ECE498: Senior Capstone Project I

Project Proposal

Smart Controlled 2-DOF Helicopters

Glenn Janiak and Kenneth Vonckx
(gjaniak, kvonckx)@mail.bradley.edu
Advisor: Dr. Suruz Miah
smiah@bradley.edu

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

***Abstract*— This paper proposes a strategy for testing and comparing three control algorithms, LQR, LQG, and ADP, to control two two-DOF helicopters from a mobile device. We will be using Raspberry Pi 3's as terminals for the wireless communication and MATLAB as our primary coding language.**

I. Introduction

Helicopters are of a paramount importance as they are used in many civilian and military applications due to their ability for vertical take-off and landing. To enable their use in such applications, intensive research has been conducted in the literature to date since helicopters involve complex nonlinear dynamics. Most of the work on helicopter-based research requires dedicated computers for controlling their motion to specific configurations and resistant to turbulent conditions. Such methods are expensive and time-consuming to develop. Implementation of motion control techniques using cost-effective hardware is still a challenge.

In this project, we are proposing an algorithm for smart control of a team of two degree-of-freedom (two-DOF) helicopters using conventional motion control in cooperation with machine learning techniques where a user will be able to configure helicopters from any initial position. Even though conventional techniques have been tested with simple platforms in the literature, the current project employs conventional motion control strategies in cooperation with machine learning technique (reinforcement learning, for instance) for a team of helicopters as well as introducing user control via mobile devices. This project is expected to encourage research in this area as well as serve as an educational tool in teaching environments.

II. Background Study

Our project requires a great deal of research as some of our tasks have not been attempted before. As a result, we have examined research papers, work complete by other projects at Bradley University, and documentation/teaching materials from Quanser Inc.

A. Review of Literature

In order to get a better understanding of importance, kinematics, and control techniques of helicopters, several articles, journal, and conference papers we examined.

The motor directly controls the angle of their respective axis using the force generated by its propeller, creating a torque on the opposite axis as an effect of air resistance. [1]

$$\tau_p = l_p K_{pp} \Omega_p^2 + l_p K_{py} \Omega_y^2 \tag{1}$$

$$\tau_y = l_y K_{yy} \Omega_y^2 + l_y K_{yp} \Omega_p^2 \tag{2}$$

where K_{pp} is the pitch motor thrust constant, K_{py} thrust constant acting on the pitch angle from the yaw motor, K_{yp} thrust constant acting on yaw angle from pitch motor, K_{yy} is the yaw motor thrust constant, Ω_p^2 and Ω_y^2 are the pitch and motor speeds squared, and l_p and l_y are the distance of each motor to the center of rotation of the helicopter.

PID controllers are a very basic technique to improve the performance of feedback systems.[2] Extended Kalman Filters (EKF) are used to help reduce noise in a system.[3]. Neural Network (NN) algorithms such as radial basis function (RBF) and multilayer neural networks (MNNs) are widely used due their ability to model nonlinear systems.[3] Fuzzy logic controllers are also common in nonlinear systems.[2] ADP (Approximate Dynamic Programming) is a data-driven method rather than based on a model.[2]

B. Review of Last Year's Project

Tony Birge and Andrew Fandel completed a similar project in the fall of 2017 and spring of 2018 as part of Bradley University's Electrical and Computer Engineering Department[4]. Their project focused on the the development of a method using APD (Approximate Dynamic Programming) for control of the Quanser Aero two-DOF helicopter. The involved the creation of a neural network to compute the gain value for the system. They implemented this on a Raspberry Pi micro-controller by generating a C program to run their algorithm. To communicate with the Quanser Aero, they created another program which utilized SPI protocol. Eventually they created a mobile application for an Android smart phone using the local network.

Their work will serve as a foundation for our project moving forward. Their documentation on SPI has helped us to implement our own control algorithms. We will also replicate their ADP algorithm to provide insight as to which control method best suits this application.

C. Quanser Aero

This project will implement control techniques on Quanser's two-DOF Aero helicopter platform. This consist of two 18 volt motors powered by a built-in PWM amplifier attached to a propeller. [5] Encoders are also attached to the motors as well as the yolk which count the number of revolutions to provide RPM and position information.

Quanser Inc. provided background teaching materials which we used as a starting point for implementing some of our control algorithms[6]. They

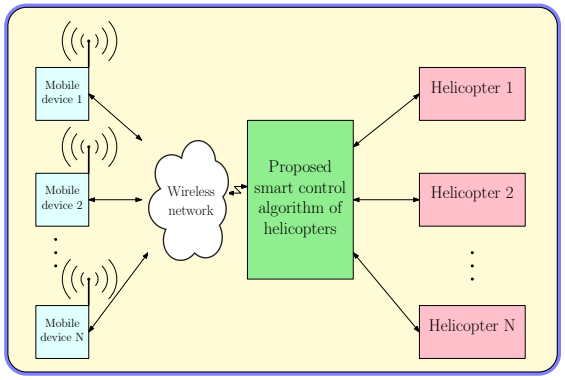


Fig. 1: General High-Level System Architecture

also provided the parameters for the Quanser Aero as shown in Table I. Using the parameters, the state-space model for the system can be created as shown in Equation 3, Equation 4, Equation 5, Equation 6

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -K_{sp}/J_p & -D_p/J_p & 0 \\ 0 & 0 & 1 & -D_y/J_y \end{bmatrix} \quad (3)$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ K_{pp}/J_p & K_{py}/J_p \\ K_{yp}/J_y & K_{yy}/J_y \end{bmatrix} \quad (4)$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (5)$$

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (6)$$

III. Functional Requirements

Over the next few sections we will describe the functional requirements of our project. This will cover our basic system architecture to the details of our communication protocols.

A. System Architecture

For our project, we consider the high level system shown in Figure 1 where mobile devices are used to to control a team of two degree of freedom helicopters. In our case we will be using one mobile device and two helicopters. Information sent by the users will be transmitted through a wireless TCP/IP network. We will be using Raspberry Pi 3's to accomplish the wireless communication between the mobile devices and the helicopters. Their inputs will interface with our proposed smart control algorithm and change the configuration of the helicopters.

Each of the helicopters used will have fixed bases as shown in Figure 2 (courtesy of Quanser

Fig. 2: Quanser Aero 2-DOF Helicopter.

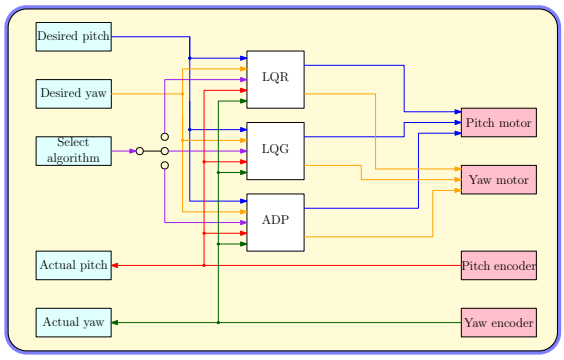


Fig. 3: Low Level Smart Control Diagram

Inc.[1]). The tail of the helicopter has a motor that controls its yaw motion. Similarly, the main rotor changes the pitch of the helicopter. The directions in which the motors spin will be determined by the polarity of the applied voltages. This will be regulated by our smart control algorithm.

B. Proposed Smart Control Algorithm

The system we propose for our project in Figure 1 has a mystery box with our smart control algorithm. Figure 3 shows our subsystem level smart control algorithm. This would be loaded to our Raspberry Pi 3 which is connected to the Quanser Aero, and wirelessly communicates with the mobile device. From the mobile device you choose a desired pitch and yaw angle for your helicopters to move from their initial positions and you select a control algorithm, which is talked about later. The algorithm selector will enable one of the three algorithms while disabling the other two. Then the enabled algorithm takes in the desired pitch and yaw angles and the actual pitch and yaw angles, which are recorded from the helicopter, as inputs. The actual pitch and yaw angles will also be displayed on the mobile device. With the inputs going into the desired algorithm, voltages are calculated for each

[1] https://www.quanser.com/products/quanser-aero/

Parameter	Description	Value	Unit
L_{body}	length of horizontal body	0.1651	[m]
m_{body}	mass of horizontal body	0.094	[kg]
J_{body}	Moment of Inertia of helicopter body	0.0026	$[\text{kg} \cdot \text{m}^2]$
m_{prop}	mass of dc motor + shield + propeller shield	0.43	[kg]
r_{prop}	distance from center of mass to center of pitch axis	0.1588	[m]
J_{prop}	moment of inertia from motor + guard assembly about pivot $(m_p * r_p^2)$	0.0108	$[\text{kg} \cdot \text{m}^2]$
J_p	equivalent moment of inertia about pitch axis (J_{body} + $2 * J_{\text{prop}}$)	0.0215	$[\text{kg} \cdot \text{m}^2]$
m_{yoke}	mass of entire yoke assembly	0.526	[kg]
r_{fork}	radius of each fork	0.02	[m]
J_{yoke}	moment of inertia of yoke fork that rotates about yaw axis $(0.5 * m_y * r_f^2)$	0.00010520	$[\text{kg} \cdot \text{m}^2]$
J_y	equivalent moment of inertia about yaw axis (J_{body} + $2 * J_{\text{prop}}$ + J_{yoke})	0.0237	$[\text{kg} \cdot \text{m}^2]$
K_{sp}	stiffness (found experimentally)	0.037463	$[\text{N} \cdot \text{m} \cdot \text{rad}^{-1}]$
K_{pp}	(found experimentally)	0.0011	$[\text{N} \cdot \text{m} \cdot \text{V}^{-1}]$
K_{yy}	(found experimentally)	0.0022	$[\text{N} \cdot \text{m} \cdot \text{V}^{-1}]$
K_{py}	thrust acting on pitch from yaw (found experimentally)	0.0021	$[\text{N} \cdot \text{m} \cdot \text{V}^{-1}]$
K_{yp}	thrust acting on yaw from pitch (found experimentally)	-0.0027	$[\text{N} \cdot \text{m} \cdot \text{V}^{-1}]$
D_p	viscous damping pitch axis (found experimentally)	0.0071116	$[\text{N} \cdot \text{m} \cdot \text{s} \cdot \text{rad}^{-1}]$
D_y	viscous damping yaw axis (found experimentally)	0.0220	$[\text{N} \cdot \text{m} \cdot \text{s} \cdot \text{rad}^{-1}]$

TABLE I: Quanser Aero Parameters

of the two motors on the helicopter to move to the desired angles.

C. Control Algorithms

We will be using three algorithms: LQR, LQG, and ADP.

1) LQR: LQR (Linear Quadratic Regulator) minimizes a cost function given by the parameters Table I for the system.

The Quadratic term, Equation 7, and Regulator term, Equation 8, are used to solve the cost function. They are solved using algebraic Riccati equation. Quanser proved the values shown in the matrices.

$$Q = \begin{bmatrix} 3500 & 0 & 0 & 0 \\ 0 & 500 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{7}$$

$$R = 0.005 * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{8}$$

The steps of the LQR technique are given in algorithm 1.

2) LQG: Once LQR is completed, we will investigate LQG (Linear Quadratic Gaussian). This method uses a Kalman filter to eliminate noise in the system.

3) ADP: Once LQG is completed, we will investigate ADP (Approximate Dynamic Programming). This method uses machine learning to optimize the system.

Algorithm 1: LQR Algorithm

begin
- Initialize state variable θ, ψ, $\dot{\theta}$, and $\dot{\psi}$
- Initialize parameters in state-space model matrices Equation 3, Equation 4, Equation 5, Equation 6
- Define cost function $\int_0^\infty (x^T Q x + u^T R u) dt$
- **repeat**
  - Compute u
  - $u = ke$
  - $e = x^d - x$
  - $\dot{e} = -\dot{x}$
  - $\dot{e} = -Ax - Bu$
  - $\dot{e} = -A(x^d - e) - Bu$
  - $\dot{e} = -Ae - Bu - Ax^d$
  - $\dot{e} = -Ae - Bke - Ax^d$
  - Apply u to helicopter actuators
  - Measure states
- **until** *done*

The steps of the ADP technique are given in algorithm 2.

D. Specifications

For the proposed system, there are a few specifications that need to be met. First, any extra materials we might need have to have an overall cost that is less than $500 in order to stay within the budget

Algorithm 2: ADP Algorithm

Input: $\mathbf{x}^{[d]}$
Output: $\mathbf{x}$
begin
  repeat
    • $t = k\tau$
    • Apply $u[k]$ to system model $\rightarrow x[k]$
    • Apply $\dot{x}(t) = Ax(t) + Bu(t)$ $\rightarrow x[k+1]$
    • Constrain ψ on $[-180°, 180°]$ and θ on $[-90°, 90°]$
    • Calculate error $\rightarrow\ e[k] = x^{\text{ref}}[k] - x[k]$
    • Calculate updated w_c if $t = T$
    repeat
      • $w_{\text{last}} = w_c$, $i = 0$
      repeat
        • $u[i] = [0, 0]^T$, $j = 0$
        repeat
          • $u_{\text{last}}[i] = u[i]$
          • Find $e[i+1]$ using collected error data and $e[k+1] = f(e[k]) + Gu[k]$ to $x[k]$
          • Compute new $u[i]$ using $u^* = -\frac{1}{2}R^{-1}G^T\nabla * (e(k+1))$
          • $j = j + 1$
        until $\|u[i] - u_{last}[i]\| < \epsilon$ *or* $j = j_{max}$
        Output: $u[i]$
        • Find $V(i)$ using $V(e(k)) = e^T(k)Qe(k) + u^T(k)Ru(k) + w_C^T\phi(e(k+1))$
        • $\Lambda(i) = \phi(e(i))$
        • $i = i + 1$
      until $i = \bar{n} - 1$
      Output: Λ,V
      • Update w_c using $w_c = (\Lambda^T\Lambda)^{-1}\Lambda^-1V$
    until $\|w_c - w_{last}\| < \epsilon_c$
    Output: w_c
    • Calculate state-feedback gain using $u^* = -\frac{1}{2}R^{-1}G^T\nabla * (e(k+1))$ and $V(e(k)) = e^T(k)Pe(k)$
    • Constrain optimal inputs on $[-18, 18]$
    • $k = k + 1$
  until $t = t_f$

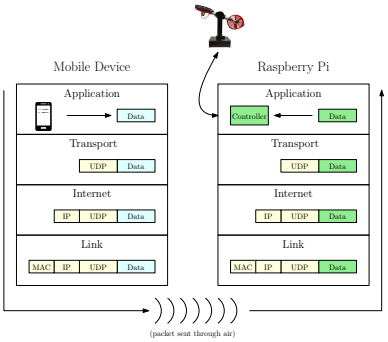


Fig. 4: Communication model.

of the department. Second, we need to have local WiFi capabilities for the connection of the mobile device to the Raspberry Pi 3. Third, we will be using MATLAB and Simulink to generate C code for the Raspberry Pi 3. Finally, we will be using an Android cell phone as our mobile device for user interfacing.

E. Communication Protocols

The smart motion control algorithm runs on a single board computer which has back and forth communication between the mobile devices and the helicopter. Fig. 4 shows the overall communication structure of the current work. The mobile device running the user application sends command signals, which are then formed into a data packet to be sent over the wireless network. In this work, the user datagram protocol (UDP), which is a TCP/IP protocol, is used for the communication between the mobile device and the helicopter. It has low-latency and more tolerant to lost packets. Next, routing information and logical addressing information are added to the packet including the internet protocol (IP) address. The link layer includes physical functions, such as the media access control address (MAC) of the network interface card (NIC) in the packet and transmits it along a medium. This framework utilized IEEE 802.11 which transmits the packet wirelessly through the air using radio waves. The single board computer then removes the extraneous information until all that is left is the original data.

1) SPI: SPI will be used to facilitate communication between our micro-controller (Raspberry Pi 3) and the Quanser Aero. Table II contains the pin functions of the Q-flex2 embedded panel used to create the connection. [5] SPI communication involves a master/slave relationship between devices. In our case the Raspberry Pi 3 is the master device while the Aero is the slave. What SPI allows is for the master device to send and receive data to the slave device bit by bit. The major issue using SPI

Wire	Color	Function
1	White	VCC (1.8V-5V)
2	Yellow	MOSI
3	Blue	MISO
4	Green	CLK
5	Gray	QCLK (Not Used)
6	Purple	CS (Digital Output Line)
7	Red	GND

TABLE II: Embedded Wiring

with the Quanser Aero is that SPI is meant for small quantities of data; however, the Aero sends about 51 bytes, which is not small. Last year's group actually came up with a solution to this issue by setting the SPI clock to have a two microsecond period.

2) Wireless: The Raspberry Pi 3 has a network card that uses IEEE 802.11ac to be able to send and receive transmission of a wireless network. We will use this to create a SSH connection between the Raspberry Pi and a desktop so we can interface with the terminal. This will also be used for the communication between the Raspberry Pi and the smartphone.

3) UDP: UDP (User Datagram Protocol) will be used to send and receive information between the Raspberry Pi and the Android smart phone. This is much faster than (TCP Transmission Control Protocol) because the sender and receiver do not make sure that the last transmission was received. This is important because we want commands to be sent as quickly as possible to the Quanser and are not concerned if one out of many packets fails to be transmitted.

IV. Preliminary Work

This project is a continuation of the work done last year by Tony Birge and Andrew Fandel. We started the lab work for this project after we had our first meeting with Dr. Miah. We choose this project because of our interest in the field of robotics. The following subsections contain the work we have already completed. It includes the implementation of the LQR algorithm on one of the Quanser Aero helicopters.

A. LQR Simulation Results

Before we implemented LQR using a USB connection, we ran simulations in MATLAB to make sure that our algorithm would run as expected. Figure 5 shows the position error of the pitch, yaw, speed of pitch, and speed of yaw. Figure 6 shows the position in degrees of the desired pitch, actual

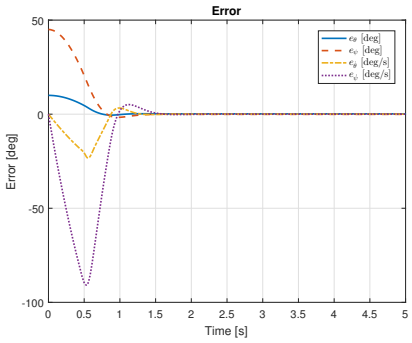


Fig. 5: LQR Simulation Error w/ Constant Signal

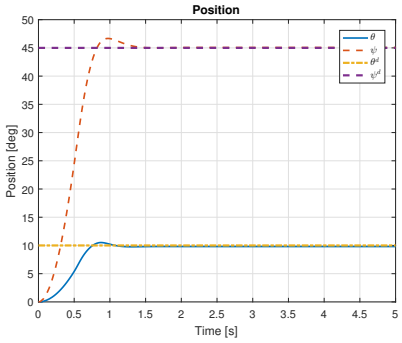


Fig. 6: LQR Simulation Position w/ Constant Signal

pitch, desired yaw, and actual yaw. Figure 7 shows the voltages applied to the motors.

B. LQR USB Implementation

Our first task that needed to be accomplished was testing and implementing the LQR algorithm onto the Aero using a USB connection. Figure 8 shows our Simulink model for controlling the Aero with the USB connection. Figure 9 through Figure 20 are our MATLAB implementation figures for testing different types of inputs to our model. The inputs we used were of three types: square wave, sine wave, and constant. We recorded the positions

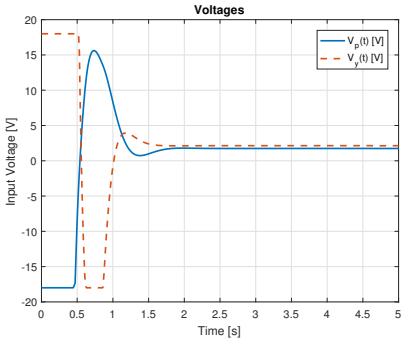


Fig. 7: LQR Simulation Voltage w/ Constant Signal

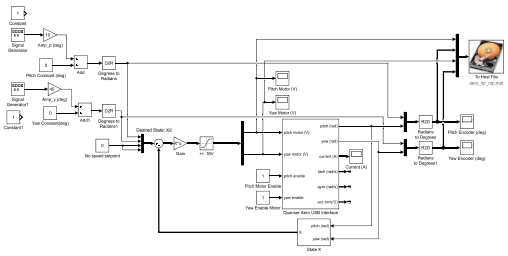


Fig. 8: LQR USB Simulink Model

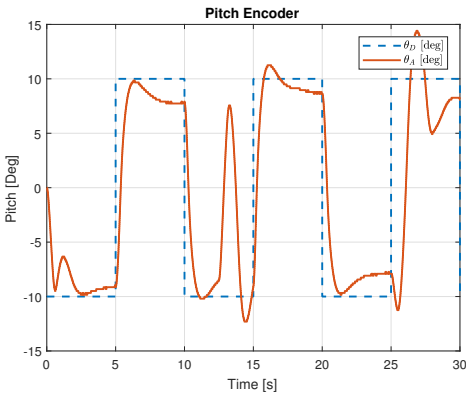


Fig. 9: LQR USB Pitch Encoder w/ Square Wave

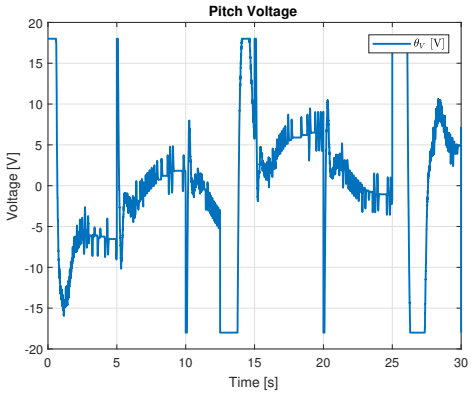


Fig. 10: LQR USB Pitch Motor w/ Square Wave

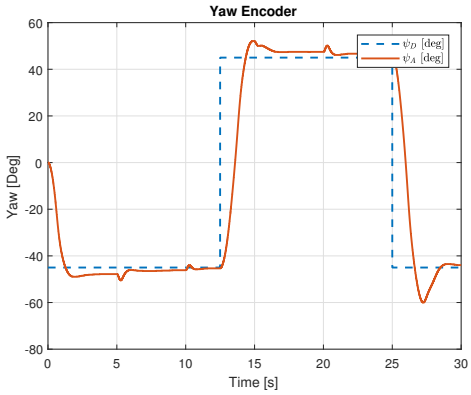


Fig. 11: LQR USB Yaw Encoder w/ Square Wave

of both pitch and yaw angles and the voltages that were applied to the motors.

C. LQR Raspberry Pi Implementation

For using the Raspberry Pi 3 we needed to use a local network for wireless communication with the PC. This took a little time, but we did achieve this. Next was the issue of using SPI communication between the Raspberry Pi and the Aero. Luckily for us, since this project is a continuation, last year's group already had a Simulink model that can implement SPI communication between the Raspberry Pi and the Aero. After recreating every block and matching every configuration, we were able to send Figure 21 to the Raspberry Pi 3. Figure 22 is the inside of the SPI Communication block in Figure 21. These Simulink models are just the basic implementation of the SPI communication. After testing it we learned a few things. One, the initial position of the Aero is set when the Aero is started up. Two, we need some type of user interface to be able to change the desired angles while the program is running. That second difficulty is what we hope to be solved by the android application.

D. LQR Android Implementation

Figure 23 is our Simulink model that will be loaded onto the Raspberry Pi 3 to control the helicopter using LQR algorithm. For the android connectivity blocks we followed the model of last year's group. Again, checking and rechecking all the configurations to make sure they are the same, except targeting one of our phones instead. Figure 24 is the Simulink model for the android application. It will have two slider bars to set desired pitch and yaw, and will display what the actual pitch and yaw of the helicopter currently is. Fig. 25(b) and Fig. 26(b) show the voltages that are applied to the pitch and yaw motors to move the helicopter to the configurations depicted in Fig. 25(a) and Fig. 26(a).

V. Future Work

Moving forward with this project, we plan on continuing with our other two algorithms. We are in the process of simulating LQG using MATLAB. Once this is completed, we will implement this on the Quanser Aero using USB. However, with the teaching materials that were provided to us by

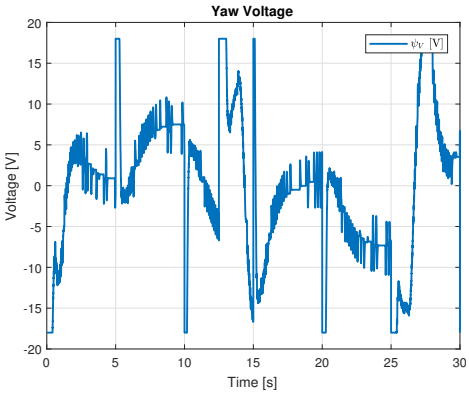


Fig. 12: LQR USB Yaw Motor w/ Square Wave

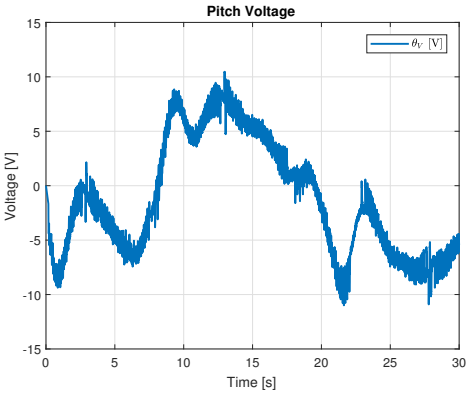


Fig. 14: LQR USB Pitch Motor w/ Sine Wave

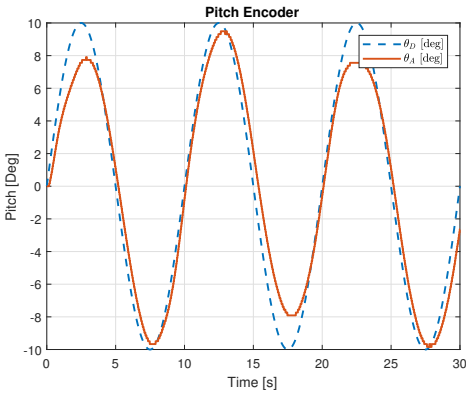


Fig. 13: LQR USB Pitch Encoder w/ Sine Wave

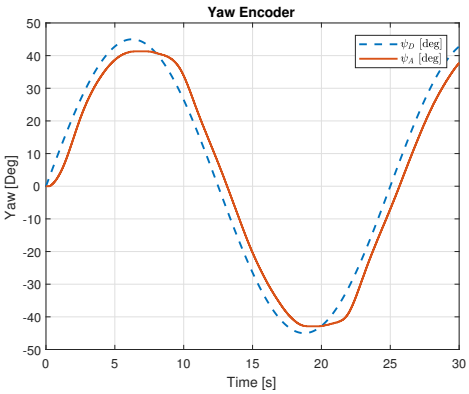


Fig. 15: LQR USB Yaw Encoder w/ Sine Wave

Quanser, we were able to implement and test an LQG algorithm that controls only pitch or yaw, but not both. So, we are currently in the process of researching a way to couple the control for both pitch and yaw angles. After we figure out this coupling issue and test it using the USB connection and wirelessly, we will implement LQG on an android phone.

After we complete implementing LQG all the way to the smart phone level, we will start simulating the ADP algorithm. Like the other two algorithms we will start testing ADP using USB, then wireless communication using the Raspberry Pi. After we get the algorithm working on the smart phone level, we will combine all three algorithms so the user may choose which algorithm they wish to use on the Aero.

VI. Parts List

For our project, we need a two-DOF helicopter platform to incorporate our algorithms. Bradley University acquired a Quanser Aero in 2016. A second one was purchased in 2018. In order to interface this platform with a micro-controller, we need a module that will facilitate communication between these two devices. The first Aero came with a Qflex2 embedded panel as part of a promotion. We were to convince Quanser to send us as second Qflex2 in September of 2018.

We have chosen to use a Raspberry Pi as our micro-controller. The ECE department has several they are willing to loan out to students. We are currently borrowing two Raspberry Pis. In order to communicate with the Raspberry Pi using the desktop, we require a wireless network card for the PC. The ECE department has loaned us two USB dongles for each PC.

We are also considering some add-ons to our project. When testing, we would like to use a fan to provide some turbulence to see how each algorithm reacts. We would also like to purchase a LCD screen that would attach to the Raspberry Pi and display the current pitch, yaw, and current algorithm. If we progress far enough, we may be able to purchase six-DOF helicopter to implement a full range of motion.

VII. Timeline

The workload for this project is split by semester. In the Fall of 2018, Figure 27, we hope to

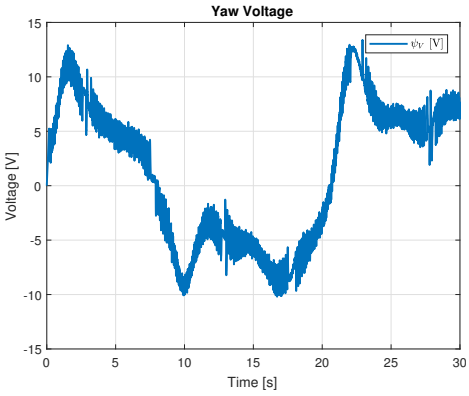


Fig. 16: LQR USB Yaw Motor w/ Sine Wave

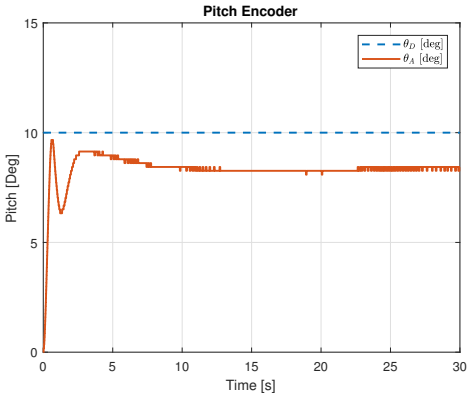


Fig. 17: LQR USB Pitch Encoder w/ Constant Input

complete LQR and LQG up to implementation on an Android phone. In the Spring of 2019, Figure 28, we plan to complete the ADP implementation as well as testing for all three methods.

REFERENCES

[1] G. Neto, F. Barbosa, and B. Angélico, "2-dof helicopter controlling by pole-placement."

[2] W. Gao and Z.-P. Jiang, "Data-driven adaptive optimal output-feedback control of a 2-dof helicopter," in *American Control Conference*, Boston Marriott Copley Place, Boston, MA, USA, July 2016.

[3] M. Hernandez-Gonzalez, A. Alanis, and E. Hernandez-Vargas, "Decentralized discrete-time neural control for a quanser 2-dof helicopter," *Applied Soft Computing*, 2012.

[4] A. Birge and A. Fandel, "Experiments on 2-dof helicopter using approximate dynamic programming," 2018.

[5] *Quanser Aero User Manual*, Quanser Inc., Markham,Ontario, 2016.

[6] *Aero 2 DOF Laboratory Guide*, Quanser Inc., Markham,Ontario, 2016, pgs. 26-39.

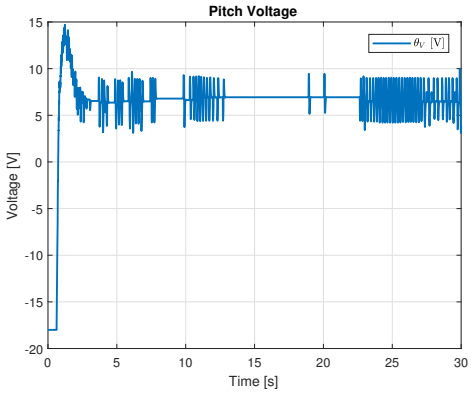


Fig. 18: LQR USB Pitch Motor w/ Constant Input

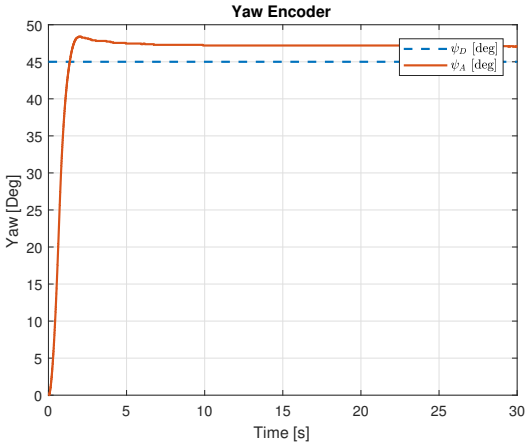


Fig. 19: LQR USB Yaw Encoder w/ Constant Input

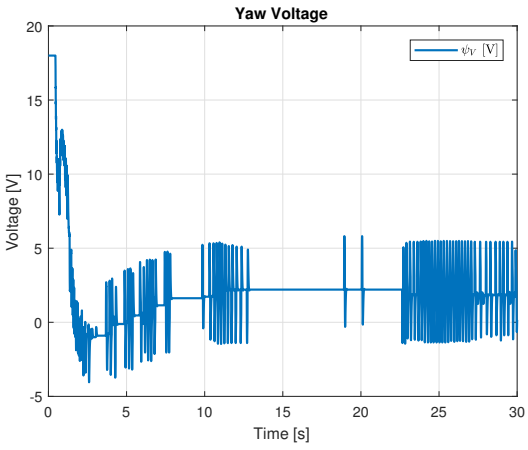


Fig. 20: LQR USB Yaw Motor w/ Constant Input

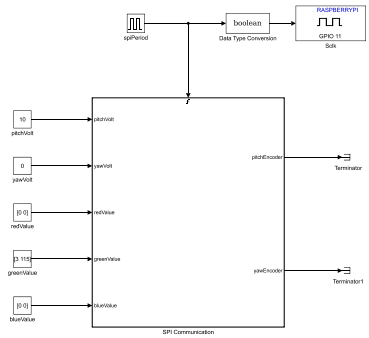


Fig. 21: Basic Top Simulink Model

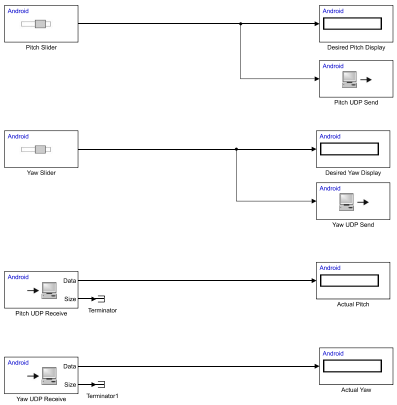


Fig. 24: Simulink Model of Phone App

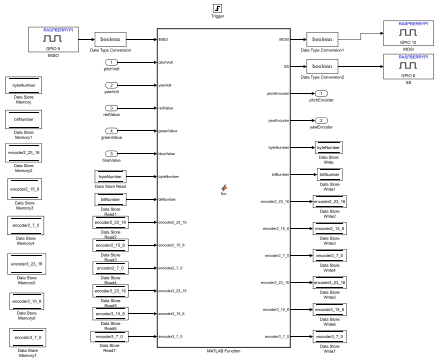
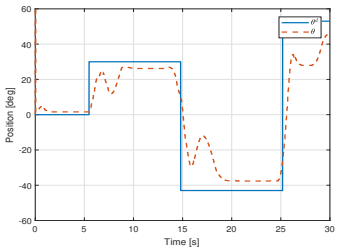
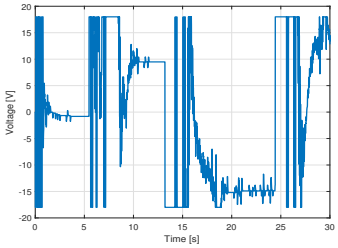


Fig. 22: SPI Communication Simulink Model



(a)



(b)

Fig. 25: Performance in following user's command (a) tracking pitch angle, and (b) pitch motor input voltage.

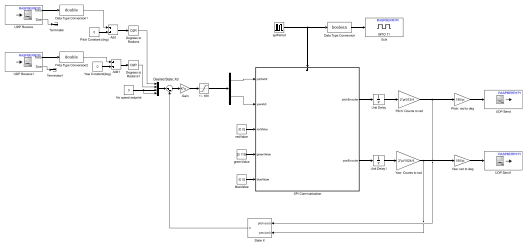
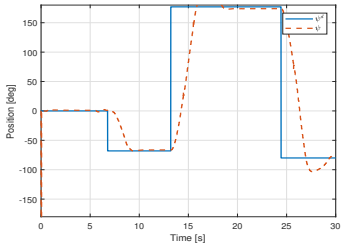
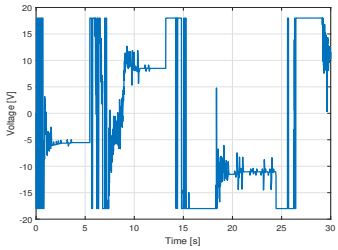


Fig. 23: Simulink Model of LQR with Android Compatibility



(a)



(b)

Fig. 26: Performance in following user's command (a) tracking yaw angle, and (b) yaw motor input voltage.

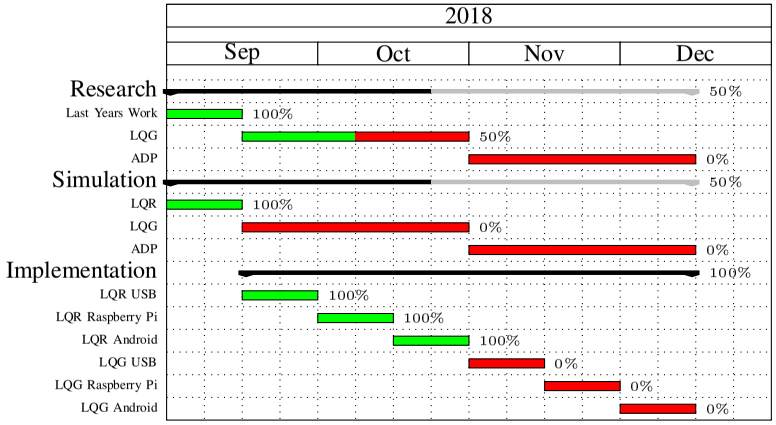


Fig. 27: Gantt chart for Fall 2018

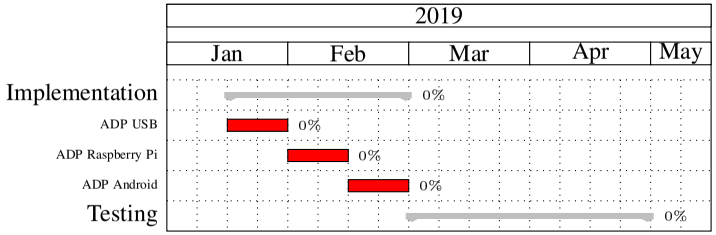


Fig. 28: Gantt Chart for Spring 2019