AAE 497 Final Report: Autopilot Design

Vishnu Vijay

Abstract

This report describes the process by which an autopilot is designed. We start with a derivation of the aircraft dynamics, generating a system of nonlinear equations of motion describing the aircraft's 12 defined states. These equations of motion are then linearized to decouple the dynamics of the aircraft into two major groups: the longitudinal and lateral dynamics. This decoupling also simplifies the design of the autopilot controller. The equilibrium, or trim, condition is also determined at this stage. The trim condition is defined to be a condition where a significant portion of the aircraft states are constant. The linearized models are defined with respect to this trim condition. Transfer function and state space models are then derived from these simplified equations. These models are used to generate our autopilot controllers: the Successive Loop Closure (SLC) Controller, constructed from the transfer function models, and the Linear-Quadratic Regulator (LQR) Controller, constructed from the state space model. Both controllers are implemented and the performance of each is evaluated in the report.

CONTENTS

I. Introduction

Computers have long aided humans read, process, and react to data at significant speeds. This is becoming increasingly true as technological advancements have made computers more efficient and powerful. It is a natural progression to combine computers with airplanes, unarguably one of the most complex machines in the world today, to help automate tasks humans previously had to do themselves.

These automation computers on aircraft are called autopilots. Autopilots take in a state command, typically the altitude, course, pitch, speed, etc. to maintain, and generate actuator commands that are sent to the aircraft's control surfaces and propulsion system. Autopilot systems are vastly important due to the potential consequences of human error in today's aviation industry.

Specifically in this report, we will assume the aircraft we are analyzing is a miniature air vehicle (MAV). Due to their small size, these aircraft behave differently than larger aircraft, affecting the assumptions made in our derivations.

In this report, we will create two different autopilot controllers. One autopilot is a Successive Loop Closure (SLC) controller. The other is a Linear-Quadratic Regulator (LQR) controller. These are two ways of creating a controller from a transfer function or state space model. Unlike the SLC autopilot, the LQR autopilot is an optimal-control controller. This means that it minimizes an objective function, typically a cost function, to produce the desired results.

Much of this analysis was taken from Randal Beard and Timothy McLain's *Small Unmanned Aircraft: Theory and Practice*. This includes the notation, derivations, and resulting equations.

In section II of the report, we will derive the the aircraft's dynamics, generating 12 equations of motion. We will consider the forces and moments acting on the MAV as well. In section III, we will perform the calculations necessary to construct the autopilots. Section IV displays the results from implementing these autopilots in a simulator. In section V, we discuss the controllers and conclude the report.

II. Aircraft Dynamics

In this section, we will derive the equations of motion that will be used as the basis for the controller design in the section to follow. First, we must define the reference frames we will use in our analysis. Then we can define the aircraft's states that the equations of motion will track. From here, we can derive the kinematics and dynamics of the aircraft, and construct the equations of motion.

A. Reference Frames and Aircraft States

We must first define an inertial frame, $\mathscr{F}^i$, which is fixed translationally in the earth. In this frame, the $\mathbf{i}^i$ vector is fixed to point north, the $\mathbf{j}^i$ vector is fixed to point east, and the $\mathbf{k}^i$ vector is fixed to point downward. The $\mathbf{k}^i$ vector is fixed as such to form a valid right-hand reference frame.

We must also define a vehicle frame, $\mathscr{F}^v$, and body frame, $\mathscr{F}^b$. The $\mathscr{F}^v$ frame is fixed translationally in the aircraft, but the frame components point in the same directions as $\mathscr{F}^i$. The $\mathscr{F}^b$ frame is fixed in the same point as the $\mathscr{F}^v$ frame, but rotated via the 3-2-1 Euler Angle sequence. This sequence is described below by the rotation matrix $\mathscr{R}_v^b(\phi,\theta,\psi)$ from $\mathscr{F}^v$ to $\mathscr{F}^b$. We define $\mathscr{R}_v^{v1}(\psi)$ to be the rotation matrix from $\mathscr{F}^v$ to an intermediately defined frame $\mathscr{F}^{v1}$ by the angle ψ. Similarly, $\mathscr{R}_{v1}^{v2}(\theta)$ is the rotation matrix from $\mathscr{F}^{v1}$ to another intermediately defined frame $\mathscr{F}^{v2}$ by angle θ, and $\mathscr{R}_{v2}^{b}(\phi)$ is the rotation matrix from $\mathscr{F}^{v2}$ to the body frame $\mathscr{F}^b$ by angle ϕ. This is represented by the following:

$$\begin{aligned}
\mathscr{R}_v^b(\phi,\theta,\psi) &= \mathscr{R}_{v2}^b(\phi)\mathscr{R}_{v1}^{v2}(\theta)\mathscr{R}_v^{v1}(\psi) \\
&= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{pmatrix} \begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{pmatrix} \begin{pmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} \cos(\theta)\cos(\psi) & \cos(\theta)\sin(\psi) & -\sin(\theta) \\ \sin(\phi)\sin(\theta)\cos(\psi)-\cos(\phi)\sin(\psi) & \sin(\phi)\sin(\theta)\sin(\psi)+\cos(\phi)\cos(\psi) & \sin(\phi)\cos(\theta) \\ \cos(\phi)\sin(\theta)\cos(\psi)+\sin(\phi)\sin(\psi) & \cos(\phi)\sin(\theta)\sin(\psi)-\sin(\phi)\cos(\psi) & \cos(\phi)\cos(\theta) \end{pmatrix}
\end{aligned}$$

With this rotation matrix, we can express the rotation from $\mathscr{F}^v$ to $\mathscr{F}^b$ in Equation 1 below.

$$\mathscr{F}^b = \mathscr{R}_v^b(\phi,\theta,\psi)\mathscr{F}^v \tag{1}$$

With the primary reference frames defined, we can now define the aircraft states. They are itemized in Table I below.

TABLE I: Aircraft States

State	Type	Reference Frame	Direction
p_n	Linear Position	$\mathscr{F}^i$	North ($\mathbf{i}^i$)
p_e	Linear Position	$\mathscr{F}^i$	East ($\mathbf{j}^i$)
p_d	Linear Position	$\mathscr{F}^i$	Down ($\mathbf{k}^i$)
u	Linear Velocity	$\mathscr{F}^b$	Forward ($\mathbf{i}^b$)
v	Linear Velocity	$\mathscr{F}^b$	Right ($\mathbf{j}^b$)
w	Linear Velocity	$\mathscr{F}^b$	Down ($\mathbf{k}^b$)
ϕ	Angular Position	$\mathscr{F}^{v2}$	Roll angle ($\mathbf{i}^{v2}$)
θ	Angular Position	$\mathscr{F}^{v1}$	Pitch angle ($\mathbf{j}^{v1}$)
ψ	Angular Position	$\mathscr{F}^v$	Yaw angle ($\mathbf{k}^v$)
p	Angular Velocity	$\mathscr{F}^b$	Roll ($\mathbf{i}^b$)
q	Angular Velocity	$\mathscr{F}^b$	Pitch ($\mathbf{j}^b$)
r	Angular Velocity	$\mathscr{F}^b$	Yaw ($\mathbf{k}^b$)

These states will be used in later derivations.

B. Kinematics

We will relate the states described in Table I in this section, first relating the linear positions and velocities and then the angular positions and velocities.

The inertial linear position vector $\left(p_n, p_e, p_d\right)^T$ can be related to the body frame velocity vector $\left(u, v, w\right)^T$ by taking the derivative and applying a rotation as shown below in Equation 2.

$$\frac{d}{dt}\mathscr{R}_v^b(\phi,\theta,\psi)\begin{pmatrix}p_n\\p_e\\p_d\end{pmatrix}=\begin{pmatrix}u\\v\\w\end{pmatrix}$$

$$\mathscr{R}_v^b(\phi,\theta,\psi)\frac{d}{dt}\begin{pmatrix}p_n\\p_e\\p_d\end{pmatrix}=\begin{pmatrix}u\\v\\w\end{pmatrix}$$

$$\frac{d}{dt}\begin{pmatrix}p_n\\p_e\\p_d\end{pmatrix}=\mathscr{R}_v^b(\phi,\theta,\psi)^T\begin{pmatrix}u\\v\\w\end{pmatrix}$$

$$\begin{pmatrix}\dot{p}_n\\\dot{p}_e\\\dot{p}_d\end{pmatrix}=\mathscr{R}_v^b(\phi,\theta,\psi)^T\begin{pmatrix}u\\v\\w\end{pmatrix} \tag{2}$$

Now we can relate the angular position vector $\left(\phi, \theta, \psi\right)^T$ to the angular velocity vector $\left(p, q, r\right)^T$. The angular velocity vector acts entirely in the body frame $\mathscr{F}^b$, while the angular position vector acts in different frames. Angle ϕ acts in $\mathscr{F}^{v2}$; angle θ acts in $\mathscr{F}^{v1}$; and angle ψ acts in $\mathscr{F}^v$. Thus, the two vectors are related by:

$$\begin{pmatrix}p\\q\\r\end{pmatrix}=\frac{d}{dt}\begin{pmatrix}\phi\\0\\0\end{pmatrix}+\mathscr{R}_{v2}^b(\phi)\frac{d}{dt}\begin{pmatrix}0\\\theta\\0\end{pmatrix}+\mathscr{R}_{v2}^b(\phi)\mathscr{R}_{v1}^{v2}(\theta)\frac{d}{dt}\begin{pmatrix}0\\0\\\phi\end{pmatrix}$$

$$\begin{pmatrix}p\\q\\r\end{pmatrix}=\begin{pmatrix}1&0&-\sin(\theta)\\0&\cos(\phi)&\sin(\phi)\cos(\theta)\\0&-\sin(\phi)&\cos(\phi)\cos(\theta)\end{pmatrix}\begin{pmatrix}\dot{\phi}\\\dot{\theta}\\\dot{\psi}\end{pmatrix}$$

This can be rewritten to express the derivative of the angular position vector as a function of the angular velocity vector. This is seen in Equation 3.

$$\begin{pmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{pmatrix} = \begin{pmatrix}1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta)\\ 0 & \cos(\phi) & -\sin(\phi)\\ 0 & \sin(\phi)\sec(\theta) & \cos(\phi)\sec(\theta)\end{pmatrix}\begin{pmatrix}p\\ q\\ r\end{pmatrix} \tag{3}$$

C. Dynamics

With the kinematic relations defined, we can derive dynamics equations by applying Newton's Second Law to the linear and rotational states. These derivations will be analyzed separately as they are based on different equations.

1) Linear Dynamics: The linear dynamics are based on

$$\mathrm{m}\frac{{}^{i}d\mathbf{V}_g}{dt} = \mathbf{f} \tag{4}$$

where V_g is the aircraft's ground velocity (i.e. velocity with respect to the inertial frame $\mathscr{F}^i$) and $\mathbf{f}$ is the net applied force on the aircraft. The expression $\frac{d^{i}\mathbf{V}_g}{dt}$ is the inertial derivative of the ground velocity vector. Since the applied forces $\mathbf{f}$ are typically expressed in the body frame $\mathscr{F}^b$, the LHS of equation 4 should be re-expressed in the same frame. This can be done by applying the Transport Theorem. The Transport Theorem states that given a vector $\mathbf{v}$ and a rotation rate from the first frame $\mathscr{F}^i$ to a second frame $\mathscr{F}^f$, represented by ${}^{i}\omega^{f}$, we can express the time derivative of the vector in the second frame by evaluating the following:

$$\frac{{}^{f}d\mathbf{v}}{dt} = \frac{{}^{i}d\mathbf{v}}{dt} + {}^{i}\omega^{f} \times \mathbf{v} \tag{5}$$

Applying Equation 5 to Equation 4 gives the following:

$$\mathrm{m}\left(\frac{{}^{b}d\mathbf{V}_g}{dt} + {}^{i}\omega^{b} \times \mathbf{V}_g\right) = \mathbf{f} \tag{6}$$

If all the expressions were to be written in terms of the body frame $\mathscr{F}^b$ and rearranged, Equation 6 can be rewritten as

$$\begin{aligned}
\mathrm{m}\left(\frac{{}^{b}d\mathbf{V}_g}{dt} + {}^{i}\omega^{b} \times \mathbf{V}_g\right) &= \mathbf{f}\\
\frac{{}^{b}d}{dt}\begin{pmatrix}u\\ v\\ w\end{pmatrix} + \begin{pmatrix}p\\ q\\ r\end{pmatrix} \times \begin{pmatrix}u\\ v\\ w\end{pmatrix} &= \frac{1}{\mathrm{m}}\begin{pmatrix}f_x\\ f_y\\ f_z\end{pmatrix}\\
\begin{pmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{pmatrix} + \begin{pmatrix}qw - rv\\ ru - pw\\ pv - qu\end{pmatrix} &= \frac{1}{\mathrm{m}}\begin{pmatrix}f_x\\ f_y\\ f_z\end{pmatrix}\\
\begin{pmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{pmatrix} &= \frac{1}{\mathrm{m}}\begin{pmatrix}f_x\\ f_y\\ f_z\end{pmatrix} - \begin{pmatrix}qw - rv\\ ru - pw\\ pv - qu\end{pmatrix}
\end{aligned} \tag{7}$$

2) Rotational Dynamics: The rotational dynamics are based on

$$\frac{{}^{i}d\mathbf{h}}{dt} = \mathbf{m} \tag{8}$$

where $\mathbf{h}$ is the angular momentum vector and $\mathbf{m}$ is the the net applied moment on the aircraft about its center of mass. The expression $\frac{{}^{i}d\mathbf{h}}{dt}$ is the inertial derivative of the angular momentum vector. Applying Transport Theorem to the LHS of Equation 8 gives

$$\frac{{}^{b}d\mathbf{h}}{dt} + {}^{i}\omega^{b} \times \mathbf{h} = \mathbf{m} \tag{9}$$

The angular momentum vector $\mathbf{h}$ can be expressed in terms of the angular velocity ${}^{i}\omega^{b}$ and the inertia tensor $\mathbf{J}$:

$$\mathbf{h} = \mathbf{J}{}^{i}\omega^{b}$$

The inertia tensor $\mathbf{J}$ is given to be:

$$\mathbf{J} = \begin{pmatrix} J_x & -J_{xy} & -J_{xz} \\ -J_{xy} & J_y & -J_{yz} \\ -Jxz & -J_{yz} & J_z \end{pmatrix}$$

Substituting the expression for $\mathbf{h}$ into Equation 9 produces the equation

$$\frac{{}^b d(\mathbf{J}^i\omega^b)}{dt} + {}^i\omega^b \times (\mathbf{J}^i\omega^b) = \mathbf{m}$$

We can simplify this expression knowing that the time derivative of the inertia tensor in the body frame is 0. That is, $\mathbf{J}$ is constant in the body frame. We can also substitute in expressions for ${}^i\omega^b$ and $\mathbf{m}$ expressed in the body frame $\mathscr{F}^b$, giving

$$\mathbf{J}\frac{{}^b d({}^i\omega^b)}{dt} + {}^i\omega^b \times (\mathbf{J}^i\omega^b) = \mathbf{m}$$

$$\mathbf{J}\frac{{}^b d}{dt}\begin{pmatrix} p \\ q \\ r \end{pmatrix} + \begin{pmatrix} p \\ q \\ r \end{pmatrix} \times \mathbf{J}\begin{pmatrix} p \\ q \\ r \end{pmatrix} = \begin{pmatrix} l \\ m \\ n \end{pmatrix}$$

$$\mathbf{J}\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} + \begin{pmatrix} p \\ q \\ r \end{pmatrix} \times \mathbf{J}\begin{pmatrix} p \\ q \\ r \end{pmatrix} = \begin{pmatrix} l \\ m \\ n \end{pmatrix}$$

where l is the applied moment about $\mathbf{i}^b$ in $\mathscr{F}^b$, m is the applied moment about $\mathbf{j}^b$ in $\mathscr{F}^b$, and n is the applied moment about $\mathbf{k}^b$ in $\mathscr{F}^b$. Rearranging,

$$\mathbf{J}\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = -\begin{pmatrix} p \\ q \\ r \end{pmatrix} \times \mathbf{J}\begin{pmatrix} p \\ q \\ r \end{pmatrix} + \begin{pmatrix} l \\ m \\ n \end{pmatrix}$$

$$\mathbf{J}\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} J_{xz}pq + (J_y - J_z)qr \\ J_{xz}(r^2 - p^2) + (J_z - J_x)pr \\ (J_x - J_y)pq - J_{xz}qr \end{pmatrix} + \begin{pmatrix} l \\ m \\ n \end{pmatrix}$$

$$\mathbf{J}\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} J_{xz}pq + (J_y - J_z)qr + l \\ J_{xz}(r^2 - p^2) + (J_z - J_x)pr + m \\ (J_x - J_y)pq - J_{xz}qr + n \end{pmatrix}$$

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \mathbf{J}^{-1}\begin{pmatrix} J_{xz}pq + (J_y - J_z)qr + l \\ J_{xz}(r^2 - p^2) + (J_z - J_x)pr + m \\ (J_x - J_y)pq - J_{xz}qr + n \end{pmatrix}$$

Computing $\mathbf{J}^{-1}$

$$\mathbf{J}^{-1} = \frac{1}{\Gamma}\begin{pmatrix} J_z & 0 & J_{xz} \\ 0 & \frac{\Gamma}{J_y} & 0 \\ J_{xz} & 0 & J_x \end{pmatrix}$$

where $\Gamma = J_x J_z - J_{xz}^2$. Finally, we have

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \frac{1}{\Gamma}\begin{pmatrix} J_z & 0 & J_{xz} \\ 0 & \frac{\Gamma}{J_y} & 0 \\ J_{xz} & 0 & J_x \end{pmatrix}\begin{pmatrix} J_{xz}pq + (J_y - J_z)qr + l \\ J_{xz}(r^2 - p^2) + (J_z - J_x)pr + m \\ (J_x - J_y)pq - J_{xz}qr + n \end{pmatrix} \tag{10}$$

D. Applied Forces and Moments

The values f_x, f_y, and f_z represent the applied forces in the body frame's $\mathbf{i}^b$, $\mathbf{j}^b$, and $\mathbf{k}^b$ directions, respectively. Similarly, the values l, m, and n represent the applied moments about the body frame's $\mathbf{i}^b$, $\mathbf{j}^b$, and $\mathbf{k}^b$ directions, respectively. Expressions will be found in this section for each of these quantities.

The applied force vector $\mathbf{f}$ consists of a gravitational component $\mathbf{f}_g$, an aerodynamic component $\mathbf{f}_a$, and a propulsion component $\mathbf{f}_p$. The applied moment vector $\mathbf{m}$ consists of an aerodynamic component $\mathbf{m}_a$ and a propulsion component $\mathbf{m}_p$.

1) Gravitational Force: The gravitational force on the aircraft in the inertial frame $\mathbf{f}_g^i$ is

$$\mathbf{f}_g^i = \begin{pmatrix} 0 \\ 0 \\ \mathrm{m}g \end{pmatrix}$$

The applied force vector is expressed in the body frame, so $\mathbf{f}_g^i$ is transformed to $\mathscr{F}^b$ as follows

$$\mathbf{f}_g^b = \mathscr{R}_i^b \begin{pmatrix} 0 \\ 0 \\ \mathrm{m}g \end{pmatrix} \tag{11}$$

2) Aerodynamic Forces and Moments: The aerodynamic forces and moments on the aircraft must first be split into its components. Looking at the longitudinal group (acting in the pitching plane) first, we can express lift, drag, and aerodynamic pitching moment with the following linear models:

$$\begin{aligned} F_{\text{lift}} &= \frac{1}{2}\rho V_a^2 S C_L(\alpha, q, \delta_e) \\ F_{\text{drag}} &= \frac{1}{2}\rho V_a^2 S C_D(\alpha, q, \delta_e) \\ m_a &= \frac{1}{2}\rho V_a^2 S c C_m(\alpha, q, \delta_e) \end{aligned}$$

where ρ is the air density, C_L is the coefficient of lift, C_D is the coefficient of drag, α is the aircraft's angle of attack relative to the airspeed vector, δ_e is the elevator deflection, and c is the wing chord length. Fully expanded, these equations become

$$\begin{aligned} F_{\text{lift}} &= \frac{1}{2}\rho V_a^2 S\left[C_{L_0} + C_{L_\alpha}\alpha + C_{L_q}\frac{c}{2V_a}q + C_{L_{\delta_e}}\delta_e\right] \\ F_{\text{drag}} &= \frac{1}{2}\rho V_a^2 S\left[C_{D_0} + C_{D_\alpha}\alpha + C_{D_q}\frac{c}{2V_a}q + C_{D_{\delta_e}}\delta_e\right] \\ m_a &= \frac{1}{2}\rho V_a^2 S c\left[C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{c}{2V_a}q + C_{m_{\delta_e}}\delta_e\right] \end{aligned}$$

This linear model is valid for small angles of attack. With larger angles of attack, the models cannot accurately predict the decrease in C_L and increase in C_D resulting in flow separation. To generate a more accurate model, we must blend the lift linear model with the flat-plate model, which is found to be reasonably accurate in predicting the coefficient of lift beyond the stall angle. The lift model becomes

$$F_{\text{lift}} = \frac{1}{2}\rho V_a^2 S\left[C_L(\alpha) + C_{L_q}\frac{c}{2V_a}q + C_{L_{\delta_e}}\delta_e\right] \tag{12}$$

$$C_L(\alpha) = \big(1 - \sigma(\alpha)\big)\left[C_{L_0} + C_{L_\alpha}\alpha\right] + \sigma(\alpha)\left[2\text{sign}(\alpha)\sin^2(\alpha)\cos(\alpha)\right] \tag{13}$$

$$\sigma(\alpha) = \frac{1 + e^{-M(\alpha-\alpha_0)} + e^{M(\alpha+\alpha_0)}}{\left(1 + e^{-M(\alpha-\alpha_0)}\right)\left(1 + e^{M(\alpha+\alpha_0)}\right)} \tag{14}$$

where Equation 13 expresses how coefficient of lift varies with angle of attack, and Equation 14 is used to blend the linear lift model with the flat-plate model.

The drag force is best predicted with a nonlinear model as well. The nonlinear model used is below

$$F_{\text{drag}} = \frac{1}{2}\rho V_a^2 S\left[C_D(\alpha) + C_{D_q}\frac{c}{2V_a}q + C_{D_{\delta_e}}\delta_e\right] \tag{15}$$

$$C_D(\alpha) = C_{D_p} + \frac{(C_{L_0} + C_{L_\alpha}\alpha)^2}{\pi e_{os} AR} \tag{16}$$

where C_{D_p} is the parasitic drag coefficient, $e_o s$ is the Oswald efficiency factor, and AR is the wing aspect ratio.

The pitching moment linear model is reasonably accurate for simulation purposes.

$$m_a = \frac{1}{2}\rho V_a^2 S c\left[C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{c}{2V_a}q + C_{m_{\delta_e}}\delta_e\right] \tag{17}$$

Now looking at the lateral group of forces, we have the following expressions for f_y, l, and n

$$\begin{aligned} f_y &= \frac{1}{2}\rho V_a^2 S C_Y(\beta, p, r, \delta_a, \delta_r) \\ l_a &= \frac{1}{2}\rho V_a^2 S b C_Y(\beta, p, r, \delta_a, \delta_r) \\ n_a &= \frac{1}{2}\rho V_a^2 S b C_Y(\beta, p, r, \delta_a, \delta_r) \end{aligned}$$

where β is the sideslip angle from the airspeed vector, δ_a is the aileron deflection, δ_r is the rudder deflection, and b is the wingspan. Expanded these equations become

$$F_y = \frac{1}{2}\rho V_a^2 S\left[C_{Y_0} + C_{Y_\beta}\beta + C_{Y_p}\frac{b}{2V_a}p + C_{Y_r}\frac{b}{2V_a}r + C_{Y_{\delta_a}}\delta_a + C_{Y_{\delta_e}}\delta_r\right] \tag{18}$$

$$l_a = \frac{1}{2}\rho V_a^2 S b\left[C_{l_0} + C_{l_\beta}\beta + C_{l_p}\frac{b}{2V_a}p + C_{l_r}\frac{b}{2V_a}r + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_e}}\delta_r\right] \tag{19}$$

$$n_a = \frac{1}{2}\rho V_a^2 S b\left[C_{n_0} + C_{n_\beta}\beta + C_{n_p}\frac{b}{2V_a}p + C_{n_r}\frac{b}{2V_a}r + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_e}}\delta_r\right] \tag{20}$$

These linear models are sufficiently accurate.

3) Propulsion Force and Moment: The propulsive force and moment component consist of thrust T_p and torque Q_p applied to the aircraft by its propulsion system. Thrust and torque act in and about $\mathbf{i}^b$, respectively. Expressions for both are as follows

$$\begin{aligned} T_p &= \rho n^2 D^4 C_T(J) \\ Q_p &= \rho n^2 D^5 C_Q(J) \end{aligned}$$

where $n = \frac{\Omega}{2\pi}$ is the propeller speed in revolutions per second, Ω is the propeller speed in radians per second, D is the propeller diameter, $J = \frac{2\pi V_a}{\Omega_p D}$ is the advance ratio, C_T is the thrust coefficient, and C_Q is the torque coefficient. The coefficients can be approximated as quadratic functions. That is,

$$\begin{aligned} C_T(J) &= C_{T2}J^2 + C_{T1}J + C_{T0} \\ C_Q(J) &= C_{Q2}J^2 + C_{Q1}J + C_{Q0} \end{aligned}$$

We assume a rigid propeller shaft and equate the expression for propulsive torque Q_p to the expression for motor torque Q_m below

$$Q_m = K_Q\left[\frac{1}{R}\big(V_{max}\delta_t - K_V\Omega\big) - i_0\right]$$

where K_Q, K_V, and i_0 are motor-specific constants, V_{max} is the maximum voltage provided by the battery, and δ_t is the throttle percent.

Equating the two expressions and solving for Ω gives us the following for operating propeller speed Ω_{op}

$$\Omega_{op} = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$\begin{aligned} a &= \frac{\rho D^5}{(2\pi)^2}C_{Q0} \\ b &= \frac{\rho D^4}{2\pi}C_{Q1}V_a + \frac{K_Q K_V}{R} \\ c &= \rho D^3 C_{Q2}V_a^2 - \frac{K_Q V_{max}\delta_t}{R} + K_Q i_0 \end{aligned}$$

Then, the expressions for thrust and torque are

$$T_p = \rho\frac{\Omega_{op}^2}{4\pi^4}D^4\big(C_{T2}J_{op}^2 + C_{T1}J_{op} + C_{T0}\big) \tag{21}$$

$$Q_p = \rho\frac{\Omega_{op}^2}{4\pi^4}D^5\big(C_{Q2}J_{op}^2 + C_{Q1}J_{op} + C_{Q0}\big) \tag{22}$$

4) Net Force and Moment: Transforming $\mathbf{f}_g$, $\mathbf{f}_a$, and $\mathbf{f}_p$ to the body frame and summing together, we get the following for net force $\mathbf{f}$

$$\mathbf{f} = \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} = \begin{pmatrix} -\mathrm{m}g\sin\theta \\ \mathrm{m}g\cos\theta\sin\phi \\ \mathrm{m}g\cos\theta\cos\phi \end{pmatrix} + \begin{pmatrix} -F_{\text{drag}}\cos\alpha + F_{\text{lift}}\sin\alpha \\ F_y \\ -F_{\text{drag}}\sin\alpha - F_{\text{lift}}\cos\alpha \end{pmatrix} + \begin{pmatrix} T_p \\ 0 \\ 0 \end{pmatrix} \tag{23}$$

Similarly, summing together $\mathbf{m}_a$ and $\mathbf{m}_p$, we get the following for net moment $\mathbf{m}$

$$\mathbf{m} = \begin{pmatrix} l \\ m \\ n \end{pmatrix} = \begin{pmatrix} l_a \\ m_a \\ n_a \end{pmatrix} + \begin{pmatrix} Q_p \\ 0 \\ 0 \end{pmatrix} \tag{24}$$

Equations 23 and 24 can be substituted into Equations 7 and 10 to complete our equations of motion derivation.

E. Implementation

Python scripts implementing the equations expressed above were implemented, and are shown in the Appendix. The Runge-Kutta 4 algorithm is used to numerically integrate the differential equations.

The Euler angle attitude representation is susceptible to a phenomenon known as "gimbal lock," where two or more of the axes align and a degree of freedom is lost. Mathematically, this is known as a singularity. In the case of our aircraft, the singularity occurs when it is pitched directly up or down (ie $\theta = \pm 90$ degrees). At this point, we are no longer able to determine the value of ψ. In the implementation of our aircraft dynamics, we forgo using Euler angles in calculations for this very reason. Rather, we use unit quaternions to represent the aircraft's attitude. Quaternion, with 4 DOF, are not susceptible to gimbal lock. We write the quaternion e in vector form as

$$e = \begin{pmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \end{pmatrix}$$

As they are of unit length, we know that $e_0^2 + e_1^2 + e_2^2 + e_3^2 = 1$

Converting from the Euler angle representation to quaternion representation of attitude, we use

$$\begin{aligned} e_0 &= \cos\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2} \\ e_1 &= \cos\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2} - \sin\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2} \\ e_2 &= \cos\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2} \\ e_3 &= \sin\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2} - \cos\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2} \end{aligned}$$

Converting from the quaternion representation to Euler angle representation of attitude

$$\begin{aligned} \phi &= \text{atan2}\big(2(e_0e_1 + e_2e_3), (e_0^2 + e_3^2 - e_1^2 - e_2^2)\big) \\ \theta &= \arcsin\big(2(e_0e_2 - e_1e_3)\big) \\ \psi &= \text{atan2}\big(2(e_0e_3 + e_1e_2), (e_0^2 + e_1^2 - e_2^2 - e_3^2)\big) \end{aligned}$$

III. Controller Design

In this section, we will use the equations of motion derived in the previous section to generate a model from which we can design an optimal-control autopilot. For much of this section, we will refer to the states of and inputs to the aircraft. We denote the states with x and inputs with u. These can also be expressed as

$$\begin{aligned} x &= \big(p_n, p_e, p_d, u, v, w, \phi, \theta, \psi, p, q, r\big)^T \\ u &= \big(\delta_e, \delta_a, \delta_r, \delta_t\big)^T \end{aligned}$$

We also define the function

$$\dot{x} = f(x, u) \tag{25}$$

to represent our system.

A. Trim Calculation

First, we will need to find the trim condition. Trim, denoted by the superscript *, is the state at which much of the aircraft's states are at equilibrium. Finding the trim condition is important as the linearized models are defined with respect to this equilibrium state. We define equilibrium to be $f(x^*, u^*) = 0$. At trim, we write that

$$\dot{x}^* = f(x^*, u^*) \tag{26}$$

where x^* is the trim states and u^* is the trim inputs. In steady level flight, all states but north and east positions are constant. That is,

$$\dot{x}^* = \big(\text{irrelevant}, \text{irrelevant}, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\big)^T$$

Assuming a constant airspeed V_a^*, a constant flight path angle γ^*, and constant turning radius R^*, we can write

$$\dot{x}^* = \begin{pmatrix} \text{irrelevant} \\ \text{irrelevant} \\ V_a^* \sin\gamma^* \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{V_a^*}{R^*} \\ 0 \\ 0 \end{pmatrix} \tag{27}$$

Finding the trim states x^* and inputs u^* will consist of solving Equation 26 with Equation 27. This is a constrained optimization problem that can be solved in Python with the `scipy.optimize.minimize()` function. Here, we use a 13-state vector with the quaternion attitude representation, over the 12-state vector with the Euler angle representation. Our constraints into the function define the following:

$$\begin{aligned} u^2 + v^2 + w^2 &= V_a^2 \\ v &= 0 \\ e_0^2 + e_1^2 + e_2^2 + e_3^2 &= 1 \\ e_1 &= 0 \\ e_3 &= 0 \\ p &= 0 \\ q &= 0 \\ r &= 0 \end{aligned}$$

Using the function, with a desired airspeed V_a^* of 25 m/s, desired flight path angle γ^* of 0 degree, and turning radius R^* of ∞ meters, our program outputs the following for trim:

```
import numpy as np
x_trim = np.array([[-0.000000, -0.000000, -0.000000, 24.968623, 0.000000, 1.252151,
        0.999686, 0.000000, 0.025051, 0.000000, 0.000000, 0.000000, 0.000000]]).T
u_trim = np.array([[-0.125044, 0.001837, -0.000303, 0.676775]]).T
Va_trim = 25.000000
alpha_trim = 0.050107
theta_trim = 0.050107
```

These are the trim values for steady level flight at 25 m/s.

B. Linearization

Before generating an autopilot, we must first linearize the equations of motion derived earlier. After linearization, we can generate transfer function and state space models for the aircraft's behavior in the lateral and longitudinal planes. We will linearize the lateral dynamics first and then proceed to the longitudinal dynamics. All linearization error constants and disturbances are listed in the Appendix.

1) Lateral - Roll: We start with the EoM for roll ϕ,

$$\dot{\phi} = p + q \sin\phi \tan\theta + r \cos\phi \tan\theta$$

Assuming small θ, we define a linearization error d_{ϕ_1}, giving

$$\dot{\phi} = p + d_{\phi_1}$$

We can differentiate the equation above and rearrange terms to give

$$\ddot{\phi} = -a_{\phi_1}\dot{\phi} + a_{\phi_2}\delta_a + d_{\phi_2} \tag{28}$$

where

$$\begin{aligned} a_{\phi_1} &= -\frac{1}{2}\rho V_a^2 SbC_{p_p}\frac{b}{2V_a} \\ a_{\phi_2} &= \frac{1}{2}\rho V_a^2 SbC_{p_{\delta_a}} \end{aligned}$$

Disturbance d_{ϕ_2} can be found in the Appendix.

2) Lateral - Course: Without wind, the course heading χ is equivalent to the yaw angle ψ. That is, $\chi = \psi$. Then we can write

$$\begin{aligned} \dot{\psi} = \dot{\chi} &= \frac{g}{V_g}\tan\phi \\ &= \frac{g}{V_g}\phi + \frac{g}{V_g}(\tan\phi - \phi) \end{aligned}$$

Defining a disturbance d_χ, we get

$$\dot{\chi} = \frac{g}{V_g}\phi + \frac{g}{V_g}d_\chi \tag{29}$$

3) Lateral - Sideslip: In the absence of wind, we can write, by definition of sideslip angle β,

$$v = V_a \sin\beta$$

Differentiating the equation and rearranging terms gives

$$\dot{\beta} = -a_{\beta_1}\beta + a_{\beta_2}\delta_r + d_\beta \tag{30}$$

where

$$\begin{aligned} a_{\beta_1} &= \frac{\rho V_a S}{2\text{m}\cos\beta}C_{Y_\beta} \\ a_{\beta_2} &= \frac{\rho V_a S}{2\text{m}\cos\beta}C_{Y_{\delta_r}} \end{aligned}$$

Disturbance d_β can be found in the Appendix.

4) Longitudinal - Pitch: We start with the EoM for pitch θ,

$$\dot{\theta} = q\cos\phi - r\sin\phi$$

We define a linearization error d_{θ_1}, giving

$$\dot{\theta} = q + d_{\theta_1}$$

We can differentiate the equation and rearrange, giving

$$\ddot{\theta} = -a_{\theta_1}\dot{\theta} - a_{\theta_2}\theta + a_{\theta_3}\delta_e + d_{\theta_2} \tag{31}$$

where

$$\begin{aligned} a_{\theta_1} &= \frac{\rho V_a^2 cS}{2J_y}C_{m_q}\frac{c}{2V_a} \\ a_{\theta_2} &= \frac{\rho V_a^2 cS}{2J_y}C_{m_\alpha} \\ a_{\theta_3} &= \frac{\rho V_a^2 cS}{2J_y}C_{m_{\delta_e}} \end{aligned}$$

Disturbance d_{θ_1} can be found in the Appendix.

5) Longitudinal - Altitude: We define altitude h to be $h = -p_d$. Then,

$$\dot{h} = -\dot{p}_d = u\sin\theta - v\sin\phi\cos\theta - w\cos\phi\cos\theta$$

Defining a disturbance d_h, we get

$$\dot{h} = V_a\theta + d_h \tag{32}$$

6) Longitudinal - Airspeed: We know that the airspeed V_a can be expressed as

$$V_a = \sqrt{u^2 + v^2 + w^2}$$

Differentiating the equation,

$$\dot{V}_a = \dot{u}\cos\alpha\cos\beta + \dot{v}\sin\beta + \dot{w}\sin\alpha\cos\beta$$

Defining a disturbance d_{V_1},

$$\dot{V}_a = \dot{u}\cos\alpha + \dot{w}\sin\alpha + d_{V_1} \tag{33}$$

Defining deviations from trim,

$$\begin{aligned}\bar{V}_a &= V_a - V_a^* \\ \bar{\theta} &= \theta - \theta^* \\ \bar{\delta}_e &= \delta_e - \delta_e^* \\ \bar{\delta}_t &= \delta_t - \delta_t^*\end{aligned}$$

We can substitute these deviations from trim into Equation 33 and rearranging gives us

$$\dot{\bar{V}}_a = -a_{V_1}\bar{V}_a + a_{V_2}\bar{\delta}_t - a_{V_3}\bar{\theta} + d_V \tag{34}$$

where

$$\begin{aligned}a_{V_1} &= \frac{\rho V_a^* S}{\mathrm{m}}[C_{D_0} + C_{D_\alpha}\alpha^* + C_{D_{\delta_e}}\delta_e^*] - \frac{1}{m}\frac{\partial T_p}{\partial V_a}(\delta_t^*, V_a^*) \\ a_{V_2} &= \frac{1}{\mathrm{m}}\frac{\partial T_p}{\partial V_a}(\delta_t^*, V_a^*) \\ a_{V_3} &= g\cos(\theta^* - \alpha^*)\end{aligned}$$

The disturbance d_V can be found in the Appendix.

C. Transfer Function Models

We will now take the Laplace transform of Equations 28, 29, 30, 31, 32, and 34.

1) Lateral - Roll: Taking the Laplace transform of Equation 28 and rearranging,

$$\phi(s) = \left(\frac{a_{\phi_2}}{s(s + a_{\phi_1})}\right)\left(\delta_a(s) + \frac{1}{a_{\phi_2}}d_{\phi_2}\right) \tag{35}$$

2) Lateral - Course: Taking the Laplace transform of Equation 29 and rearranging,

$$\chi(s) = \frac{\left(\frac{g}{V_g}\right)}{s}\big(\phi(s) + d_\chi(s)\big) \tag{36}$$

3) Lateral - Sideslip: Taking the Laplace transform of Equation 30 and rearranging,

$$\beta(s) = \frac{a_{\beta_2}}{s + a_{\beta_1}}\big(\delta_r(s) + d_\beta(s)\big) \tag{37}$$

4) Longitudinal - Pitch: Taking the Laplace transform of Equation 31 and rearranging,

$$\theta(s) = \left(\frac{a_{\theta_3}}{s^2 + a_{\theta_1} s + a_{\theta_2}}\right)\left(\delta_e(s) + \frac{1}{a_{\theta_3}} d_{\theta_2}(s)\right) \tag{38}$$

5) Longitudinal - Altitude: Taking the Laplace transform of Equation 32 and rearranging,

$$h(s) = \frac{V_a}{s}\left(\theta + \frac{1}{V_a} d_h\right) \tag{39}$$

6) Longitudinal - Airspeed: Taking the Laplace transform of Equation 34 and rearranging,

$$\bar{V}_a(s) = \frac{1}{s + a_{V_1}}\left(a_{V_2}\bar{\delta}_t(s) - a_{V_3}\bar{\theta}(s) + d_V(s)\right) \tag{40}$$

D. State Space Models

We define the lateral states and inputs to be

$$x_{\text{lat}} = \begin{pmatrix} v \\ p \\ r \\ \phi \\ \psi \end{pmatrix}$$

$$u_{\text{lat}} = \begin{pmatrix} \delta_a \\ \delta_r \end{pmatrix}$$

We define the longitudinal states and inputs to be

$$x_{\text{lon}} = \begin{pmatrix} u \\ w \\ q \\ \theta \\ h \end{pmatrix}$$

$$u_{\text{lon}} = \begin{pmatrix} \delta_e \\ \delta_t \end{pmatrix}$$

We defined Equation 26 to be true earlier. We also defined Equation 27. Defining $\bar{x}$ to be the deviation from trim, we can write

$$\begin{aligned} \bar{x} &= x - x^* \\ \dot{\bar{x}} &= \dot{x} - \dot{x}^* \\ &= f(x, u) - f(x^*, u^*) \\ &= f(x^* + \bar{x}, u^* + \bar{u}) - f(x^*, u^*) \end{aligned}$$

From here we can use a first-order Taylor series approximation to give

$$\begin{aligned} \dot{\bar{x}} &\approx f(x^*, u^*) + \frac{\partial f(x^*, u^*)}{\partial x}\bar{x} + \frac{\partial f(x^*, u^*)}{\partial u}\bar{u} - f(x^*, u^*) \\ &= \frac{\partial f(x^*, u^*)}{\partial x}\bar{x} + \frac{\partial f(x^*, u^*)}{\partial u}\bar{u} \\ \dot{\bar{x}} &= A\bar{x} + B\bar{u} \end{aligned}$$

For the lateral and longitudinal systems, we have

$$\dot{\bar{x}}_{\text{lat}} = A_{\text{lat}}\bar{x}_{\text{lat}} + B_{\text{lat}}\bar{u}_{\text{lat}}$$
$$\dot{\bar{x}}_{\text{lon}} = A_{\text{lon}}\bar{x}_{\text{lon}} + B_{\text{lon}}\bar{u}_{\text{lon}}$$

where

$$A_{\text{lat}} = \frac{\partial f_{\text{lat}}}{\partial x_{\text{lat}}}$$
$$B_{\text{lat}} = \frac{\partial f_{\text{lat}}}{\partial u_{\text{lat}}}$$

and

$$A_{\text{lon}} = \frac{\partial f_{\text{lon}}}{\partial x_{\text{lon}}}$$
$$B_{\text{lon}} = \frac{\partial f_{\text{lon}}}{\partial u_{\text{lon}}}$$

The Jacobian matrices A_i and B_i were numerically computed according to the equations below:

$$\frac{\partial f}{\partial x} = \begin{pmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \dots & \frac{\partial f}{\partial x_n} \end{pmatrix}$$

$$\frac{\partial f}{\partial x_i} = \begin{pmatrix} \frac{\partial f_1}{\partial x_i} \\ \frac{\partial f_2}{\partial x_i} \\ \dots \\ \frac{\partial f_m}{\partial x_i} \end{pmatrix}$$

$$\frac{\partial f}{\partial x_i} \approx \frac{f(x + \epsilon e_i) - f(x)}{\epsilon}$$

where ϵ is a small number ($\epsilon << 1$), and e_i represents the state that the partial derivative is taken with respect to.

The Jacobian matrices were computed around the trim condition and can be found in the Appendix.

E. Successive Loop Closure (SLC) Controller

In this section we will create our first autopilot based on the Successive Loop Closure Method, where we look at a system's block diagram and "close" successive loops, starting at the innermost loop and moving outwards.

An example closed-loop system is shown below:

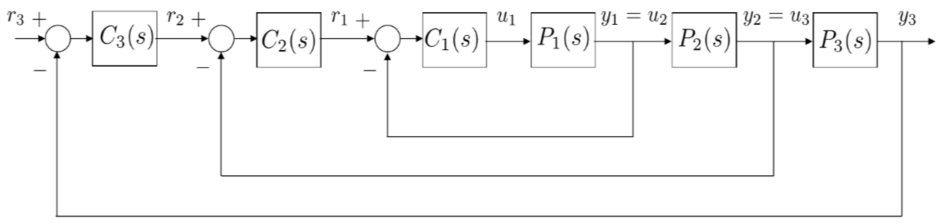


Fig. 1: Simple Closed Loop System [1]

We design the inner loop to operate at a bandwidth ω_{BW1}. When analyzing the system at a frequency substantially lower than this defined bandwidth, we can approximate the inner loop with a unity gain block, as seen below:

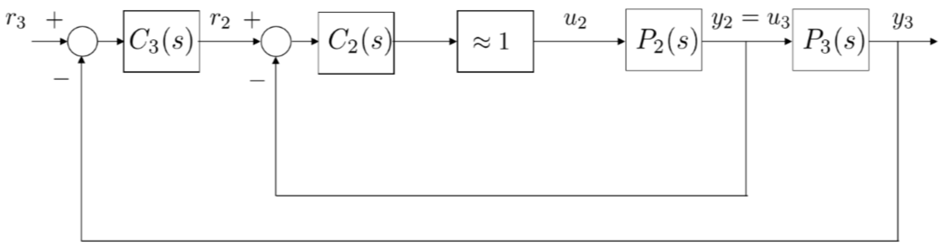


Fig. 2: Simple Closed Loop System - First Closure [1]

This is because at these lower frequencies, we can assume that the inner loop has already reached its desired state. That is, the input is equivalent to the output. Typically, we approximate the loop as a unity gain block when frequencies are lower by a factor of 10 or more. Now we can apply the SLC method to our lateral and longitudinal autopilots.

Using the transfer function models generated earlier, we can create the following block diagrams for the lateral autopilot

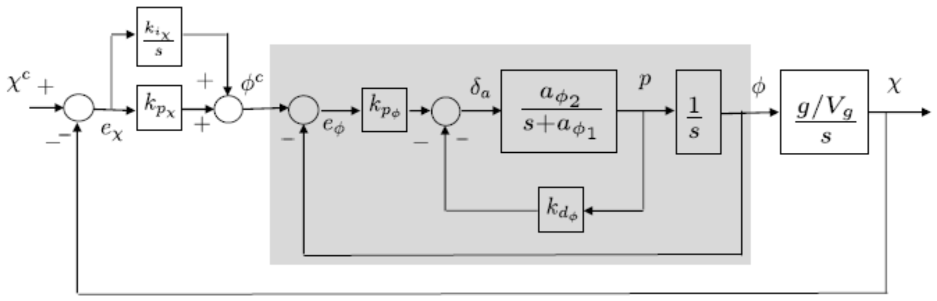
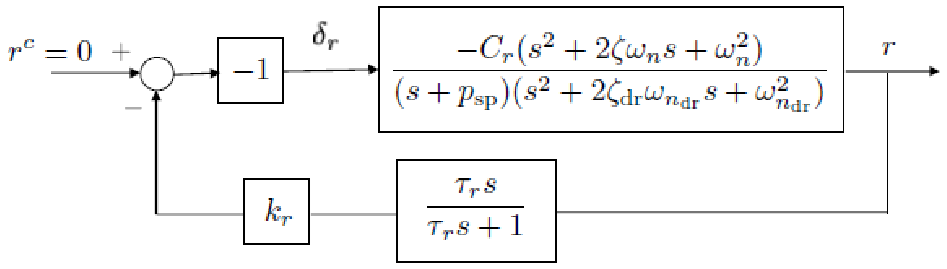


Fig. 3: Course-Roll Autopilot [1]



$r^c = 0$
$+$
$-$
-1
δ_r
$\frac{-C_r(s^2+2\zeta\omega_n s+\omega_n^2)}{(s+p_{\text{sp}})(s^2+2\zeta_{\text{dr}}\omega_{n_{\text{dr}}}s+\omega_{n_{\text{dr}}}^2)}$
r
k_r
$\frac{\tau_r s}{\tau_r s+1}$

Fig. 4: Yaw Damper [1]

The darkened section in Figure 3 is the roll hold algorithm, while the other blocks are used for the course hold algorithm. First looking at the roll hold loop, we can create a transfer function from commanded roll to true roll

$$G(s) = \frac{k_{p_\phi} a_{\phi_2}}{s^2 + (a_{\phi_1} + a_{\phi_2} k_{d_\phi}) + k_{p_\phi} a_{\phi_2}}$$

Setting this equation equal to the canonical second order transfer function, we find the gains to be

$$k_{p_\phi} = \frac{\omega_{n_\phi}^2}{a_{\phi_2}}$$
$$k_{d_\phi} = \frac{2\zeta_\phi \omega_{n_\phi} - a_{\phi_2}}{a_{\phi_2}}$$

For frequencies lower than ω_{n_ϕ} by a factor of 10 or more, we can approximate the roll loop as the unity gain block. We will define this factor to be W_χ.

Now looking at the course loop, we can rewrite Equation 36 and equate it to the canonical second order transfer function to find the gains and values

$$\omega_{n_\chi} = \frac{1}{W_\chi} \omega_{n_\phi}$$
$$k_{p_\chi} = 2\zeta_\chi \omega_{n_\chi} \frac{V_g}{g}$$
$$k_{i_\chi} = \omega_{n_\chi}^2 \frac{V_g}{g}$$

We can also generate the following block diagrams for the longitudinal autopilot

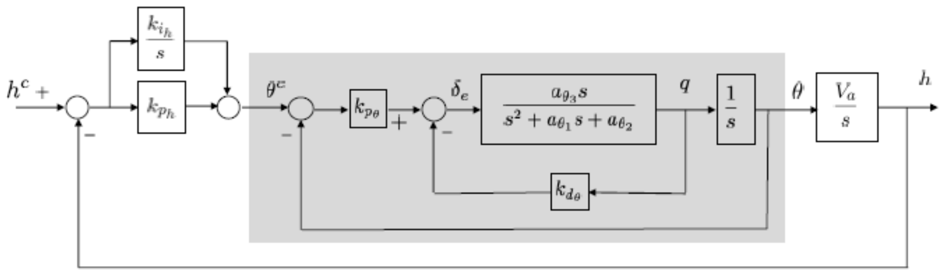


Fig. 5: Altitude-Pitch Autopilot [1]

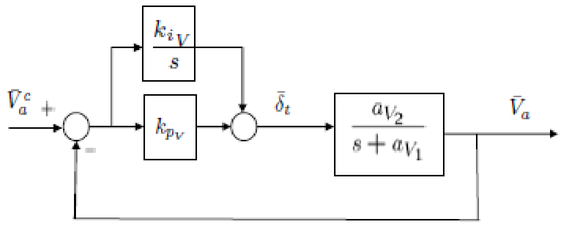


Fig. 6: Airspeed Autopilot [1]

The darkened section of Figure 5 is the pitch hold algorithm, while the other blocks are used for the altitude hold algorithm. First looking at the pitch hold, we can create a transfer function from commanded pitch to true pitch

$$G(s) = \frac{k_{p\theta} a_{\theta_3}}{s^2 + (a_{\theta_1} + k_{d\theta} a_{\theta_3})s + (a_{\theta_2} + k_{p\theta} a_{\theta_3})}$$

Equating the expression with the canonical second order transfer function, we find the gains to be

$$\begin{aligned} k_{p\theta} &= \frac{\omega_{n_\theta}^2 - a_{\theta_2}}{a_{\theta_3}} \\ k_{d\theta} &= \frac{2\zeta_\theta \omega_{n_\theta} - a_{\theta_1}}{a_{\theta_3}} \\ K_{\theta_{DC}} &= \frac{k_{p\theta} a_{\theta_3}}{\omega_{n_\theta}^2} \end{aligned}$$

We can approximate the pitch loop as a unity gain block for sufficiently lower frequencies than ω_{n_θ}. We define the factor to be W_h.

We can rewrite Equation 39 and equate it to the canonical second order transfer function to find the gains and values below

$$\begin{aligned} \omega_{n_h} &= \frac{1}{W_h} \omega_{n_\theta} \\ k_{i_h} &= \frac{\omega_{n_h}^2}{K_{\theta_{DC}} V_a} \\ k_{p_h} &= \frac{2\zeta_h \omega_{n_h}}{K_{\theta_{DC}} V_a} \end{aligned}$$

For airspeed hold, we can rewrite Equation 34 and equate it to the canonical second order transfer function to find

$$\begin{aligned} k_{i_V} &= \frac{\omega_{n_V}^2}{a_{V_2}} \\ k_{p_V} &= \frac{2\zeta_V \omega_{n_V} - a_{V_1}}{a_{V_2}} \end{aligned}$$

Thus, we tune the lateral autopilot by choosing values for ω_{n_ϕ}, ζ_ϕ, W_χ, ζ_χ, τ_r, and k_r. We tune the longitudinal autopilot by choosing values for ω_{n_θ}, ζ_θ, W_h, ζ_h, ω_{n_V}, and ζ_V. Values for these design parameters are chosen, implemented, and discussed in section IV.

F. Linear-Quadratic Regulator (LQR) Controller

We are given the system

$$\dot{\bar{x}} = A\bar{x} + B\bar{u}$$

and a state cost matrix Q and input cost R. The gains used by the LQR controller are given by

$$K = R^{-1}B^{T}P$$

where P is the solution to the Algebraic Riccati Equation.

The system inputs are determined by

$$u^{*}(t) = -Kx(t)$$

where $x(t)$ is the deviation from the desired state.

We augment the lateral states with an integral state for course χ. We also augment the longitudinal states with an integral state for height h and airspeed V_a. These specific states are augmented because they are the ones we are providing commands for. Then the augmented lateral states ζ_{lat} and longitudinal states ζ_{lon} are as follows

$$\begin{aligned}\zeta_{\text{lat}} &= \left(\tilde{v}, p, r, \phi, \tilde{\chi}, \textstyle\int \tilde{\chi}\right)^{T}\\ \zeta_{\text{lon}} &= \left(\tilde{u}, \tilde{w}, q, \theta, \tilde{h}, \textstyle\int \tilde{h}, \int \tilde{V}_a\right)^{T}\end{aligned}$$

where the $\tilde{}$ denotes the deviation from commanded state. All other states are being driven to 0.

The state space systems become

$$\begin{aligned}\dot{\zeta}_{\text{lat}} &= \bar{A}_{\text{lat}}\zeta_{\text{lat}} + \bar{B}_{\text{lat}}u_{\text{lat}}\\ \dot{\zeta}_{\text{lon}} &= \bar{A}_{\text{lon}}\zeta_{\text{lon}} + \bar{B}_{\text{lon}}u_{\text{lon}}\end{aligned}$$

Our state cost and input cost matrices defined by

$$\begin{aligned}Q_{\text{lat}} &= diag\left([q_v, q_p, q_r, q_\phi, q_\chi, q_{I\chi}]\right)^{T}\\ R_{\text{lat}} &= diag\left([r_{\delta_a}, r_{\delta_r}]\right)^{T}\\ Q_{\text{lon}} &= diag\left([q_u, q_w, q_q, q_\theta, q_h, q_{Ih}, q_{IV_a}]\right)^{T}\\ R_{\text{lon}} &= diag\left([r_{\delta_e}, r_{\delta_t}]\right)^{T}\end{aligned}$$

Implementation of the LQR controller is discussed in the following section.

IV. Simulation Results

The autopilots described in the previous section were implemented and the results are described in the subsections below. The autopilot accepts commands for altitude, airspeed, and course heading. The aircraft starts at an altitude of 0 meters, an airspeed of 25 m/s, and a course heading of 0 degrees. The altitude command instantaneously increases to 15 meters at a time of 0 seconds. The airspeed command instantaneously increases to 28 m/s at a time of 2 seconds. The course command instantaneously changes to 45 degrees at a time of 5 seconds.

The altitude command h^c, airspeed command V_a^c, and course command χ^c functions are expressed below:

$$\begin{aligned}h^c(t) &= \begin{cases}0 & \text{if } t < 0\\ 15 & \text{if } t \geq 0\end{cases}\\ V_a^c(t) &= \begin{cases}25 & \text{if } t < 2\\ 28 & \text{if } t \geq 2\end{cases}\\ \chi^c(t) &= \begin{cases}0 & \text{if } t < 5\\ 45 & \text{if } t \geq 5\end{cases}\end{aligned}$$

The simulations are conducted in the absence of steady-state winds, but low-altitude and low-turbulence gusts are present. This is evident in the plots to follow.

As the scope of this report does not deal with a specific application of an aircraft autopilot, we cannot determine quantitative desired performance characteristics. Rather, we will use a qualitative performance characteristic to determine the value of the controller. Specifically, we will judge whether the true states of altitude, airspeed, and course heading tend towards the reference states in a reasonable and steady fashion. The keyword "reasonable" entails reaching the reference state in minimal time and minimizing the true states' maximum overshoot. The keyword "steady" entails a relatively smooth change from the current state to the commanded state.

An outline of the scripts used for these autopilots can be found in the Appendix in addition to a link to a GitHub repository storing the Python files. The aircraft's physical, aerodynamic, motor, and propeller parameters used in the simulation can be found in the Appendix.

A. *Successive Loop Closure (SLC) Controller*

The design parameters were taken from Beard and McLain's *Small Unmanned Aircraft: Theory and Practice*. The SLC lateral autopilot was implemented with the following design parameters:

Design Parameter	ω_{n_ϕ}	ζ_ϕ	W_χ	ζ_χ	τ_r	k_r
Value	20	$\frac{\sqrt{2}}{2}$	20	1	$\frac{20}{9}$	0.2

The SLC longitudinal autopilot was implemented with the following design parameters:

Design Parameter	ω_{n_θ}	ζ_θ	W_h	ζ_h	ω_{n_V}	ζ_V
Value	24	$\frac{\sqrt{2}}{2}$	30	1	8	2

The results of implementing an SLC autopilot with the design parameters listed above are shown below in Figure 7. The figure shows the true state of the altitude h, airspeed V_a, and course χ graphed against the commanded state.

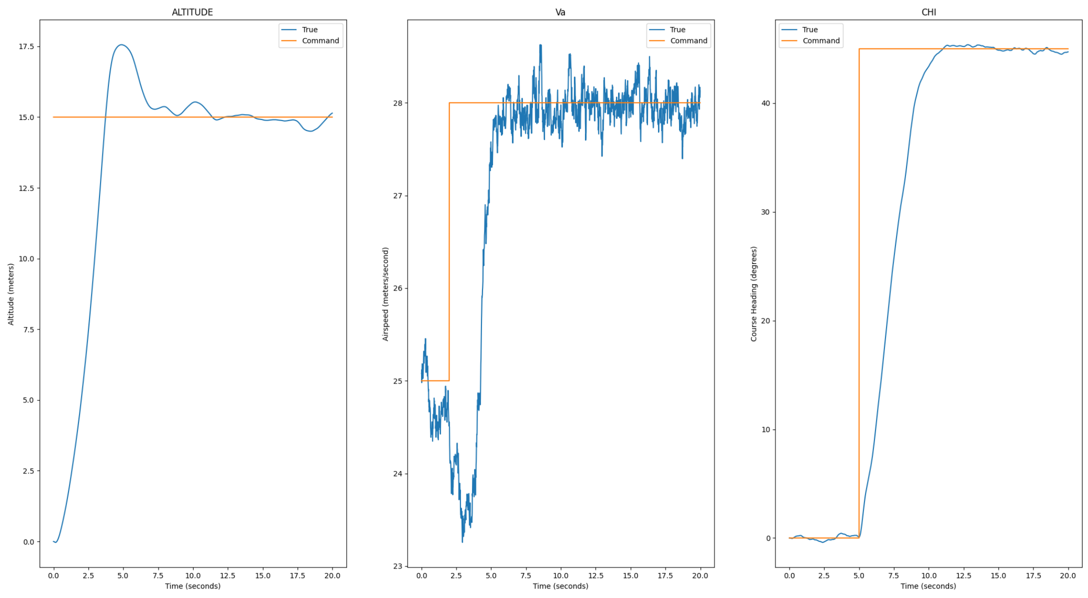


Fig. 7: SLC Autopilot Results - Original Values

It can be seen that all three states tend towards the commanded value as time increases. The true altitude overshoots the commanded state by about 2 meters. The true airspeed lags behind the commanded value until the aircraft reaches the desired altitude. This is expected as both states act in the longitudinal system, and there should be interdependence between the states. It can be seen that the true course heading slowly increases to reach the reference state. The smooth change in course is due to a saturation applied to the course error, ensuring that the absolute error cannot be greater than 15 degrees. A similar saturation is applied to the altitude error, ensuring that the error cannot be greater than ± 2 meters.

A link to a video of this simulation can be found in the Appendix.

We could further tune the design parameters to reduce the overshoot error in the altitude and to have the airspeed reach the reference state faster. However, these would have to be done with respect to quantitative performance characteristics. As seen in the figure, the states tend to the commanded values in a reasonable and steady fashion, therefore our primary performance characteristic was met and fine-tuning is unnecessary in this case.

B. *Linear-Quadratic Regulator (LQR) Controller*

The Q and R matrices for the lateral and longitudinal autopilots presented in Beard and McLain's book are as follows:

$$Q_{\text{lat}} = diag([0.001, 0.01, 0.1, 100, 1, 100])$$
$$R_{\text{lat}} = diag([1, 1])$$
$$Q_{\text{lon}} = diag([10, 10, 0.001, 0.01, 10, 100, 100])$$
$$R_{\text{lon}} = diag([1, 1])$$

A plot of this autopilot's behavior can be seen below in Figure 8.

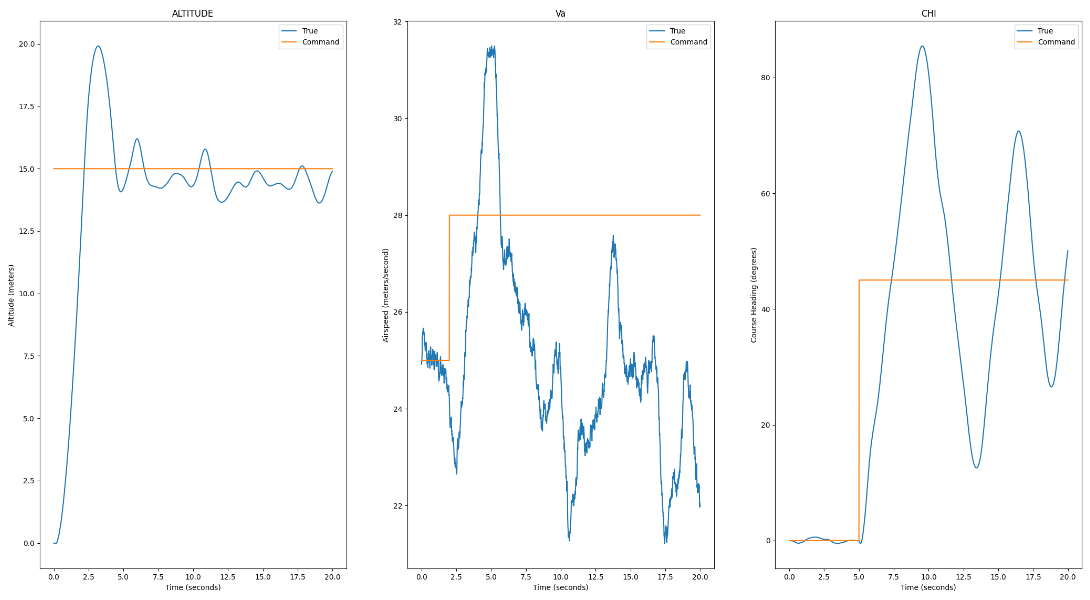


Fig. 8: Graph of LQR Autopilot Results - Original Values

While the true altitude is reasonably close to the reference state, the other two commanded states are not. The airspeed is declining below the initial airspeed and the course heading is oscillating about the reference state with significant error. Comparing Figure 8 with the SLC autopilot result in Figure 7 shows a stark contrast between the two.

The qualitative performance characteristic was not met and thus tuning is required. Tuning the Q and R matrices consisted of observing the aircraft's behavior and adjusting the error weights in Q and input costs in R depending on how the aircraft tended towards the reference state.

For example, in the original configuration of Q and R, aircraft did not bank when turning. Rather, the craft relied on its rudder to turn. Looking at the original Q_{lat}, we see that a relatively large weight is placed the roll error ($q_\phi = 100$). This inhibited the aircraft's ability to bank as the ϕ state was being driven to 0. With a lower value for q_ϕ, the aircraft turns as expected. Similar tuning was done for the aircraft's other states.

As a result, the following matrices were constructed:

$$Q_{\text{lat}} = diag([0.1, 1, 0.1, 1, 10, 0.1])$$
$$R_{\text{lat}} = diag([10, 1])$$
$$Q_{\text{lon}} = diag([10, 10, 0.01, 0.1, 1000, 100, 100])$$
$$R_{\text{lon}} = diag([1, 1])$$

This configuration produces the graph seen in Figure 9 below.

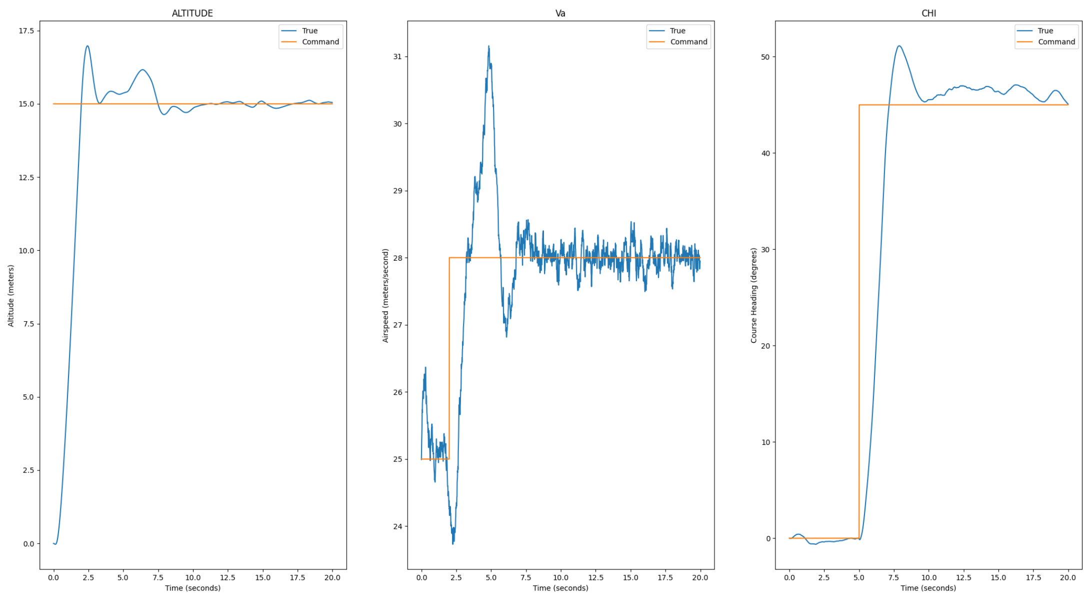


Fig. 9: Graph of LQR Autopilot Results - Iteration 1

We can still see overshooting on all 3 states, with more significant overshooting on the course and airspeed states. This was not seen on the SLC controller. This is likely due to a phenomenon known as integral wind-up error. Integral wind-up error becomes prevalent when a relatively large error is introduced instantaneously. This causes the integral controller to accumulate a significant amount of error, either positive or negative. This causes the controller output to overshoot the reference state to eliminate the accumulated error on the integrator.

As such, to make the LQR results look similar to the SLC autopilot results, the weights on the integrators were set to 0. That is,

$$\begin{aligned} q_{I_\chi} &= 0 \\ q_{I_h} &= 0 \\ q_{I_{V_a}} &= 0 \end{aligned}$$

This effectively removes all integral states from affecting the autopilot. The Q and R matrices are as follows

$$\begin{aligned} Q_{\text{lat}} &= diag([0.1, 1, 0.1, 1, 10, 0]) \\ R_{\text{lat}} &= diag([10, 1]) \\ Q_{\text{lon}} &= diag([10, 10, 0.01, 0.1, 1000, 0, 0]) \\ R_{\text{lon}} &= diag([1, 1]) \end{aligned}$$

The results of this configuration are shown below in Figure 10.

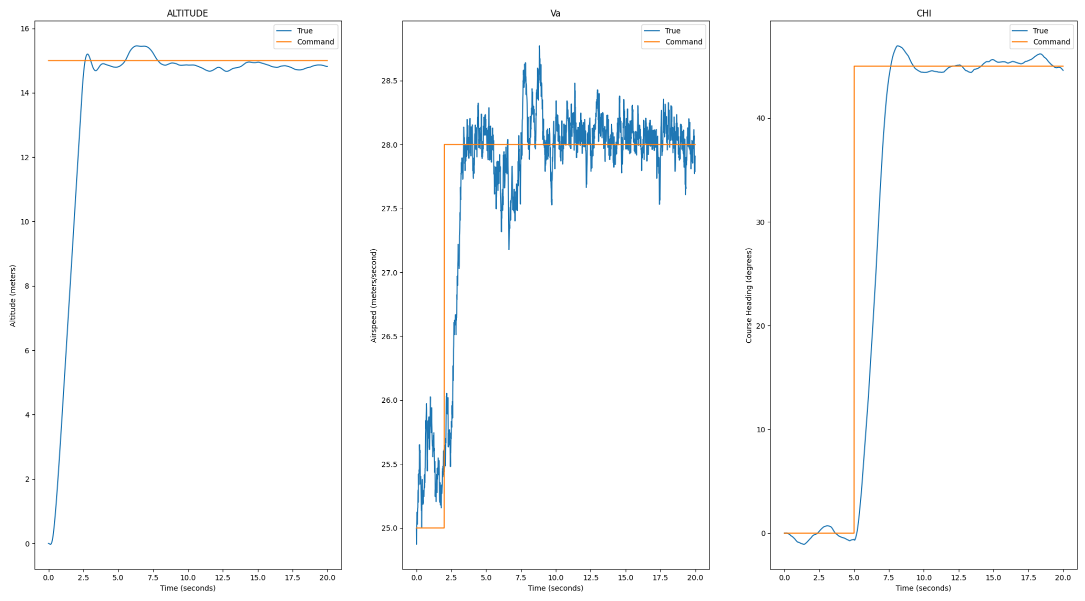


Fig. 10: Graph of LQR Autopilot Results - Iteration 2

As seen in the figure, this autopilot satisfies our desired performance characteristic. All states tend towards the reference state in a reasonable and steady fashion. The plot looks more similar to the SLC state plot in Figure 7 than the first LQR state plot in Figure 8.

A link to a video of this simulation can be found in the Appendix.

This autopilot could be improved by implementing an anti-windup algorithm to minimize errors introduced by the presence of an integral controller. It could be further improved by further augmenting the aircraft's states with a derivative state, thereby implementing a LQR-PID controller.

V. Conclusion

In this report, we produced two autopilot controllers. Neither controller was quantitatively tuned but could be if provided with performance characteristics. The derivation of the equations of motion, and linearization of said equations, were presented. Transfer function models and state space models for the lateral- and longitudinal-directions were derived. The results of the controllers designed were displayed and discussed. As the autopilots are not tuned using quantitative desired performance characteristics, there is room for improvement in the performance of both controllers.

A next step to improve on the designed controllers is to properly implementing the augmented integral states for the LQR autopilot. If implemented correctly, the controller's performance could greatly benefit from the integral controllers as it would help drive any steady state error to 0.

References

[1] R. Beard and T. McLain, *Small Unmanned Aircraft: Theory and Practice*. Princeton, NJ, USA: Princeton University Press, 2012.

[2] R. Beard. “randybeard/uavbook.” Github. https://github.com/randybeard/uavbook (retrieved Aug. 3, 2022).

Appendices

A. Script Outline

Outline

Script Name	Description
autopilot_cmds.py	Message-type file storing commands for the autopilot to follow
autopilot_LQR.py	LQR autopilot algorithm - accepts commanded states and generates actuator commands
autopilot_SLC.py	SLC autopilot algorithm - accepts commanded states and generates actuator commands
chX_project.py	Project file for chapter X - implements other modules
compute_models.py	Computes transfer functions and state space models from chapter 5
control_parameters.py	Calculates and stores controller gains used by the SLC autopilot
delta_state.py	Message-type file storing actuator commands
helper.py	Functions generating rotation matrices, and converting between Euler angles and quaternions
mav_body_parameter.py	Stores MAV's physical and aerodynamic parameters
mav_dynamics.py	Calculates the dynamics of the MAV according to actuator commands and current states
mav_state.py	Message-type file storing the 12 states, and more, of the MAV
mav.py	Accepts the MAV state and generates a 3D model of the aircraft
model_coeff.py	Stores the transfer function and state space models
pd_control_with_rate.py	Generic PD controller if rate data is known
pi_control.py	Generic PI controller
pid_control.py	Generic PID controller
signals.py	Generates a signal according to input parameters (e.g. square wave, step function)
transfer_function.py	Handles transfer functions - used for wind simulation
trim.py	Finds trim condition, with inputs of airspeed, flight path angle, and turning radius
wind_simulation.py	Simulates steady-state winds and gusts
wrap.py	Function ensures course heading is between -180 and 180 degrees

GitHub Repository: https://github.com/VishnuVijay56/AAE497-Summer2022

B. Aircraft Parameters

Physical Parameters

Variable	Description	Value
`m`	Mass	11
`Jx`	Moment of Inertia	0.8244
`Jy`	Moment of Inertia	1.135
`Jz`	Moment of Inertia	1.759
`Jxz`	Product of Inertia	0.1204
`gravity`	Gravity	9.81
`S_wing`	Wing surface area	0.55
`b`	Wingspan	2.8956
`c`	Chord length	0.18994
`S_prop`	Propeller surface area	0.2027
`rho`	Density	1.2682
`e`	Oswald efficiency factor	0.9
`AR`	Aspect ratio	`b**2 / S_wing`

Aerodynamic Parameters

Variable	Value
C_L_0	0.23
C_D_0	0.043
C_m_0	0.0135
C_L_alpha	5.61
C_D_alpha	0.03
C_m_alpha	−2.74
C_L_q	7.95
C_D_q	0.0
C_m_q	−38.21
C_L_delta_e	0.13
C_D_delta_e	0.0135
C_m_delta_e	−0.99
M	50
alpha0	0.47
C_D_p	0
C_Y_0	0
C_l_0	0
C_n_0	0
C_Y_beta	−0.98
C_l_beta	−0.13
C_n_beta	0.073
C_Y_p	0
C_l_p	−0.51
C_n_p	0.069
C_Y_r	0
C_l_r	0.25
C_n_r	−0.095
C_Y_delta_a	0.075
C_l_delta_a	0.17
C_n_delta_a	−0.011
C_Y_delta_r	0.19
C_l_delta_r	0.0024
C_n_delta_r	−0.069

Motor and Propeller Parameters

Variable	Value
D_prop	0.508
KVstar	145
KV	60 / (2 * np.pi * KVstar)
KQ	KV
R_motor	0.042
i0	1.5
ncells	12
V_max	3.7 * ncells
C_Q2	−0.01664
C_Q1	0.004970
C_Q0	0.005230
C_T2	−0.1079
C_T1	−0.06044
C_T0	0.09357

C. Equations

Rotational Dynamics

$$
\begin{aligned}
\Gamma &= J_x J_z - J_{xz}^2 \\
\Gamma_1 &= \frac{J_{xz}(J_x - J_y + J_z}{\Gamma} \\
\Gamma_2 &= \frac{J_z(J_z - J_y) + J_{xz}^2}{\Gamma} \\
\Gamma_3 &= \frac{J_z}{\Gamma} \\
\Gamma_4 &= \frac{J_{xz}}{\Gamma} \\
\Gamma_5 &= \frac{J_z - J_x}{J_y} \\
\Gamma_6 &= \frac{J_{xz}}{J_y} \\
\Gamma_7 &= \frac{(J_x - J_y)J_x + J_{xz}^2}{\Gamma} \\
\Gamma_8 &= \frac{J_x}{\Gamma}
\end{aligned}
$$

Linearization - Disturbances

$$
\begin{aligned}
d_{\phi_1} &= q \sin\phi \tan\theta + r \cos\phi \tan\theta \\
d_{\phi_2} &= \Gamma_1 pq - \Gamma_2 qr + \frac{1}{2}\rho V_a^2 Sb \left[C_{p_0} + C_{p_\beta}\beta - C_{p_p}\frac{b}{2V_a} d_{\phi_1} + C_{p_r}\frac{br}{2V_a} + C_{p_{\delta_r}}\delta_r \right] + \dot{d}_{\phi_1} \\
d_\chi &= \tan\phi - \phi \\
d_\beta &= \frac{1}{V_a}(pq - ru + g\cos\theta\sin\theta) + \frac{\rho V_a S}{2\text{m}}\left[C_{Y_0} + C_{Y_p}\frac{bp}{2V_a} + C_{Y_r}\frac{br}{2V_a} + C_{Y_{\delta_a}}\delta_a \right] \\
d_{\theta_1} &= q(\cos\phi - 1) \\
d_{\theta_2} &= \Gamma_6(r^2 - p^2) + \Gamma_5 pr + \frac{\rho V_a^2 cS}{2J_y}\left[C_{m_0} + C_{m_\alpha}\gamma - C_{m_q}\frac{c}{2V_a} d_{\theta_1} \right] + \dot{d}_{\theta_1} \\
d_h &= u\sin\theta - V_a\theta - v\sin\phi\cos\theta - w\cos\phi\cos\theta \\
d_{V_1} &= -\dot{u}(1 - \cos\beta)\cos\alpha - \dot{w}(1 - \cos\beta)\sin\alpha + \dot{v}\sin\beta \\
d_{V_2} &= (rV_a\cos\alpha - pV_a\sin\alpha)\sin\beta - g\sin\alpha\cos\theta(1 - \cos\phi) + \frac{1}{\text{m}}T_p(\delta_t, V_a)(\cos\alpha - 1) + d_{V_1} \\
d_V &= d_{V_2} + [\text{ linearization error }]
\end{aligned}
$$

D. Simulation Videos

SLC Controller: https://youtu.be/W3AbCgBHNvo

LQR Controller: https://youtu.be/5aT46j5yP6Q