: Motor voltage inputs using the approximate dynamic programming algorithm and LQR.

From the simulation results, we can see that LQR control method seems to be outperforming the approximate dynamic programming algorithm in terms of response time. In theory, the approximate dynamic programming algorithm should provide better results than the LQR method. Further research will need to be performed in order to find how to accomplish better results. We may need to research the weight matrices in the cost function or the error model.

Even though the approximate dynamic programming algorithm does not out perform LQR, the MATLAB simulations seem acceptable. The MATLAB simulations use the error model of the system to propagate random errors for data collection for the actor-critic neural network. When we physically implement the approximate dynamic programming algorithm to the Quanser AERO, the error data for the actor-critic neural network will only come from measurements. The error model will not be needed for implementation. If LQR is used in implementation, though, the state-space model matrices will be needed prior to implementation to determine the state-feedback gains. This will be a drawback of LQR compared to the approximate dynamic programming algorithm. We will need to consider the drawbacks of both control methods when implementing them.

5 Parts List

We are fortunate to have all of the necessary equipment readily available. Dr. Miah had already purchased a Quanser AERO, so it was only a matter of contacting Quanser for a separate license to operate the Quanser AERO using our laptops. The software we are using, whether it be MATLAB, V-REP, or QUARC, is readily available or was used in other classes. As for the single-board microcomputers, Dr. Miah also has a few BeagleBones and Raspberry Pies readily available.

6 Timeline

The Gantt chart in Figure 15 shows our current and expected progress for the fall semester. The simulations in both MATLAB and V-REP have been taking longer than anticipated this semester hindering our progress. The Gantt chart in Figure 16 shows our desired timeline for the spring semester. The spring semester will consist of mainly integrating the proposed algorithm to the physical Quanser AERO either through Simulink or a single-board microcomputer such as a BeagleBone or Raspberry Pi.

References

[1] Q. Ahmed, A. I. Bhatti, S. Iqbal, and I. H. Kazmi, "2-sliding mode based robust control for 2-dof helicopter," in *2010 11th International Workshop on Variable Structure Systems (VSS)*, June 2010, pp. 481–486.

[2] J. L. H. Chang, W.; Moon, "Fuzzy model-based output-tracking control for 2 degree-of-freedom helicopter," *Journal of Electrical Engineering Technology*, vol. 12.00, no. 1, pp. 1921–1928, 2017, quanser product(s): 2 DOF Helicopter. [Online]. Available: http://www.jeet.or.kr/LTKPSWeb/uploadfiles/be/201705/29052017095734400375O.pdf

[3] W. Gao and Z. P. Jiang, "Data-driven adaptive optimal output-feedback control of a 2-dof helicopter," in *2016 American Control Conference (ACC)*, July 2016, pp. 2512–2517.

[4] M. Hernandez-Gonzalez, A. Alanis, and E. Hernandez-Vargas, "Decentralized discrete-time neural control for a quanser 2-dof helicopter," in *Applied Soft Computing*, February 2012, pp. 2462–2469.

[5] E. Kayacan and M. Khanesar, "Recurrent interval type-2 fuzzy control of 2-dof helicopter with finite time training algorithm," in *IFAC-PapersOnLine*, July 2016, pp. 293–299.

[6] K. Subbarao, P. Nuthi., and G. Atmeh, "Reinforcement learning based computational adaptive optimal control and system identification for linear systems," in *Annual Reviews in Control*, September 2016, pp. 319–331.

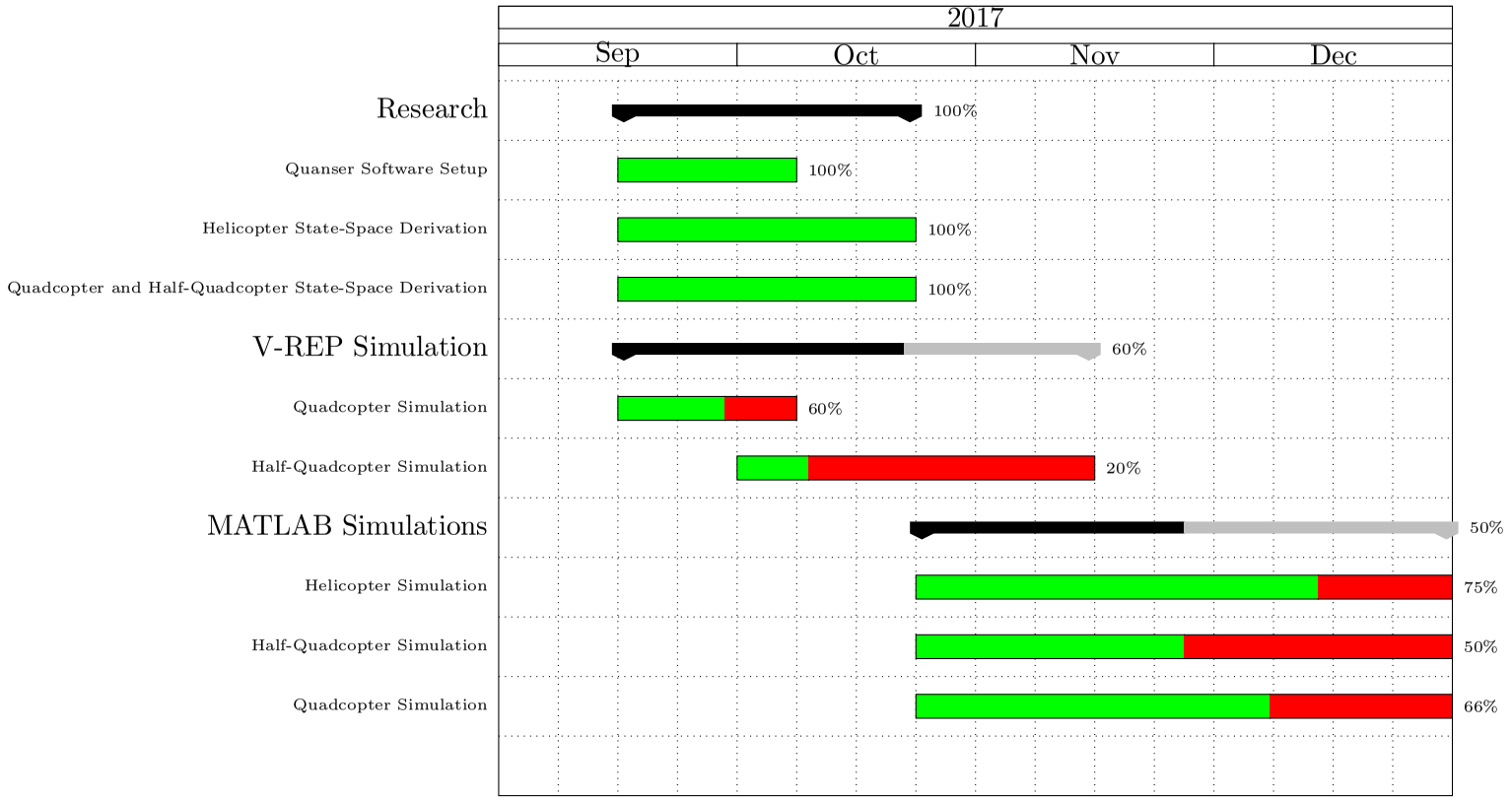

Figure 15: Gantt Chart for Fall 2017

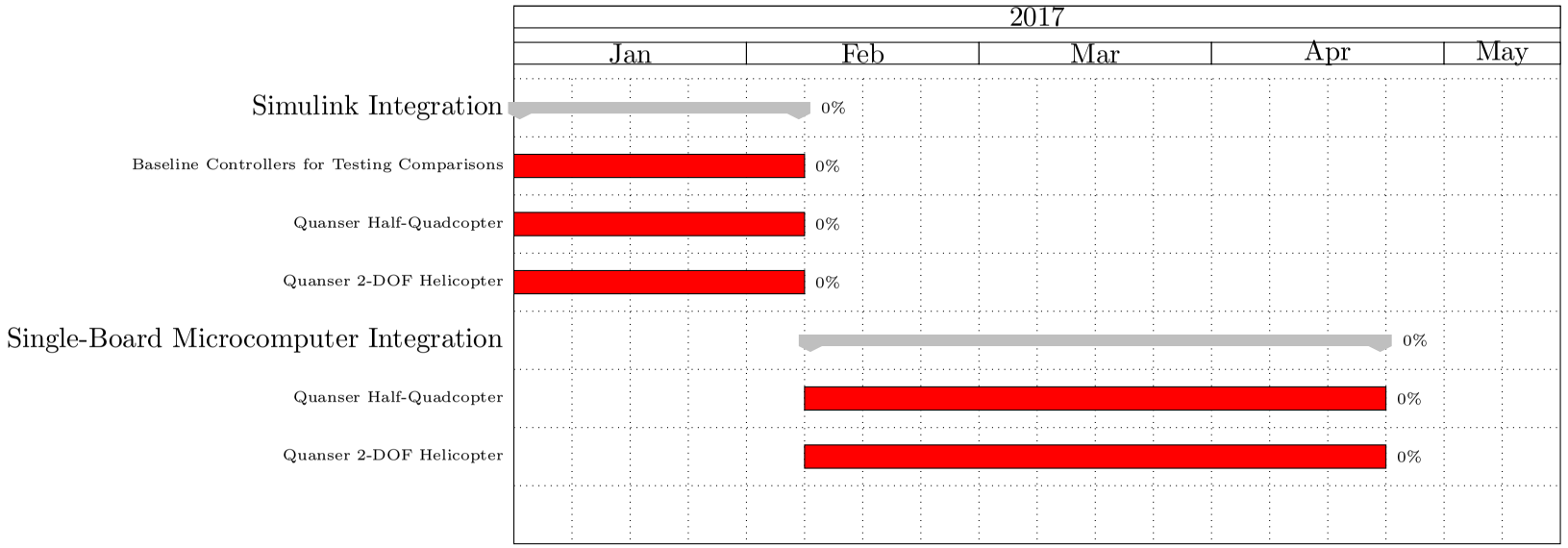

Figure 16: Gantt Chart for Spring 2018

[7] R. Subramanian and V. Elumalai, "Robust mrac augmented baseline lqr for tracking control of 2-dof helicopter," in *Robotics and Autonomous Systems*, August 2016, pp. 70–77.

[8] T. Jinec, "Stabilization and control of unmanned quadcopter," in *Lule University of Technology - Department of Computer Science, Electrical and Space Engineering*, May 2011, pp. 913–919.

[9] T. Luukkonen, "Modelling and control of quadcopter," in *2014 American Control Conference*, vol. Aalto School of Science, June 2011.

[10] E. Gopalakrishnan, "Quadcopter flight mechanics model and control algorithms," in *Czech Technical University - Department of Control Engineering*, May 2016.

[11] P. Bhatkhande and T. C. Havens, "Real time fuzzy controller for quadrotor stability control," in *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, July 2014, pp. 913–919.

[12] S. Miah, "Value iteration based approximate dynamic programming for mobile robot trajectory with persistent inputs."