Analysis of ”Gain Scheduled Attitude Control of Fixed-Wing UAV with Automatic Controller Tuning.”

Adam Lobo and Brendan Neal

***Abstract*—Control of fixed wing unmanned aerial vehicles (UAVs) is a widely researched and growing topic in the field of robotics today. Applications for fixed wing UAVs include: military, agriculture, search and rescue, environmental and infrastructure surveying, and entertainment. The goal of this paper is to summarize, derive, recreate, and explain the results found in [10] through the use of online resources to further research the subject and MATLAB/Simulink to perform the experiments outlined in the paper.**

I. Introduction

A. Motivation

Fixed-wing UAVs have a lot applications in the world today, from conducting surveillance of land to delivering goods across a long distance. They are used extensively by the United States Government and Military to complete tasks and operations that would otherwise be very difficult to impossible by manned aircraft. Many of these operations are crucial to the safety, security, and continuous day-to-day life of the United States. As the name implies, UAVs are unmanned, meaning there is no onboard pilot flying them. They are flown either directly by an operator on the ground, or, in most cases, automatically by software. Due to the important role these devices play in our society, maintaining full and continuous control of UAVs is vital in not only the success of their operation, but also the safety of civilization. Since these vehicles are flying high through airspace at a high rate of speed, they pose a significant danger to other aircraft and life on the ground below if control is lost even briefly. Therefore, proper design of robust control algorithms is highly important in the continued success of the UAV [7].

B. Background

PID controllers are simple yet robust solutions to control systems and are easy to integrate into linear applications. PID controllers have three components: proportional, integral, and derivative. The proportional component addresses the present behavior of the system by taking the current error into account and scaling it by a constant to reduce that error. The integral component examines past behavior by adding all previous errors and scaling it by a constant. The derivative component predicts future behavior by utilizing the slope or rate of change of error and scaling it by a constant. Each component has a different linear constant that is tuned to help minimize the overall error. Most linear control systems use one or a combination of these components (i.e., P, PI, PD) depending on the system's complexity and sources of error. Standard PID controllers work well in linear control systems, but do not perform in nonlinear systems due to different varying system conditions and parameters [12].

The flight control of a UAV is a nonlinear system. Different flight and ambient conditions, such as airspeed, wind direction, attitude, and altitude/air pressure affect the flight dynamics of the UAV. A standard PID controller may perform well in one particular condition and poorly in a different condition, so utilizing this configuration is not ideal. One approach to addressing the non-linearity of a system is gain scheduling. Instead of using fixed P, I, and D constants, gain scheduling creates an array of different constant values that correspond to different operating conditions. Threshold limit values are set for each operating condition. When the dynamics of a system cross the threshold into a different operating condition, P, I and D constants are adjusted to the values tuned for the new operation condition.

One can either tune these controllers based on calculations derived from the dynamic equations of the UAV or through a new automatic controller tuning technique proposed in [10]. This involves performing oscillatory experiments under relay feedback under different operating conditions in order to examine the frequency response and calculate controller gains.

II. Derivations

A. Model of a Fixed-Wing UAV

The first step replicating the results in [10] is to model the UAV's dynamics as a function of its control inputs. Pictured below in Figure 1 is the standard notation for aerodynamic forces/moments and linear/angular velocities in the body frame. Additionally, the origin of this coordinate frame is at the center of mass of the UAV.

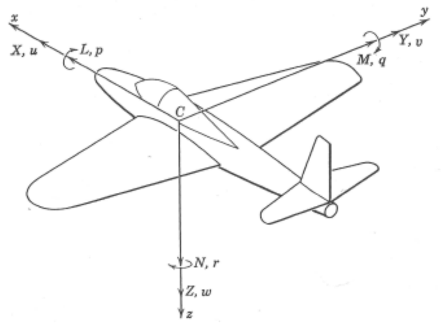

Fig. 1: Aircraft Model [3]

Fixed-wing UAVs (as well as other fixed wing aircraft) have three main control surfaces: the elevator, rudder, and ailerons. Each of these control surfaces are capable of producing moments around a particular body-frame axis, thus changing the attitude of the UAV. The attitude of a UAV is described as the combination of pitch, roll, and yaw angles. Pitch is the angle measured with respect to the body frame y-axis, the roll is the angle measured with respect to the body frame x-axis, and the yaw is measured with respect to the body frame z-axis. The elevator controls pitch, the ailerons control roll, and the rudder controls yaw. Additionally in Figure 1 are the terms u, v, and w which describe the components of instantaneous linear velocity in their respective directions [3]. The terms L, M, and N describe components of aerodynamic moments in X, Y, and Z respectively [3]. The terms p, q, and r represent the components of instantaneous angular velocity in X, Y, and Z respectively [3]. Though not depicted in Figure 1, the terms ϕ, θ, and ψ represent the roll, pitch, and yaw angles respectively. This then defines $p = \dot{\phi}(t)$, $q = \dot{\theta}(t)$, and $r = \dot{\psi}(t)$.

Now that the basics terms have been defined for fixed-wing UAVs, the control architecture in [10] is defined for attitude control of a fixed wing UAV. In this section, we will define the dynamic equations for the attitude of a fixed wing UAV. Since the control surfaces of the UAV produce moments around their respective axes, we can model the attitude of the UAV as a set of coupled nonlinear differential equations [1]:

$$\begin{aligned} L &= \dot{p}I_x - \dot{r}I_{xz} + qr(I_z - I_y) - pqI_{xz} \\ M &= \dot{q}I_y + pr(I_x - I_z) + (p^2 - r^2)I_{xz} \\ N &= \dot{r}I_z - \dot{p}I_{xz} + pq(I_y - I_x) + qrI_{xz} \end{aligned} \tag{1}$$

where I_x, I_y, and I_z are the moments of inertia about x, y, and z respectively. I_{xz} is the product of I_x and I_z.

Now, simplifying 1 and solving for $\dot{p}$, $\dot{q}$, and $\dot{r}$ yields:

$$\begin{aligned} \dot{p} &= \frac{1}{I_xI_z - (I_{xz})^2}(I_z(L + (I_y - I_z)qr) \\ &\quad + I_{xz} * (N + (I_x - I_y + I_z)pq - I_{xz}qr) \\ \dot{q} &= \frac{1}{I_y} * (M + pr(I_z - I_x) + (r^2 - p^2)I_{xz}) \\ \dot{r} &= \frac{1}{I_xI_z - (I_{xz})^2}(I_x(N + (I_x - I_y)pq) \\ &\quad + I_{xz} * (L + (I_x - I_y + I_z)qr - I_{xz}pq) \end{aligned} \tag{2}$$

[10] suggests the following constraints to simplify the equations moving forward:

$$\begin{aligned} \Gamma &= I_xI_z - (I_{xz})^2 \\ \Gamma_1 &= \frac{I_{xz}(I_x - I_y + I_z}{\Gamma} \\ \Gamma_2 &= \frac{I_z(I_z - I_y) + (I_{xz})^2}{\Gamma} \\ \Gamma_3 &= \frac{I_z}{\Gamma} \\ \Gamma_4 &= \frac{I_{xz}}{\Gamma} \\ \Gamma_5 &= \frac{I_z - I_x}{I_y} \\ \Gamma_6 &= \frac{I_{xz}}{I_y} \\ \Gamma_7 &= \frac{(I_x - I_y)I_x + (I_{xz})^2}{\Gamma} \\ \Gamma_8 &= \frac{I_x}{\Gamma} \end{aligned} \tag{3}$$

Simplifying 2 and substituting in the constants defined in 3 we yield:

$$\begin{aligned} \dot{p} &= \Gamma_1 pq - \Gamma_2 qr + \Gamma_3 L + \Gamma_4 N \\ \dot{q} &= \Gamma_5 pr - \Gamma_6(p^2 - r^2) + \frac{1}{I_y}M \\ \dot{r} &= \Gamma_7 pq - \Gamma_1 qr + \Gamma_4 L + \Gamma_8 N \end{aligned} \tag{4}$$

For control system design, we need to introduce the control surfaces into the dynamic equations of the system. We will use δ_a, δ_e, and δ_r as the aileron, elevator, and rudder control surface variables. From [10], the aerodynamic moments L, M, and N are functions of p, q, r, δ_a, δ_e, and δ_r as follows:

$$\begin{aligned} L &= l_p p + l_r r + l_{\delta_a}\delta_a + l_{\delta_r}\delta_r \\ M &= m_q q + m_{\delta_e}\delta_e + m_{\delta_t}\delta_t \\ N &= n_p p + n_r r + n_{\delta_a}\delta_a + n_{\delta_r}\delta_r \\ \delta_t &= \delta_a - \delta_r \end{aligned} \tag{5}$$

where l, m, and r are the force coefficients for the roll, pitch and yaw axes given a system condition.

In order to combine 4 and 5, we first need to define these force coefficients. From [3] and echoed in [8], we can define these force coefficients in non-dimensional aerodynamic forms. First, we define dynamic pressure, Q:

$$Q = \frac{1}{2}\rho_{SL}V_a^2 \tag{6}$$

where V_a is the equivalent airspeed of the UAV in standard sea-level air density (ρ_{SL}) [8]. From [3], using S as the wing platform area, b as the wingspan, β_c as the course angle, and c as the mean chord, we can write these coefficients as:

$$\begin{aligned} l_p &= C_{l_0}QSb \\ m_q &= C_{m_0}QSc \\ n_r &= C_{n_0}Qsb \end{aligned} \tag{7}$$

Next, we define the control derivative constants:

$$\begin{aligned} l_{\delta_a} &= \frac{Qsb}{I_x} C_{l_{\delta_a}} \\ l_{\delta_r} &= \frac{Qsb}{I_z} C_{l_{\delta_r}} \\ m_{\delta_e} &= \frac{Qsc}{I_y} C_{m_{\delta_e}} \\ n_{\delta_a} &= \frac{Qsb}{I_x} C_{n_{\delta_a}} \\ n_{\delta_r} &= \frac{Qsb}{I_z} C_{n_{\delta_r}} \end{aligned} \tag{8}$$

Where:

- l_{δ_a} is the rolling moment due to aileron deflection.
- l_{δ_r} is the rolling moment due to rudder deflection.
- m_{δ_e} is the pitching moment due to elevator deflection.
- n_{δ_a} is the yawing moment due to aileron deflection (adverse yaw).
- n_{δ_r} is the yawing moment due to rudder deflection.

Now, we can fully define the attitude of the UAV in terms of its manipulated control variables by combining 4, 5, 6, 7, and 8 [10]:

$$\begin{aligned} \dot{p} &= \Gamma_1 pq - \Gamma_2 qr + \frac{1}{2}\rho V_a^2 Sb * \\ &(C_{l_0} + C_{l_{\beta_c}}\beta_c + C_{l_p}\frac{bp}{2V_a} + C_{l_r}\frac{br}{2V_a} \\ &+ C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r) \end{aligned} \tag{9}$$

$$\begin{aligned} \dot{q} &= \Gamma_5 pr - \Gamma_6(p^2 - r^2) + \rho\frac{V_a^2 Sc}{2I_y} * \\ &(C_{m_0} + C_{m_a} + C_{m_q}\frac{cq}{2V_a} + C_{m_{\delta_e}}\delta_e) \end{aligned} \tag{10}$$

$$\begin{aligned} \dot{r} &= \Gamma_7 pq - \Gamma_1 qr + \frac{1}{2}\rho V_a^2 Sb * \\ &(C_{n_0} + C_{n_{\beta_c}}\beta_c + C_{n_p}\frac{bp}{2V_a} + C_{n_r}\frac{br}{2V_a} \\ &+ C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r) \end{aligned} \tag{11}$$

To convert the body rates to Euler rates, we use an equation found in [6]:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & sin(\phi)tan(\theta) & cos(\phi)tan(\theta) \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi)sec(\theta) & cos(\phi)sec(\theta) \end{bmatrix} * \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{12}$$

[10] does not provide all the numeric values for many of the constants and coefficients presented above, and we will address that in a later section using information garnered from more supporting sources.

B. Control Architecture

1) General Control Structure: To control the angular velocities of the UAV, three PI controllers are used. The equations that determine the control inputs are given as follows:

$$\begin{aligned} \delta_a(t) &= K_c^p(p^* - p) + \frac{K_c^p}{\tau_c^p}\int_0^t (p^* - p)\, d\tau \\ \delta_e(t) &= K_c^q(q^* - q) + \frac{K_c^q}{\tau_c^q}\int_0^t (q^* - q)\, d\tau \\ \delta_r(t) &= K_c^r(r^* - r) + \frac{K_c^r}{\tau_c^r}\int_0^t (r^* - r)\, d\tau \end{aligned} \tag{13}$$

According to [9], for this scale of aircraft, the pitch angle θ is very small, and thus the first row of 12 can be assumed to be:

$$\dot{\phi} = p + qsin(\phi)tan(\theta) + rcos(\phi)tan(\theta) \approx p \tag{14}$$

Because of this fact, only two more PI controllers are required to control the pitch and roll Euler angles. The control signals are designed as follows:

$$\begin{aligned} p^*(t) &= K_c^\phi(\phi^* - \phi) + \frac{K_c\phi}{\tau_c^\phi}\int_0^t (\phi^* - \phi)\, d\tau \\ q^*(t) &= K_c^\theta(\theta^* - \theta) + \frac{K_c\theta}{\tau_c^\theta}\int_0^t (\theta^* - \theta)\, d\tau \end{aligned} \tag{15}$$

Here, p^* and q^* are the reference signals in the inner-loop control of the UAV.

2) Automatic Tuning of Cascade PI Controllers: As [10] mentions, one can tune the controllers via analyzing the dynamic equations presented above. However, part of the novelty of this paper stems from tuning controllers without analyzing the dynamic equations. The auto-tuning algorithm for these controllers that [10] chose is a method known as relay feedback. Relay feedback involved substituting a relay to replace the controller inside of a desired control loop. Relay feedback works particularly well with systems that include a time delay. UAV controllers include time delay due to the actuator dynamics: there is a time delay between when a control surface is actuated and the UAV attitude changes. The general working principle of relay feedback is as follows [4]:

1) The plant input u is increased by some h.
2) The output has time delay τ, and only after τ does the output begin to change.
3) At a preset hysteresis condition set for the output, the relay turns to the opposite position.
4) The output will lag the input by a certain phase shift and the system will oscillate continuously.

The amplitude and period of oscillation can be used to calculate the proper controller gains required to control the system.

In order to perform relay feedback controller tuning, [10] mounted their aircraft such that the UAV was fixed around the roll axis, as depicted in Figure 2

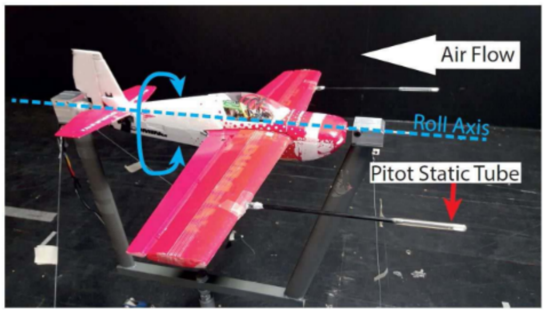

Fig. 2: Inner Loop Relay Feedback Rig [10]

Figure 3 depicts the block diagram setup for the inner loop relay feedback experiment setup:

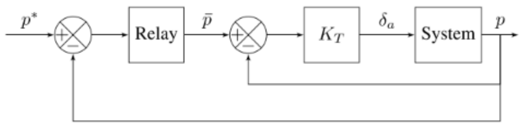

Fig. 3: Inner Loop Relay Feedback Block Diagram [10]

Figure 4 depicts the block diagram setup for the outer loop relay feedback experiment setup:

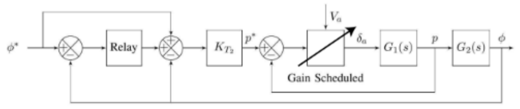

Fig. 4: Outer Loop Relay Feedback Block Diagram [10]

[10] uses the relay feedback method to tune both the inner loop controller and outer loop controller, so moving forward in this section, we will define the process by which someone finds controller gains through relay feedback.

Because of the cascaded control nature presented in [10], the inner loop PI controller is tuned first. Because of the experimental setup, the variation of q is around zero in the testing environment. Thus, the "system" of this experiment (derived from equation 9 because this is roll rate control) changes to be a single integrator with time delay. This will be explored more in the simulations section. Additionally, in 3, a proportional controller is implemented to maintain stability of the aircraft.

In order to find the controller gains, the data for the relay control $\bar{p}$ and output signal p are collected [10]. Once this data has been collected, we can use the techniques outlined in [9] to calculate the inner loop controller gains.

The key idea behind this approach is to find the frequency response of the unknown open loop system ($G(e^{j\omega})$) by estimating the frequency response to the closed loop control system ($T(e^{j\omega})$) by evaluating the following equation [9]:

$$G(e^{j\omega}) = \frac{1}{K_T} * \frac{T(e^{j\omega})}{1 - T(e^{j\omega})} \tag{16}$$

The first step to this approach is to take an average of the oscillating input data and discrete the fundamental frequency according to the number of samples (N) as follows:

$$\omega_d = \frac{2\pi}{N} \tag{17}$$

Because the proportional controller stabilizes the system, we can write the closed loop transfer function T(z) in a frequency sampling filter form [9]:

$$T(z) = \sum_{l=-\frac{N-1}{2}}^{\frac{N-1}{2}} (T(e^{jl\omega_d})F^l(z) \tag{18}$$

Where $F^l(z^{-1})$ is the lth frequency sampling filter [9]:

$$F^l(z^{-1}) = \frac{1}{N} * \frac{1 - z^{-N}}{1 - e^{jl\omega_d} z^{-1}} \tag{19}$$

[9] states that "the frequency sampling filter model describes the stable system using fundamental frequency parameters." Take $f^l(k)$ as the lth output of the frequency sampling filter. We can define:

$$f^l(k) = F^l(q^{-1})u(k) \tag{20}$$

where q^{-1} is the backward shifting operator defined in [9] as:

$$q^{-1}x(k) = x(k-1) \tag{21}$$

Next, the output of the relay feedback closed loop system in the time domain is defined as:

$$y(k) = \sum_{l=-\frac{N-1}{2}}^{\frac{N-1}{2}} (T(e^{jl\omega_d})F^l(k) + v(k) \tag{22}$$

$v(k)$ is defined as the output measurement noise. [9] states that when the input signal $u(k)$ is a perfectly periodic signal with period N, the Fourier analysis demonstrates that it exclusively includes odd frequencies, and the magnitude weakens as the count increases. By ignoring these high frequency terms, we can write the output signal Θ and its corresponding regression vector ϕ as [9]:

$$\begin{aligned} \Theta = [&T(0)T(e^{j\omega_d})T(e^{-j\omega_d})T(e^{2j\omega_d}) \\ &T(e^{-2j\omega_d})T(e^{3j\omega_d})T(e^{-3j\omega_d})]^* \\ \phi = [&f^0(k)f^1(k)f^{-1}(k)f^2(k)f^{-2}(k)f^3(k)f^{-3}(k)]^* \end{aligned} \tag{23}$$

The $*$ denotes that the Θ and ϕ matrices are the conjugate transposes of Θ and ϕ. Using this output, [9] proposes a recursive least squares function to estimate the frequency response parameters:

$$\begin{aligned} P(k-1) &= P(k-2) - \frac{P(k-2)^T\phi(k)\phi(k)^T P(k-2)}{1 + \phi(k)^T P(k-2)\phi(k)} \\ \hat{\Theta}(k) &= \hat{\Theta}(k-1) + P(k-1)\phi(k)(y(k) - \phi(k)^T\hat{\Theta}(k-1)) \end{aligned} \tag{24}$$

Using the frequency response parameters, we can now look to evaluate equation 16. The discrete time frequency response is close to the the continuous time frequency response due to the fast sampling environment. Thus, the fundamental frequency in the continuous time environment is: $\omega_1 = \frac{\omega_d}{\Delta t}$ [9]. Because of this, we can write:

$$G_p(j\omega_1) \approx G(e^{j\omega_d}) \tag{25}$$

Because both [10] and [9] model the UAV dynamics for the inner loop as a single integrator with time delay, we can write:

$$G_p(s) = \frac{K_p e^{-ds}}{s} \tag{26}$$

For a time delay integrator system, a single frequency is enough to calculate the gain K_p and time delay d [9]. Equating equations 25 and 26 yields:

$$\frac{K_p e^{-jd\omega_1}}{j\omega_1} = G_p(j\omega_1) \tag{27}$$

To calculate the gain K_p, we compare both magnitudes:

$$K_p = \omega_1 |G_p(\omega_1)| \tag{28}$$

To calculate the time delay we compare the phase angles:

$$d = -\frac{1}{\omega_1} arctan(\frac{IMAG(jG_p(j\omega_1))}{REAL(jG_p(j\omega_1))}) \tag{29}$$

Finally, to get the controller parameters, we have to normalize the parameters following these rules outlined in [11]:

$$\begin{gathered} K_c = \frac{\hat{K_c}}{dK_p} \\ \tau_I = d\hat{\tau}_I \\ \tau_d = d\hat{\tau}_d \end{gathered} \tag{30}$$

[10] calculates their controller parameters based on a damping coefficient of 1 and a performance factor β.

3) Gain Scheduled PI Controller: Gain scheduling is a form of adaptive control that involves adapting a controller's parameters based off of a certain operating condition. Gain scheduling works well to control nonlinear systems by interpolating between a set of linear controllers [2]. For fixed-wing UAVs (a nonlinear system), the operating condition is the airspeed. In the case of [10]'s work, the PI controller gains are set between three different airspeeds: low (7 m/s), medium (10 m/s), and high (15 m/s). [10] defines the controller parameters as the following:

- Low Airspeed (V_a^L): K_c^L and τ_I^L
- Medium Airspeed (V_a^M): K_c^M and τ_I^M
- High Airspeed (V_a^H): K_c^H and τ_I^H

[10] interpolates between these parameters based off the derivative of the control signal, $\dot{u}(t)$:

$$\begin{aligned} \dot{u}(t) = {} & \lambda^L(K_c^L \dot{e}(t) + \frac{K_c^L}{\tau_I^L} e(t)) \\ & +\lambda^M(K_c^M \dot{e}(t) + \frac{K_c^M}{\tau_I^M} e(t)) \\ & +\lambda^H(K_c^H \dot{e}(t) + \frac{K_c^H}{\tau_I^H} e(t)) \end{aligned} \tag{31}$$

where $e(t) = p^*(t) - p(t)$.

λ_L, λ_M, and λ_H are the weighting parameters and are set based on the airspeed. They vary between 0 and 1 and are calculated based on the following principles:

- If $V_a(t) \le V_a^L$: $\lambda^L = 1$, $\lambda^M = 0$, and $\lambda^H = 0$
- If $V_a^M - \delta \le V_a(t) \le V_a^M + \delta$: $\lambda^L = 0$, $\lambda^M = 1$, and $\lambda^H = 0$
- If $V_a(t) \ge V_a^H$: $\lambda^L = 0$, $\lambda^M = 0$, and $\lambda^H = 1$

where δ is a pre-set band around the medium airspeed that allows for smooth transition in the interpolation phase.

For airpseeds that fall between the conditions above, [10] offers these interpolation equations to "split" the control signal between two different bands:

If $V_a^L \le V_a(t) \le V_a^M - \delta$:

$$\begin{aligned} \lambda^H &= 0 \\ \lambda^M &= \frac{V_a(t) - V_a^L - \delta}{V_a^M - V_a^L - 2\delta} \\ \lambda^L &= 1 - \lambda^M \end{aligned} \tag{32}$$

If $V_a^M + \delta \le V_a(t) \le V_a^H$:

$$\begin{aligned} \lambda^H &= \frac{V_a(t) - V_a^M - \delta}{V_a^H - V_a^M - 2\delta} \\ \lambda^M &= 1 - \lambda^H \\ \lambda^L &= 0 \end{aligned} \tag{33}$$

This method ensures that all operating conditions will be covered by the gain scheduled controller. In order to properly calculate the control signal, we must discretize 31. Assume the sampling interval is: $\Delta t = t_i - t_{i-1}$. At t_i, we can approximate:

$$\dot{u}(t_i) \approx \frac{u(t_i) - u(t_{i-1})}{\Delta t}$$

$$\dot{e}(t_i) \approx \frac{e(t_i) - e(t_{i-1})}{\Delta t}$$

Using this approximation, we can write 31 in the discrete time form as follows:

$$\begin{aligned} & u(t_i) = u(t_{i-1}) + \Delta t* \\ & [\lambda^L(K_c^L(e(t_i) - e(t_{i-1}) + \frac{K_c^L}{\tau_I^L} e(t_i)\Delta t) \\ & +\lambda^M(K_c^M(e(t_i) - e(t_{i-1}) + \frac{K_c^M}{\tau_I^M} e(t_i)\Delta t) \\ & +\lambda^H(K_c^H(e(t_i) - e(t_{i-1}) + \frac{K_c^H}{\tau_I^H} e(t_i)\Delta t)] \end{aligned} \tag{34}$$

Finally, [10] uses an anti-windup mechanism to avoid over-stressing the system actuators:

- If $\bar{u}(t_i) < u_{min}$, then $u(t_i) = u_{min}$
- If $u_{min} \le \bar{u}(t_i) \le u_{max}$ then $u(t_i) = u(t_i)$
- If $\bar{u}(t_i) > u_{max}$, then $u(t_i) = u_{max}$

III. Simulations

The experiments outlined in [10] were performed on an actual UAV inside of a wind tunnel. We aimed to recreate these experiments in simulation via MATLAB and Simulink.

A. Simulation Setup and Parameters

Because the authors of [10] are using a physical aircraft, our first step is to create a virtual aircraft model in simulation. The only physical specifications that [10] gives are listed below in Table 1:

Table 1: UAV Specifications

a) :

Parameter	Value
Airframe	Slick 360
Airfoil	NACA0012
Wing Span	0.490m
Fuselage Length	0.421m
Avg. Chord	0.0885m
Weight	130g
Cruise Speed	10m/s

Because of this, important constants and coefficients used in the dynamic equations presented in 9, 10, and 11 are not given in the paper. Thus, we had to do a bit of extra research to discern these specifications. First, we found data for the moment of inertia values from a similar size and shape model aircraft through another academic paper: [5]. Pictured below in Table 2 are these moment of inertia specifications:

Table 2: Moments of Inertia

b) :

Parameter	Value
I_x	0.8869 $\frac{kg}{m^2}$
I_y	1.0353 $\frac{kg}{m^2}$
I_z	1.2138 $\frac{kg}{m^2}$

B. Inner Loop Relay Feedback Tests

The first step to creating a gain scheduled attitude controller was to tune the inner loop controller. In order to complete this, we first had to mimic the roll-rig setup depicted in Figure 2 2. Because the UAV is fixed on the roll axis, we can model the system by using equation 9. Because of the fixed roll rig, [10] claims that the variation of q in the testing environment is around 0. Additionally, the primary control surface that will be controlling the roll of the UAV will be the ailerons (i.e. zero rudder deflection in the tesing environment). Finally, since the UAV has a 0 course angle, we can ignore terms involving β_c. The model of the plant now becomes:

$$\dot{p} = \frac{1}{2}\rho V_a^2 Sb * (C_{l_0} + C_{l_p}\frac{bp}{2V_a} + C_{l_{\delta_a}}\delta_a) \qquad (35)$$

[10] does not explicitly state the numeric values of C_{l_0}, C_{l_p}, or $C_{l_{\delta_a}}$, therefore we had to use a systematic approach to discern these values and replicate the results. [3] provides example values for these constants for a Boeing 737. Though the numbers are not applicable for a UAV this size, they do provide guidance as to what the signs and magnitude relative to each other of these constants should be. After a systematic grid search process, we came to the conclusion that to best replicate the results across all tests that $C_{l_0} = 1.108$, $C_{l_p} = -11.5$, and $C_{l_{\delta_a}} = 0.65$. Obviously this ad-hoc method will lead to discrepancies in the results, but that will be discussed in further detail in the "Discussion" section.

[10] states that the plant can be modeled as a single integrator with time delay, which tracks with equation 35 above. We modeled the block diagram given in Figure 3 in Simulink. Below in Figure 5 is our model of the block diagram:

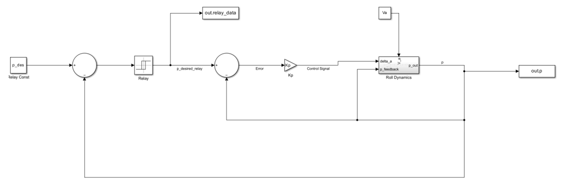

Fig. 5: Inner Loop Relay Feedback Simulink Diagram

For more information on how we modeled the plant, below in Figure 6 is the roll-rig plant subsystem where F(u) is the function stated in 35 and u are the properly indexed u inputs.

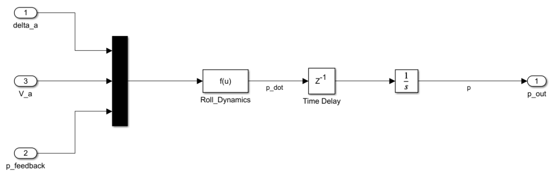

Fig. 6: Roll Rig Plant Subsystem

[10] implements a proportional controller, Kp, of 0.3 in order to ensure the safety of the aircraft. Additionally, the relay reference angular velocity for these tests is 100 deg/s. [10] implemented the switch condition (hysterisis) of 30 deg/s in order to prevent relay switching from noise. Finally, [10] conducted three separate tests (one for each airspeed operating condition). Using the Simulink model depicted in 5, we recreated these tests.

Below in Figure 7 are the results of the 7m/s test:

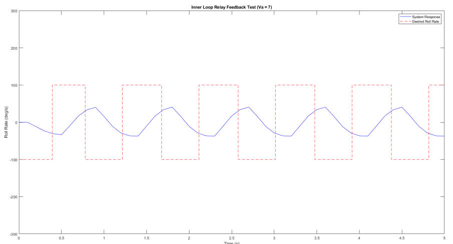

Fig. 7: Inner Loop Relay Feedback 7m/s Test

Below in Figure 8 are the results of the 10m/s test:

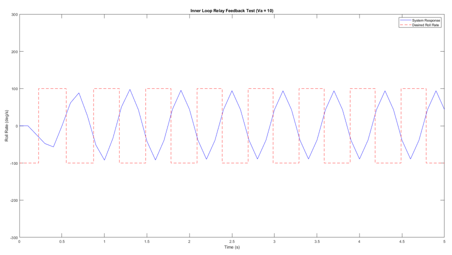

Fig. 8: Inner Loop Relay Feedback 10m/s Test

Below in Figure 9 are the results of the 15m/s test:

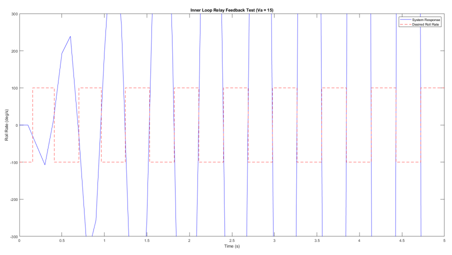

Fig. 9: Inner Loop Relay Feedback 15m/s Test

From these tests implemented in Simulink, we get results similar to [10]. Thus, following the controller tuning procedure outlined in the Derivations section, we can back out the Inner Loop Controller Gains:

a) :

Scheduled Controller Gains		
V_a	K_p	τ_I
7	0.92	0.095
10	0.23	0.146
15	0.19	0.178

C. Outer Loop Relay Feedback Tests

Now that the inner loop, gain scheduled controller has been properly tuned, we aim to tune the outer loop controller using the same roll-rig setup. This means that the system dynamics are still represented by 35. We modeled the block diagram given in Figure 4 in Simulink. Below in 10 is our model of the block diagram:

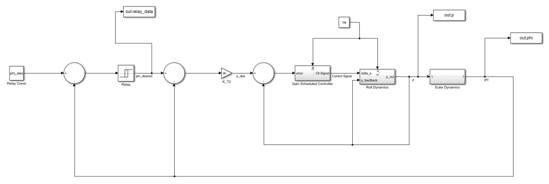

Fig. 10: Outer Loop Relay Feedback Block Diagram

In order to replicate the gain scheduling architecture, we had to get a bit creative. Since Simulink does not have a direct gain scheduling block, we had to create our own. We accomplished this by creating a subsystem that takes the scheduling variable (the airspeed) and uses two 1-Dimensional lookup tables to implement the controller gain conditions mentioned in the Derivations section. The lookup tables will sense the scheduling variable and then match the controller gains either exactly as set or through the interpolation function defined earlier. Below in Figure 11 is the gain scheduled controller subsystem:

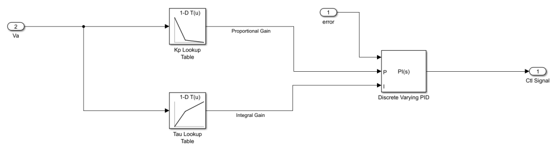

Fig. 11: Gain Scheduled Controller

Additionally, we had to model the dynamics of the outer loop plant given in equation 12. Since we are focused on roll, only the first row multiplication applies from this equation. Since q and r do not vary in this roll rig environment, the dynamic equation for the outer loop plant becomes:

$$\dot{\phi} = q \tag{36}$$

[10] implements a proportional controller, K_{T2}, of 3 in order to ensure the safety of the aircraft. Additionally, the relay reference angle for these tests is 20 deg. [10] implemented the switch condition (hysterisis) of 10 deg in order to prevent relay switching from noise. Finally, [10] conducted three separate tests (one for each airspeed operating condition). Using the Simulink model depicted in Figure 10, we recreated these tests.

Below in Figure 12 are the results of the 7m/s test:

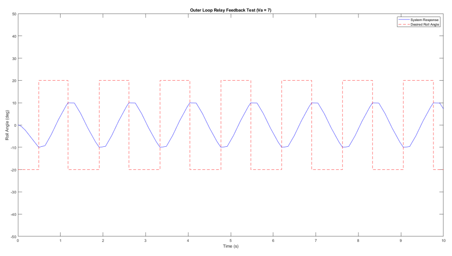

Fig. 12: Outer Loop Relay Feedback 7m/s Test

Below in Figure 13 are the results of the 10m/s test:

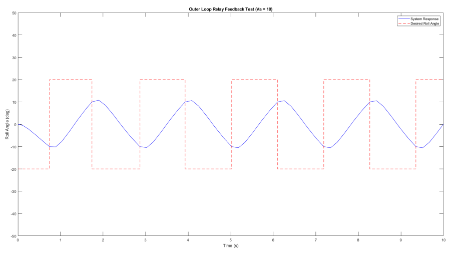

Fig. 13: Outer Loop Relay Feedback 10m/s Test

Below in Figure 14 are the results of the 15m/s test:

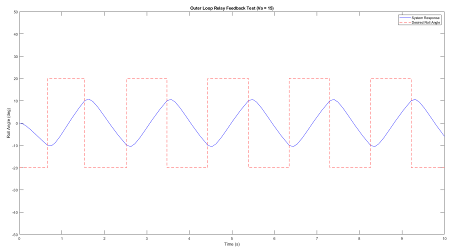

Fig. 14: Outer Loop Relay Feedback 15m/s Test

From these tests implemented in Simulink, we get results similar to [10]. Thus, following the controller tuning procedure outlined in the Derivations section, we can back out the Outer Loop Controller Gains:

a) :

OL Controller Gains		
V_a	K_p	τ_I
7	7.38	23.2
10	7.67	24.12
15	8	24.2

Because of the inner loop tuning and the fact that each of the responses above have similar amplitudes and frequency, the outer loop controllers do not have to be gain scheduled [10]. Thus, moving forward, the outer loop PI controller gains are selected from the 10 m/s operating condition.

D. Performance Evaluation - Setup

In order to evaluate the performance of the gain scheduled controller, [10] implements a traditional cascaded PI controller in order to present a baseline case to compare their control architecture to. The architecture is based on the roll rig experiments described above. The controller gains for this cascaded PI architecture are as follows: $K_{p1} = 0.23$, $\tau_{L1} = 0.146$, $K_{p2} = 7.76$, and $\tau_{L2} = 24.12$. Depicted in Figure 15 is the cascaded PI controller block diagram that [10] uses:

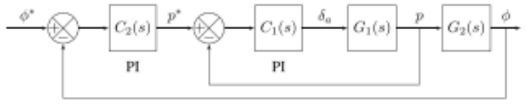

Fig. 15: Cascaded PI Controller Block Diagram [10]

We recreated this cascaded PI control architecture in Simulink. Below in Figure 16 is our recreation:

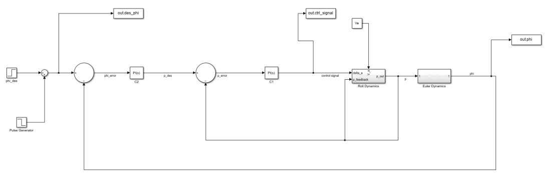

Fig. 16: Cascaded PI Controller Simulink Diagram

Additionally, [10] creates a gain scheduled control architecture in order to compare to the traditional cascaded PI architecture depicted above. Depicted in 17 is the Gain Scheduled PI controller block diagram that [10] uses:

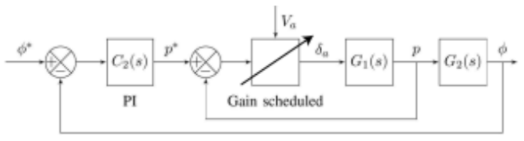

Fig. 17: Gain Scheduled Controller Block Diagram [10]

We recreated this Gain Scheduled PI control architecture in Simulink. Below in Figure 18 is our recreation:

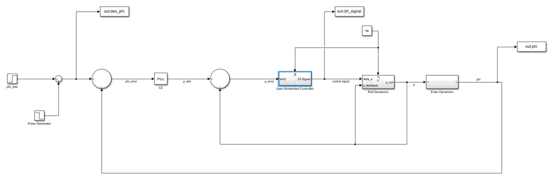

Fig. 18: Gain Scheduled Controller Simulink Diagram

[10] puts these two architectures under two tests at the three operating speeds. These two tests are a step response test as well as a zero reference test. For the step response tests, the desired roll angle is 30 degrees. Below in the next few sections are the results of these tests.

E. Traditional Cascaded PI Step Response Tests

Below in Figure 19 is the system response of the Cascaded PI step response test:

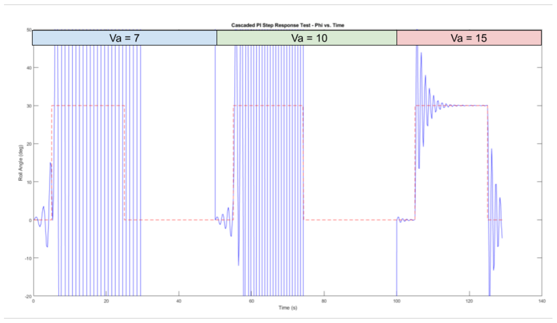

Fig. 19: Cascaded PI Step Response

Below in Figure 20 is the control input of the Cascaded PI step response test:

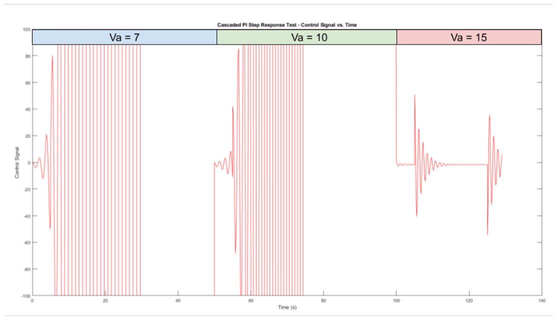

Fig. 20: Cascaded PI Step Control Input

F. *Traditional Cascaded PI Zero Reference Tests*

Below in Figure 21 is the system response of the Cascaded PI zero reference test:

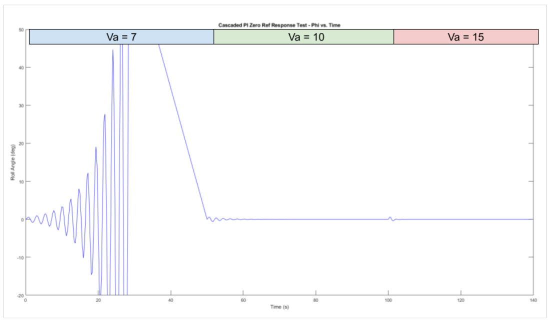

Fig. 21: Cascaded PI Zero Reference Response

Below in Figure 22 is the control input of the Cascaded PI zero reference test:

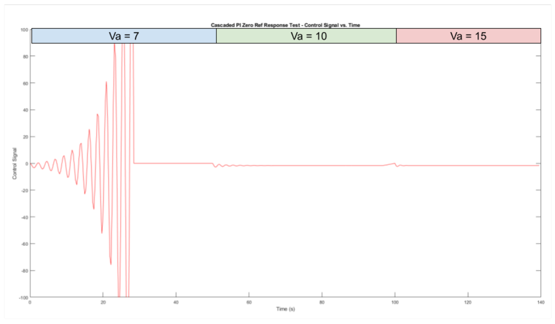

Fig. 22: Cascaded PI Zero Reference Control Input

G. *Gain Scheduled PI Step Response Tests*

Below in Figure 23 is the system response of the Gain Scheduled PI step response test:

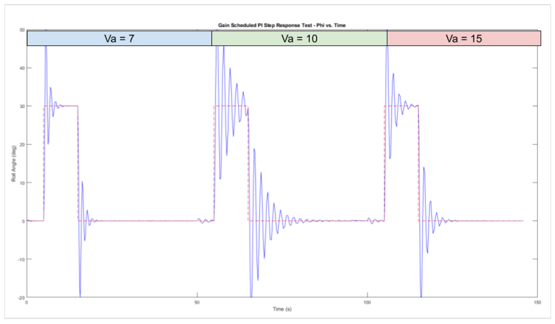

Fig. 23: Gain Scheduled Step Response

Below in Figure 24 is the control input of the Gain Scheduled PI step response test:

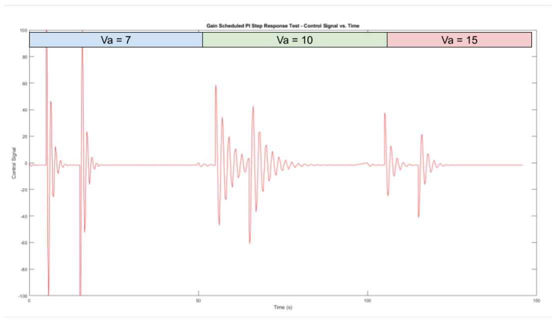

Fig. 24: Gain Scheduled Step Control Input

Additionally, we observed the control signals to ensure that the gain scheduling subsystem was functioning properly. In

Figure 25 is the plot of the proportional and integral controller gains during Gain Scheduled the step response test:

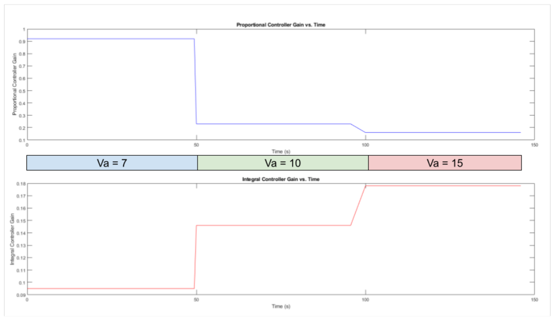

Fig. 25: Gain Scheduled Controller Gains

H. Gain Scheduled PI Zero Reference Tests

Below in Figure 26 is the system response of the Gain Scheduled PI zero reference test:

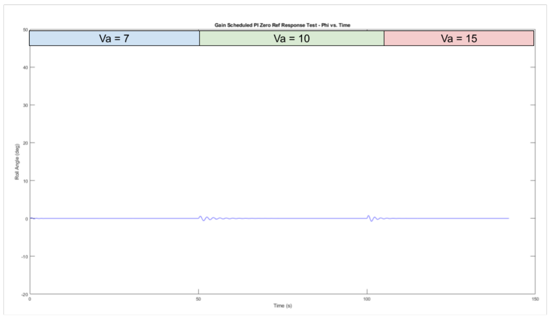

Fig. 26: Gain Scheduled Zero Reference Response

Below in Figure 27 is the control input of the Gain Scheduled PI zero reference test:

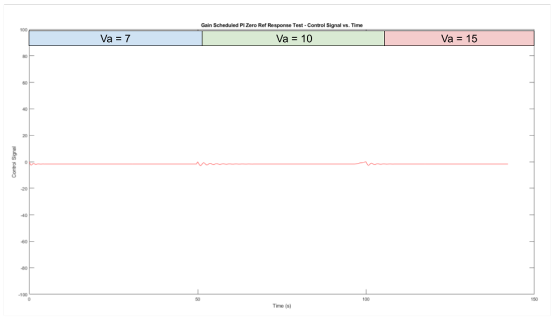

Fig. 27: Gain Scheduled Zero Reference Control Inputs

IV. Discussion

Unfortunately we were unsuccessful in perfectly recreating the results presented in [10]. The biggest obstacle that prevented us from recreating these results came in the form of the missing aerodynamic derivative coefficients C_{l_0}, C_{l_p}, and $C_{l_{\delta_a}}$. As explained earlier, we used outside resources in order to get the relative magnitude and sign information for these coefficients, but past that information, we performed a manual grid search to determine these coefficients. After many iterations of searching, the numbers we came up with corresponded with the ”best” matching of results across all tests presented in the paper. For example, we would almost perfectly match the Inner Loop Relay Feedback tests with a set of coefficients, but upon applying those coefficients to the Outer Loop Relay Feedback tests we would either get worse results or unstable behavior. We had to repeat these process across all the experiments performed above, and the overall best results came from $C_{l_0} = 1.108$, $C_{l_p} = -11.5$, and $C_{l_{\delta_a}} = 0.65$. The reason that [10] does not give us these coefficients is because of the novel controller tuning procedure described above. This proposed controller tuning procedure allows the user to tune controllers directly based on hardware without examining/calculating the dynamic equations. For this project, we tried contacting the authors of this paper, but got no response as of the time this project is due.

Another reason for us not being able to perfectly replicate the results was that the experiments in [10] were performed on physical hardware. Our results across all tests were ”smoother” than what was found in the paper. This is because simulation yields ideal cases while hardware experiments introduce more uncontrolled variables. For example, the air speed in a wind tunnel is not perfectly set at 7, 10, or 15 m/s. Furthermore, our simulations assume that the state of the system is known perfectly at all times. In [10], there is a time delay between where the airspeed is detected by the pitot-static tube and the actuators can actuate. [10] tried to offset this by placing the pitot-static tube ahead of the plane, but this still will not be as ideal as a simulation would be. Another minor reason that our results (especially in the zero reference tests) do not match [10] is that we did not have full control over the airspeed as the original authors did (i.e. perform smooth transition tests). We had to set the airspeed as a constant operating condition for our simulations to work. Furthermore, there was no detailed description of [10]’s techniques for implementation of their controller architecture. All they provided were block diagrams without any explanation of how they were implemented. As a result, our choice of blocks in Simulink may not have exactly matched what the authors of [10] used to control their UAV.

Even though our results do not perfectly match [10], our results do follow extremely similar patterns that are seen in the [10]’s results, which means that the driving factor for the discrepancies was not having full information about the plant. For example, in the Inner Loop Relay Feedback tests, each increase in airspeed condition in [10] yielded an increase in amplitude in the system response, even verging on unstable in the 15 m/s test. Our results presented in Figures 7-9 follow that exact same pattern. An increase in airspeed leads to an increase in amplitude. In the Outer Loop Relay Feedback tests, all three responses for all three airspeeds in [10] had similar amplitude and frequency. Our results in Figures 12-14 showed a similar pattern. For the performance evaluation between a traditional cascaded PI controller and the gain scheduled controller, the exact response patterns were not replicated, but the same conclusions were drawn. In the cascaded PI step and zero reference response tests, the responses were unstable at points in our results. This does not match what was found in [10], as the cascaded PI controller only exhibited instability during the 15m/s operating condition. However, our gain scheduled controller step and zero reference responses were stable and exhibited similar behavior to what was found in [10]. We can draw the exact same conclusions from our results as theirs: that the gain scheduled controller is necessary to not only stabilize the system under various operating conditions but provides a response that is faster settling and requires less control input (therefore preventing over-stress on the actuators) than a traditional cascaded PI controller.

A. Notes on Tethered Wind Tunnel Test

We attempted to recreate the results from the tethered wind tunnel test toward the end of [10] to prove the viability of the gain scheduled control architecture in a less-inhibited environment. The UAV's center of mass is mounted to a string inside of the wind tunnel and uses a larger-scale gain-scheduled control architecture to control the UAV under different airspeed conditions. However, the issues faced earlier when grid searching for three plant coefficients only amplified in the recreation of this particular test. We had to recreate the control architecture in the above figure. We first had to extend the inner-loop plant model to include all the equations for 9, 10, and 11. We also had to readjust the outer loop plant to be fully-encompassing of equation 12. This led to us having 22 unknown crucial parameters. There were 16 unknown aerodynamic derivative coefficients (including the three we found earlier, which were not ideal to start with), and 6 controller gain values (encompassing the controllers for the pitch and yaw of the UAV).

We recreated the control architecture presented in [10] in Simulink and began a manual grid search to attempt to discern these unknown parameters. Even with the assistance of [3] in determining the relative magnitude and sign information for these coefficients, we simply were unable to yield any results at all. Every time we would run a test, Simulink would throw errors. These errors were either unbound derivatives or unbound integrals. Because the system dynamics are highly coupled and nonlinear, Simulink might have had trouble passing information from one block to another at the correct time. Another source of error could be the 22 unknown parameters causing signals to be too high or too low, thus hitting Simulink's computational limits. Either way, because we could not get graphical results, we were unable to set up an automatic grid search to find these parameters. Since we have no results, we neglected the tethered wind tunnel test from the Simulations section. However, we have included the Simulink model in our code submission for inspection and proof-of-effort.

Regardless of our failure in the tethered wind tunnel test, the claims made earlier in this section still hold based on the results from the 10 other simulations we performed.

V. Conclusion

[10] provides two novel techniques that aid in the designing controllers and controlling fixed-wing UAVs. The first of which is an approach to automatically tune the controllers, which involves performing oscillatory relay-feedback tests and examining the frequency responses in order to calculate controller gains. The second technique is for the control of fixed wing UAVs through a gain-scheduled controller. Because of the non-linearity of UAV dynamics, a single controller may not be sufficient to control the system safely under different operating conditions. Hence, tuning a gain scheduled controller to handle a wide band of operating conditions proves useful in maintaining the safe operation of this particular fixed-wing UAV. These concepts are proven in [10]. In our process of recreating these results, we have discovered the exact same benefits and conclusions found in [10] even without perfectly replicating their results.

References

[1] Shabban Ali-Salman, Jinyoung Choi, and S.G. Anavatti. “Attitude Dynamics Identification of Unmanned Aircraft Vehicle”. In: *International Journal of Control Automation and Systems* (2010).

[2] Christopher J. Bett. “The Electrical Engineering Handbook”. In: Academic Press, 2005. Chap. 9, pp. 1107–1114.

[3] David A. Caughey. *Introduction to Aircraft Stability and Control Course Notes for MAE 5070*. Sibley School of Mechanical and Aerospace Engineering Cornell University, 2011.

[4] Wenqing Cui et al. “A Relay Feedback Method for the Tuning of Linear Active Disturbance Rejection Controllers”. In: (2020).

[5] Or Dantsker et al. “High Fidelity Moment of Inertia Testing of Unmanned Aircraft”. In: *American Institute of Aeronautics and Astronautics* (2011).

[6] James Diebel. “Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors”. In: (2006).

[7] *Everything You Need to Know About Fixed Wing Drone*. URL: https://www.zenadrone.com/everything-you-need-to-know-about-fixed-wing-drones/.

[8] Howard Loewen. “Stability Derivatives - What they are and how they are used”. In: (2013).

[9] Parkorn Poksawat, Liuping Wang, and Abdulghani Mohamed. “Automatic Tuning of Attitude Control System for Fixed-Wing Unmanned Aerial Vehicles”. In: *IET Control Theory and Applications* (2016).

[10] Parkorn Poksawat, Liuping Wang, and Abdulghani Mohamed. “Gain Scheduled Attitude Control of Fixed-Wing UAV with Automatic Controller Tuning”. In: *IEEE Transactions on Control Systems Technology* 26.4 (2018), pp. 1192–1203.

[11] L Wang and W Cluett. “Tuning PID controllers for integrating processes”. In: *IEE Proceedings on Control Theory Applications* 144.5 (1997), pp. 385–392.

[12] Scott Zhuge. *PID Control Theory*. URL: https://www.crystalinstruments.com/blog/2020/8/23/pid-control-theory#:~:text=It%20computes%20the%20output%20value,it%20by%20a%20constant%20K&text=Integral%20term%3A%20take%20the%20cumulative,it%20by%20a%20constant%20K&text=Derivative%20term%3A%20take%20the%20rate,it%20by%20a%20constant%20K.