Small Unmanned Aircraft

Small Unmanned Aircraft

Theory and Practice
Supplement

Randal W. Beard
Timothy W. McLain

PRINCETON UNIVERSITY PRESS
PRINCETON AND OXFORD

Contents

Chapter One

Introduction

1.1 SYSTEM ARCHITECTURE

1.2 DESIGN MODELS

1.3 DESIGN PROJECT

Chapter Two

Coordinate Frames

2.1 ROTATION MATRICES

2.2 MAV COORDINATE FRAMES

2.2.1 The inertial frame $\mathcal{F}^i$

2.2.2 The vehicle frame $\mathcal{F}^v$

2.2.3 The vehicle-1 frame $\mathcal{F}^{v1}$

2.2.4 The vehicle-2 frame $\mathcal{F}^{v2}$

2.2.5 The body frame $\mathcal{F}^b$

2.2.6 The stability frame $\mathcal{F}^s$

2.2.7 The wind frame $\mathcal{F}^w$

2.3 AIRSPEED, WIND SPEED, AND GROUND SPEED

MODIFIED MATERIAL: Combining expressions, we can express the airspeed vector body-frame components in terms of the airspeed magnitude, angle of attack, and sideslip angle as

$$\begin{aligned}\mathbf{V}_a^b = \begin{pmatrix} u_r \\ v_r \\ w_r \end{pmatrix} &= \mathcal{R}_w^b \begin{pmatrix} V_a \\ 0 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} \cos\beta\cos\alpha & -\sin\beta\cos\alpha & -\sin\alpha \\ \sin\beta & \cos\beta & 0 \\ \cos\beta\sin\alpha & -\sin\beta\sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} V_a \\ 0 \\ 0 \end{pmatrix},\end{aligned}$$

2.4 THE WIND TRIANGLE

MODIFIED MATERIAL: Similarly, the airspeed vector in the inertial frame can be expressed as

$$\mathbf{V}_a^i = V_a \begin{pmatrix} \cos(\psi+\beta)\cos\gamma_a \\ \sin(\psi+\beta)\cos\gamma_a \\ -\sin\gamma_a \end{pmatrix},$$

where $V_a = \|\mathbf{V}_a\|$. Under the assumption that the sideslip angle is zero, the wind triangle can be expressed in inertial coordinates as

$$V_g \begin{pmatrix} \cos\chi\cos\gamma \\ \sin\chi\cos\gamma \\ -\sin\gamma \end{pmatrix} = V_a \begin{pmatrix} \cos(\psi+\beta)\cos\gamma_a \\ \sin(\psi+\beta)\cos\gamma_a \\ -\sin\gamma_a \end{pmatrix} + \begin{pmatrix} w_n \\ w_e \\ w_d \end{pmatrix}. \tag{2.1}$$

2.5 DIFFERENTIATION OF A VECTOR

2.6 CHAPTER SUMMARY

NOTES AND REFERENCES

2.7 DESIGN PROJECT

Chapter Three

Kinematics and Dynamics

3.1 STATE VARIABLES

3.2 KINEMATICS

3.3 RIGID-BODY DYNAMICS

3.3.1 Rotational Motion

3.4 CHAPTER SUMMARY

NOTES AND REFERENCES

3.5 DESIGN PROJECT

MODIFIED MATERIAL:

3.1 Implement the MAV equations of motion given in equations (**??**) through (**??**). Assume that the inputs to the block are the forces and moments applied to the MAV in the body frame. Changeable parameters should include the mass, the moments and products of inertia, and the initial conditions for each state. Use the parameters given in Appendix E.

3.2 Verify that the equations of motion are correct by individually setting the forces and moments along each axis to a nonzero value and convincing yourself that the motion is appropriate.

3.3 Since J_{xz} is non-zero, there is gyroscopic coupling between roll and yaw. To test your simulation, set J_{xz} to zero and place nonzero moments on l and n and verify that there is no coupling between the roll and yaw axes. Verify that when J_{xz} is not zero, there is coupling between the roll and yaw axes.

Chapter Four

Forces and Moments

4.1 GRAVITATIONAL FORCES

NEW MATERIAL: Using a unit quaternion to represent attitude instead of Euler angles, the gravity force can be expressed as

$$\mathbf{f}_g^b = \mathsf{m}g \begin{pmatrix} 2(e_x e_z - e_y e_0) \\ 2(e_y e_z + e_x e_0) \\ e_z^2 + e_0^2 - e_x^2 - e_y^2 \end{pmatrix}.$$

4.2 AERODYNAMIC FORCES AND MOMENTS

4.2.1 Control Surfaces

MODIFIED MATERIAL: Mathematically, we can convert between ruddervators and rudder-elevator signals as

$$\begin{pmatrix} \delta_e \\ \delta_r \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} \delta_{rr} \\ \delta_{rl} \end{pmatrix}.$$

MODIFIED MATERIAL: Mathematically, we can convert between elevons and aileron-elevator signals with

$$\begin{pmatrix} \delta_e \\ \delta_a \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} \delta_{er} \\ \delta_{el} \end{pmatrix}.$$

4.2.2 Longitudinal Aerodynamics

MODIFIED MATERIAL: The parameter e_{os} is the Oswald efficiency factor, which ranges between 0.8 and 1.0 [1].

4.2.3 Lateral Aerodynamics

MODIFIED MATERIAL: The coefficient C_{Y_0} is the value of the lateral force coefficient C_Y when $\beta = p = r = \delta_a = \delta_r = 0$.

4.2.4 Aerodynamic Coefficients

4.3 PROPULSION FORCES AND MOMENTS

NEW MATERIAL:

This section describes a model for the thrust and torque produced by a motor/propeller pair. The inputs to the thrust and torque model will be the airspeed of the aircraft V_a and the throttle setting $\delta_t \in [0, 1]$. We assume that the thrust and torque vectors produced by the propeller/motor is aligned with the rotation axes of the motor and denote the magnitude of the thrust by T_p and the magnitude of the torque by Q_p.

Based on propeller theory [], the standard model for the thrust and torque produced by a propeller is given by

$$T_p(\Omega_p, C_T) = \frac{\rho D^4}{4\pi^2}\Omega_p^2 C_T$$
$$Q_p(\Omega_p, C_Q) = \frac{\rho D^5}{4\pi^2}\Omega_p^2 C_Q,$$

where ρ is the density of air, D is the propeller diameter, Ω_p is the propeller speed in radians per second, and C_T and C_Q are non-dimensional aerodynamic coefficients. The aerodynamic coefficients are found experimentally and typical plots are shown in figures 4.1 and 4.2, where C_T and C_Q are plotted as a function of the nondimensional advanced ratio

$$J(\Omega_p, V_a) = \frac{2\pi V_a}{\Omega_p D}.$$

As shown in figures 4.1 and 4.2, aerodynamic coefficients can be approximated as quadratic functions of J. Accordingly we have

$$C_T(J) \approx C_{T2}J^2 + C_{T1}J + C_{T0}$$
$$C_Q(J) \approx C_{Q2}J^2 + C_{Q1}J + C_{Q0},$$

where the coefficients C_{T*} and C_{Q*} are unitless coefficients that are determined experimentally from data. The quadratic approximations are shown as solid lines in figures 4.1 and 4.2. Combining these equations, the thrust

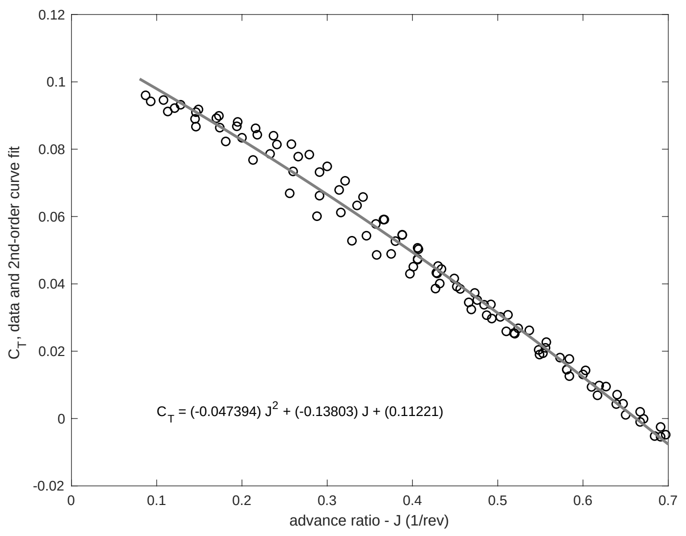


Figure 4.1: Thrust coefficient versus advance ratio for APC Thin Electric 10x5 propeller [**?**].

and torque produced by the propeller are given by

$$\begin{aligned} T_p(\Omega_p, V_a) &= \frac{\rho D^4}{4\pi^2}\Omega_p^2\left(C_{T2}J^2(\Omega_p, V_a) + C_{T1}J(\Omega_p, V_a) + C_{T0}\right) \\ &= \left(\frac{\rho D^4 C_{T0}}{4\pi^2}\right)\Omega_p^2 + \left(\frac{\rho D^3 C_{T1} V_a}{2\pi}\right)\Omega_p + (\rho D^2 C_{T2} V_a^2) \end{aligned} \tag{4.1}$$

$$\begin{aligned} Q_p(\Omega_p, V_a) &= \frac{\rho D^5}{4\pi^2}\Omega_p^2\left(C_{Q2}J^2(\Omega_p, V_a) + C_{Q1}J(\Omega_p, V_a) + C_{Q0}\right) \\ &= \left(\frac{\rho D^5 C_{Q0}}{4\pi^2}\right)\Omega_p^2 + \left(\frac{\rho D^4 C_{Q1} V_a}{2\pi}\right)\Omega_p + (\rho D^3 C_{Q2} V_a^2). \end{aligned} \tag{4.2}$$

The speed of the propeller Ω_p will be determined by the torque applied to the propeller by the motor. For a DC motor, the steady-state torque generated for a given input voltage V_{in} is given by

$$Q_m = K_Q\left[\frac{1}{R}(V_{in} - K_V\Omega_m) - i_0\right], \tag{4.3}$$

where K_Q is the motor torque constant, R is the resistance of the motor windings, K_V is the back-emf voltage constant, Ω_m is the angular speed of the motor, and i_0 is the zero-torque or no-load current. Both K_Q and K_V represent how power is transformed between the mechanical and electrical

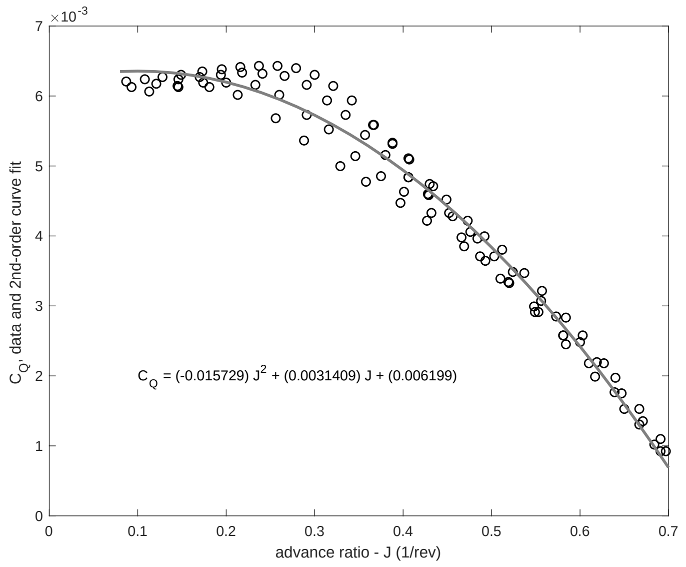


Figure 4.2: Torque coefficient versus advance ratio for APC Thin Electric 10x5 propeller [**?**].

energy domains. If consistent units (such as SI units) are utilized, K_Q and K_T are identical in value.[1] It should be noted that when a voltage change is applied to the motor, the motor changes speed according to the dynamic behavior of the motor. For a fixed-wing aircraft, these transients are significantly faster than the aircraft dynamics and we can neglect them without negatively effecting the accuracy of the model.

When a DC motor drives the propeller we have that $\Omega_p = \Omega_m$ and $Q_m = Q_p$. Therefore, setting equation (4.2) equal to equation (4.3) and grouping like terms gives

$$\left(\frac{\rho D^5}{(2\pi)^2} C_{Q0}\right)\Omega_p^2 + \left(\frac{\rho D^4}{2\pi} C_{Q1} V_a + \frac{K_Q^2}{R}\right)\Omega_p + \left(\rho D^3 C_{Q2} V_a^2 - \frac{K_Q}{R} V_{in} + K_Q i_0\right) = 0. \tag{4.4}$$

[1]We assume K_Q has units of N-m/A and K_V has units of V-sec/rad. In RC aircraft motor datasheets, it is common for K_V to be specified in units of rpm/V.

Denoting the coefficients of this quadratic equation as

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

the positive root given by

$$\Omega_p(V_{in}, V_a) = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \tag{4.5}$$

defines the operating speed of the propeller as a function of airspeed V_a and motor voltage V_{in}.

Using the approach above to find the operating speed, the thrust and torque characteristics of the motor/propeller combination over the full range of voltage and airspeed inputs can be quantified. A family of curves, one for each level of voltage input, is plotted versus airspeed and shows the thrust and torque generated by a specific motor/propeller pair in figures 4.3 and 4.4. Although the voltages are plotted at discrete intervals, the voltage input is assumed to be continuously variable. Figure 4.3 in particular illustrates the full range of thrust performance for the selected motor/propeller pair. As expected, the maximum thrust for a given voltage setting occurs at zero airspeed and the thrust produced goes down as the airspeed increases.

Note that these plots model the windmilling effect that can occur at excessively high advance ratios, where both the thrust and torque become negative and the propeller produces drag instead of thrust. This condition can occur when an aircraft is gliding and descending in a motor-off state and the airspeed is being controlled using the pitch angle of the aircraft.

In this book, we will approximate the input voltage as a linear function of the throttle command, which is roughly true for PWM commands applied to an RC speed controller. Accordingly,

$$V_{in} = V_{\max} \delta_t, \tag{4.6}$$

where $V_{\max}$ is the maximum voltage that can be supplied by the battery.

In summary, the magnitude of the thrust and torque produced by a propeller/motor pair is computed using Algorithm 4.1.

Assuming that the center of the propeller is on the body frame x-axis and that the direction of the propeller is also aligned with the body frame x-axis,

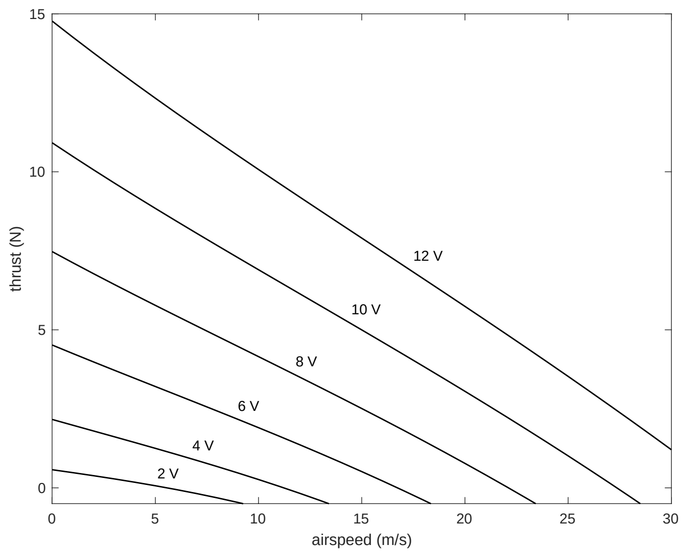


Figure 4.3: Propeller thrust versus airspeed for various motor voltage settings.

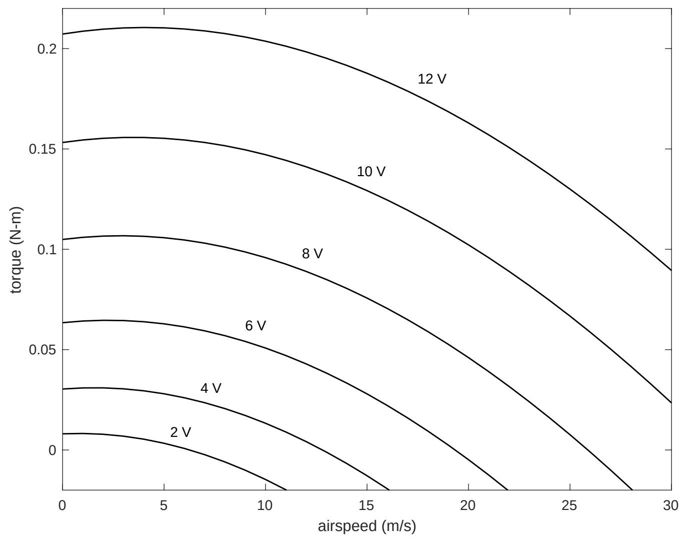


Figure 4.4: Motor torque versus airspeed for various motor voltage settings.

Algorithm 1 Calculate $T_p(\delta_t, V_a)$ and $Q_p(\delta_t, V_a)$

```
    Compute $V_{in}$ using equation (4.6)
    Compute $\Omega_p$ using equation (4.5)
    Compute $T_p$ using equation (4.1)
    Compute $Q_p$ using equation (4.2)
```

then the force and torque vectors due to the propeller are given by

$$\mathbf{f}_p = \begin{pmatrix} T_p, & 0, & 0 \end{pmatrix}^\top$$
$$\mathbf{m}_p = \begin{pmatrix} Q_p, & 0, & 0 \end{pmatrix}^\top .$$

4.4 ATMOSPHERIC DISTURBANCES

MODIFIED MATERIAL: In this section, we will discuss atmospheric disturbances, such as wind, and describe how these disturbances enter into the dynamics of the aircraft. In chapter 2, we defined $\mathbf{V}_g$ as the velocity of the airframe relative to the ground, $\mathbf{V}_a$ as the velocity of the airframe relative to the surrounding air mass, and $\mathbf{V}_w$ as the velocity of the air mass relative to the ground, or in other words, the wind velocity. As shown in equation (**??**), the relationship between ground velocity, air velocity, and wind velocity is given by

$$\mathbf{V}_g = \mathbf{V}_a + \mathbf{V}_w. \tag{4.7}$$

For simulation purposes, we will assume that the total wind vector can be represented as

$$\mathbf{V}_w = \mathbf{V}_{w_s} + \mathbf{V}_{w_g},$$

where $\mathbf{V}_{w_s}$ is a constant vector that represents a steady ambient wind, and $\mathbf{V}_{w_g}$ is a stochastic process that represents wind gusts and other atmospheric disturbances. The ambient (steady) wind is typically expressed in the *inertial* frame as

$$\mathbf{V}^i_{w_s} = \begin{pmatrix} w_{n_s} \\ w_{e_s} \\ w_{d_s} \end{pmatrix},$$

where w_{n_s} is the speed of the steady wind in the north direction, w_{e_s} is the speed of the steady wind in the east direction, and w_{d_s} is the speed of the steady wind in the down direction. The stochastic (gust) component of the wind is typically expressed in the aircraft *body* frame because the atmospheric effects experienced by the aircraft in the direction of its forward motion occur at a higher frequency than do those in the lateral and down directions. The gust portion of the wind can be written in terms of its body-

Table 4.1: Dryden gust model parameters [?].

gust description	altitude (m)	$L_u = L_v$ (m)	L_w (m)	$\sigma_u = \sigma_v$ (m/s)	σ_w (m/s)
low altitude, light turbulence	50	200	50	1.06	0.7
low altitude, moderate turbulence	50	200	50	2.12	1.4
medium altitude, light turbulence	600	533	533	1.5	1.5
medium altitude, moderate turbulence	600	533	533	3.0	3.0

frame components as

$$\mathbf{V}_{w_g}^b = \begin{pmatrix} u_{w_g} \\ v_{w_g} \\ w_{w_g} \end{pmatrix}.$$

Experimental results indicate that a good model for the non-steady gust portion of the wind model is obtained by passing white noise through a linear time-invariant filter given by the von Karmen turbulence spectrum in [2]. Unfortunately, the von Karmen spectrum does not result in a rational transfer function. A suitable approximation of the von Karmen model is given by the Dryden transfer functions

$$H_u(s) = \sigma_u \sqrt{\frac{2V_a}{\pi L_u}} \frac{1}{s + \frac{V_a}{L_u}}$$

$$H_v(s) = \sigma_v \sqrt{\frac{3V_a}{\pi L_v}} \frac{\left(s + \frac{V_a}{\sqrt{3}L_v}\right)}{\left(s + \frac{V_a}{L_v}\right)^2}$$

$$H_w(s) = \sigma_w \sqrt{\frac{3V_a}{\pi L_w}} \frac{\left(s + \frac{V_a}{\sqrt{3}L_w}\right)}{\left(s + \frac{V_a}{L_w}\right)^2},$$

where σ_u, σ_v, and σ_w are the intensities of the turbulence along the vehicle frame axes; and L_u, L_v, and L_w are spatial wavelengths; and V_a is the airspeed of the vehicle. The Dryden models are typically implemented assuming a constant nominal airspeed V_{a_0}. The parameters for the Dryden gust model are defined in MIL-F-8785C. Suitable parameters for low and medium altitudes and light and moderate turbulence were presented in [?] and are shown in Table 4.1.

Figure 4.5 shows how the steady wind and atmospheric disturbance components enter into the equations of motion. White noise is passed through the Dryden filters to produce the gust components expressed in the vehicle frame. The steady components of the wind are rotated from the inertial frame into the body frame and added to the gust components to produce the total wind in the body frame. The combination of steady and gust terms can be expressed mathematically as

$$\mathbf{V}_w^b = \begin{pmatrix} u_w \\ v_w \\ w_w \end{pmatrix} = \mathcal{R}_v^b(\phi,\theta,\psi)\begin{pmatrix} w_{n_s} \\ w_{e_s} \\ w_{d_s} \end{pmatrix} + \begin{pmatrix} u_{w_g} \\ v_{w_g} \\ w_{w_g} \end{pmatrix},$$

where $\mathcal{R}_v^b$ is the rotation matrix from the vehicle to the body frame given in equation (**??**). From the components of the wind velocity $\mathbf{V}_w^b$ and the

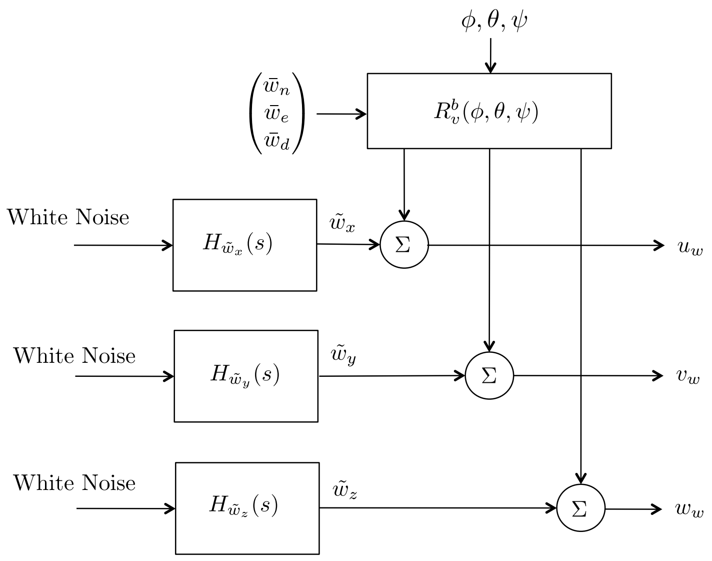


Figure 4.5: The wind is modeled as a constant wind field plus turbulence. The turbulence is generated by filtering white noise with a Dryden model.

ground velocity $\mathbf{V}_g^b$, we can calculate the body-frame components of the airspeed vector as

$$\mathbf{V}_a^b = \begin{pmatrix} u_r \\ v_r \\ w_r \end{pmatrix} = \begin{pmatrix} u - u_w \\ v - v_w \\ w - w_w \end{pmatrix}.$$

From the body-frame components of the airspeed vector, we can calculate the airspeed magnitude, the angle of attack, and the sideslip angle according to equation (**??**) as

$$\begin{aligned}
V_a &= \sqrt{u_r^2 + v_r^2 + w_r^2} \\
\alpha &= \tan^{-1}\left(\frac{w_r}{u_r}\right) \\
\beta &= \sin^{-1}\left(\frac{v_r}{\sqrt{u_r^2 + v_r^2 + w_r^2}}\right).
\end{aligned}$$

These expressions for V_a, α, and β are used to calculate the aerodynamic forces and moments acting on the vehicle. The key point to understand is that wind and atmospheric disturbances affect the airspeed, the angle of attack, and the sideslip angle. It is through these parameters that wind and atmospheric effects enter the calculation of the aerodynamic forces and moments and thereby influence the motion of the aircraft.

Note that to implement the system

$$Y(s) = \frac{as+b}{s^2+cs+d}U(s),$$

first put the system into control canonical form

$$\begin{aligned}
\dot{x} &= \begin{pmatrix} -c & -d \\ 1 & 0 \end{pmatrix} x + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u \\
y &= \begin{pmatrix} a & b \end{pmatrix} x,
\end{aligned}$$

and then convert to discrete time using

$$\begin{aligned}
x_{k+1} &= x_k + T_s(Ax_k + Bu_k) \\
y_k &= Cx_k
\end{aligned}$$

where T_s is the sample rate, to get

$$\begin{aligned}
x_{k+1} &= \begin{pmatrix} 1 - T_s c & -T_s d \\ T_s & 1 \end{pmatrix} x_k + \begin{pmatrix} T_s \\ 0 \end{pmatrix} u_k \\
y_k &= \begin{pmatrix} a & b \end{pmatrix} x_k.
\end{aligned}$$

For the Dryden model gust models, the input u_k will be zero mean Gaussian noise with unity variance.

Python code that implements the transfer function above is given as follows:

```
import numpy as np

class transfer_function:
    def __init__(self, Ts):
        self.ts = Ts
        # set initial conditions
        self._state = np.array([[0.0], [0.0]])
        # define state space model
        self._A = np.array([[1-Ts*c, -Ts*d], [Ts, 1]])
        self._B = np.array([[Ts], [0]])
        self._C = np.array([[a, b]])

    def update(self, u):
        '''Update state space model'''
        self._state = self._A @ self._state + self._B * u
        y = self._C @ self._state
        return y

# initialize the system
Ts = 0.01  # simulation step size
system = transfer_function(Ts)

# main simulation loop
sim_time = 0.0
while sim_time < 10.0:
    u=np.random.randn()  # (white noise)
    y = system.update(u)  # update based on current input
    sim_time += Ts   # increment the simulation time
```

4.5 CHAPTER SUMMARY

MODIFIED MATERIAL: The total forces on the MAV can be summarized as follows:

$$\begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} = \begin{pmatrix} -\mathsf{m}g\sin\theta \\ \mathsf{m}g\cos\theta\sin\phi \\ \mathsf{m}g\cos\theta\cos\phi \end{pmatrix} + \begin{pmatrix} T_p \\ 0 \\ 0 \end{pmatrix}$$
$$+ \frac{1}{2}\rho V_a^2 S \begin{pmatrix} C_X(\alpha) + C_{X_q}(\alpha)\frac{c}{2V_a}q \\ C_{Y_0} + C_{Y_\beta}\beta + C_{Y_p}\frac{b}{2V_a}p + C_{Y_r}\frac{b}{2V_a}r \\ C_Z(\alpha) + C_{Z_q}(\alpha)\frac{c}{2V_a}q \end{pmatrix}$$
$$+ \frac{1}{2}\rho V_a^2 S \begin{pmatrix} C_{X_{\delta_e}}(\alpha)\delta_e \\ C_{Y_{\delta_a}}\delta_a + C_{Y_{\delta_r}}\delta_r \\ C_{Z_{\delta_e}}(\alpha)\delta_e \end{pmatrix}, \quad (4.8)$$

where

$$\begin{aligned}
C_X(\alpha) &\stackrel{\triangle}{=} -C_D(\alpha)\cos\alpha + C_L(\alpha)\sin\alpha \\
C_{X_q}(\alpha) &\stackrel{\triangle}{=} -C_{D_q}\cos\alpha + C_{L_q}\sin\alpha \\
C_{X_{\delta_e}}(\alpha) &\stackrel{\triangle}{=} -C_{D_{\delta_e}}\cos\alpha + C_{L_{\delta_e}}\sin\alpha \\
C_Z(\alpha) &\stackrel{\triangle}{=} -C_D(\alpha)\sin\alpha - C_L(\alpha)\cos\alpha \\
C_{Z_q}(\alpha) &\stackrel{\triangle}{=} -C_{D_q}\sin\alpha - C_{L_q}\cos\alpha \\
C_{Z_{\delta_e}}(\alpha) &\stackrel{\triangle}{=} -C_{D_{\delta_e}}\sin\alpha - C_{L_{\delta_e}}\cos\alpha,
\end{aligned} \tag{4.9}$$

and where $C_L(\alpha)$ is given by equation (**??**) and $C_D(\alpha)$ is given by equation (**??**). The subscripts X and Z denote that the forces act in the X and Z directions in the body frame, which correspond to the directions of the $\mathbf{i}^b$ and the $\mathbf{k}^b$ vectors.

The total torques on the MAV can be summarized as follows:

$$\begin{aligned}
\begin{pmatrix} l \\ m \\ n \end{pmatrix} &= \frac{1}{2}\rho V_a^2 S \begin{pmatrix} b\left[C_{l_0} + C_{l_\beta}\beta + C_{l_p}\frac{b}{2V_a}p + C_{l_r}\frac{b}{2V_a}r\right] \\ c\left[C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{c}{2V_a}q\right] \\ b\left[C_{n_0} + C_{n_\beta}\beta + C_{n_p}\frac{b}{2V_a}p + C_{n_r}\frac{b}{2V_a}r\right] \end{pmatrix} \\
&\quad + \frac{1}{2}\rho V_a^2 S \begin{pmatrix} b\left[C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r\right] \\ c\left[C_{m_{\delta_e}}\delta_e\right] \\ b\left[C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r\right] \end{pmatrix} + \begin{pmatrix} Q_p \\ 0 \\ 0 \end{pmatrix}.
\end{aligned} \tag{4.10}$$

NOTES AND REFERENCES

4.6 DESIGN PROJECT

MODIFIED MATERIAL:

4.1 Add simulation of the wind to the mavsim simulator. The wind element should produce wind gust along the body axes, and steady state wind along the NED inertial axes.

4.2 Add forces and moments to the dynamics of the MAV. The inputs to the MAV should now be elevator, throttle, aileron, and rudder. The aerodynamic coefficients are given in Appendix E.

4.3 Verify your simulation by setting the control surface deflections to different values. Observe the response of the MAV. Does it behave as you think it should?

Chapter Five

Linear Design Models

5.1 SUMMARY OF NONLINEAR EQUATIONS OF MOTION

MODIFIED MATERIAL: A variety of models for the aerodynamic forces and moments appear in the literature ranging from linear, uncoupled models to highly nonlinear models with significant cross coupling. In this section, we summarize the six-degree-of-freedom, 12-state equations of motion with the quasi-linear aerodynamic and propulsion models developed in chapter 4. We characterize them as quasi-linear because the lift and drag terms are nonlinear in the angle of attack, and the propeller thrust is nonlinear in the throttle command. For completeness, we will also present the linear models for lift and drag that are commonly used. Incorporating the aerodynamic and propulsion models described in chapter 4 into equations (**??**)-(**??**), we get the following equations of motion:

$$\begin{aligned}
\dot{p}_n &= (\cos\theta\cos\psi)u + (\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi)v \\
&\quad + (\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi)w && (5.1)\\
\dot{p}_e &= (\cos\theta\sin\psi)u + (\sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi)v \\
&\quad + (\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi)w && (5.2)\\
\dot{h} &= u\sin\theta - v\sin\phi\cos\theta - w\cos\phi\cos\theta && (5.3)\\
\dot{u} &= rv - qw - g\sin\theta + \frac{\rho V_a^2 S}{2\mathsf{m}}\Big[C_X(\alpha) + C_{X_q}(\alpha)\frac{cq}{2V_a} \\
&\quad + C_{X_{\delta_e}}(\alpha)\delta_e\Big] + \frac{T_p(\delta_t, V_a)}{\mathsf{m}} && (5.4)\\
\dot{v} &= pw - ru + g\cos\theta\sin\phi + \frac{\rho V_a^2 S}{2\mathsf{m}}\Big[C_{Y_0} + C_{Y_\beta}\beta \\
&\quad + C_{Y_p}\frac{bp}{2V_a} + C_{Y_r}\frac{br}{2V_a} + C_{Y_{\delta_a}}\delta_a + C_{Y_{\delta_r}}\delta_r\Big] && (5.5)\\
\dot{w} &= qu - pv + g\cos\theta\cos\phi + \frac{\rho V_a^2 S}{2\mathsf{m}}\Big[C_Z(\alpha) \\
&\quad + C_{Z_q}(\alpha)\frac{cq}{2V_a} + C_{Z_{\delta_e}}(\alpha)\delta_e\Big] && (5.6)\\
\dot{\phi} &= p + q\sin\phi\tan\theta + r\cos\phi\tan\theta && (5.7)\\
\dot{\theta} &= q\cos\phi - r\sin\phi && (5.8)\\
\dot{\psi} &= q\sin\phi\sec\theta + r\cos\phi\sec\theta && (5.9)\\
\dot{p} &= \Gamma_1 pq - \Gamma_2 qr + \frac{1}{2}\rho V_a^2 Sb\Big[C_{p_0} + C_{p_\beta}\beta + C_{p_p}\frac{bp}{2V_a} \\
&\quad + C_{p_r}\frac{br}{2V_a} + C_{p_{\delta_a}}\delta_a + C_{p_{\delta_r}}\delta_r\Big] && (5.10)\\
\dot{q} &= \Gamma_5 pr - \Gamma_6(p^2 - r^2) + \frac{\rho V_a^2 Sc}{2J_y}\Big[C_{m_0} + C_{m_\alpha}\alpha \\
&\quad + C_{m_q}\frac{cq}{2V_a} + C_{m_{\delta_e}}\delta_e\Big] && (5.11)\\
\dot{r} &= \Gamma_7 pq - \Gamma_1 qr + \frac{1}{2}\rho V_a^2 Sb\Big[C_{r_0} + C_{r_\beta}\beta + C_{r_p}\frac{bp}{2V_a} \\
&\quad + C_{r_r}\frac{br}{2V_a} + C_{r_{\delta_a}}\delta_a + C_{r_{\delta_r}}\delta_r\Big], && (5.12)
\end{aligned}$$

MODIFIED MATERIAL: Further, it is common to model drag as a nonlinear quadratic function of the lift as

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

where e_{os} is the Oswald efficiency factor and AR is the aspect ratio of the wing.

5.2 COORDINATED TURN

MODIFIED MATERIAL: The centrifugal force is calculated using the angular rate $\dot{\chi}$ about the inertial frame $\mathbf{k}^i$ axis and the horizontal component of the groundspeed, $V_g \cos\gamma$.

5.3 TRIM CONDITIONS

MODIFIED MATERIAL: For fixed-wing aircraft, the states are given by

$$x \overset{\triangle}{=} (p_n, p_e, p_d, u, v, w, \phi, \theta, \psi, p, q, r)^\top \tag{5.13}$$

and the inputs are given by

$$u \overset{\triangle}{=} (\delta_e, \delta_a, \delta_r, \delta_t)^\top, \tag{5.14}$$

5.4 TRANSFER FUNCTION MODELS

5.4.1 Lateral Transfer Functions

5.4.2 Longitudinal Transfer Functions

MODIFIED MATERIAL: To complete the longitudinal models, we derive the transfer functions from throttle and pitch angle to airspeed. Toward that objective, note that if wind speed is zero, then $V_a = \sqrt{u^2 + v^2 + w^2}$, which implies that

$$\dot{V}_a = \frac{u\dot{u} + v\dot{v} + w\dot{w}}{V_a}.$$

Using equation (**??**), we get

$$\begin{aligned} \dot{V}_a &= \dot{u}\cos\alpha\cos\beta + \dot{v}\sin\beta + \dot{w}\sin\alpha\cos\beta \\ &= \dot{u}\cos\alpha + \dot{w}\sin\alpha + d_{V_1}, \end{aligned} \tag{5.15}$$

where

$$d_{V_1} = -\dot{u}(1-\cos\beta)\cos\alpha - \dot{w}(1-\cos\beta)\sin\alpha + \dot{v}\sin\beta.$$

Note that when $\beta = 0$, we have $d_{V1} = 0$. Substituting equations (5.4) and (5.6) in equation (5.15), we obtain

$$\begin{aligned}\dot{V}_a = \cos\alpha \Big\{ & rv - qw + r - g\sin\theta \\ & + \frac{\rho V_a^2 S}{2\mathsf{m}}\Big[-C_D(\alpha)\cos\alpha + C_L(\alpha)\sin\alpha + (-C_{D_q}\cos\alpha + C_{L_q}\sin\alpha)\frac{cq}{2V_a} \\ & + (-C_{D_{\delta_e}}\cos\alpha + C_{L_{\delta_e}}\sin\alpha)\delta_e \Big] + \frac{1}{\mathsf{m}}T_p(\delta_t, V_a) \Big\} \\ & + \sin\alpha \Big\{ qu_r - pv_r + g\cos\theta\cos\phi \\ & + \frac{\rho V_a^2 S}{2\mathsf{m}}\Big[-C_D(\alpha)\sin\alpha - C_L(\alpha)\cos\alpha + (-C_{D_q}\sin\alpha - C_{L_q}\cos\alpha)\frac{cq}{2V_a} \\ & + (-C_{D_{\delta_e}}\sin\alpha - C_{L_{\delta_e}}\cos\alpha)\delta_e \Big] \Big\} + d_{V_1}\end{aligned}$$

where $T_p(\delta_t, V_a)$ is the thrust produced by the motor. Using equations (**??**) and the linear approximation $C_D(\alpha) \approx C_{D_0} + C_{D_\alpha}\alpha$, and simplifying, we get

$$
\begin{aligned}
\dot{V}_a &= rV_a\cos\alpha\sin\beta - pV_a\sin\alpha\sin\beta \\
&\quad - g\cos\alpha\sin\theta + g\sin\alpha\cos\theta\cos\phi \\
&\quad + \frac{\rho V_a^2 S}{2\mathsf{m}}\left[-C_{D_0} - C_{D_\alpha}\alpha - C_{D_q}\frac{cq}{2V_a} - C_{D_{\delta_e}}\delta_e\right] \\
&\quad + \frac{1}{\mathsf{m}}T_p(\delta_t, V_a)\cos\alpha + d_{V_1} \\
&= (rV_a\cos\alpha - pV_a\sin\alpha)\sin\beta \\
&\quad - g\sin(\theta-\alpha) - g\sin\alpha\cos\theta(1-\cos\phi) \\
&\quad + \frac{\rho V_a^2 S}{2\mathsf{m}}\left[-C_{D_0} - C_{D_\alpha}\alpha - C_{D_q}\frac{cq}{2V_a} - C_{D_{\delta_e}}\delta_e\right] \\
&\quad + \frac{1}{\mathsf{m}}T_p(\delta_t, V_a)\cos\alpha + d_{V_1} \\
&= -g\sin\gamma + \frac{\rho V_a^2 S}{2\mathsf{m}}\left[-C_{D_0} - C_{D_\alpha}\alpha - C_{D_q}\frac{cq}{2V_a} - C_{D_{\delta_e}}\delta_e\right] \\
&\quad + \frac{1}{\mathsf{m}}T_p(\delta_t, V_a) + d_{V_2}, \qquad (5.16)
\end{aligned}
$$

where

$$
\begin{aligned}
d_{V_2} = (rV_a\cos\alpha - pV_a\sin\alpha)\sin\beta - g\sin\alpha\cos\theta(1-\cos\phi) \\
+ \frac{1}{\mathsf{m}}T_p(\delta_t, V_a)(\cos\alpha - 1) + d_{V_1}.
\end{aligned}
$$

Again note that in level flight $d_{V_2} \approx 0$.

When considering airspeed V_a, there are two inputs of interest: the throttle setting δ_t and the pitch angle θ. Since equation (5.16) is nonlinear in V_a and δ_t, we must first linearize before we can find the desired transfer functions. Following the approach outlined in Section 5.5.1, we can linearize equation (5.16) by letting $\bar{V}_a \stackrel{\triangle}{=} V_a - V_a^*$ be the deviation of V_a from trim, $\bar{\theta} \stackrel{\triangle}{=} \theta - \theta^*$ be the deviation of θ from trim, and $\bar{\delta}_t \stackrel{\triangle}{=} \delta_t - \delta_t^*$ be the deviation of the throttle from trim. Equation (5.16) can then be linearized around the wings-level, constant-altitude ($\gamma^* = 0$) trim condition to give

$$\begin{aligned}
\dot{\bar{V}}_a &= -g\cos(\theta^* - \alpha^*)\bar{\theta} \qquad (5.17)\\
&\quad + \left\{ \frac{\rho V_a^* S}{\mathsf{m}} \left[-C_{D_0} - C_{D_\alpha}\alpha^* - C_{D_{\delta_e}}\delta_e^* \right] + \frac{1}{\mathsf{m}}\frac{\partial T_p}{\partial V_a}(\delta_t^*, V_a^*) \right\} \bar{V}_a \\
&\quad\quad + \frac{1}{\mathsf{m}}\frac{\partial T_p}{\partial \delta_t}(\delta_t^*, V_a^*)\bar{\delta}_t + d_V \\
&= -a_{V_1}\bar{V}_a + a_{V_2}\bar{\delta}_t - a_{V_3}\bar{\theta} + d_V, \qquad (5.18)
\end{aligned}$$

where

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

and d_V includes d_{V_2} as well as the linearization error.

5.5 LINEAR STATE-SPACE MODELS

5.5.1 Linearization

5.5.2 Lateral State-Space Equations

MODIFIED MATERIAL:

5.5.3 Longitudinal State-Space Equations

MODIFIED MATERIAL: Assuming that the lateral states are zero (i.e., $\phi = p = r = \beta = v = 0$) and the windspeed is zero and substituting

$$\begin{aligned}
\alpha &= \tan^{-1}\left(\frac{w}{u}\right) \\
V_a &= \sqrt{u^2 + w^2}
\end{aligned}$$

Table 5.1: Lateral State-Space Model Coefficients.

Lateral	Formula
Y_v	$\frac{\rho S b v^*}{4\mathsf{m} V_a^*}\left[C_{Y_p} p^* + C_{Y_r} r^*\right]$
	$+\frac{\rho S v^*}{\mathsf{m}}\left[C_{Y_0} + C_{Y_\beta}\beta^* + C_{Y_{\delta_a}}\delta_a^* + C_{Y_{\delta_r}}\delta_r^*\right]$
	$+\frac{\rho S C_{Y_\beta}}{2\mathsf{m}}\sqrt{u^{*2} + w^{*2}}$
Y_p	$w^* + \frac{\rho V_a^* S b}{4\mathsf{m}} C_{Y_p}$
Y_r	$-u^* + \frac{\rho V_a^* S b}{4\mathsf{m}} C_{Y_r}$
Y_{δ_a}	$\frac{\rho V_a^{*2} S}{2\mathsf{m}} C_{Y_{\delta_a}}$
Y_{δ_r}	$\frac{\rho V_a^{*2} S}{2\mathsf{m}} C_{Y_{\delta_r}}$
L_v	$\frac{\rho S b^2 v^*}{4 V_a^*}\left[C_{p_p} p^* + C_{p_r} r^*\right]$
	$+\rho S b v^*\left[C_{p_0} + C_{p_\beta}\beta^* + C_{p_{\delta_a}}\delta_a^* + C_{p_{\delta_r}}\delta_r^*\right]$
	$+\frac{\rho S b C_{p_\beta}}{2}\sqrt{u^{*2} + w^{*2}}$
L_p	$\Gamma_1 q^* + \frac{\rho V_a^* S b^2}{4} C_{p_p}$
L_r	$-\Gamma_2 q^* + \frac{\rho V_a^* S b^2}{4} C_{p_r}$
L_{δ_a}	$\frac{\rho V_a^{*2} S b}{2} C_{p_{\delta_a}}$
L_{δ_r}	$\frac{\rho V_a^{*2} S b}{2} C_{p_{\delta_r}}$
N_v	$\frac{\rho S b^2 v^*}{4 V_a^*}\left[C_{r_p} p^* + C_{r_r} r^*\right]$
	$+\rho S b v^*\left[C_{r_0} + C_{r_\beta}\beta^* + C_{r_{\delta_a}}\delta_a^* + C_{r_{\delta_r}}\delta_r^*\right]$
	$+\frac{\rho S b C_{r_\beta}}{2}\sqrt{u^{*2} + w^{*2}}$
N_p	$\Gamma_7 q^* + \frac{\rho V_a^* S b^2}{4} C_{r_p}$
N_r	$-\Gamma_1 q^* + \frac{\rho V_a^* S b^2}{4} C_{r_r}$
N_{δ_a}	$\frac{\rho V_a^{*2} S b}{2} C_{r_{\delta_a}}$
N_{δ_r}	$\frac{\rho V_a^{*2} S b}{2} C_{r_{\delta_r}}$
A_{14}	$g\cos\theta^*\cos\phi^*$
A_{43}	$\cos\phi^*\tan\theta^*$
A_{44}	$q^*\cos\phi^*\tan\theta^* - r^*\sin\phi^*\tan\theta^*$
A_{53}	$\cos\phi^*\sec\theta^*$
A_{54}	$p^*\cos\phi^*\sec\theta^* - r^*\sin\phi^*\sec\theta^*$

from equation (**??**) gives

$$\begin{aligned}
\dot{u} &= -qw - g\sin\theta + \frac{\rho(u^2+w^2)S}{2\mathsf{m}}\left[C_{X_0} + C_{X_\alpha}\tan^{-1}\left(\frac{w}{u}\right) + C_{X_{\delta_e}}\delta_e\right] \\
&\quad + \frac{\rho\sqrt{u^2+w^2}S}{4\mathsf{m}}C_{X_q}cq + T_p(\delta_t, \sqrt{u^2+w^2}) && (5.19) \\
\dot{w} &= qu + g\cos\theta + \frac{\rho(u^2+w^2)S}{2\mathsf{m}}\left[C_{Z_0} + C_{Z_\alpha}\tan^{-1}\left(\frac{w}{u}\right) + C_{Z_{\delta_e}}\delta_e\right] \\
&\quad + \frac{\rho\sqrt{u^2+w^2}S}{4\mathsf{m}}C_{Z_q}cq && (5.20) \\
\dot{q} &= \frac{1}{2J_y}\rho(u^2+w^2)cS\left[C_{m_0} + C_{m_\alpha}\tan^{-1}\left(\frac{w}{u}\right) + C_{m_{\delta_e}}\delta_e\right] \\
&\quad + \frac{1}{4J_y}\rho\sqrt{u^2+w^2}SC_{m_q}c^2q && (5.21) \\
\dot{\theta} &= q && (5.22) \\
\dot{h} &= u\sin\theta - w\cos\theta. && (5.23)
\end{aligned}$$

MODIFIED MATERIAL:

5.5.4 Reduced-order Modes

5.5.4.1 Short-period Mode

5.5.4.2 Phugoid Mode

5.5.4.3 Roll Mode

5.5.4.4 Spiral-divergence Mode

5.5.4.5 Dutch-roll Mode

NEW MATERIAL: Using the fact that for the state-space equations

$$\begin{aligned}
\dot{x} &= Ax + Bu \\
y &= Cx
\end{aligned}$$

the associated transfer function is

$$Y(s) = C(sI - A)^{-1}BU(s),$$

Table 5.2: Longitudinal State-Space Model Coefficients.

Longitudinal	Formula
X_u	$\frac{u^*\rho S}{\mathsf{m}}\left[C_{X_0}+C_{X_\alpha}\alpha^*+C_{X_{\delta_e}}\delta_e^*\right] -\frac{\rho S w^* C_{X_\alpha}}{2\mathsf{m}}+\frac{\rho S c C_{X_q} u^* q^*}{4\mathsf{m}V_a^*}+\frac{\partial T_p}{\partial u}(\delta_t^*, V_a^*)$
X_w	$-q^*+\frac{w^*\rho S}{\mathsf{m}}\left[C_{X_0}+C_{X_\alpha}\alpha^*+C_{X_{\delta_e}}\delta_e^*\right] +\frac{\rho S c C_{X_q} w^* q^*}{4\mathsf{m}V_a^*}+\frac{\rho S C_{X_\alpha} u^*}{2\mathsf{m}}+\frac{\partial T_p}{\partial w}(\delta_t^*, V_a^*)$
X_q	$-w^*+\frac{\rho V_a^* S C_{X_q} c}{4\mathsf{m}}$
X_{δ_e}	$\frac{\rho V_a^{*2} S C_{X_{\delta_e}}}{2\mathsf{m}}$
X_{δ_t}	$\frac{\partial T_p}{\partial \delta_t}(\delta_t^*, V_a^*)$
Z_u	$q^*+\frac{u^*\rho S}{\mathsf{m}}\left[C_{Z_0}+C_{Z_\alpha}\alpha^*+C_{Z_{\delta_e}}\delta_e^*\right] -\frac{\rho S C_{Z_\alpha} w^*}{2\mathsf{m}}+\frac{u^*\rho S C_{Z_q} c q^*}{4\mathsf{m}V_a^*}$
Z_w	$\frac{w^*\rho S}{\mathsf{m}}\left[C_{Z_0}+C_{Z_\alpha}\alpha^*+C_{Z_{\delta_e}}\delta_e^*\right] +\frac{\rho S C_{Z_\alpha} u^*}{2\mathsf{m}}+\frac{\rho w^* S c C_{Z_q} q^*}{4\mathsf{m}V_a^*}$
Z_q	$u^*+\frac{\rho V_a^* S C_{Z_q} c}{4\mathsf{m}}$
Z_{δ_e}	$\frac{\rho V_a^{*2} S C_{Z_{\delta_e}}}{2\mathsf{m}}$
M_u	$\frac{u^*\rho S c}{J_y}\left[C_{m_0}+C_{m_\alpha}\alpha^*+C_{m_{\delta_e}}\delta_e^*\right] -\frac{\rho S c C_{m_\alpha} w^*}{2J_y}+\frac{\rho S c^2 C_{m_q} q^* u^*}{4J_y V_a^*}$
M_w	$\frac{w^*\rho S c}{J_y}\left[C_{m_0}+C_{m_\alpha}\alpha^*+C_{m_{\delta_e}}\delta_e^*\right] +\frac{\rho S c C_{m_\alpha} u^*}{2J_y}+\frac{\rho S c^2 C_{m_q} q^* w^*}{4J_y V_a^*}$
M_q	$\frac{\rho V_a^* S c^2 C_{m_q}}{4J_y}$
M_{δ_e}	$\frac{\rho V_a^{*2} S c C_{m_{\delta_e}}}{2J_y}$

and letting $C = \begin{pmatrix} 0, & 1 \end{pmatrix}$, the transfer function from $\bar{\delta}_r$ to $\bar{r}$ is given by

$$\begin{aligned}\bar{r}(s) &= \begin{pmatrix} 0 & 1 \end{pmatrix} \left(sI - \begin{pmatrix} Y_v & \frac{Y_r}{V_a^* \cos\beta^*} \\ N_v V_a^* \cos\beta^* & N_r \end{pmatrix} \right)^{-1} \begin{pmatrix} \frac{Y_{\delta_r}}{V_a^* \cos\beta^*} \\ N_{\delta_r} \end{pmatrix} \bar{\delta}_r(s) \\ &= \left(\frac{N_{\delta_r} s + (Y_{\delta_r} N_v - N_{\delta_r} Y_v)}{s^2 - (Y_v + N_r)s + (Y_v N_r - Y_r N_v)} \right) \bar{\delta}_r(s). \end{aligned} \tag{5.24}$$

5.6 CHAPTER SUMMARY

NOTES AND REFERENCES

5.7 DESIGN PROJECT

MODIFIED MATERIAL:

5.1 Create a function that computes the trim state and the trim inputs for a desired airspeed of V_a and a desired flight path angle of $\pm\gamma$. Set the initial condition in your simulation to the trim state and trim input, and verify that the aircraft maintains trim until numerical errors cause it to drift.

5.2 Create a function that computes the transfer function models described in this chapter, linearized about the trim state and trim inputs.

5.3 Create a function that computes the longitudinal and lateral state space models described in this chapter, linearized around trim.

5.4 Compute eigenvalues of `A_lon` and notice that one of the eigenvalues will be zero and that there are two complex conjugate pairs. Using the formula

$$(s+\lambda)(s+\lambda^*) = s^2 + 2\Re\lambda s + |\lambda|^2 = s^2 + 2\zeta\omega_n s + \omega_n^2,$$

extract ω_n and ζ from the two complex conjugate pairs of poles. The pair with the larger ω_n correspond to the short-period mode, and the pair with the smaller ω_n correspond to the phugoid mode. The phugoid and short-period modes can be excited by starting the simulation in a wings-level, constant-altitude trim condition, and placing an impulse on the elevator. By placing an impulse on the elevator, convince yourself that the eigenvalues of `A_lon` adequately predict the short period and phugoid modes.

5.5 Compute eigenvalues of `A_lat` and notice that there is an eigenvalue at zero, a real eigenvalue in the right half plane, a real eigenvalue in the left half plane, and a complex conjugate pair. The real eigenvalue in the right half plane is the spiral-divergence mode, the real eigenvalue in the left half plane is the roll mode, and the complex eigenvalues are the dutch-roll mode. The lateral modes can be excited by starting the simulation in a wings-level, constant-altitude trim condition, and placing a unit doublet on the aileron or on the rudder. By simulating the doublet, convince yourself that the eigenvalues of `A_lat` adequately predict the roll, spiral-divergence, and dutch-roll modes.

Chapter Six

Autopilot Design

6.1 SUCCESSIVE LOOP CLOSURE

MODIFIED MATERIAL: The primary goal in autopilot design is to control the inertial position (p_n, p_e, h) and attitude (ϕ, θ, χ) of the MAV. For most flight maneuvers of interest, autopilots designed on the assumption of decoupled dynamics yield good performance. In the discussion that follows, we will assume that the longitudinal dynamics (forward speed, pitching, climbing/descending motions) are decoupled from the lateral dynamics (rolling, yawing motions). This simplifies the development of the autopilot significantly and allows us to utilize a technique commonly used for autopilot design called successive loop closure.

The basic idea behind successive loop closure is to close several simple feedback loops in succession around the open-loop plant dynamics rather then designing a single (presumably more complicated) control system. To illustrate how this approach can be applied, consider the open-loop system shown in figure 6.1. The open-loop dynamics are given by the product of three transfer functions in series: $P(s) = P_1(s)P_2(s)P_3(s)$. Each of the transfer functions has an output (y_1, y_2, y_3) that can be measured and used for feedback. Typically, each of the transfer functions, $P_1(s)$, $P_2(s)$, $P_3(s)$, is of relatively low order — usually first or second order. In this case, we are interested in controlling the output y_3. Instead of closing a single feedback loop with y_3, we will instead close feedback loops around y_1, y_2, and y_3 in succession, as shown in figure 6.2. We will design the compensators $C_1(s)$, $C_2(s)$, and $C_3(s)$ in succession. A necessary condition in the design process is that the inner loop has the highest bandwidth, with each successive loop bandwidth a factor of 5 to 10 times smaller in frequency.

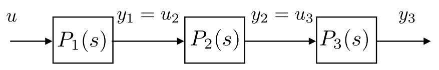

$$\begin{tikzcd} {} \arrow[r, "u"] & \boxed{P_1(s)} \arrow[r, "y_1 = u_2"] & \boxed{P_2(s)} \arrow[r, "y_2 = u_3"] & \boxed{P_3(s)} \arrow[r, "y_3"] & {} \end{tikzcd}$$

Figure 6.1: Open-loop transfer function modeled as a cascade of three transfer functions.

Examining the inner loop shown in figure 6.2, the goal is to design a closed-loop system from r_1 to y_1 having a bandwidth $\omega_{\mathrm{BW}1}$. The key as-

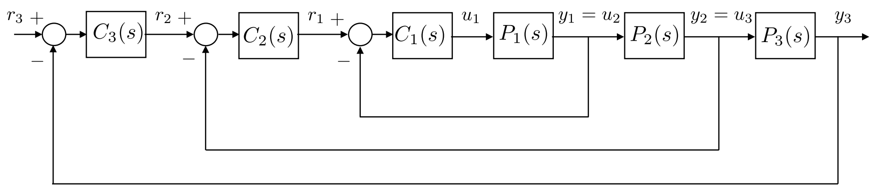


Figure 6.2: Three-stage successive loop closure design.

sumption we make is that for frequencies well below ω_{BW1}, the closed-loop transfer function $y_1(s)/r_1(s)$ can be modeled as a gain of 1. This is depicted schematically in figure 6.3. With the inner-loop transfer function modeled as a gain of 1, design of the second loop is simplified because it includes only the plant transfer function $P_2(s)$ and the compensator $C_2(s)$. The critical step in closing the loops successively is to design the bandwidth of the next loop so that it is a factor of W smaller than the preceding loop, where S is typically in the range of 5-10. In this case, we require $\omega_{\mathrm{BW2}} < \frac{1}{W}\omega_{\mathrm{BW1}}$ thus ensuring that the unity gain assumption on the inner loop is not violated over the range of frequencies that the middle loop operates.

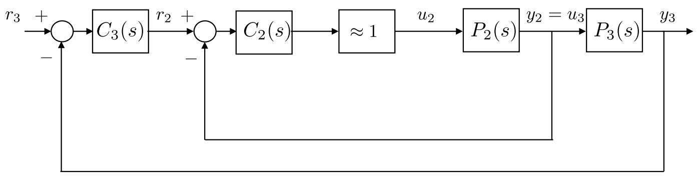


Figure 6.3: Successive loop closure design with inner loop modeled as a unity gain.

With the two inner loops operating as designed, $y_2(s)/r_2(s) \approx 1$ and the transfer function from $r_2(s)$ to $y_2(s)$ can be replaced with a gain of 1 for the design of the outermost loop, as shown in figure 6.4. Again, there is a bandwidth constraint on the design of the outer loop: $\omega_{\mathrm{BW3}} < \frac{1}{W_2}\omega_{\mathrm{BW2}}$. Because each of the plant models $P_1(s)$, $P_2(s)$, and $P_3(s)$ is first or second order, conventional PID or lead-lag compensators can be employed effectively. Transfer-function-based design methods such as root-locus or loop-shaping approaches are commonly used.

The following sections discuss the design of a lateral autopilot and a longitudinal autopilot. Transfer functions modeling the lateral and longitudinal dynamics were developed in Section 5.4 and will be used to design the autopilots in this chapter.

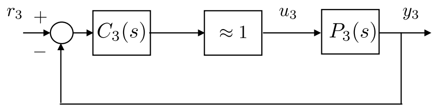
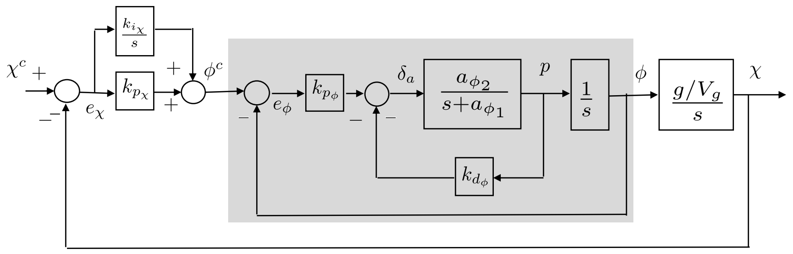


Figure 6.4: Successive-loop-closure design with two inner loops modeled as a unity gain.

6.1.1 Lateral-directional Autopilot

MODIFIED MATERIAL:

Figure 6.5 shows the block diagram for a lateral autopilot using successive loop closure. There are five gains associated with the lateral autopilot. The derivative gain k_{d_ϕ} provides roll rate damping for the innermost loop. The roll attitude is regulated with the proportional gain k_{p_ϕ}. The course angle is regulated with the proportional and integral gains k_{p_χ} and k_{i_χ}. And the dutch-roll mode is effectively damped using the yaw damper with gain k_r. The idea with successive loop closure is that the gains are successively chosen beginning with the inner loop and working outward. In particular, k_{d_ϕ} and k_{p_ϕ} are usually selected first, and k_{p_χ} and k_{i_χ} are usually chosen second. The gain k_r is selected independently of the other gains.



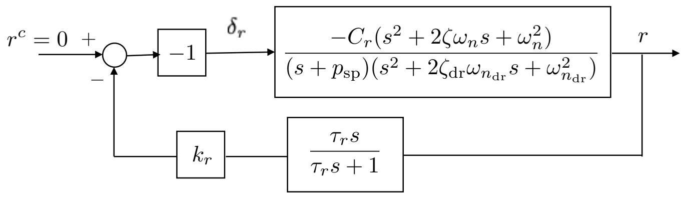

$r^c = 0$
+
−
-1
δ_r
$\frac{-C_r(s^2+2\zeta\omega_n s+\omega_n^2)}{(s+p_{\rm sp})(s^2+2\zeta_{\rm dr}\omega_{n_{\rm dr}}s+\omega_{n_{\rm dr}}^2)}$
r
k_r
$\frac{\tau_r s}{\tau_r s+1}$

Figure 6.5: Autopilot for lateral control using successive loop closure.

The following sections describe the design of the lateral autopilot using successive loop closure.

6.1.1.1 Roll Hold using the Aileron

The inner loop of the lateral autopilot is used to control roll angle and roll rate, as shown in figure 6.6.

MODIFIED MATERIAL:

If the transfer function coefficients a_{ϕ_1} and a_{ϕ_2} are known, then there is a systematic method for selecting the control gains k_{d_ϕ} and k_{p_ϕ} based on the desired response of closed-loop dynamics. From figure 6.6, the transfer

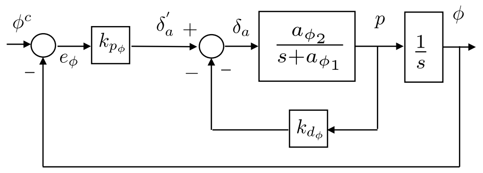


Figure 6.6: Roll attitude hold control loops.

function from ϕ^c to ϕ is given by

$$H_{\phi/\phi^c}(s) = \frac{k_{p_\phi} a_{\phi_2}}{s^2 + (a_{\phi_1} + a_{\phi_2} k_{d_\phi})s + k_{p_\phi} a_{\phi_2}}.$$

Note that the DC gain is equal to one. If the desired response is given by the canonical second-order transfer function

$$H^d_{\phi/\phi^c}(s) = \frac{\omega^2_{n_\phi}}{s^2 + 2\zeta_\phi \omega_{n_\phi} s + \omega^2_{n_\phi}},$$

then equating denominator polynomial coefficients, we get

$$\omega^2_{n_\phi} = k_{p_\phi} a_{\phi_2} \tag{6.1}$$

$$2\zeta_\phi \omega_{n_\phi} = a_{\phi_1} + a_{\phi_2} k_{d_\phi}. \tag{6.2}$$

Accordingly, the proportional and derivative gains are given by

$$k_{p_\phi} = \frac{\omega^2_{n_\phi}}{a_{\phi_2}}$$

$$k_{d_\phi} = \frac{2\zeta_\phi \omega_{n_\phi} - a_{\phi_1}}{a_{\phi_2}}.$$

where the natural frequency ω_{n_ϕ} and the damping ratio ζ_ϕ is a design parameter.

The output of the roll attitude hold loop is

$$\delta_a(t) = k_{p_\phi}(\phi^c(t) - \phi(t)) - k_{d_\phi} p(t).$$

6.1.1.2 Course Hold using Commanded Roll

MODIFIED MATERIAL:

The next step in the successive-loop-closure design of the lateral autopilot is to design the course-hold outer loop. If the inner loop from ϕ^c to ϕ has been adequately tuned, then $H_{\phi/\phi^c} \approx 1$ over the range of frequencies from 0 to ω_{n_ϕ}. Under this condition, the block diagram of figure 6.5 can be simplified to the block diagram in figure 6.7 for the purposes of designing the outer loop.

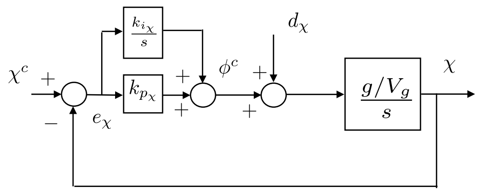


Figure 6.7: Course hold outer feedback loop.

The objective of the course hold design is to select k_{p_χ} and k_{i_χ} in figure 6.5 so that the course χ asymptotically tracks steps in the commanded course χ^c. From the simplified block diagram, the transfer functions from the inputs χ^c and d_χ to the output χ are given by

$$\chi = \frac{g/V_g s}{s^2 + k_{p_\chi} g/V_g s + k_{i_\chi} g/V_g} d_\chi + \frac{k_{p_\chi} g/V_g s + k_{i_\chi} g/V_g}{s^2 + k_{p_\chi} g/V_g s + k_{i_\chi} g/V_g} \chi^c. \quad (6.3)$$

Note that if d_χ and χ^c are constants, then the final value theorem implies that $\chi \to \chi^c$. The transfer function from χ^c to χ has the form

$$H_\chi = \frac{2\zeta_\chi \omega_{n_\chi} s + \omega_{n_\chi}^2}{s^2 + 2\zeta_\chi \omega_{n_\chi} s + \omega_{n_\chi}^2}. \quad (6.4)$$

As with the inner feedback loops, we can choose the natural frequency and damping of the outer loop and from those values calculate the feedback gains k_{p_χ} and k_{i_χ}. Figure 6.8 shows the frequency response and the step response for H_χ. Note that because of the numerator zero, the standard intuition for the selection of ζ does not hold for this transfer function. Larger ζ results in larger bandwidth and smaller overshoot.

Comparing coefficients in equations (6.3) and (6.4), we find

$$\begin{aligned} \omega_{n_\chi}^2 &= g/V_g k_{i_\chi} \\ 2\zeta_\chi \omega_{n_\chi} &= g/V_g k_{p_\chi}. \end{aligned}$$

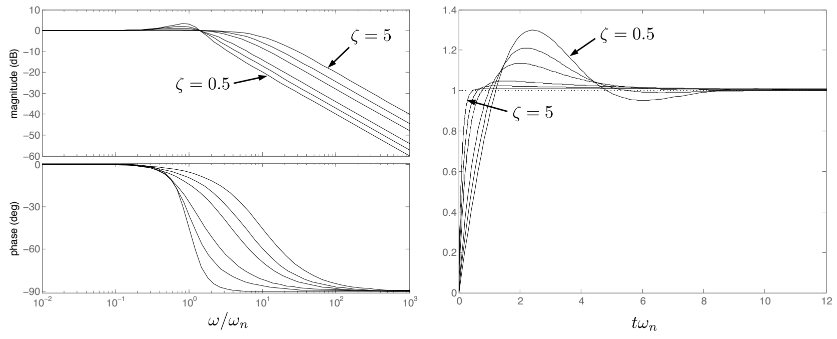


Figure 6.8: Frequency and step response for a second-order system with a transfer function zero for $\zeta = 0.5$, 0.7, 1, 2, 3, 5.

Solving these expressions for k_{p_χ} and k_{i_χ}, we get

$$k_{p_\chi} = 2\zeta_\chi \omega_{n_\chi} V_g/g \tag{6.5}$$

$$k_{i_\chi} = \omega_{n_\chi}^2 V_g/g. \tag{6.6}$$

To ensure proper function of this successive-loop-closure design, it is essential that there be sufficient bandwidth separation between the inner and outer feedback loops. Adequate separation can be achieved by letting

$$\omega_{n_\chi} = \frac{1}{W_\chi}\omega_{n_\phi},$$

where the separation W_χ is a design parameter that is usually chosen to be greater than five. Generally, more bandwidth separation is better. More bandwidth separation requires either slower response in the χ loop (lower ω_{n_χ}), or faster response in the ϕ loop (higher ω_{n_ϕ}). Faster response usually comes at the cost of requiring more actuator control authority, which may not be possible given the physical constraints of the actuators.

The output of the course hold loop is

$$\phi^c(t) = k_{p_\chi}(\chi^c(t) - \chi(t)) + k_{i_\chi}\int_{-\infty}^{t}(\chi^c(\tau) - \chi(\tau))d\tau.$$

6.1.1.3 Yaw Damper using the Rudder

NEW MATERIAL: As discussed in the previous section, the aileron is used to control the roll angle, and subsequently the course angle. In deriving the transfer function from the roll angle to the course angle we assumed that the side-slip angle is zero. While the rudder could be used to regulate the side-slip angle, this is impractical for small aircraft since the side-slip angle is not measured directly. An alternative is to use the rudder to regulate the yaw rate to zero. The difficulty with doing so, however, is that in a turn we do not desire the yaw rate to be zero and so a direct control implementation would create competing objectives between the course controller and the yaw rate controller. The solution to this dilemma is to employ a so-called washout filter to only allow feedback from yaw rate to the rudder for high frequency yaw rate. A wash-out filter is simply a high-pass filter.

Recall from Equation (5.24) that the transfer function from the rudder to the yaw rate is given by

$$r(s) = \left(\frac{N_{\delta_r} s + (Y_{\delta_r} N_v - N_{\delta_r} Y_v)}{s^2 - (Y_v + N_r)s + (Y_v N_r - Y_r N_v)} \right) \delta_r(s). \tag{6.7}$$

The yaw-damper control loop is shown in Figure 6.9, where p_{wo} is the pole of the washout-filter, and $k_r > 0$ is the control gain from yaw-rate to rudder. The effect of the washout filter is to remove the feedback signal over the

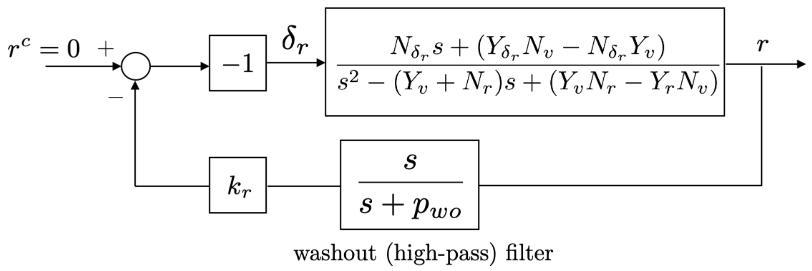

$r^c = 0$
$+$
$-$
-1
δ_r
$\frac{N_{\delta_r} s + (Y_{\delta_r} N_v - N_{\delta_r} Y_v)}{s^2 - (Y_v + N_r)s + (Y_v N_r - Y_r N_v)}$
r
k_r
$\frac{s}{s + p_{wo}}$
washout (high-pass) filter

Figure 6.9: Yaw damper with washout filter.

frequency band below p_{wo} and to pass with unity gain the feedback signal for frequencies above p_{wo}. The negative sign in the top path is due to the fact that negative δ_r results in a postive yaw rate (i.e., $N_{\delta_r} < 0$).

The selection of p_{wo} and k_r can be found from physical considerations. As discussed in Chapter 5, the poles of the transfer function from δ_r to r shown in Equation 6.7 correspond to the lightly-damped dutch roll mode. Essentially, the objective of the yaw damper is damp out the dutch-roll mode. Therefore, the pole of the washout filter should be selected to be below the natural frequency of the dutch-roll mode. The canonical expression $s^2 + 2\zeta_{dr}\omega_{n_{dr}} s + \omega_{n_{dr}}^2 = s^2 - (Y_v + N_r)s + (Y_v N_r - Y_r N_v)$ approximates the natural frequency of the dutch-roll mode as

$$\omega_{n_{dr}} \approx \sqrt{Y_v N_r - Y_r N_v}.$$

A rule-of-thumb is to select $p_{wo} = \omega_{n_{dr}}/10$.

For frequencies over p_{wo}, when the washout filter is active, the closed-loop transfer function from r^c to r is given by

$$r(s) = \left(\frac{-(N_{\delta_r} s + (Y_{\delta_r} N_v - N_{\delta_r} Y_v))}{s^2 + (-Y_v - N_r - k_r N_{\delta_r})s + (Y_v N_r - Y_r N_v - k_r (Y_{\delta_r} N_v - N_{\delta_r} Y_v))} \right) r^c(s),$$

where the closed-loop poles are given by

$$p_{\text{yaw-damper}} = \frac{Y_v + N_r + k_r N_{\delta_r}}{2} \pm j\sqrt{(Y_v N_r - Y_r N_v - k_r (Y_{\delta_r} - N_v N_{\delta_r} Y_v) - \left(\frac{Y_v + N_r + k_r N_{\delta_r}}{2}\right)^2}.$$

A reasonable approach is to select k_r so that the damping ratio of the closed-loop system is $\zeta = 0.707$. This occurs when the real and imaginary parts of closed-loop poles have the same magnitude, i.e., when

$$\left(\frac{Y_v + N_r + k_r N_{\delta_r}}{2}\right)^2 = (Y_v N_r - Y_r N_v - k_r (Y_{\delta_r} - N_v N_{\delta_r} Y_v) - \left(\frac{Y_v + N_r + k_r N_{\delta_r}}{2}\right)^2.$$

Simplifying, we get that k_r is the positive root of the quadratic equation

$$N_{\delta_r}^2 k_r^2 + 2(N_r N_{\delta_r} + Y_{\delta_r} N_v) k_r + (Y_v^2 + N_r^2 + 2 Y_r N_v) = 0,$$

resulting in

$$k_r = -\frac{N_r N_{\delta_r} + Y_{\delta_r} N_v}{N_{\delta_r}^2} + \sqrt{\left(\frac{N_r N_{\delta_r} + Y_{\delta_r} N_v}{N_{\delta_r}^2}\right)^2 - \left(\frac{Y_v^2 + N_r^2 + 2 Y_r N_v}{N_{\delta_r}^2}\right)}.$$

For the Aerosonde model given in the appendix, we get $p_{wo} = 0.45$ and $k_r = 0.196$.

Figure 6.10 shows the lateral-directional performance of the Aerosonde aircraft in response to a doublet roll command, both without the yaw damper and with the yaw damper. The upper plot shows the roll response of the aircraft to the doublet roll command. Without the yaw damper, the lightly damped dutch-roll dynamics couple into the closed-loop roll dynamics causing the roll dynamics to be less-damped than anticipated. The lower plot shows the yaw (heading) angle of the aircraft with and without the yaw damper. The improved response resulting from the yaw damper is clear. This is particularly true for the Aerosonde aircraft model used, which has an influential non-minimum-phase zero in the transfer function from aileron deflection to yaw angle. This can be seen at time $t = 1$ s where the positive roll response (caused by a positive aileron deflection) causes the aircraft to yaw negatively initially. This is commonly referred to as *adverse yaw* and is caused by the negative yawing moment produced by the positive aileron deflection as modeled by the N_{δ_r} aerodynamic coefficient. This non-minimum phase zero further aggravates the negative effect of the lightly damped dutch-roll mode, making the implementation of the yaw damper a necessity for well-behaved flight.

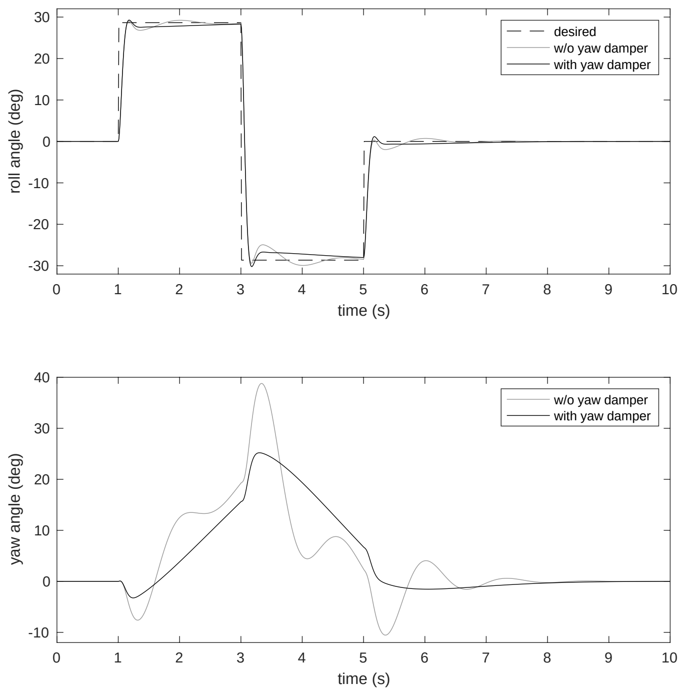


Figure 6.10: Turning performance with and without yaw damper.

A final comment on the effect of the dutch-roll mode on the design of the lateral-directional control of the aircraft is that it may be necessary to take the speed of the dutch-roll mode into consideration when using successive-loop closure to set the bandwidth of the outer-loop course control. Rather than keeping the bandwidth of the outer course loop at least a factor of ten below the bandwidth of the roll loop. Better performance may result from setting the outer course-loop bandwidth at least a factor of ten slower than the frequency of the dutch-roll mode.

6.1.1.4 Yaw Damper Implementation

NEW MATERIAL:

The yaw damper in the Laplace domain is given by

$$\delta_r(s) = k_r \left(\frac{s}{s + p_{wo}} \right) r(s).$$

Converting to the time domain results in the differential equation

$$\dot{\delta}_r = -p_{wo}\,\delta_r + k_r \dot{r}.$$

Integrating once gives

$$\delta_r(t) = -p_{wo} \int_{-\infty}^{t} \delta_r(\sigma) d\sigma + k_r r(t). \tag{6.8}$$

A block diagram that corresponds to equation (6.8) is shown in figure 6.11.

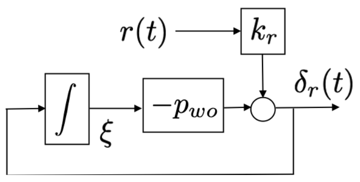


Figure 6.11: Block diagram for yaw damper transfer function.

If the output of the integrator is ξ, then the block diagram corresponds to the state-space equations

$$\begin{aligned} \dot{\xi} &= -p_{wo}\,\xi + k_r r \\ \delta_r &= -p_{wo}\,\xi + k_r r. \end{aligned}$$

Using a Euler approximation for the derivative, the discrete-time equations are

$$\begin{aligned} \xi_k &= -p_{wo}\,\xi_{k-1} + k_r r_{k-1} \\ \delta_{rk} &= -p_{wo}\,\xi_k + k_r r_k. \end{aligned}$$

The corresponding Python code is shown below.

```
class yawDamper:
    def __init__(self, k_r, p_wo, Ts):
        self.xi = 0.
        self.Ts = Ts
        self.k_r = k_r
        self.p_wo = p_wo
    def update(self, u):
        self.xi = self.xi
                + self.Ts * (-self.p_wo * self.xi
                             + self.k_r * r)
        delta_r = -p_wo * self.xi + self.k_r * r
        return delta_r
```

6.1.2 Longitudinal Autopilot

MODIFIED MATERIAL: Figure 6.12 shows the block diagram for the longitudinal autopilot using successive loop closure. There are six gains associated with the longitudinal autopilot. The derivative gain k_{d_θ} provides pitch rate damping for the innermost loop. The pitch attitude is regulated with the proportional gain k_{p_θ}. The altitude is regulated with the proportional and integral gains k_{p_h} and k_{i_h}. The airpseed is regulated using the proportional and integral gains $k_{p_{V_a}}$ and $k_{i_{V_a}}$. The idea with successive loop closure is that the gains are successively chosen beginning with the inner loop and working outward. In particular, k_{d_θ} and k_{p_θ} are usually selected first, and k_{p_h} and k_{i_h} are usually chosen second. The gains $k_{p_{V_a}}$ and $k_{i_{V_a}}$ are selected independently of the other gains.

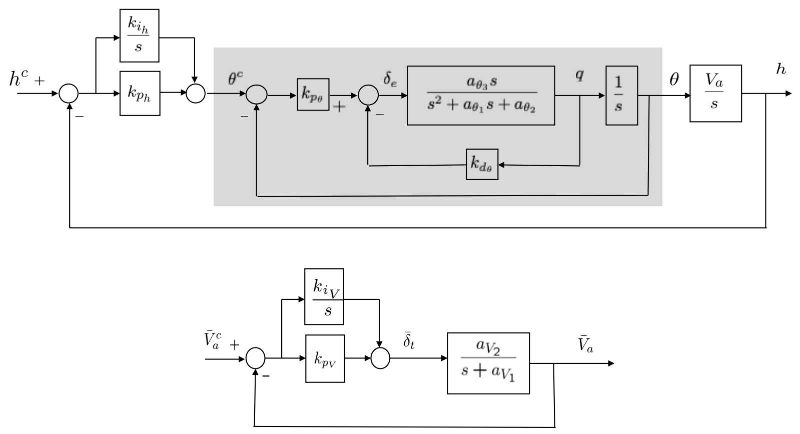


Figure 6.12: Autopilot for longitudinal control using successive loop closure.

The following sections describe the design of the longitudinal autopilot using successive loop closure.

6.1.2.1 Pitch Hold using the Elevator

MODIFIED MATERIAL: The pitch attitude hold loop is similar to the roll attitude hold loop, and we will follow a similar line of reasoning in its development. From figure 6.13, the transfer function from θ^c to θ is given by

$$H_{\theta/\theta^c}(s) = \frac{k_{p_\theta} a_{\theta_3}}{s^2 + (a_{\theta_1} + k_{d_\theta} a_{\theta_3})s + (a_{\theta_2} + k_{p_\theta} a_{\theta_3})}. \tag{6.9}$$

Note that in this case, the DC gain is not equal to one.

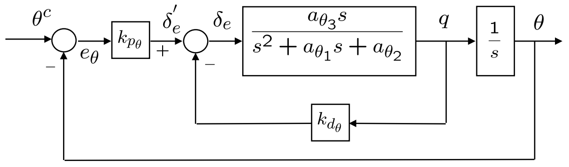


Figure 6.13: Pitch attitude hold feedback loops.

If the desired response is given by the canonical second-order transfer function

$$H^d_{\theta/\theta^c} = \frac{K_{\theta_{\rm DC}} \omega^2_{n_\theta}}{s^2 + 2\zeta_\theta \omega_{n_\theta} s + \omega^2_{n_\theta}},$$

then, equating denominator coefficients, we get

$$\omega^2_{n_\theta} = a_{\theta_2} + k_{p_\theta} a_{\theta_3} \tag{6.10}$$
$$2\zeta_\theta \omega_{n_\theta} = a_{\theta_1} + k_{d_\theta} a_{\theta_3} \tag{6.11}$$
$$K_{\theta_{\rm DC}} \omega^2_{n_\theta} = k_{p_\theta} a_{\theta_3}. \tag{6.12}$$

Solving for k_{p_θ}, k_{d_θ}, and $K_{\theta_{\rm DC}}$ we get

$$k_{p_\theta} = \frac{\omega^2_{n_\theta} - a_{\theta_2}}{a_{\theta_3}}$$
$$k_{d_\theta} = \frac{2\zeta_\theta \omega_{n_\theta} - a_{\theta_1}}{a_{\theta_3}}$$
$$K_{\theta_{\rm DC}} = \frac{k_{p_\theta} a_{\theta_3}}{\omega^2_{n_\theta}},$$

where ω_{n_θ} and ζ_θ are design parameters.

An integral feedback term could be employed to ensure unity DC gain on the inner loop. The addition of an integral term, however, can severely limit the bandwidth of the inner loop, and therefore is not used. Note however, that in the design project, the actual pitch angle will not converge to the commanded pitch angle. This fact will be taken into account in the development of the altitude hold loop.

The output of the pitch loop is therefore

$$\delta_e(t) = k_{p_\theta}\left(\theta^c(t) - \theta(t)\right) - k_{d_\theta} q(t).$$

6.1.2.2 Altitude Hold using Commanded Pitch

MODIFIED MATERIAL: The altitude-hold autopilot utilizes a successive-loop-closure strategy with the pitch attitude hold autopilot as an inner loop, as shown in figure 6.12. Assuming that the pitch loop functions as designed and that $\theta \approx K_{\theta_{\text{DC}}}\theta^c$, the altitude hold loop using the commanded pitch can be approximated by the block diagram shown in figure 6.14.

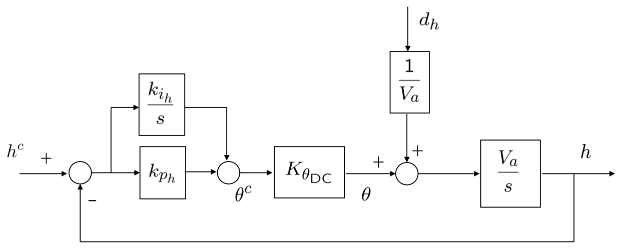


Figure 6.14: The altitude hold loop using the commanded pitch angle.

In the Laplace domain, we have

$$h(s) = \left(\frac{K_{\theta_{\text{DC}}} V_a k_{p_h}\left(s + \frac{k_{i_h}}{k_{p_h}}\right)}{s^2 + K_{\theta_{\text{DC}}} V_a k_{p_h} s + K_{\theta_{\text{DC}}} V_a k_{i_h}}\right) h^d(s) + \left(\frac{s}{s^2 + K_{\theta_{\text{DC}}} V_a k_{p_h} s + K_{\theta_{\text{DC}}} V_a k_{i_h}}\right) d_h(s),$$

where again we see that the DC gain is equal to one, and constant disturbances are rejected. The closed-loop transfer function is again independent of aircraft parameters and is dependent only on the known airspeed. The gains k_{p_h} and k_{i_h} should be chosen such that the bandwidth of the altitude-from-pitch loop is less than the bandwidth of the pitch-attitude-hold loop. Similar to the course loop, let

$$\omega_{n_h} = \frac{1}{W_h}\omega_{n_\theta},$$

where the bandwidth separation W_h is a design parameter that is usually between five and fifteen. If the desired response of the altitude hold loop is given by the canonical second-order transfer function

$$H^d_{h/h^c} = \frac{\omega^2_{n_h}}{s^2 + 2\zeta_h \omega_{n_h} s + \omega^2_{n_h}},$$

then, equating denominator coefficients, we get

$$\begin{aligned} \omega^2_{n_h} &= K_{\theta_{\rm DC}} V_a k_{i_h} \\ 2\zeta_h \omega_{n_h} &= K_{\theta_{\rm DC}} V_a k_{p_h}. \end{aligned}$$

Solving these expressions for k_{i_h} and k_{p_h}, we get

$$k_{i_h} = \frac{\omega^2_{n_h}}{K_{\theta_{\rm DC}} V_a} \tag{6.13}$$

$$k_{p_h} = \frac{2\zeta_h \omega_{n_h}}{K_{\theta_{\rm DC}} V_a}. \tag{6.14}$$

Therefore, selecting the desired damping ratio ζ_h and the bandwidth separation W_h fixes the value for k_{p_h} and k_{i_h}.

The commanded pitch angle is therefore

$$\theta^c(t) = k_{p_h} \left(h^c(t) - h(t)\right) + k_{i_h} \int_{-\infty}^{t} (h^c(\tau) - h(\tau)) d\tau.$$

6.1.2.3 Airspeed Hold using Throttle

MODIFIED MATERIAL: Using PI control for the airspeed loop results in the closed-loop system is shown in figure 6.15. If we use proportional

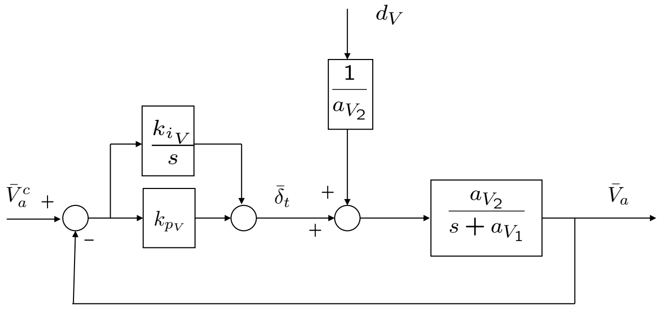


Figure 6.15: Airspeed hold using throttle.

control, then

$$\bar{V}_a(s) = \left(\frac{a_{V_2} k_{p_V}}{s + (a_{V_1} + a_{V_2} k_{p_V})}\right) \bar{V}^c_a(s) + \left(\frac{1}{s + (a_{V_1} + a_{V_2} k_{p_V})}\right) d_V(s).$$

Note that the DC gain is not equal to one and that step disturbances are not rejected. If, on the other hand, we use proportional-integral control, then

$$\bar{V}_a = \left(\frac{a_{V_2}(k_{p_V} s + k_{i_V})}{s^2 + (a_{V_1} + a_{V_2} k_{p_V})s + a_{V_2} k_{i_V}} \right) \bar{V}_a^c + \left(\frac{1}{s^2 + (a_{V_1} + a_{V_2} k_{p_V})s + a_{V_2} k_{i_V}} \right) d_V.$$

It is clear that using a PI controller results in a DC gain of one, with step disturbance rejection. If a_{V_1} and a_{V_2} are known, then the gains k_{p_V} and k_{i_V} can be determined using the same technique we have used previously. Equating the closed-loop transfer function denominator coefficients with those of a canonical second-order transfer function, we get

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

Inverting these expressions gives the control gains

$$k_{i_V} = \frac{\omega_{n_V}^2}{a_{V_2}} \tag{6.15}$$

$$k_{p_V} = \frac{2\zeta_V \omega_{n_V} - a_{V_1}}{a_{V_2}}. \tag{6.16}$$

The design parameters for this loop are the damping coefficient ζ_V and the natural frequency ω_{n_V}.

Note that since $\bar{V}_a^c = V_a^c - V_a^*$ and $\bar{V}_a = V_a - V_a^*$, the error signal in figure 6.15 is

$$e = \bar{V}_a^c - \bar{V}_a = V_a^c - V_a.$$

Therefore, the control loop shown in figure 6.15 can be implemented without knowledge of the trim velocity V_a^*. If the throttle trim value δ_t^* is known, then the throttle command is

$$\delta_t = \delta_t^* + \bar{\delta}_t.$$

However, if δ_t^* is not precisely known, then the error in δ_t^* can be thought of as a step disturbance, and the integrator will wind up to reject the disturbances.

The throttle command is therefore

$$\delta_t(t) = k_{p_V} \left(V_a^c(t) - V_a(t) \right) + k_{i_V} \int_{-\infty}^{t} (V_a^c(\tau) - V_a(\tau)) d\tau.$$

6.1.3 Digital Implementation of PID Loops

NEW MATERIAL:

A Python class that implements a general PID loop is shown below.

```
import numpy as np

class pid_control:
    def __init__(self, kp=0.0, ki=0.0, kd=0.0, Ts=0.01,
                       sigma=0.05, limit=1.0):
        self.kp = kp
        self.ki = ki
        self.kd = kd
        self.Ts = Ts
        self.limit = limit
        self.integrator = 0.0
        self.error_delay_1 = 0.0
        self.error_dot_delay_1 = 0.0
        # gains for differentiator
        self.a1 = (2.0 * sigma - Ts) / (2.0 * sigma + Ts)
        self.a2 = 2.0 / (2.0 * sigma + Ts)

    def update(self, y_ref, y, reset_flag=False):
        if reset_flag == True:
            self.integrator = 0.0
            self.error_delay_1 = 0.0
            self.y_dot = 0.0
            self.y_delay_1 = 0.0
            self.y_dot_delay_1 = 0.0
        # compute the error
        error = y_ref - y
        # update the integrator using trapazoidal rule
        self.integrator = self.integrator \
                + (self.Ts/2) * (error + self.error_delay_1)
        # update the differentiator
        error_dot = self.a1 * self.error_dot_delay_1 \
                + self.a2 * (error - self.error_delay_1)
        # PID control
        u = self.kp * error \
            + self.ki * self.integrator \
            + self.kd * error_dot
        # saturate PID control at limit
        u_sat = self._saturate(u)
        # integral anti-windup
        #   adjust integrator to keep u out of saturation
        if np.abs(self.ki) > 0.0001:
            self.integrator = self.integrator \
                + (1.0 / self.ki) * (u_sat - u)
        # update the delayed variables
        self.error_delay_1 = error
        self.error_dot_delay_1 = error_dot
        return u_sat

    def update_with_rate(self, y_ref, y, ydot,
                         reset_flag=False):
        if reset_flag == True:
            self.integrator = 0.0
            self.error_delay_1 = 0.0
        # compute the error
        error = y_ref - y
        # update the integrator using trapazoidal rule
        self.integrator = self.integrator \
                + (self.Ts/2) * (error + self.error_delay_1)
        # PID control
        u = self.kp * error \
            + self.ki * self.integrator \
            - self.kd * ydot
        # saturate PID control at limit
        u_sat = self._saturate(u)
        # integral anti-windup
        #   adjust integrator to keep u out of saturation
        if np.abs(self.ki) > 0.0001:
            self.integrator = self.integrator \
                + (1.0 / self.ki) * (u_sat - u)
        self.error_delay_1 = error
        return u_sat

    def _saturate(self, u):
        # saturate u at +- self.limit
        if u >= self.limit:
            u_sat = self.limit
        elif u <= -self.limit:
            u_sat = -self.limit
        else:
            u_sat = u
        return u_sat
```

STOP NEW MATERIAL:
NEW MATERIAL:

6.2 TOTAL ENERGY CONTROL

One of the disadvantages of the longitudinal autopilot discussed in the book is the number of required loops and the state machine for altitude control shown in page 113. In this note we will describe a simplier scheme based on the energy states of the system. The ideas in this supplement are motivated by [**?**, **?**].

The longitudinal control system is complicated by the fact that both altitude and airspeed need to be regulated, but that these quantities are strongly coupled. If we assume a low level autopilot on the pitch angle, then the primary control signals are the throttle and the commanded pitch angle. Both of these quantities have a significant effect on both altitude and airspeed. Rather than attempt to decouple these effects by operating different loops in different flight regimes, the total energy control method changes the regulated outputs from altitude and airspeed to total energy and energy balance, which produces a natural decoupling in the longitudinal motion.

The kinetic energy of a body in motion is given by $K = \frac{1}{2}m\left\|\mathbf{v}\right\|^2$. If we use the velocity of the aircraft relative to the air mass, then $K = \frac{1}{2}mV_a^2$. The reference kinetic energy is given by $K_{\text{ref}} = \frac{1}{2}m(V_a^c)^2$. Therefore, the error in kinetic energy is given by

$$K_{\text{error}} \overset{\triangle}{=} K_{\text{ref}} - K = \frac{1}{2}m((V_a^c)^2 - V_a^2).$$

The potential energy of a body with mass m is given by $U = U_0 + mgh$ where U_0 is the potential of ground level when the altitude $h = 0$. The reference potential energy is given by $U_{\text{ref}} = U_0 + mgh^c$. Therefore, the error in potential energy is given by

$$U_{\text{error}} = mg(h^c - h). \tag{6.17}$$

The total energy (error) is given by

$$E = U_{\text{error}} + K_{\text{error}}.$$

The energy (error) balance is given by

$$B = U_{\text{error}} - K_{\text{error}}.$$

As mentioned previously, the throttle is used to control the total energy using a PI controller:

$$\delta_t(t) = k_{p_E}E(t) + k_{i_E}\int_{-\infty}^{t} E(\tau)\,d\tau.$$

The pitch command is used to regulate the energy balance using a PI controller:

$$\theta^c(t) = k_{p_B}B(t) + k_{i_B}\int_{-\infty}^{t} B(\tau)\,d\tau.$$

As a matter of practical consideration, the TECS scheme works better for large deviations in altitude if the altitude error in (6.17) is saturated as

$$U_{\text{error}} = mg\ \text{sat}_{\bar{h}_e}(h^c - h),$$

where $\bar{h}_e > 0$ is the largest altitude error used to compute U_{error}, and sat is the saturation function.

NEW MATERIAL:

6.3 LQR CONTROL

Given the state space equation

$$\dot{x} = Ax + Bu$$

and the symmetric positive semi-definite matrix Q, and the symmetric positive definite matrix R, the LQR problem is to minimize the cost index

$$J(x_0) = \min_{u(t), t\geq 0} \int_0^\infty x^\top(\tau)Qx(\tau) + u^\top(\tau)Ru(\tau)d\tau.$$

If (A, B) is controllable, and $(A, Q^{1/2})$ is observable, then a unique optimal control exists and is given in linear feedback form as

$$u_{lqr}(t) = -K_{lqr}x(t),$$

where the LQR gain is given by

$$K_{lqr} = R^{-1}B^\top P,$$

and where P is the symmetric positive definite solution of the Algebraic Riccati Equation

$$PA + A^\top P + Q - PBR^{-1}B^\top P = 0.$$

Note that K_{lqr} are the optimal feedback gains given Q and R. The controller is tuned by changing Q and R. Typically we choose Q and R to be diagonal matrices

$$Q = \begin{pmatrix} q_1 & 0 & \dots & 0 \\ 0 & q_2 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & q_n \end{pmatrix} \qquad R = \begin{pmatrix} r_1 & 0 & \dots & 0 \\ 0 & r_2 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & r_m \end{pmatrix},$$

where n is the number of states, m is the number of inputs, and $q_i \geq 0$ ensures Q is positive semi-definite, and $r_i > 0$ ensure R is positive definite.

For the lateral and longitudinal autopilots, we have seen that we need integral control for course, altitude, and airspeed. Given the state space system

$$\dot{x} = Ax + Bu$$
$$z = Hx$$

where z represents the controlled output. Suppose that the objective is to drive z to a reference signal z^c and further suppose that z^c is a step, i.e., $\dot{z}^c = 0$. The first step is to augment the state with the integrator

$$x_I = \int_{-\infty}^{t} (z(\tau) - z^c)\, d\tau.$$

Defining the augmented state as $\xi = (x^\top, x_I^\top)^\top$, results in the augmented state space equations

$$\dot{\xi} = \bar{A}\xi + \bar{B}u,$$

where

$$\bar{A} = \begin{pmatrix} A & 0 \\ H & 0 \end{pmatrix} \qquad \bar{B} = \begin{pmatrix} B \\ 0 \end{pmatrix}.$$

The LQR design process then proceeds as normal.

6.3.1 Lateral Autopilot using LQR

As derived in chapter 5, the state space equations for the lateral equation of motion are given by

$$\dot{x}_{lat} = A_{lat}x_{lat} + B_{lat}u_{lat},$$

where $x_{lat} = (v, p, r, \phi, \psi)^\top$ and $u_{lat} = (\delta_a, \delta_r)^\top$. The objective of the lateral autopilot is to drive the course χ to the commanded course χ^c. Therefore, we augment the state with

$$x_I = \int (\chi - \chi^c) dt.$$

Since $\chi \approx \psi$, we approximate x_I as

$$x_I = \int (H_{lat}x_{lat} - \chi^c) dt,$$

where $H_{lat} = (0, 0, 0, 0, 1)$.

The augmented lateral state equations are therefore

$$\dot{\xi}_{lat} = \bar{A}_{lat}\xi_{lat} + \bar{B}_{lat}u_{lat},$$

where

$$\bar{A}_{lat} = \begin{pmatrix} A_{lat} & 0 \\ H_{lat} & 0 \end{pmatrix} \qquad \bar{B}_{lat} = \begin{pmatrix} B_{lat} \\ 0 \end{pmatrix}$$

The LQR controller is designed using

$$Q = \text{diag}\left([q_v, q_p, q_r, q_\phi, q_\chi, q_I]\right)$$
$$R = \text{diag}\left([r_{\delta_a}, r_{\delta_r}]\right).$$

6.3.2 Longitudinal Autopilot using LQR

As derived in chapter 5, the state space equations for the longitudinal equations of motion are given by

$$\dot{x}_{lon} = A_{lon} x_{lon} + B_{lon} u_{lon},$$

where $x_{lon} = (u, w, q, \theta, h)^\top$ and $u_{lat} = (\delta_e, \delta_t)^\top$. The objective of the longitudinal autopilot is to drive the altitude h to the commanded altitude h^c, and the airspeed V_a to commanded airspeed V_a^c. Therefore, we augment the state with

$$x_I = \begin{pmatrix} \int (h - h^c) dt \\ \int (V_a - V_a^c) dt \end{pmatrix}$$
$$= \int \left(H_{lon} x_{lon} - \begin{pmatrix} h^c \\ V_a^c \end{pmatrix} \right) dt,$$

where

$$H_{lon} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ \frac{u^*}{V_a} & \frac{w^*}{V_a} & 0 & 0 & 0 \end{pmatrix}.$$

The augmented longitudinal state equations are therefore

$$\dot{\xi}_{lon} = \bar{A}_{lon} \xi_{lon} + \bar{B}_{lon} u_{lon},$$

where

$$\bar{A}_{lon} = \begin{pmatrix} A_{lon} & 0 \\ H_{lon} & 0 \end{pmatrix} \qquad \bar{B}_{lon} = \begin{pmatrix} B_{lon} \\ 0 \end{pmatrix}$$

The LQR controller is designed using

$$Q = \text{diag}\left([q_u, q_w, q_q, q_\theta, q_h, q_I]\right)$$
$$R = \text{diag}\left([r_{\delta_e}, r_{\delta_t}]\right).$$

6.4 CHAPTER SUMMARY

NOTES AND REFERENCES

6.5 DESIGN PROJECT

The objective of this assignment is to implement the lateral and longitudinal autopilots, as described in this chapter. You can use either successive loop closure, total energy control, or LQR.

6.1 Implement the longitudinal autopilot and tune the gains. The input the longitudinal autopilot is the commanded airspeed and the commanded altitude. To make the process easier, you may want to initialize the system to trim inputs and then tune airspeed control first, followed by the pitch loop, and then the altitude loop.

6.2 Implement the lateral autopilot. The input to the lateral autopilot is the commanded course angle.

6.3 Test the autopilot using a variety of different step inputs on the commanded airspeed, altitude, and course angle. Be sure to command course angles that are greater than ± 180 degrees.

Chapter Seven

Sensors for MAVs

7.1 ACCELEROMETERS

MODIFIED MATERIAL: As an alternative, we can substitute from equations (5.4), (5.5), and (5.6) to obtain

$$\begin{aligned}
y_{\text{accel},x} &= \frac{\rho V_a^2 S}{2\mathsf{m}}\left[C_X(\alpha) + C_{X_q}(\alpha)\frac{\bar{c}q}{2V_a} + C_{X_{\delta_e}}(\alpha)\delta_e\right] \\
&\quad + T_p(\delta_t, V_a) + \eta_{\text{accel},x} \\
y_{\text{accel},y} &= \frac{\rho V_a^2 S}{2\mathsf{m}}\left[C_{Y_0} + C_{Y_\beta}\beta + C_{Y_p}\frac{bp}{2V_a} + C_{Y_r}\frac{br}{2V_a}\right. \\
&\quad \left. + C_{Y_{\delta_a}}\delta_a + C_{Y_{\delta_r}}\delta_r\right] + \eta_{\text{accel},y} \qquad (7.1) \\
y_{\text{accel},z} &= \frac{\rho V_a^2 S}{2\mathsf{m}}\left[C_Z(\alpha) + C_{Z_q}(\alpha)\frac{\bar{c}q}{2V_a} + C_{Z_{\delta_e}}(\alpha)\delta_e\right] + \eta_{\text{accel},z}.
\end{aligned}$$

7.2 RATE GYROS

7.3 PRESSURE SENSORS

7.3.1 Altitude Measurement

7.3.2 Airspeed Sensor

7.4 DIGITAL COMPASSES

7.5 GLOBAL POSITIONING SYSTEM

MODIFIED MATERIAL:

Because of clock synchronization errors between the satellites (whose atomic clocks are almost perfectly synchronized) and the receiver, four independent pseudo-range measurements are required to trilaterate the position of the receiver as depicted in figure 7.1. Why are four pseudo-range measurements required? One range measurement narrows the receiver position to a sphere around the satellite. A second measurement from another satellite narrows the possible positions to the the intersection of the the two spheres around the satellites, which is a circle. A third measurement narrows the possible positions to the intersection of the three spheres around the satellites, which is two points. The point closest to the surface of the earth is taken as the position. To resolve position in three dimensions with a receiver clock offset error, at least four measurements are needed. The geometry associated with the pseudo-range measurements from four different satellites form a system of four nonlinear algebraic equations in four unknowns: latitude, longitude, and altitude of the GPS receiver, and receiver clock time offset [?].

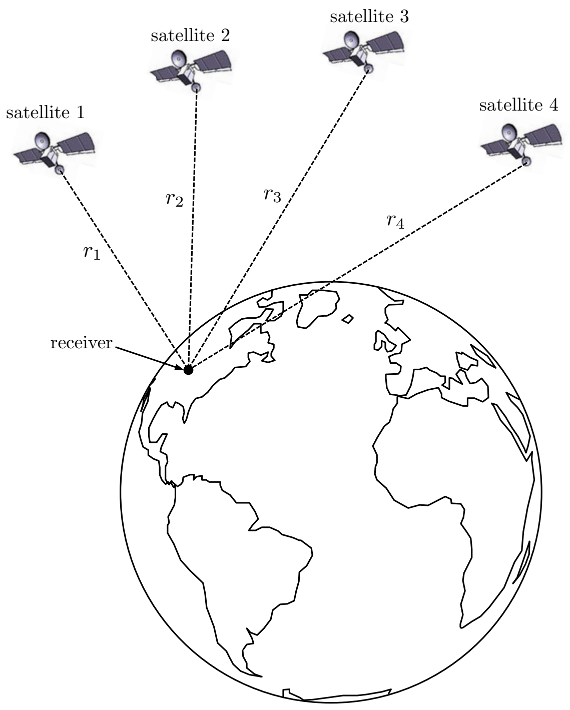


Figure 7.1: Pseudo-range measurements from four satellites used to trilaterate the position of a receiver.

7.5.1 GPS Measurement Error

7.5.2 Transient Characteristics of GPS Positioning Error

7.5.3 GPS Velocity Measurements

MODIFIED MATERIAL: Horizontal ground speed and course are calculated from the North and East velocity components from GPS as

$$V_g = \sqrt{V_n^2 + V_e^2} \tag{7.2}$$

$$\chi = \text{atan2}\left(V_e, V_n\right), \tag{7.3}$$

where $V_n = V_a \cos\psi + w_n$ and $V_e = V_a \sin\psi + w_e$.

7.6 CHAPTER SUMMARY

NOTES AND REFERENCES

7.7 DESIGN PROJECT

Chapter Eight

State Estimation

8.1 BENCHMARK MANEUVER

8.2 LOW-PASS FILTERS

8.3 STATE ESTIMATION BY INVERTING THE SENSOR MODEL

8.3.1 Angular Rates

8.3.2 Altitude

8.3.3 Airspeed

8.3.4 Roll and Pitch Angles

8.3.5 Position, Course, and Groundspeed

8.4 COMPLEMENTARY FILTER

8.4.1 Model Free Complementary Filter

NEW MATERIAL: The objective of the complementary filter described in this section is to produce estimates of the Euler angles ϕ, θ, and ψ, when they are small. The complementary filter fuses two types of measurements. The first measurement for each angle comes from the rate gyros which measure the angular rates plus a bias. From equation (**??**), we see that when ϕ and θ are small that

$$\begin{aligned}\dot{\phi} &\approx p \\ \dot{\theta} &\approx q \\ \dot{\psi} &\approx r,\end{aligned}$$

where $\boldsymbol{\omega}_{b/i}^{b} = (p, q, r)^\top$ is the angular rate of the vehicle resolved in the body frame. Resolving equation (**??**) along each body axes we get

$$\begin{aligned}
y_{gyro,x} &= p + b_p + \eta_p \\
y_{gyro,y} &= q + b_q + \eta_q \\
y_{gyro,z} &= r + b_r + \eta_r,
\end{aligned}$$

where b_* is a slowly varying bias term, and η_* is a zero mean Gaussian random variable with known variance. Integrating the output of the rate gyros gives

$$\begin{aligned}
\hat{\phi}_{gyro}(t) &\triangleq \int_{-\infty}^{t} y_{gyro,x}(\tau)d\tau = \phi(t) + \int_{-\infty}^{t} \left(b_p(\tau) + \eta_p(\tau)\right) d\tau \\
&= \phi(t) + \beta_\phi(t) \\
\hat{\theta}_{gyro}(t) &\triangleq \int_{-\infty}^{t} y_{gyro,y}(\tau)d\tau = \theta(t) + \int_{-\infty}^{t} \left(b_q(\tau) + \eta_q(\tau)\right) d\tau \\
&= \theta(t) + \beta_\theta(t) \\
\hat{\psi}_{gyro}(t) &\triangleq \int_{-\infty}^{t} y_{gyro,z}(\tau)d\tau = \psi(t) + \int_{-\infty}^{t} \left(b_r(\tau) + \eta_r(\tau)\right) d\tau \\
&= \psi(t) + \beta_\psi(t),
\end{aligned}$$

where $\beta_*(t)$ are slowly varying signals, or signals with low frequency content.

The second measurement that will be used in the model-free complementary filter either comes from the accelerometers in the case of the roll and pitch angles, or from a magnetometer, in the case of the yaw angle. Letting $\mathbf{v}^b = (u, v, w)^\top$, using equation (**??**) for the rotation matrix in terms of the Euler angles, and resolving equation (**??**) along each body axis, we get

$$\begin{aligned}
y_{accel,x} &= \dot{u} + qw - rv + g\sin\theta + b_x + \eta_x \\
y_{accel,y} &= \dot{v} + ru - pw + g\cos\theta\sin\phi + b_y + \eta_y \\
y_{accel,z} &= \dot{w} + pv - qu + g\cos\theta\cos\phi + b_z + \eta_z,
\end{aligned}$$

where b_* are slowly varying biases, and η_* represent zero mean Gaussian noise. If the bias terms are known, for example through pre-flight calibration, then the roll and pitch angles can be approximated as

$$\begin{aligned}
\hat{\phi}_{accel}(t) &\overset{\triangle}{=} \tan^{-1}\left(\frac{y_{accel,y} - b_y}{y_{accel,z} - b_z}\right) \\
&= \tan^{-1}\left(\frac{\dot{v} + ru - pw + g\cos\theta\sin\phi + \eta_y}{\dot{w} + pv - qu + g\cos\theta\cos\phi + \eta_z}\right) \\
&= \phi(t) + \nu_\phi(t) + \eta_\phi \\
\hat{\theta}_{accel}(t) &\overset{\triangle}{=} \sin^{-1}\left(\frac{y_{accel,x} - b_x}{g}\right) \\
&= \sin^{-1}\left(\frac{\dot{u} + qw - rv + g\sin\theta + \eta_x}{g}\right) \\
&= \theta(t) + \nu_\theta(t) + \eta_\theta .
\end{aligned}$$

The approximation of ϕ and θ given by the accelerometers will be most accurate when the vehicle is not accelerating, i.e., when $\dot{u} + qw - rv = \dot{v} + ru - pw = \dot{w} + pv - qu = 0$. Since these terms are high frequency signals that are linked through the dynamics to the true roll and pitch angles, ϕ_{accel} and θ_{accel} are only good approximations at low frequencies. Therefore, we assume that ν_ϕ and ν_θ are signals with high frequency content. We note that this assumption is violated in many flight scenarios for fixed wing aircraft like when the vehicle is in a fixed loiter configuration where ν_ϕ and ν_θ are constant.

In the case of the magnetometer, the measurement is first processed by subtracting any known biases, rotating to remove the inclination and declination angles, and then rotating to the body level frame as

$$\begin{aligned}
\mathbf{m}^{v1} &= R_b^{v1}(\phi,\theta) R_z^\top(\delta) R_y^\top(\iota)(\mathbf{y}_{\text{mag}} - \mathbf{b}_{\text{mag}}) && (8.1) \\
&= R_b^{v1}(\phi,\theta) R_y(\iota) R_z(\delta)\left(\mathbf{m}^b + \boldsymbol{\nu}_{\text{mag}} + \boldsymbol{\eta}_{\text{mag}}\right) && (8.2) \\
&= \begin{pmatrix} c_\theta & s_\theta s_\phi & s_\theta c_\phi \\ 0 & c_\phi & -s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{pmatrix} \begin{pmatrix} c_\iota & 0 & s_\iota \\ 0 & 1 & 0 \\ -s_\iota & 0 & c_\iota \end{pmatrix} \begin{pmatrix} c_\delta & s_\delta & 0 \\ -s_\delta & c_\delta & 0 \\ 0 & 0 & 1 \end{pmatrix} && (8.3) \\
&\quad \cdot \left(\mathbf{m}^b + \boldsymbol{\nu}_{\text{mag}} + \boldsymbol{\eta}_{\text{mag}}\right), && (8.4)
\end{aligned}$$

where δ is the declination angle, ι is the inclination angle, $\boldsymbol{\eta}$ is zero mean Gaussian noise, and $\boldsymbol{\nu}$ represents all other magnetic interference that comes from, for example, the motor of the vehicle or flying over power lines, etc. The estimate of the heading ψ is therefore given by

$$\begin{aligned}
\hat{\psi}_{\text{mag}}(t) &= -\text{atan2}(m_y^{v1}, m_x^{v1}) \\
&= \psi(t) + \nu_\psi(t) + \eta_\psi .
\end{aligned}$$

We will assume that ν_ψ is a high frequency signal, and therefore, that a low pass filtered version of $\hat{\psi}_{\text{mag}}$ is essentially ψ over low frequencies.

We will present the derivation of the simple complementary filter for the roll angle ϕ. The derivation for the pitch and yaw angles is similar. The intuitive idea of the complementary filter is to estimate the roll angle by blending a high pass version of $\hat{\phi}_{gyro}$ and a low pass version of $\hat{\phi}_{accel}$ as

$$\hat{\phi} = H_{HPF}(s)\hat{\phi}_{gyro} + H_{LPF}(s)\hat{\phi}_{accel}$$

where $H_{HPF}(s)$ is a high pass filter and $H_{LPF}(s)$ is a low pass filter. Since

$$\begin{aligned}\hat{\phi} &= H_{HPF}(s)\hat{\phi}_{gyro} + H_{LPF}(s)\hat{\phi}_{accel} \\ &= H_{HPF}(s)\left[\phi + \beta_\phi\right] + H_{LPF}(s)\left[\phi + \nu_\phi + \eta_\phi\right] \\ &= \left[H_{HPF}(s) + H_{LPF}(s)\right]\phi + H_{HPF}(s)\beta_\phi + H_{LPF}(s)\left[\nu_\phi + \eta_\phi\right],\end{aligned}$$

if the frequency content of β_ϕ is below the cut-off frequency of H_{HPF} and the frequency content of $\nu_\phi + \eta_\phi$ is above the cut-off frequency of H_{LPF} then

$$\hat{\phi} = \left[H_{HPF}(s) + H_{LPF}(s)\right]\phi,$$

which implies that the filters H_{HPF} and H_{LPF} need to be selected so that

$$H_{HPF}(s) + H_{LPF}(s) = 1.$$

For example, if $H_{LPF}(s) = \frac{k_p}{s+k_p}$ then we need to select $H_{HPF}(s) = \frac{s}{s+k_p}$. The block diagram for a naive implementation of the complementary filter is shown in figure 8.1.

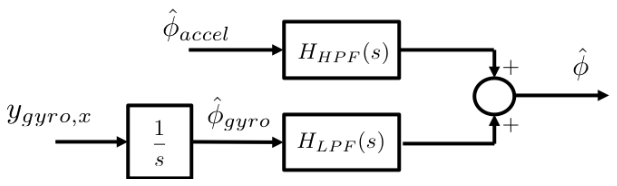


Figure 8.1: Naive complementary filter for roll angle estimation

The implementation of the complementary filter shown in figure 8.1 has several drawbacks that include the need to implement two filters and also the fact that bias rejection properties are not obvious. A better implementation strategy is to use a feedback configuration, as explained below. Consider the feedback loop show in figure 8.2. Following standard block diagram manipulation we get that

$$y(s) = \left(\frac{1}{1+PC}\right)d_o(s) + \left(\frac{P}{1+PC}\right)d_i(s) + \left(\frac{PC}{1+PC}\right)r(s)$$

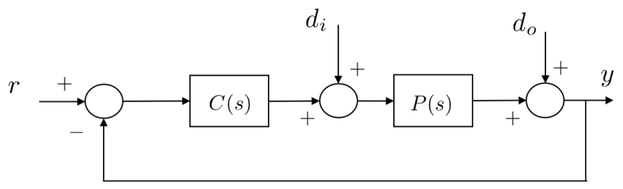


Figure 8.2: Standard feedback loop

where y is the output, r is the reference input, d_o is an output disturbance, and d_i is an input disturbance. The transfer function

$$S = \frac{1}{1+PC}$$

is called the sensitivity function, and the tranfer function

$$T = \frac{PC}{1+PC}$$

is called the complementary sensitivity function. Note that

$$S(s) + T(s) = 1.$$

If $P(s)C(s)$ is a standard loopshape that is large ($>> 1$) at low frequency and small ($<< 1$) at high frequency, then the sensitivity function $S(s)$ is a high pass filter and the complementary sensitivity function $T(s)$ is a low pass filter. Therefore, the feedback structure can be used to implement a complementary filter for the roll angle as shown in figure 8.3.

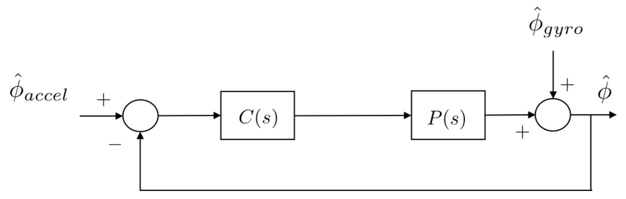


Figure 8.3: Feedback loop implementation of the complementary filter.

In order to get a first order filter where

$$S(s) = \frac{1}{1+PC} = \frac{s}{s+k_p} = \frac{1}{1+\frac{k_p}{s}}$$

we set $P(s) = 1/s$ and $C(s) = k_p$ as shown in figure 8.4. Figure 8.4 also indicates that $\hat{\phi}_{gyro}$ is the integral of the measurement $y_{gyro,x}$. Clearly, the output disturbance of $\hat{\phi}_{gyro}$ is equivalent to an input disturbance of $y_{gyro,x}$ as shown in figure 8.5.

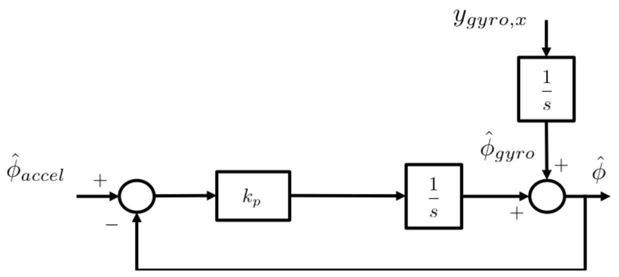


Figure 8.4: Feedback loop implementation of the complementary filter.

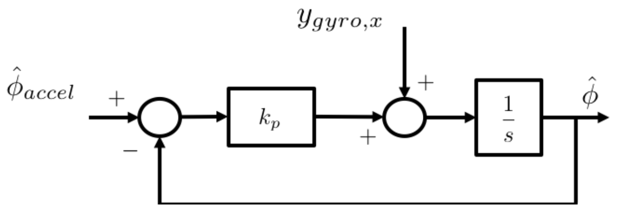


Figure 8.5: Feedback loop implementation of the complementary filter.

As mentioned above, the gyro measurement contains the true roll rate p plus a nearly constant bias b_ϕ. We can use the final value theorem to determine the response of the feedback system shown above to a constant b_ϕ as the input disturbance. The relevant transfer function is

$$\hat{\phi}(s) = \frac{P}{1+PC} b_\phi(s).$$

The final value theorem gives

$$\begin{aligned}\hat{\phi}_{ss} &= \lim_{t\to\infty} \hat{\phi}(t)\\ &= \lim_{s\to 0} s\left(\frac{P}{1+PC}\right)\frac{b}{s}\\ &= \lim_{s\to 0} \frac{bP}{1+PC}.\end{aligned}$$

When $P = \frac{1}{s}$ and $C = k_p$ we get

$$\begin{aligned}\hat{\phi}_{ss} &= \lim_{s\to 0} \frac{\frac{b}{s}}{1+\frac{k_p}{s}}\\ &= \lim_{s\to 0} \frac{b}{s+k_p}\\ &= \frac{b}{k_p}.\end{aligned}$$

Therefore, the effect of the bias can be reduced by increasing k_p but cannot be eliminated. However, using a PI structure for $C(s)$ we can completely remove the effect of the bias. The resulting architecture is shown in figure 8.6. When $C(s) = k_p + k_i/s$ the steady state response to a constant bias is

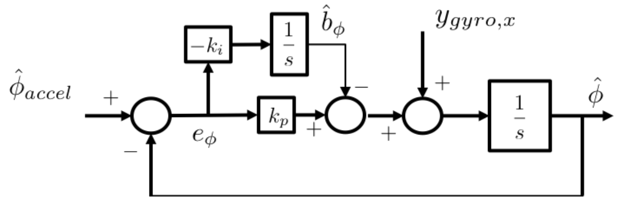


Figure 8.6: Feedback loop implementation of the complementary filter.

$$\begin{aligned}\hat{\phi}_{ss} &= \lim_{s\to 0} \frac{\frac{b}{s}}{1 + \frac{k_p + k_i/s}{s}} \\ &= \lim_{s\to 0} \frac{bs}{s^s + k_p s + k_i} \\ &= 0.\end{aligned}$$

From the block diagram, we see that the differential equations that describe the complementary filter are given by

$$\begin{aligned}\dot{\hat{b}}_\phi &= -k_i(\hat{\phi}_{accel} - \hat{\phi}) \\ \dot{\hat{\phi}} &= (y_{gyro,x} - \hat{b}_\phi) + k_p(\hat{\phi}_{accel} - \hat{\phi}).\end{aligned} \tag{8.5}$$

Note that we have introduced negative signs in the implementation of the integrator to emphasize the fact that the role of the integrator is to estimate the bias and to subtract the bias from the gyro measurement.

We also note that a Lyapunov argument can be used to prove the stability of the complementary filter given in equation (8.5) in the absence of noise. Indeed, consider the Lyapunov function candidate

$$V = \frac{1}{2}(\phi - \hat{\phi})^2 + \frac{1}{2k_i}(b_\phi - \hat{b}_\phi)^2.$$

Differentiating and using the system and filter dynamics gives

$$\begin{aligned}
\dot{V} &= (\phi - \hat{\phi})(\dot{\phi} - \dot{\hat{\phi}}) + \frac{1}{k_i}(b_\phi - \hat{b}_\phi)(\dot{b}_\phi - \dot{\hat{b}}_\phi) \\
&= (\phi - \hat{\phi})(p - (y_{gyro,x} - \hat{b}) - k_p(\hat{\phi}_{accel} - \hat{\phi})) \\
&\quad - \frac{1}{k_i}(b_\phi - \hat{b}_\phi)(-k_i)(\hat{\phi}_{accel} - \hat{\phi}) \\
&= (\phi - \hat{\phi})((y_{gyro,x} - b) - (y_{gyro,x} - \hat{b}) - k_p(\phi + \nu_\phi - \hat{\phi})) \\
&\quad + (b_\phi - \hat{b}_\phi)(\phi + \nu_\phi - \hat{\phi}) \\
&\leq -k_p(\phi - \hat{\phi})^2 + \gamma|\nu_\phi| \left\| \begin{pmatrix} \phi - \hat{\phi} \\ b - \hat{b} \end{pmatrix} \right\|.
\end{aligned}$$

If $\nu_\phi = 0$ then LaSalle's invariance principle can be used to show asymptotic convergence of the complementary filter. For non-zero ν_ϕ, the filter error is bounded by a function of the size of ν_ϕ.

We note also that for Euler angle representation for pitch and yaw, a similar derivation results in the complementary filters

$$\begin{aligned}
\dot{\hat{b}}_\theta &= -k_i(\hat{\theta}_{accel} - \hat{\theta}) \\
\dot{\hat{\theta}} &= (y_{gyro,y} - \hat{b}_\theta) + k_p(\hat{\theta}_{accel} - \hat{\theta}),
\end{aligned} \tag{8.6}$$

$$\begin{aligned}
\dot{\hat{b}}_\psi &= -k_i(\hat{\psi}_{mag} - \hat{\psi}) \\
\dot{\hat{\psi}} &= (y_{gyro,z} - \hat{b}_\psi) + k_p(\hat{\psi}_{mag} - \hat{\psi}).
\end{aligned} \tag{8.7}$$

8.4.1.1 Digital Implementation of the Simple Complementary Filter

NEW MATERIAL: Using the Euler approximation

$$\dot{z}(t) \approx \frac{z(t) - z(t - T_s)}{T_s}$$

where T_s is the sample rate, the simple complementary filter can be implemented using the following pseudo-code.

Inputs:

- The rate gyro measurements $y_{gyro,x}$, $y_{gyro,y}$, $y_{gyro,z}$.
- The accelerometer measurements $y_{accel,x}$, $y_{accel,y}$, and $y_{accel,z}$.
- The (processed) magnetometer measurement $y_{mag,\psi}$.

- Accelerometer and magnetometer biases b_x, b_y, b_z, b_ψ.
- The sample rate T_s.

Initialization:

- Initialize the estimate of the biases $\hat{b}_\phi[0]$, $\hat{b}_\theta[0]$, $\hat{b}_\psi[0]$.
- Initialize the estimate of the Euler angles $\hat{\phi}[0]$, $\hat{\theta}[0]$, $\hat{\psi}[0]$.

Step 1: Process the accelerometers and magnetometer:

- $\hat{\phi}_{accel}[n] = \tan^{-1}\left(\frac{y_{accel,y}[n]-b_y}{y_{accel,z}[n]-b_z}\right)$
- $\hat{\theta}_{accel}[n] = \sin^{-1}\left(\frac{y_{accel,x}[n]-b_x}{g}\right)$
- $\hat{\psi}_{mag}[n] = y_{mag,\psi}[n] - b_\psi$.

Step 2: Compute the errors

- $e_\phi[n] = \hat{\phi}_{accel}[n] - \hat{\phi}[n-1]$
- $e_\theta[n] = \hat{\theta}_{accel}[n] - \hat{\theta}[n-1]$
- $e_\psi[n] = \hat{\psi}_{mag}[n] - \hat{\psi}[n-1]$

Step 3: Update the bias estimates:

- $\hat{b}_\phi[n] = \hat{b}_\phi[n-1] - T_s k_i e_\phi[n]$
- $\hat{b}_\theta[n] = \hat{b}_\theta[n-1] - T_s k_i e_\theta[n]$
- $\hat{b}_\psi[n] = \hat{b}_\psi[n-1] - T_s k_i e_\psi[n]$

Step 4: Update Euler angle estimates:

- $\hat{\phi}[n] = \hat{\phi}[n-1] = T_s\Big((y_{gyro,x}[n] - \hat{b}_\phi[n]) + k_p e_\phi[n]\Big)$
- $\hat{\theta}[n] = \hat{\theta}[n-1] = T_s\Big((y_{gyro,y}[n] - \hat{b}_\theta[n]) + k_p e_\theta[n]\Big)$
- $\hat{\psi}[n] = \hat{\psi}[n-1] = T_s\Big((y_{gyro,z}[n] - \hat{b}_\psi[n]) + k_p e_\psi[n]\Big)$

RWB: Change the previous into a python implementation.

Algorithm 2 Continuous-discrete Observer

```
1: Initialize: $\hat{x} = 0$.
2: Pick an output sample rate $T_{out}$ that is less than the sample rates of the
   sensors.
3: At each sample time $T_{out}$:
4: for $i = 1$ to $N$ do {Propagate the state equation.}
5:     $\hat{x} = \hat{x} + \left(\frac{T_{out}}{N}\right) f(\hat{x}, u)$
6: end for
7: if A measurement has been received from sensor $i$ then {Measurement
   Update}
8:     $\hat{x} = \hat{x} + L_i\left(y_i - h_i(\hat{x})\right)$
9: end if
```

8.5 DYNAMIC-OBSERVER THEORY

MODIFIED MATERIAL:

8.6 DERIVATION OF THE CONTINUOUS-DISCRETE KALMAN FILTER

MODIFIED MATERIAL: ξ is a zero-mean Gaussian random process with covariance $E\{\xi(t)\xi(\tau)\} = Q\delta(t,\tau)$ where $\delta(t,\tau)$ is the Dirac delta function, and $\eta[n]$ is a zero-mean Gaussian random variable with covariance R.

8.6.0.1 *Between Measurements:*

MODIFIED MATERIAL: We can compute $E\{\tilde{x}\xi^\top\}$ as

$$\begin{aligned}
E\{\tilde{x}\xi^\top\} &= E\Big\{e^{(A-LC)t}\tilde{x}_0\xi^\top(t) + \int_0^t e^{(A-LC)(t-\tau)}\xi(\tau)\xi^\top(\tau)\,d\tau \\
&\qquad - \int_0^t e^{(A-LC)(t-\tau)}L\eta(\tau)\xi^\top(\tau)\,d\tau\Big\} \\
&= \int_0^t e^{(A-LC)(t-\tau)}Q\delta(t-\tau)\,d\tau \\
&= \frac{1}{2}Q,
\end{aligned}$$

where the $\frac{1}{2}$ is because we only use half of the area inside the delta function. Therefore, since Q is symmetric we have that P evolves between measurements as

$$\dot{P} = AP + PA^\top + Q.$$

8.6.0.2 At Measurements:

8.7 COVARIANCE UPDATE EQUATIONS

NEW MATERIAL:

Between measurements, the covariance evolves according to the equation

$$\dot{P} = AP + PA^\top + Q.$$

One way to approximate this equation numerically is to use the Euler approximation

$$P_{k+1} = P_k + T_s \left(AP_k + P_k A^\top + Q \right).$$

However, given the numerical inaccuracy due to the Euler approximation, P may not remain positive definite. We can solve this problem by discretizing in a different way. Recall that the estimation error is given by

$$\tilde{x}(t) = e^{A(t-t_0)}\tilde{x}(t_0) + \int_{t_0}^{t} e^{A(t-\tau)}\xi(\tau)d\tau.$$

Letting $t = kT_s$ and using the notation $\tilde{x}_k = \tilde{x}(kT_s)$, and assuming that $\xi(\tau)$ is held constant over each time interval and that $\xi_k = \xi(kT_s)$, we get

$$\tilde{x}_{k+1} = e^{AT_s}\tilde{x}_k + (\int_0^{T_s} e^{A\tau} d\tau)\xi_k.$$

Using the approximation

$$A_d = e^{AT_s} \approx I + AT_s + A^2\frac{T_s^2}{2}$$
$$B_d = (\int_0^{T_s} e^{A\tau} d\tau) \approx \int_0^{T_s} I d\tau = T_s I,$$

we get

$$\tilde{x}_{k+1} = A_d\tilde{x}_k + T_s\xi_k.$$

Defining the error covariance as

$$P_k = E\{\tilde{x}_k\tilde{x}_k^\top\},$$

the covariance update can be approximated as

$$\begin{aligned}P_{k+1} &= E\{\tilde{x}_{k+1}\tilde{x}_{k+1}^\top\}\\ &= E\{(A_d\tilde{x}_k + T_s\xi_k)(A_d\tilde{x}_k + T_s\xi_k)^\top\}\\ &= E\{A_d\tilde{x}_k\tilde{x}_k^\top A_d + T_s\xi_k\tilde{x}_k^\top A_d^\top + A_d\tilde{x}_k + T_s\xi_k\xi_k^\top + T_s^2\xi_k\xi_k^\top\}\\ &= A_dE\{\tilde{x}_k\tilde{x}_k^\top\ A_d^\top + T_sE\{P\xi_k\tilde{x}_k^\top\}A_d^\top + T_sA_dE\{\tilde{x}_k\xi_k^\top\} + T_s^2E\{\xi_k\xi_k^\top\}\\ &= A_dP_kA_d^\top + T_s^2Q,\end{aligned}$$

where we have made the assumption that the discrete estimation error and the process noise are uncorrelated. Note that the discrete evolution equation

$$P_{k+1} = A_dP_kA_d^\top + T_s^2Q$$

ensures that the error covariance remains positive definite.

During the measurement update equation, the covariance is updated as

$$P^+ = (I - LC)P^-. \qquad (8.8)$$

The difficulty with this form of the update equation is that numerical error may cause $(I - LC)$ to be such that P^+ is not positive definite, even if P^- is positive definite. To address this issue, we go back to the equation for covariance update given in equation (**??**):

$$P^+ = P^- - P^-C^\top L^\top - LCP^- + LCP^-C^\top L^\top + LRL^\top.$$

Rearranging we get

$$P^+ = (I - LC)P^-(I - LC)^\top + LRL^\top.$$

Note that since P^- and R are positive definite, that P^+ will also be positive definite. Therefore, rather than using equation (8.8), it is more numerically stable to use the so-called Joseph's Stabilized form:

$$P^+ = (I - LC)P^-(I - LC)^\top + LRL^\top.$$

The revised algorithm is given by Algorithm 3.

8.8 MEASUREMENT GATING

NEW MATERIAL: The performance of the Kalman filter is negatively impacted by measurement outliers. The performance of the algorithm can often be improved by testing for outliers and discarding those measurements.

Algorithm 3 Continuous-discrete Extended Kalman Filter

```
1: Initialize: $\hat{x} = 0$.
2: Pick an output sample rate $T_{out}$ which is much less than the sample
   rates of the sensors.
3: At each sample time $T_{out}$:
4: for $i = 1$ to $N$ do {Prediction Step}
5:     $T_p = T_{out}/N$
6:     $\hat{x} = \hat{x} + \left(\frac{T_{out}}{N}\right) f(\hat{x}, u)$
7:     $A = \frac{\partial f}{\partial x}(\hat{x}, u)$
8:     $A_d = I + AT_p + A^2 T_p^2$
9:     $P = A_d P A_d^\top + T_p^2 Q$
10: end for
11: if Measurement has been received from sensor $i$ then {Measurement
    Update}
12:     $C_i = \frac{\partial h_i}{\partial x}(\hat{x}, u[n])$
13:     $L_i = PC_i^\top (R_i + C_i P C_i^\top)^{-1}$
14:     $P = (I - L_i C_i) P (I - L_i C_i)^\top + L_i R_i L_i^\top$
15:     $\hat{x} = \hat{x} + L_i \left(y_i[n] - h(\hat{x}, u[n])\right)$.
16: end if
```

Define the innovation sequence

$$\begin{aligned} v_k &= y_k - C\hat{x}_k^- \\ &= Cx_k + \eta_k - C\hat{x}_k^- \\ &= C\tilde{x}_k^- + \eta_k, \end{aligned}$$

and note that since $\tilde{x}_k^-$ and η_k are zero mean that v_k is zero mean. Therefore, the covariance of v_k is given by

$$\begin{aligned} S_k &= E\{v_k v_k^\top\} \\ &= E\{(\eta_k + C\tilde{x}_k^-)(\eta_k + C\tilde{x}_k^-)^\top\} \\ &= R + CP_k^- C^\top, \end{aligned}$$

since η_k and $\tilde{x}_k^-$ are uncorrelated.

Note that the Kalman gain can be written as

$$L_k = P_k^- C^\top S_k^{-1}.$$

The random variable

$$z_k = S_k^{-\frac{1}{2}} v_k$$

is therefore an m-dimensional Gaussian random variable with zero mean and covariance of I. Therefore the random variable $w_k = z_k^\top z_k$ is a χ^2 (chi-squared) random variable with m degrees-of-freedom. We can therefore compute the probability that w_k comes from a chi-squared distribution with m- degrees-of-freedom, and only do a measurement update if the probability is above a threshold.

For example, in Python, implementation of measurement gating, where the measurement update is only performed when there is a 99% likelihood that the measurement is not an outlier is given by

```
import numpy as np
from scipy import stats
def measurement_update(self, measurement, state):
    # measurement updates
    h = self.h(self.xhat, measurement, state)
    C = jacobian(self.h, self.xhat, measurement, state)
    y = np.array([[measurement.accel_x,
                   measurement.accel_y,
                   measurement.accel_z]]).T
    S_inv = np.linalg.inv(self.R_accel + C @ self.P @ C.T)
    if stats.chi2.sf((y - h).T @ S_inv @ (y - h), df=3)>0.01:
        L = self.P @ C.T @ S_inv
        tmp = np.eye(2) - L @ C
        self.P = tmp@self.P@tmp.T + L@self.R_accel@L.T
        self.xhat = self.xhat + L @ (y - h)
```

NEW MATERIAL: The function `stats.chi2.sf(x, df=3)` is the survival function for the χ^2 distribution with `df` degrees-of-freedom and returns the area of the tail of the probability density function above x. In this implementation, the survival function must be computed for every time step. An alternative is to compute a threshold on w_k using the inverse survival function, and to only compute this once. For example, in the class constructor we could use

```
    self.accel_threshold = stats.chi2.isf(q=0.01, df=3)
```

and then use gating thereshold

```
    if (y - h).T @ S_inv @ (y - h) < self.accel_threshold
```

In matlab, we could use either

```
if chi2cdf((y-h)' * S_inv * (y-h), 3) < 0.99
```

or

```
self.accel_threshold = chi2inv(0.99, 3);
if (y-h)' * S_inv * (y-h) < self.accel_threshold
```

8.9 ATTITUDE ESTIMATION

MODIFIED MATERIAL: Implementation of the Kalman filter requires the Jacobians $\frac{\partial f}{\partial x}$ and $\frac{\partial h}{\partial x}$. Accordingly we have

$$\frac{\partial f}{\partial x} = \begin{pmatrix} q\cos\phi\tan\theta - r\sin\phi\tan\theta, & \frac{q\sin\phi + r\cos\phi}{\cos^2\theta} \\ -q\sin\phi - r\cos\phi, & 0 \end{pmatrix}$$

$$\frac{\partial h}{\partial x} = \begin{pmatrix} 0 & qV_a\cos\theta + g\cos\theta \\ -g\cos\phi\cos\theta & -rV_a\sin\theta - pV_a\cos\theta + g\sin\phi\sin\theta \\ g\sin\phi\cos\theta & (qV_a + g\cos\phi)\sin\theta \end{pmatrix}.$$

8.10 GPS SMOOTHING

MODIFIED MATERIAL: Assuming that wind and airspeed are constant we get

$$\dot{V}_g = \frac{V_a\dot{\psi}(w_e\cos\psi - w_n\sin\psi)}{V_g}.$$

MODIFIED MATERIAL: The Jacobian of f is given by

$$\frac{\partial f}{\partial x} = \begin{pmatrix} 0 & 0 & \cos\chi & -V_g\sin\chi & 0 & 0 & 0 \\ 0 & 0 & \sin\chi & V_g\cos\chi & 0 & 0 & 0 \\ 0 & 0 & -\frac{\dot{V}_g}{V_g} & 0 & \frac{-\dot{\psi}V_a\sin\psi}{V_g} & \frac{\dot{\psi}V_a\cos\psi}{V_g} & \frac{\partial\dot{V}_g}{\partial\psi} \\ 0 & 0 & -\frac{g}{V_g^2}\tan\phi & 0 & 0 & \dot{\psi}V_a\cos\psi & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

MODIFIED MATERIAL:
where $y_{GPS} = (y_{GPS,n}, y_{GPS,e}, y_{GPS,V_g}, y_{GPS,\chi}, y_{\text{wind,n}}, y_{\text{wind,e}})$, $u = \hat{V}_a$, and

8.11 FULL STATE EKF

NEW MATERIAL: In this section we will expand upon the previous discussion to include the full aircraft state.

8.11.1 Aircraft and sensor models

The model for the dynamics of the aircraft are similar to that presented in the book. If we define $\mathbf{p} = (p_n, p_e, p_d)^\top$, $\mathbf{v} = (u, v, w)^\top$, $\Theta = (\phi, \theta, \psi)^\top$, $\boldsymbol{\omega} = (p, q, r)^\top$, then we can write the dynamics as

$$\begin{aligned}
\dot{\mathbf{p}} &= R(\Theta)\mathbf{v} \\
\dot{\mathbf{v}} &= \mathbf{v}^\times \boldsymbol{\omega} + \mathbf{a} + R^\top(\Theta)\mathbf{g} \\
\dot{\Theta} &= S(\Theta)\boldsymbol{\omega} \\
\dot{\mathbf{b}} &= 0_{3\times 1} \\
\dot{\mathbf{w}} &= 0_{2\times 1},
\end{aligned}$$

where we have used the notation

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix}^\times = \begin{pmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{pmatrix},$$

and where

$$\begin{aligned}
R(\Theta) &= \begin{pmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\phi s_\theta c_\psi - c_\phi s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi c_\theta \\ c_\phi s_\theta c_\psi + s_\phi s_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi & c_\phi c_\theta \end{pmatrix} \\
S(\Theta) &= \begin{pmatrix} 1 & \sin\phi \tan\theta & \cos\phi \tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi \sec\theta & \cos\phi \sec\theta \end{pmatrix},
\end{aligned}$$

and where we have used the model for the accelerometer in equation (7.1) to get

$$\frac{1}{\mathsf{m}}\mathbf{f} = \mathbf{a} + R^\top(\Theta)\mathbf{g},$$

where $\mathbf{g} = (0, 0, g)^\top$.

We will assume that the sensors that are available are the gyroscopes, the accelerometers, the static and differential pressure sensors, the GPS north and east measurements, and the GPS ground-speed and course measurement. The models for the sensors are given by

$$
\begin{aligned}
\mathbf{y}_{\text{accel}} &= \mathbf{a} + \boldsymbol{\eta}_{\text{accel}} \\
\mathbf{y}_{\text{gyro}} &= \boldsymbol{\omega} + \mathbf{b} + \boldsymbol{\eta}_{\text{gyro}} \\
y_{\text{abs pres}} &= \rho g h_{\text{AGL}} + \eta_{\text{abs pres}} \\
y_{\text{diff pres}} &= \frac{\rho V_a^2}{2} + \eta_{\text{diff pres}} \\
y_{\text{GPS},n} &= p_n + \nu_n[n] \\
y_{\text{GPS},e} &= p_e + \nu_e[n] \\
y_{\text{GPS},V_g} &= \sqrt{(V_a \cos\psi + w_n)^2 + (V_a \sin\psi + w_e)^2} + \eta_V \\
y_{\text{GPS},\chi} &= \text{atan2}(V_a \sin\psi + w_e, V_a \cos\psi + w_n) + \eta_\chi.
\end{aligned}
$$

8.11.2 Propagation model for the EKF

Using the gyro and accelerometer models, the equations of motion can be written as

$$
\begin{aligned}
\dot{\mathbf{p}} &= R(\Theta)\mathbf{v} \\
\dot{\mathbf{v}} &= \mathbf{v}^{\times}(\mathbf{y}_{\text{gyro}} - \mathbf{b} - \boldsymbol{\eta}_{\text{gyro}}) + (\mathbf{y}_{\text{accel}} - \boldsymbol{\eta}_{\text{accel}}) + R^{\top}(\Theta)\mathbf{g} \\
\dot{\Omega} &= S(\Theta)(\mathbf{y}_{\text{gyro}} - \mathbf{b} - \boldsymbol{\eta}_{\text{gyro}}) \\
\dot{\mathbf{b}} &= 0_{3\times 1} \\
\dot{\mathbf{w}} &= 0_{2\times 1}.
\end{aligned}
$$

Defining

$$
\begin{aligned}
\mathbf{x} &= (\mathbf{p}^{\top}, \mathbf{v}^{\top}, \Theta^{\top}, \mathbf{b}^{\top}, \mathbf{w}^{\top})^{\top} \\
\mathbf{y} &= (\mathbf{y}_{\text{accel}}^{\top}, \mathbf{y}_{\text{gyro}}^{\top})^{\top},
\end{aligned}
$$

we get

$$
\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{y}) + G_g(\mathbf{x})\boldsymbol{\eta}_{\text{gyro}} + G_a \boldsymbol{\eta}_{\text{accel}} + \boldsymbol{\eta},
$$

where

$$f(\mathbf{x},\mathbf{y}) = \begin{pmatrix} R(\Theta)\mathbf{v} \\ \mathbf{v}^{\times}(\mathbf{y}_{\text{gyro}} - \mathbf{b}) + \mathbf{y}_{\text{accel}} + R^{\top}(\Theta)\mathbf{g} \\ S(\Theta)(\mathbf{y}_{\text{gyro}} - \mathbf{b}) \\ 0_{3\times 1} \\ 0_{2\times 1}, \end{pmatrix}$$

$$G_g(\mathbf{x}) = \begin{pmatrix} 0_{3\times 3} \\ -\mathbf{v}^{\times} \\ -S(\Theta) \\ 0_{3\times 3} \\ 0_{2\times 3} \end{pmatrix}$$

$$G_a = \begin{pmatrix} 0_{3\times 3} \\ -I_{3\times 3} \\ 0_{3\times 3} \\ 0_{3\times 3} \\ 0_{2\times 3} \end{pmatrix},$$

and where $\boldsymbol{\eta}$ is the general process noise associated with the model uncertainty and assumed to have covariance Q.

The Jacobian of f is

$$A(\mathbf{x},\mathbf{y}) \triangleq \frac{\partial f}{\partial x} = \begin{pmatrix} 0_{3\times 3} & R(\Theta) & \frac{\partial[R(\Theta)\mathbf{v}]}{\partial\Theta} & 0_{3\times 3} & 0_{3\times 2} \\ 0_{3\times 3} & -(\mathbf{y}_{\text{gyro}} - \mathbf{b})^{\times} & \frac{\partial[R^{\top}(\Theta)\mathbf{g}]}{\partial\Theta} & -\mathbf{v}^{\times} & 0_{3\times 2} \\ 0_{3\times 3} & 0_{3\times 3} & \frac{\partial[S(\Theta)(\mathbf{y}_{\text{gyro}} - \mathbf{b})]}{\partial\Theta} & -S(\Theta) & 0_{3\times 2} \\ 0_{3\times 3} & 0_{3\times 3} & 0_{3\times 3} & 0_{3\times 3} & 0_{3\times 2} \\ 0_{2\times 3} & 0_{2\times 3} & 0_{2\times 3} & 0_{2\times 3} & 0_{2\times 2} \end{pmatrix}.$$

It is straightforward to show that when $\mathbf{z} \in \mathbb{R}^3$ and $A(\mathbf{z})$ is a 3×3 matrix whose elements are functions of $\mathbf{z}$, then for any $\mathbf{w} \in \mathbb{R}^3$

$$\frac{\partial\left[A(\mathbf{z})\mathbf{w}\right]}{\partial\mathbf{z}} = \left(\left(\frac{\partial A(\mathbf{z})}{\partial z_1}\right)\mathbf{w},\quad \left(\frac{\partial A(\mathbf{z})}{\partial z_2}\right)\mathbf{w},\quad \left(\frac{\partial A(\mathbf{z})}{\partial z_3}\right)\mathbf{w}\right).$$

Therefore

$$\begin{aligned}
\frac{\partial\left[R(\Theta)\mathbf{v}\right]}{\partial\Theta} &= \left(\frac{\partial R(\Theta)}{\partial\phi}\mathbf{v},\quad \frac{\partial R(\Theta)}{\partial\theta}\mathbf{v},\quad \frac{\partial R(\Theta)}{\partial\psi}\mathbf{v}\right) \\
\frac{\partial\left[R^\top(\Theta)\mathbf{g}\right]}{\partial\Theta} &= \left(\frac{\partial R^\top(\Theta)}{\partial\phi}\mathbf{g},\quad \frac{\partial R^\top(\Theta)}{\partial\theta}\mathbf{g},\quad \frac{\partial R^\top(\Theta)}{\partial\psi}\mathbf{g}\right) \\
&= g\begin{pmatrix} 0 & -\cos\theta & 0 \\ \cos\theta\cos\phi & -\sin\theta\sin\phi & 0 \\ -cos\theta\sin\phi & -\sin\theta\cos\phi & 0 \end{pmatrix} \\
\frac{\partial\left[S(\Theta)(\mathbf{y}_{\text{gyro}}-\mathbf{b})\right]}{\partial\Theta} &= \begin{pmatrix} \frac{\partial S(\Theta)}{\partial\phi}(\mathbf{y}_{\text{gyro}}-\mathbf{b}) \\ \frac{\partial S(\Theta)}{\partial\theta}(\mathbf{y}_{\text{gyro}}-\mathbf{b}) \\ \frac{\partial S(\Theta)}{\partial\psi}(\mathbf{y}_{\text{gyro}}-\mathbf{b}) \end{pmatrix},
\end{aligned}$$

where $\frac{\partial R(\Theta)}{\partial\phi}$ is the matrix where each element of $R(\Theta)$ has been differentiated by ϕ.

The covariance of the noise term $G_g\boldsymbol{\eta}_{\text{gyro}} + G_a\boldsymbol{\eta}_{\text{accel}} + \boldsymbol{\eta}$ is

$$\begin{aligned}
&E[(G_g\boldsymbol{\eta}_{\text{gyro}} + G_a\boldsymbol{\eta}_{\text{accel}} + \boldsymbol{\eta})(G_g\boldsymbol{\eta}_{\text{gyro}} + G_a\boldsymbol{\eta}_{\text{accel}} + \boldsymbol{\eta})^\top] \\
&\qquad\qquad = G_g Q_{\text{gyro}} G_g^\top + G_a Q_{\text{accel}} G_a^\top + Q,
\end{aligned}$$

where

$$\begin{aligned}
Q_{\text{gyro}} &= \text{diag}([\sigma^2_{\text{gyro},x}, \sigma^2_{\text{gyro}_y}, \sigma^2_{\text{gyro}_z}]) \\
Q_{\text{accel}} &= \text{diag}([\sigma^2_{\text{accel},x}, \sigma^2_{\text{accel}_y}, \sigma^2_{\text{accel}_z}]),
\end{aligned}$$

and where

$$Q = \text{diag}([\sigma^2_{p_n}, \sigma^2_{p_e}, \sigma^2_{p_d}, \sigma^2_u, \sigma^2_v, \sigma^2_w, \sigma^2_\phi, \sigma^2_\theta, \sigma^2_\psi, \sigma^2_{b_x}, \sigma^2_{b_y}, \sigma^2_{b_z}, \sigma^2_{w_n}, \sigma^2_{w_e}])$$

are tuning parameters.

8.11.3 Sensor Update Equations

8.11.3.1 Static Pressure Sensor

The model for the static pressure sensor is

$$h_{\text{static}}(\mathbf{x}) = \rho g h = -\rho g p_d.$$

Therefore, the Jacobian is given by

$$C_{\text{static}}(\mathbf{x}) = (0, 0, -\rho g, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0).$$

8.11.3.2 Differential Pressure Sensor

The model for the differential pressure sensor is

$$h_{\text{diff}}(\mathbf{x}) = \frac{1}{2}\rho V_a^2,$$

where

$$\begin{aligned} V_a^2 &= \left(\mathbf{v} - R^\top(\Theta)\mathbf{w}\right)^\top \left(\mathbf{v} - R^\top(\Theta)\mathbf{w}\right) \\ \mathbf{w} &= (w_n, w_e, 0)^\top. \end{aligned}$$

Therefore

$$h_{\text{diff}}(\mathbf{x}) = \frac{1}{2}\rho \left(\mathbf{v} - R^\top(\Theta)\mathbf{w}\right)^\top \left(\mathbf{v} - R^\top(\Theta)\mathbf{w}\right).$$

The associated Jacobian is given by

$$C_{\text{diff}}(\mathbf{x}) = \rho \begin{pmatrix} 0_{3\times1} \\ \mathbf{v} - R^\top(\Theta)\mathbf{w} \\ -\frac{\partial\left[R^\top(\Theta)\mathbf{w}\right]}{\partial\Theta}^\top \left(\mathbf{v} - R^\top(\Theta)\mathbf{w}\right) \\ 0_{3\times1} \\ -\left(I_{2\times2}, \quad 0_{2\times1}\right) R(\Theta) \left(\mathbf{v} - R^\top(\Theta)\mathbf{w}\right) \end{pmatrix}^\top.$$

8.11.3.3 GPS North sensor

The model for the GPS north sensor is

$$h_{\text{GPS,n}}(\mathbf{x}) = p_n$$

with associated Jacobian

$$C_{\text{GPS,n}}(\mathbf{x}) = (1,0,0,0,0,0,0,0,0,0,0,0,0,0).$$

8.11.3.4 GPS East sensor

The model for the GPS east sensor is

$$h_{\text{GPS,e}}(\mathbf{x}) = p_e$$

with associated Jacobian

$$C_{\text{GPS,e}}(\mathbf{x}) = (0,1,0,0,0,0,0,0,0,0,0,0,0,0).$$

8.11.3.5 GPS ground-speed sensor

The inertial velocity in the world frame, projected onto the north-east plane is given by

$$\mathbf{v}_{g\perp} = \begin{bmatrix} I_{2\times 2}, & 0_{2\times 1} \end{bmatrix} R(\Theta)\mathbf{v}. \tag{8.9}$$

The ground-speed is given by

$$V_g(\mathbf{v}, \Theta) = \|\mathbf{v}_{g\perp}\|.$$

The model for the sensor is therefore

$$h_{\text{GPS},V_g}(\mathbf{x}) = \|PR(\Theta)\mathbf{v}\|,$$

where $P = \begin{bmatrix} I_{2\times 2}, & 0_{2\times 1} \end{bmatrix}$, and the associated Jacobian is given by

$$C_{\text{GPS},V_g}(\mathbf{x}) = \left(0_{1\times 3}, \quad \mathbf{v}^\top R^\top(\Theta) P^\top P R(\Theta), \quad \frac{\partial V_g(\mathbf{v},\Theta)}{\partial \Theta}^\top, \quad 0_{1\times 3}, \quad 0_{1\times 2}\right),$$

where $\frac{\partial V_g(\mathbf{v},\Theta)}{\partial \Theta}^\top$ is computed numerically as

$$\frac{\partial V_g(\mathbf{v},\Theta)}{\partial \Theta_i} = \frac{V_g(\mathbf{v}, \Theta + \Delta \mathbf{e}_i) - V_g(\mathbf{v},\Theta)}{\Delta},$$

where $\mathbf{e}_i$ is the 3×1 vector with one in the i^{th} location and zeros elsewhere, and Δ is a small number.

8.11.3.6 GPS course

The course angle is the direction of $\mathbf{v}_{g\perp} = (v_{g\perp n}, v_{g\perp e})^\top$ given in equation (8.9). Therefore, the course angle is given by

$$\chi(\mathbf{v}, \Theta) = \text{atan2}(v_{g\perp e}, v_{g\perp n}).$$

The model for the sensor is therefore

$$h_{\text{GPS},\chi}(\mathbf{x}) = \text{atan2}(v_{g\perp e}, v_{g\perp n}),$$

and the associated Jacobian is given by

$$C_{\text{GPS},\chi}(\mathbf{x}) = \left(0_{1\times 3}, \quad \frac{\partial \chi(\mathbf{v},\mathbf{v})}{\partial \mathbf{v}}^\top, \quad \frac{\partial \chi(\mathbf{v},\Theta)}{\partial \Theta}^\top, \quad 0_{1\times 3}, \quad 0_{1\times 2}\right),$$

where $\frac{\partial \chi(\mathbf{v},\mathbf{v})}{\partial \mathbf{v}}^\top$ and $\frac{\partial \chi(\mathbf{v},\Theta)}{\partial \Theta}^\top$ are computed numerically as

$$\frac{\partial \chi(\mathbf{v},\Theta)}{\partial \mathbf{v}_i} = \frac{\chi(\mathbf{v} + \Delta \mathbf{e}_i, \Theta) - \chi(\mathbf{v},\Theta)}{\Delta}$$
$$\frac{\partial \chi(\mathbf{v},\Theta)}{\partial \Theta_i} = \frac{\chi(\mathbf{v}, \Theta + \Delta \mathbf{e}_i) - \chi(\mathbf{v},\Theta)}{\Delta}.$$

8.11.3.7 Pseudo Measurement for Side Slip Angle

With the given sensors, the side-slip angle, or side-to-side velocity is not observable and can therefore drift. To help correct this situation, we can add a pseudo measurement on the side-slip angle by assuming that it is zero.

The side slip angle is given by

$$\beta = \frac{v_r}{V_a},$$

therefore an equivalent condition is that $v_r = 0$. The airspeed vector is given by

$$\mathbf{v}_a = \mathbf{v} - R(\Theta)^\top \mathbf{w},$$

which implies that

$$v_r(\mathbf{v}, \Theta, \mathbf{w}) = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \left(\mathbf{v} - R(\Theta)^\top \mathbf{w} \right).$$

Therefore, the model for the pseudo-sensor is given by

$$h_\beta(\mathbf{x}) = v_r(\mathbf{v}, \Theta, \mathbf{w}),$$

and the associated Jacobian is given by

$$C_\beta(\mathbf{x}) = \left(0_{1\times 3}, \quad \frac{\partial v_r(\mathbf{v},\Theta,\mathbf{w})}{\partial \mathbf{v}}^\top, \quad \frac{\partial v_r(\mathbf{v},\Theta,\mathbf{w})}{\partial \Theta}^\top, \quad 0_{1\times 3}, \quad \frac{\partial v_r(\mathbf{v},\Theta,\mathbf{w})}{\partial \mathbf{w}}^\top, \quad \right),$$

where the partial derivatives are computed numerically as

$$\begin{aligned}
\frac{\partial v_r(\mathbf{v},\Theta,\mathbf{w})}{\partial \mathbf{v}_i} &= \frac{v_r(\mathbf{v}+\Delta \mathbf{e}_i,\Theta,\mathbf{w}) - v_r(\mathbf{v},\Theta,\mathbf{w})}{\Delta} \\
\frac{\partial v_r(\mathbf{v},\Theta,\mathbf{w})}{\partial \Theta_i} &= \frac{v_r(\mathbf{v},\Theta+\Delta \mathbf{e}_i,\mathbf{w}) - v_r(\mathbf{v},\Theta,\mathbf{w})}{\Delta} \\
\frac{\partial v_r(\mathbf{v},\Theta,\mathbf{w})}{\partial \mathbf{w}_i} &= \frac{v_r(\mathbf{v},\Theta,\mathbf{w}+\Delta \mathbf{e}_i) - v_r(\mathbf{v},\Theta,\mathbf{w})}{\Delta}.
\end{aligned}$$

The measurement used in the correction step is $y_\beta = 0$.

8.11.4 Algorithm for Direct Kalman Filter

We assume that the gyro, accelerometers, and pressure sensors are sampled at rate T_s, which is the same rate that state estimates are required by the controller. We also assume that GPS is sampled at $T_{\text{GPS}} >> T_s$.

The algorithm for the direct Kalman filter is given as follows.

0. Initialize Filter.

Set

$$\hat{x} = (p_n(0), p_e(0), p_d(0), V_a(0), 0, 0, 0, 0, \psi(0), 0, 0, 0, 0, 0)^\top,$$
$$P = \text{diag}\left([e_{p_n}^2, e_{p_e}^2, e_{p_d}^2, e_u^2, e_v^2, e_w^2, e_\phi^2, e_\theta^2, e_\psi^2, e_{b_x}^2, e_{b_y}^2, e_{b_z}^2, e_{w_n}^2, e_{w_e}^2]\right),$$

where e_* is the standard deviation of the expected error in $*$.

1. At the fast sample rate T_s

1.a. Propagate $\hat{x}$ and P according to

$$\dot{\hat{x}} = f(\hat{x}, y)$$
$$\dot{P} = A(\hat{x}, y)P + PA^\top(\hat{x}, y) + G_g(\hat{x})Q_{\text{gyros}}G_g^\top(\hat{x}) + G_aQ_{\text{accel}}G_a^\top + Q$$

1.b. Update $\hat{x}$ and P with the static pressure sensor according to

$$L = P^- C_{\text{static}}^\top(\hat{x}^-)/(\sigma_{\text{abs pres}}^2 + C_{\text{static}}(\hat{x}^-)P^- C_{\text{static}}^\top(\hat{x}^-))$$
$$P^+ = (I - LC_{\text{static}}(\hat{x}^-))P^-$$
$$\hat{x}^+ = \hat{x}^- + L(y_{\text{abs pres}} - h_{\text{static}}(\hat{x}^-)).$$

1.c. Update $\hat{x}$ and P with the differential pressure sensor according to

$$L = P^- C_{\text{diff}}^\top(\hat{x}^-)/(\sigma_{\text{diff pres}}^2 + C_{\text{diff}}(\hat{x}^-)P^- C_{\text{diff}}^\top(\hat{x}^-))$$
$$P^+ = (I - LC_{\text{diff}}(\hat{x}^-))P^-$$
$$\hat{x}^+ = \hat{x}^- + L(y_{\text{diff pres}} - h_{\text{diff}}(\hat{x}^-)).$$

1.d. Update $\hat{x}$ and P with the side-slip pseudo measurement according to

$$L = P^- C_\beta^\top(\hat{x}^-)/(\sigma_\beta^2 + C_\beta(\hat{x}^-)P^- C_\beta^\top(\hat{x}^-))$$
$$P^+ = (I - LC_\beta(\hat{x}^-))P^-$$
$$\hat{x}^+ = \hat{x}^- + L(0 - h_\beta(\hat{x}^-)).$$

2. When GPS measurements are received at T_{GPS}:

2.a. Update $\hat{x}$ and P with the GPS north measurement according to

$$L = P^- C_{\text{GPS},n}^\top(\hat{x}^-)/(\sigma_{\text{GPS},n}^2 + C_{\text{GPS},n}(\hat{x}^-)P^- C_{\text{GPS},n}^\top(\hat{x}^-))$$
$$P^+ = (I - LC_{\text{GPS},n}(\hat{x}^-))P^-$$
$$\hat{x}^+ = \hat{x}^- + L(y_{\text{GPS},n} - h_{\text{GPS},n}(\hat{x}^-)).$$

2.b. Update $\hat{x}$ and P with the GPS east measurement according to

$$L = P^- C_{\text{GPS},e}^\top(\hat{x}^-)/(\sigma_{\text{GPS},e}^2 + C_{\text{GPS},e}(\hat{x}^-)P^- C_{\text{GPS},e}^\top(\hat{x}^-))$$
$$P^+ = (I - LC_{\text{GPS},e}(\hat{x}^-))P^-$$
$$\hat{x}^+ = \hat{x}^- + L(y_{\text{GPS},e} - h_{\text{GPS},e}(\hat{x}^-)).$$

2.c. Update $\hat{x}$ and P with the GPS groundspeed measurement according to

$$
\begin{aligned}
L &= P^- C^\top_{\text{GPS},V_g}(\hat{x}^-)/(\sigma^2_{\text{GPS},V_g} + C_{\text{GPS},V_g}(\hat{x}^-)P^- C^\top_{\text{GPS},V_g}(\hat{x}^-)) \\
P^+ &= (I - LC_{\text{GPS},V_g}(\hat{x}^-))P^- \\
\hat{x}^+ &= \hat{x}^- + L(y_{\text{GPS},V_g} - h_{\text{GPS},V_g}(\hat{x}^-)).
\end{aligned}
$$

2.d. Update $\hat{x}$ and P with the GPS course measurement according to

$$
\begin{aligned}
L &= P^- C^\top_{\text{GPS},\chi}(\hat{x}^-)/(\sigma^2_{\text{GPS},\chi} + C_{\text{GPS},\chi}(\hat{x}^-)P^- C^\top_{\text{GPS},\chi}(\hat{x}^-)) \\
P^+ &= (I - LC_{\text{GPS},\chi}(\hat{x}^-))P^- \\
\hat{x}^+ &= \hat{x}^- + L(y_{\text{GPS},\chi} - h_{\text{GPS},V\chi}(\hat{x}^-)).
\end{aligned}
$$

8.12 CHAPTER SUMMARY

NOTES AND REFERENCES

8.13 DESIGN PROJECT

MODIFIED MATERIAL:

8.1 Download the simulation files for this chapter from the web..

8.2 Implement the simple schemes described in Section 8.3 using low-pass filters and model inversion to estimate the states p_n, p_e, h, V_a, ϕ, θ, χ, p, q, and r. Tune the bandwidth of the low-pass filter to observe the effect. Different states may require a different filter bandwidth.

8.3 Modify the observer file to implement the extended Kalman filter for roll and pitch angles described in Section 8.9. Tune the filter until you are satisfied with the performance.

8.4 Modify the observer file to implement the extended Kalman filter for position, heading, and wind described in Section 8.10. Tune the filter until you are satisfied with the performance.

8.5 Change the simulation to use the estimated state in the autopilot as opposed to the true states. Tune autopilot and estimator gains if necessary. By changing the bandwidth of the low-pass filter, note that the stability of the closed loop system is heavily influenced by this value.

Chapter Nine

Design Models for Guidance

9.1 AUTOPILOT MODEL

9.2 KINEMATIC MODEL OF CONTROLLED FLIGHT

9.2.1 Coordinated Turn

9.2.2 Accelerating Climb

MODIFIED MATERIAL: To derive the dynamics for the flight-path angle, we will consider a pull-up maneuver in which the aircraft climbs along an arc. Define the two dimensional plane $\mathcal{P}$ as the plane containing the velocity vector $\mathbf{v}_g$ and the vector from the center of mass of the aircraft to the instantaneous center of the circle defined by the pull-up maneuver. The free-body diagram of the MAV in $\mathcal{P}$ is shown in figure 9.1. Since the airframe is rolled

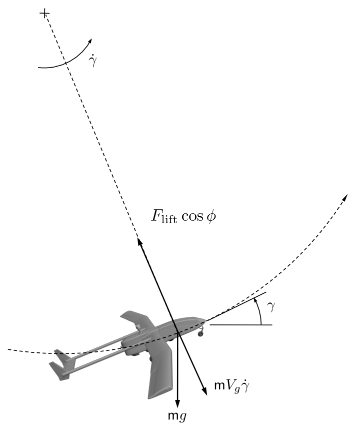


Figure 9.1: Free-body diagram for a pull-up maneuver. The MAV is at a roll angle of ϕ.

at an angle of ϕ, the projection of the lift vector onto $\mathcal{P}$ is $F_{\text{lift}}\cos\phi$. The centripetal force due to the pull-up maneuver is $\mathsf{m}V_g\dot{\gamma}$. Therefore, summing the forces in the $\mathbf{i}^b$-$\mathbf{k}^b$ plane gives

$$F_{\text{lift}}\cos\phi = \mathsf{m}V_g\dot{\gamma} + \mathsf{m}g\cos\gamma. \tag{9.1}$$

Solving for $\dot{\gamma}$ gives

$$\dot{\gamma} = \frac{g}{V_g}\left(\frac{F_{\text{lift}}}{\mathsf{m}g}\cos\phi - \cos\gamma\right). \tag{9.2}$$

9.3 KINEMATIC GUIDANCE MODELS

9.4 DYNAMIC GUIDANCE MODEL

NEW MATERIAL:

9.5 THE DUBINS AIRPLANE MODEL

Unmanned aircraft, particularly smaller systems, fly at relatively low airspeeds causing wind to have a significant effect on their performance. Since wind effects are not known prior to the moment they act on an aircraft, they are typically treated as a disturbance to be rejected in real time by the flight control system, rather than being considered during the path planning. It has been shown that vector-field-based path following methods, such as those employed in this chapter, are particularly effective at rejecting wind disturbances [3]. Treating wind as a disturbance also allows paths to be planned relative to the inertial environment, which is important as UAVs are flown in complex 3D terrain. Accordingly, when the Dubins airplane model is used for path planning, the effects of wind are not accounted for when formulating the equations of motion. In this case, the airspeed V is the same as the groundspeed, the heading angle ψ is the same as the course angle (assuming zero sideslip angle), and the flight-path angle γ is the same as the air-mass-referenced flight-path angle [4].

Figure 9.2 depicts a UAV flying with airspeed V, heading angle ψ and flight-path angle γ. Denoting the inertial position of the UAV as $\mathbf{p} = (r_n, r_e, r_d)^\top$, the kinematic relationship between the inertial velocity, $\mathbf{v} = (\dot{r}_n, \dot{r}_e, \dot{r}_d)^\top$, and the airspeed, heading angle, and flight-path angle can be easily visualized as

$$\begin{pmatrix} \dot{r}_n \\ \dot{r}_e \\ \dot{r}_d \end{pmatrix} = \begin{pmatrix} V\cos\psi\cos\gamma \\ V\sin\psi\cos\gamma \\ -V\sin\gamma \end{pmatrix},$$

where $V = \|\mathbf{v}\|$.

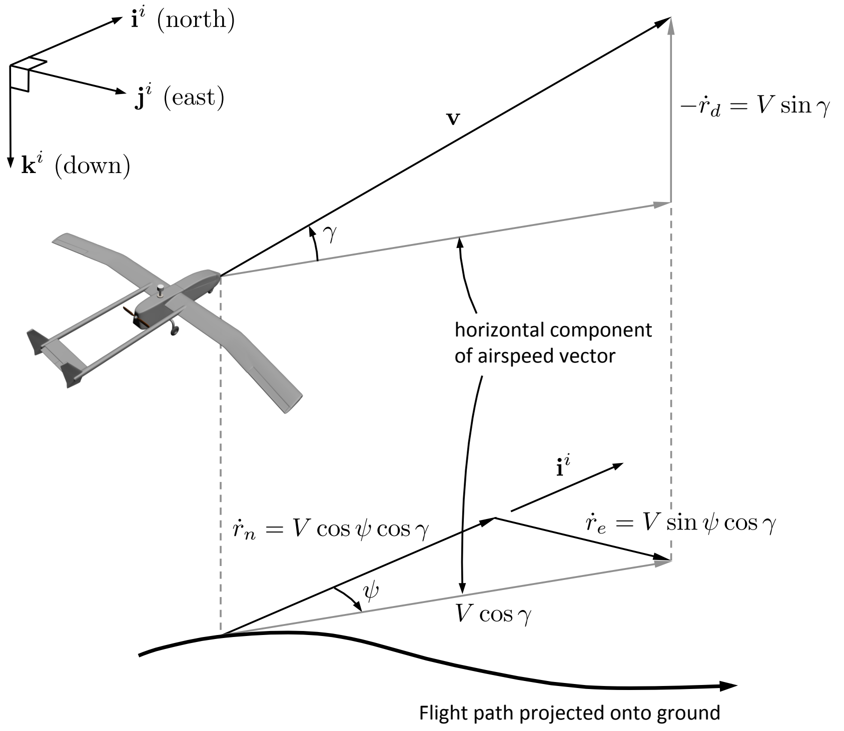

$\mathbf{i}^i$ (north)
$\mathbf{j}^i$ (east)
$\mathbf{k}^i$ (down)
$\mathbf{v}$
$-\dot{r}_d = V \sin\gamma$
γ
horizontal component of airspeed vector
$\mathbf{i}^i$
$\dot{r}_n = V \cos\psi \cos\gamma$
$\dot{r}_e = V \sin\psi \cos\gamma$
ψ
$V \cos\gamma$
Flight path projected onto ground

Figure 9.2: Graphical representation of aircraft kinematic model.

This chapter assumes that a low-level autopilot regulates the airspeed V to a constant commanded value V^c, the flight-path angle γ to the commanded flight-path angle γ^c, and the bank angle ϕ to the commanded bank angle ϕ^c. In addition, the dynamics of the flight-path angle and bank angle loops are assumed to be sufficiently fast that they can be ignored for the purpose of path following. The relationship between the heading angle ψ and the bank angle ϕ is given by the coordinated turn condition [4]

$$\dot{\psi} = \frac{g}{V} \tan\phi,$$

where g is the acceleration due to gravity.

Under the assumption that the autopilot is well tuned and the airspeed, flight-path angle, and bank angle states converge with the desired response to their commanded values, then the following kinematic model is a good description of the UAV motion

$$\begin{aligned}\dot{r}_n &= V\cos\psi\cos\gamma^c \\ \dot{r}_e &= V\sin\psi\cos\gamma^c \\ \dot{r}_d &= -V\sin\gamma^c \\ \dot{\psi} &= \frac{g}{V}\tan\phi^c.\end{aligned} \tag{9.3}$$

Physical capabilities of the aircraft place limits on the achievable bank and flight-path angles that can be commanded. These physical limits on the aircraft are represented by the following constraints

$$|\phi^c| \leq \bar{\phi} \tag{9.4}$$

$$|\gamma^c| \leq \bar{\gamma}. \tag{9.5}$$

The kinematic model given by (9.3) with the input constraints (9.4) and (9.5) will be referred to as the Dubins airplane. This model builds upon the model originally proposed for the Dubins airplane in [**?**], which is given by

$$\begin{aligned}\dot{r}_n &= V\cos\psi \\ \dot{r}_e &= V\sin\psi \\ \dot{r}_d &= u_1 \qquad |u_1| \leq 1 \\ \dot{\psi} &= u_2 \qquad |u_2| \leq 1.\end{aligned} \tag{9.6}$$

Although (9.3) is similar to (9.6), it captures the aircraft kinematics with greater accuracy and provides greater insight into the aircraft behavior, and is more consistent with commonly used aircraft guidance models. Note however, that (9.3) is only a kinematic model that does not include aerodynamics, wind effects, or engine/thrust limits. While it is not sufficiently accurate for low-level autopilot design, it is well suited for for high level path planning and path following control design. In-depth discussions of aircraft dynamic models can be found in [5, 6, 7, 1].

9.6 CHAPTER SUMMARY

NOTES AND REFERENCES

9.7 DESIGN PROJECT

MODIFIED MATERIAL:

The objective of the assignment in this chapter is to estimate the autopilot constants b_* and to develop a reduced-order design models that can be used to test and debug the guidance algorithm discussed in later chapters, prior to implementation on the full simulation model. We will focus primarily on the models given in equations (**??**) and (**??**).

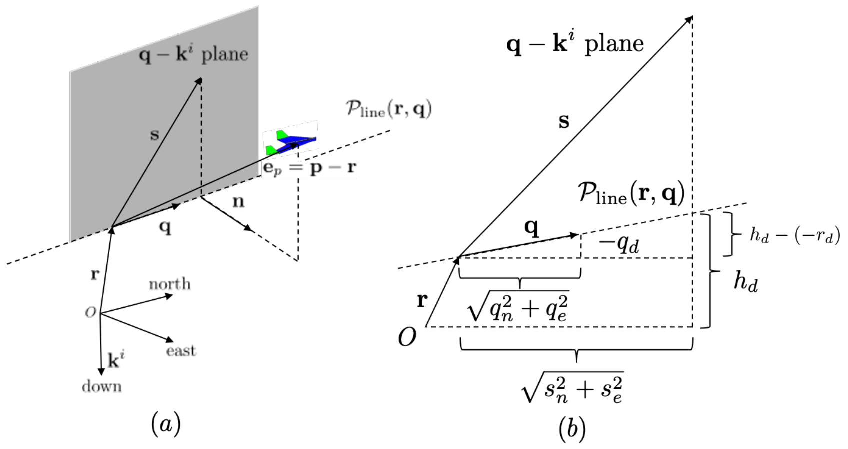
9.1 Create a Simulink S-function or Python/Matlab class that implements the model given in equation (**??**) and insert it in your MAV simulator. For different inputs χ^c, h^c, and V_a^c, compare the output of the two models, and tune the autopilot coefficients b_{V_a}, $b_{\dot{h}}$, b_h, $b_{\dot{\chi}}$, and b_χ to obtain similar behavior. You may need to re-tune the autopilot gains obtained from the previous chapter.

Chapter Ten

Straight-line and Orbit Following

10.1 STRAIGHT-LINE PATH FOLLOWING

MODIFIED MATERIAL: MODIFIED MATERIAL: Referring to the ver-



$\mathbf{q}-\mathbf{k}^i$ plane
$\mathbf{s}$
$\mathcal{P}_{\text{line}}(\mathbf{r},\mathbf{q})$
$\mathbf{e}_p = \mathbf{p} - \mathbf{r}$
$\mathbf{n}$
$\mathbf{q}$
$\mathbf{r}$
north
O
east
$\mathbf{k}^i$
down
(a)
$\mathbf{q}-\mathbf{k}^i$ plane
$\mathbf{s}$
$\mathcal{P}_{\text{line}}(\mathbf{r},\mathbf{q})$
$\mathbf{q}$
$-q_d$
$h_d - (-r_d)$
h_d
$\mathbf{r}$
$\sqrt{q_n^2+q_e^2}$
O
$\sqrt{s_n^2+s_e^2}$
(b)

Figure 10.1: Desired altitude calculation for straight-line path following in longitudinal direction.

tical plane containing the path shown in figure 10.1(b) and using similar triangles, we have the relationship

$$\frac{h_d + r_d}{\sqrt{s_n^2 + s_e^2}} = \frac{-q_d}{\sqrt{q_n^2 + q_e^2}}.$$

MODIFIED MATERIAL: From figure 10.1(b), the desired altitude for an aircraft at $\mathbf{p}$ following the straight-line path $\mathcal{P}_{\text{line}}(\mathbf{r},\mathbf{q})$ is given by

$$h_d(\mathbf{r},\mathbf{p},\mathbf{q}) = -r_d - \sqrt{s_n^2 + s_e^2}\left(\frac{q_d}{\sqrt{q_n^2 + q_e^2}}\right). \tag{10.5}$$

10.1.1 Longitudinal Guidance Strategy for Straight-line Following

10.1.2 Lateral Guidance Strategy for Straight-line Following

10.2 ORBIT FOLLOWING

NEW MATERIAL:

10.2.1 Feedforward with no wind

The commanded course angle for orbit following is given in equation (10.13) as

$$\chi^c(t) = \varphi + \lambda \left[\frac{\pi}{2} + \tan^{-1}\left(k_{\text{orbit}} \left(\frac{d-\rho}{\rho} \right) \right) \right].$$

One of the disadvantages of this strategy is that if the roll angle is used to command the course, and if the course angle is currently correct, then the roll angle will be zero. The controller can be improved by using a feedforward term on the roll angle.

If the UAV is on the orbit and there is no wind, then the desired heading rate is given by

$$\dot{\psi}^d = \lambda \frac{V_a}{R}.$$

Assuming a coordinated turn condition, the kinematics of the UAV are given by equation (9.14) as

$$\dot{\psi} = \frac{g}{V_a} \tan\phi.$$

Setting these expressions equal to each other, and solving for the roll angle, gives the feedforward term

$$\phi_{ff} = \lambda \tan^{-1}\left(\frac{V_a^2}{gR} \right). \tag{10.6}$$

In other words, if the UAV is currently on the orbit and is banked at ϕ_{ff}, and assuming that airspeed and altitude are being maintained by the autopilot, then the UAV will continue to fly on the orbit.

10.2.2 Feedforward with wind

When wind is present, the situation becomes more complicated. We will assume in this section that the wind vector $\mathbf{w} = (w_n, w_e, w_d)^\top$ is known. In the wind case, a stationary orbit of radius R relative the ground is described by the expression

$$\dot{\chi}^d(t) = \lambda \frac{V_g(t)}{R},$$

where V_g is the time varying ground speed. From equation (9.10) in the book, the coordinated turn condition in wind is given by

$$\dot{\chi} = \frac{g}{V_g} \tan\phi \cos(\chi - \psi).$$

Equating these expressions and solving for ϕ results in the feedforward term

$$\phi_{ff} = \lambda \tan^{-1}\left(\frac{V_g^2}{gR\cos(\chi - \psi)}\right).$$

From the wind triangle expression given in equation (2.12) we have that

$$\sin(\chi - \psi) = \frac{1}{V_a \cos\gamma_a}(w_e \cos\chi - w_n \sin\chi).$$

Using a simple identity from trigonometry we get

$$\cos(\chi - \psi) = \sqrt{1 - \left(\frac{1}{V_a \cos\gamma_a}(w_e \cos\chi - w_n \sin\chi)\right)^2}.$$

In a constant altitude orbit, the flight path angle $\gamma = 0$, and therefore equation (2.11), which comes from the wind triangle, becomes

$$\sin\gamma_a = \frac{w_d}{V_a},$$

from which we get that

$$\cos\gamma_a = \sqrt{1 - \left(\frac{w_d}{V_a}\right)^2},$$

which implies that

$$\cos(\chi - \psi) = \sqrt{1 - \frac{(w_e \cos\chi - w_n \sin\chi)^2}{V_a^2 - w_d^2}}.$$

In a constant altitude orbit, the flight path angle $\gamma = 0$, and therefore equation (2.10), which comes from the wind triangle, becomes

$$V_g^2 - 2\left(w_n \cos\chi + w_e \sin\chi\right) V_g + (V_w^2 - V_a^2) = 0.$$

Taking the positive root for V_g gives

$$\begin{aligned} V_g &= (w_n \cos\chi + w_e \sin\chi) + \sqrt{(w_n \cos\chi + w_e \sin\chi)^2 - V_w^2 + V_a^2} \\ &= (w_n \cos\chi + w_e \sin\chi) + \sqrt{V_a^2 - (w_n \sin\chi - w_e \cos\chi)^2 - w_d^2}. \end{aligned}$$

The term under the square root will be positive when the airspeed is greater than the windspeed, ensuring a positive groundspeed.

Therefore, the feedforward roll angle is given by

$$\phi_{ff} = \lambda \tan^{-1}\left(\frac{\left((w_n \cos\chi + w_e \sin\chi) + \sqrt{V_a^2 - (w_n \sin\chi - w_e \cos\chi)^2 - w_d^2}\right)^2}{gR\sqrt{\frac{V_a^2 - (w_n \sin\chi - w_e \cos\chi)^2 - w_d^2}{V_a^2 - w_d^2}}} \right), \tag{10.7}$$

which depends only on the wind speed, the current course angle, and the airspeed. When the wind is zero, i.e., $w_n = w_e = w_d = 0$, then equation (10.7) simplifies to equation (10.6).

NEW MATERIAL:

10.3 3D VECTOR-FIELD PATH FOLLOWING

This section shows how to develop guidance laws to ensure that the kinematic model (9.3) follows straight-line and helical paths. Section **??** shows how straight-line and helical paths are combined to produce minimum distance paths between start and end configurations.

10.3.1 Vector-field Methology

The guidance strategy will use the vector-field methodology proposed in [8], and this section provides a brief overview. The path to be followed in $\mathbb{R}^3$ is specified as the intersection of two two-dimensional manifolds given by $\alpha_1(\mathbf{r}) = 0$ and $\alpha_2(\mathbf{r}) = 0$ where α_1 and α_2 have bounded second partial derivatives, and where $\mathbf{r} \in \mathbb{R}^3$. An underlying assumption is that the path given by the intersection is connected and one-dimensional. Defining the function

$$V(\mathbf{r}) = \frac{1}{2}\alpha_1^2(\mathbf{r}) + \frac{1}{2}\alpha_2^2(\mathbf{r}),$$

gives

$$\frac{\partial V}{\partial \mathbf{r}} = \alpha_1(\mathbf{r})\frac{\partial \alpha_1}{\partial \mathbf{r}}(\mathbf{r}) + \alpha_2(\mathbf{r})\frac{\partial \alpha_2}{\partial \mathbf{r}}(\mathbf{r}).$$

Note that $-\frac{\partial V}{\partial \mathbf{r}}$ is a vector that points toward the path. Therefore following $-\frac{\partial V}{\partial \mathbf{r}}$ will transition the Dubins airplane onto the path. However simply following $-\frac{\partial V}{\partial \mathbf{r}}$ is insufficient since the gradient is zero on the path. When the Dubins airplane is on the path, its direction of motion should be perpendicular to both $\frac{\partial \alpha_1}{\partial \mathbf{r}}$ and $\frac{\partial \alpha_2}{\partial \mathbf{r}}$. Following [8] the desired velocity vector $\mathbf{u}' \in \mathbb{R}^3$ can be chosen as

$$\mathbf{u}' = -K_1 \frac{\partial V}{\partial \mathbf{r}} + K_2 \frac{\partial \alpha_1}{\partial \mathbf{r}} \times \frac{\partial \alpha_2}{\partial \mathbf{r}}, \tag{10.8}$$

where K_1 and K_2 are symmetric tuning matrices. It is shown in [8] that the dynamics $\dot{\mathbf{r}} = \mathbf{u}'$ where $\mathbf{u}'$ is given by equation (10.8), results in $\mathbf{r}$ asymptotically converging to a trajectory that follows the intersection of α_1 and α_2 if K_1 is positive definite, and where the definiteness of K_2 determines the direction of travel along the trajectory.

The problem with equation (10.8) is that the magnitude of the desired velocity $\mathbf{u}'$ may not equal V, the velocity of the Dubin's airplane. Therefore $\mathbf{u}'$ is normalized as

$$\mathbf{u} = V \frac{\mathbf{u}'}{\|\mathbf{u}'\|}. \tag{10.9}$$

Fortunately, the stability proof in [8] is still valid when $\mathbf{u}'$ is normalized as in equation (10.9).

Setting the NED components of the velocity of the Dubins airplane model given in equation (9.3) to $\mathbf{u} = (u_1, u_2, u_3)^\top$ gives

$$\begin{aligned} V \cos\psi^d \cos\gamma^c &= u_1 \\ V \sin\psi^d \cos\gamma^c &= u_2 \\ -V \sin\gamma^c &= u_3. \end{aligned}$$

Solving for the commanded flight-path angle γ^c, and the desired heading angle ψ^d results in the expressions

$$\begin{aligned} \gamma^c &= -\text{sat}_{\bar{\gamma}} \left[\sin^{-1}\left(\frac{u_3}{V}\right)\right] \\ \psi^d &= \text{atan2}(u_2, u_1), \end{aligned} \tag{10.10}$$

where atan2 is the four quadrant inverse tangent, and where the saturation function is defined as

$$\text{sat}_a[x] = \begin{cases} a & \text{if } x \geq a \\ -a & \text{if } x \leq -a \\ x & \text{otherwise} \end{cases}.$$

Assuming the inner-loop lateral-directional dynamics are accurately modeled by the coordinated-turn equation, roll-angle commands yielding desirable turn performance can be obtained from the expression

$$\phi^c = \text{sat}_{\bar{\phi}}\left[k_\phi(\psi^d - \psi)\right], \tag{10.11}$$

where k_ϕ is a positive constant.

Sections 10.3.2 and 10.3.3 applies the framework described in this section to straight-line following and helix following, respectively.

10.3.2 Straight-line Paths

A straight-line path is described by the direction of the line and a point on the line. Let $\mathbf{c}_\ell = (c_n, c_e, c_d)^\top$ be an arbitrary point on the line, and let the direction of the line be given by the desired heading angle from north ψ_ℓ, and the desired flight-path angle γ_ℓ measured from the inertial north-east plane. Therefore

$$\mathbf{q}_\ell = \begin{pmatrix} q_n \\ q_e \\ q_d \end{pmatrix} \triangleq \begin{pmatrix} \cos\psi_\ell \cos\gamma_\ell \\ \sin\psi_\ell \cos\gamma_\ell \\ -\sin\gamma_\ell \end{pmatrix}$$

is a unit vector that points in the direction of the desired line. The straight-line path is given by

$$\mathcal{P}_{\text{line}}(\mathbf{c}_\ell, \psi_\ell, \gamma_\ell) = \left\{\mathbf{r} \in \mathbb{R}^3 : \mathbf{r} = \mathbf{c}_\ell + \sigma\mathbf{q}_\ell, \sigma \in \mathbb{R}\right\}. \tag{10.12}$$

A unit vector that is perpendicular to the longitudinal plane defined by $\mathbf{q}_\ell$ is given by

$$\mathbf{n}_{\text{lon}} \triangleq \begin{pmatrix} -\sin\psi_\ell \\ \cos\psi_\ell \\ 0 \end{pmatrix}.$$

Similarly, a unit vector that is perpendicular to the lateral plane defined by $\mathbf{q}_\ell$ is given by

$$\mathbf{n}_{\text{lat}} \triangleq \mathbf{n}_{\text{lon}} \times \mathbf{q}_\ell = \begin{pmatrix} -\cos\psi_\ell \sin\gamma_\ell \\ -\sin\psi_\ell \sin\gamma_\ell \\ -\cos\gamma_\ell \end{pmatrix}.$$

It follows that $\mathcal{P}_{\text{line}}$ is given by the intersection of the surfaces defined by

$$\alpha_{\text{lon}}(\mathbf{r}) \triangleq \mathbf{n}_{\text{lon}}^\top(\mathbf{r} - \mathbf{c}_\ell) = 0 \tag{10.13}$$

$$\alpha_{\text{lat}}(\mathbf{r}) \triangleq \mathbf{n}_{\text{lat}}^\top(\mathbf{r} - \mathbf{c}_\ell) = 0. \tag{10.14}$$

Figure 10.2 shows $\mathbf{q}_\ell$, $\mathbf{c}_\ell$, and the surfaces defined by $\alpha_{\text{lon}}(\mathbf{r}) = 0$ and $\alpha_{\text{lat}}(\mathbf{r}) = 0$.

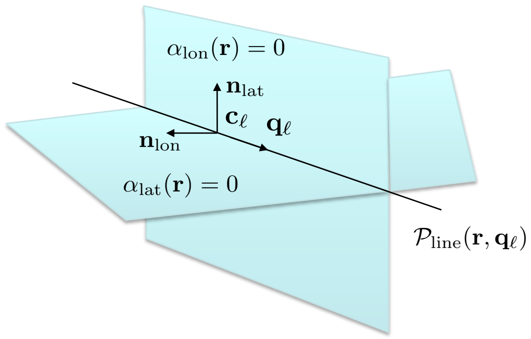

$\alpha_{\text{lon}}(\mathbf{r}) = 0$
$\mathbf{n}_{\text{lat}}$
$\mathbf{c}_\ell$
$\mathbf{q}_\ell$
$\mathbf{n}_{\text{lon}}$
$\alpha_{\text{lat}}(\mathbf{r}) = 0$
$\mathcal{P}_{\text{line}}(\mathbf{r}, \mathbf{q}_\ell)$

Figure 10.2: This figure shows how the straight-line path $\mathcal{P}_{\text{line}}(\mathbf{c}_\ell, \psi_\ell, \gamma_\ell)$ is defined by the intersection of the two surfaces given by $\alpha_{\text{lon}}(\mathbf{r}) = 0$ and $\alpha_{\text{lat}}(\mathbf{r}) = 0$.

The gradients of α_{lon} and α_{lat} are given by

$$\frac{\partial \alpha_{\text{lon}}}{\partial \mathbf{r}} = \mathbf{n}_{\text{lon}}$$
$$\frac{\partial \alpha_{\text{lat}}}{\partial \mathbf{r}} = \mathbf{n}_{\text{lat}}.$$

Therefore, before normalization, the desired velocity vector is given by

$$\mathbf{u}'_{\text{line}} = K_1 \left(\mathbf{n}_{\text{lon}}\mathbf{n}_{\text{lon}}^\top + \mathbf{n}_{\text{lat}}\mathbf{n}_{\text{lat}}^\top\right)(\mathbf{r} - \mathbf{c}_\ell) + K_2 \left(\mathbf{n}_{\text{lon}} \times \mathbf{n}_{\text{lat}}\right). \quad (10.15)$$

10.3.3 Helical Paths

A time parameterized helical path is given by

$$\mathbf{r}(t) = \mathbf{c}_h + \begin{pmatrix} R_h \cos(\lambda_h t + \psi_h) \\ R_h \sin(\lambda_h t + \psi_h) \\ -t R_h \tan \gamma_h \end{pmatrix}, \quad (10.16)$$

where $\mathbf{r}(t) = \begin{pmatrix} r_n \\ r_e \\ r_d \end{pmatrix}(t)$ is the position along the path, $\mathbf{c}_h = (c_n, c_e, c_d)^\top$ is the center of the helix, and the initial position of the helix is

$$\mathbf{r}(0) = \mathbf{c}_h + \begin{pmatrix} R_h \cos \psi_h \\ R_h \sin \psi_h \\ 0 \end{pmatrix},$$

and where R_h is the radius, $\lambda_h = +1$ denotes a clockwise helix ($N \to E \to S \to W$), and $\lambda_h = -1$ denotes a counter-clockwise helix ($N \to W \to S \to E$), and where γ_h is the desired flight-path angle along the helix.

To find two surfaces that define the helical path, the time parameterization in (10.16) needs to be eliminated. Equation (10.16) gives

$$(r_n - c_n)^2 + (r_e - c_e)^2 = R_h^2.$$

In addition, divide the east component of $\mathbf{r} - \mathbf{c}_h$ by the north component to get

$$\tan(\lambda_h t + \psi_h) = \frac{r_e - c_e}{r_n - c_n}$$

Solving for t and plugging into the third component of (10.16) gives

$$r_d - c_d = -\frac{R_h \tan\gamma_h}{\lambda_h}\left(\tan^{-1}\left(\frac{r_e - c_e}{r_n - c_n}\right) - \psi_h\right).$$

Therefore, normalizing these equations by R_h results in

$$\begin{aligned}\alpha_{\text{cyl}}(\mathbf{r}) &= \left(\frac{r_n - c_n}{R_h}\right)^2 + \left(\frac{r_e - c_e}{R_h}\right)^2 - 1\\ \alpha_{\text{pl}}(\mathbf{r}) &= \left(\frac{r_d - c_d}{R_h}\right) + \frac{\tan\gamma_h}{\lambda_h}\left(\tan^{-1}\left(\frac{r_e - c_e}{r_n - c_n}\right) - \psi_h\right).\end{aligned}$$

Normalization by R_h makes the gains on the resulting control strategy invariant to the size of the orbit.

A helical path is then defined as

$$\mathcal{P}_{\text{helix}}(\mathbf{c}_h, \psi_h, \lambda_h, R_h, \gamma_h) = \{\mathbf{r} \in \mathbb{R}^3 : \alpha_{\text{cyl}}(\mathbf{r}) = 0 \text{ and } \alpha_{\text{pl}}(\mathbf{r}) = 0\}. \tag{10.17}$$

The two surfaces $\alpha_{\text{cyl}}(\mathbf{r}) = 0$ and $\alpha_{\text{pl}}(\mathbf{r}) = 0$ are shown in figure 10.3 for parameters $\mathbf{c}_h = (0,0,0)^\top$, $R_h = 30$ m, $\gamma_h = \frac{15\pi}{180}$ rad, and $\lambda_h = +1$. The associated helical path is the intersection of the two surfaces.
The gradients of α_{cyl} and α_{pl} are given by

$$\begin{aligned}\frac{\partial\alpha_{\text{cyl}}}{\partial\mathbf{r}} &= \left(2\frac{r_n - c_n}{R_h}, \quad 2\frac{r_e - c_e}{R_h}, \quad 0\right)^\top\\ \frac{\partial\alpha_{\text{pl}}}{\partial\mathbf{r}} &= \left(\frac{\tan\gamma_h}{\lambda_h}\frac{-(r_e - c_e)}{(r_n - c_n)^2 + (r_e - c_e)^2}, \quad \frac{\tan\gamma_h}{\lambda_h}\frac{(r_n - c_n)}{(r_n - c_n)^2 + (r_e - c_e)^2}, \quad \frac{1}{R_h}\right)^\top.\end{aligned}$$

Before normalization, the desired velocity vector is given by

$$\mathbf{u}'_{\text{helix}} = K_1\left(\alpha_{\text{cyl}}\frac{\partial\alpha_{\text{cyl}}}{\partial\mathbf{r}} + \alpha_{\text{pl}}\frac{\partial\alpha_{\text{pl}}}{\partial\mathbf{r}}\right) + \lambda K_2\left(\frac{\partial\alpha_{\text{cyl}}}{\partial\mathbf{r}} \times \frac{\partial\alpha_{\text{pl}}}{\partial\mathbf{r}}\right), \tag{10.18}$$

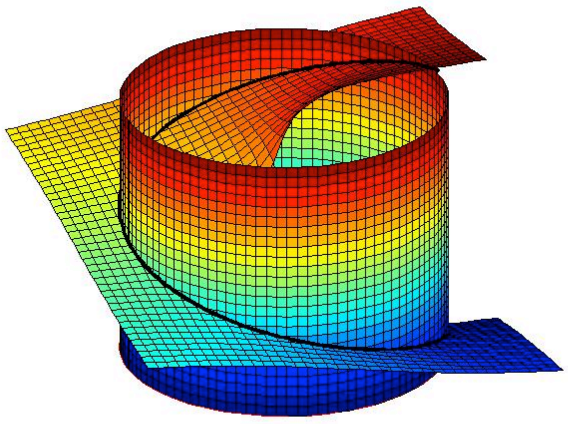


Figure 10.3: A helical path for parameters $\mathbf{c}_h = (0,0,0)^\top$, $R_h = 30$ m, $\gamma_h = \frac{15\pi}{180}$ rad, and $\lambda_h = +1$.

where

$$\frac{\partial \alpha_{\text{cyl}}}{\partial \mathbf{r}} \times \frac{\partial \alpha_{\text{pl}}}{\partial \mathbf{r}} = \frac{2}{R_h} \left(\tfrac{r_e - c_e}{R_h}, \quad -\tfrac{r_n - c_n}{R_h}, \quad \lambda_h \tan \gamma_h \right)^\top.$$

10.4 CHAPTER SUMMARY

NOTES AND REFERENCES

10.5 DESIGN PROJECT

Chapter Eleven

Path Manager

11.1 TRANSITIONS BETWEEN WAYPOINTS

MODIFIED MATERIAL: In other words, the guidance algorithm would switch at the first time instant when

$$\|\mathbf{p}(t) - \mathbf{w}_i\| \leq b,$$

where b is the size of the ball and $\mathbf{p}(t)$ is the location of the MAV. MODIFIED MATERIAL: MODIFIED MATERIAL: When the state machine

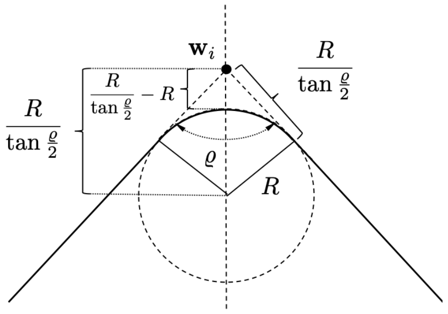

$\mathbf{w}_i$
$\frac{R}{\tan\frac{\varrho}{2}}$
$\frac{R}{\tan\frac{\varrho}{2}} - R$
$\frac{R}{\tan\frac{\varrho}{2}}$
ϱ
R

Figure 11.1: The geometry associated with inserting a fillet between waypoint segments.

is in state `state=1`, the MAV is commanded to follow the straight-line path along $\overline{\mathbf{w}_{i-1}\mathbf{w}_i}$, which is parameterized by $\mathbf{r} = \mathbf{w}_{i-1}$, and $\mathbf{q} = \mathbf{q}_{i-1}$, which are assigned in Lines **??**–**??**. Lines **??**–**??** test to see if the MAV has transitioned into the half plane shown as $\mathcal{H}_1$ in figure **??**. If the MAV has transitioned into $\mathcal{H}_1$, then the state machine is updated to `state=2`. MODIFIED MATERIAL: To be precise, let

$$|\mathcal{W}| \triangleq \sum_{i=2}^{N} \|\mathbf{w}_i - \mathbf{w}_{i-1}\|$$

be defined as the length of the waypoint path $\mathcal{W}$. Define $|\mathcal{W}|_F$ as the path length of the fillet-corrected waypoint path that will be obtained using Algorithm **??**. From figure 11.1 we see that the length of the fillet traversed by the corrected path is $R(\pi - \varrho_i)$. In addition, it is clear that the length of the straight-line segment removed from $|\mathcal{W}|$ by traversing the fillet is $2R/\tan\frac{\varrho_i}{2}$. Therefore,

$$|\mathcal{W}|_F = |\mathcal{W}| + \sum_{i=2}^{N} \left(R(\pi - \varrho_i) - \frac{2R}{\tan\frac{\varrho_i}{2}} \right), \tag{11.1}$$

where ϱ_i is given in equation (**??**).

11.2 DUBINS PATHS

11.2.1 Definition of Dubins Path

11.2.2 Path Length Computation

11.2.2.1 Case I: R-S-R

11.2.2.2 Case II: R-S-L

11.2.2.3 Case III: L-S-R

11.2.2.4 Case IV: L-S-L

11.2.3 Algorithm for Tracking Dubins Paths

MODIFIED MATERIAL:

11.3 CHAPTER SUMMARY

NOTES AND REFERENCES

11.4 DESIGN PROJECT

Algorithm 4 Find Dubins Parameters:
$(L, \mathbf{c}_s, \lambda_s, \mathbf{c}_e, \lambda_e, \mathbf{z}_1, \mathbf{q}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{q}_3)$
$= \mathsf{findDubinsParameters}(\mathbf{p}_s, \chi_s, \mathbf{p}_e, \chi_e, R)$

```
Input: Start configuration $(\mathbf{p}_s, \chi_s)$, End configuration $(\mathbf{p}_e, \chi_e)$, Radius $R$.
Require: $\|\mathbf{p}_s - \mathbf{p}_e\| \geq 3R$.
Require: $R$ is larger than minimum turn radius of MAV.
1: $\mathbf{c}_{rs} \leftarrow \mathbf{p}_s + R\mathcal{R}_z\left(\frac{\pi}{2}\right)(\cos\chi_s, \sin\chi_s, 0)^\top$,
2: $\mathbf{c}_{ls} \leftarrow \mathbf{p}_s + R\mathcal{R}_z\left(\frac{-\pi}{2}\right)(\cos\chi_s, \sin\chi_s, 0)^\top$
3: $\mathbf{c}_{re} \leftarrow \mathbf{p}_e + R\mathcal{R}_z\left(\frac{\pi}{2}\right)(\cos\chi_e, \sin\chi_e, 0)^\top$,
4: $\mathbf{c}_{le} \leftarrow \mathbf{p}_e + R\mathcal{R}_z\left(-\frac{\pi}{2}\right)(\cos\chi_e, \sin\chi_e, 0)^\top$
5: Compute $L_1$, $L_2$, $L_3$, and $L_3$ using equations (??) through (??).
6: $L \leftarrow \min\{L_1, L_2, L_3, L_4\}$.
7: if $\arg\min\{L_1, L_2, L_3, L_4\} = 1$ then
8:     $\mathbf{c}_s \leftarrow \mathbf{c}_{rs}$,   $\lambda_s \leftarrow +1$,   $\mathbf{c}_e \leftarrow \mathbf{c}_{re}$,   $\lambda_e \leftarrow +1$
9:     $\mathbf{q}_1 \leftarrow \frac{\mathbf{c}_e - \mathbf{c}_s}{\|\mathbf{c}_e - \mathbf{c}_s\|}$,
10:    $\mathbf{z}_1 \leftarrow \mathbf{c}_s + R\mathcal{R}_z\left(-\frac{\pi}{2}\right)\mathbf{q}_1$,
11:    $\mathbf{z}_2 \leftarrow \mathbf{c}_e + R\mathcal{R}_z\left(-\frac{\pi}{2}\right)\mathbf{q}_1$
12: else if $\arg\min\{L_1, L_2, L_3, L_4\} = 2$ then
13:    $\mathbf{c}_s \leftarrow \mathbf{c}_{rs}$,   $\lambda_s \leftarrow +1$,   $\mathbf{c}_e \leftarrow \mathbf{c}_{le}$,   $\lambda_e \leftarrow -1$
14:    $\ell \leftarrow \|\mathbf{c}_e - \mathbf{c}_s\|$,
15:    $\vartheta \leftarrow \text{angle}(\mathbf{c}_e - \mathbf{c}_s)$,
16:    $\vartheta_2 \leftarrow \vartheta - \frac{\pi}{2} + \sin^{-1}\frac{2R}{\ell}$
17:    $\mathbf{q}_1 \leftarrow \mathcal{R}_z\left(\vartheta_2 + \frac{\pi}{2}\right)\mathbf{e}_1$,
18:    $\mathbf{z}_1 \leftarrow \mathbf{c}_s + R\mathcal{R}_z(\vartheta_2)\mathbf{e}_1$,
19:    $\mathbf{z}_2 \leftarrow \mathbf{c}_e + R\mathcal{R}_z(\vartheta_2 + \pi)\mathbf{e}_1$
20: else if $\arg\min\{L_1, L_2, L_3, L_4\} = 3$ then
21:    $\mathbf{c}_s \leftarrow \mathbf{c}_{ls}$,   $\lambda_s \leftarrow -1$,   $\mathbf{c}_e \leftarrow \mathbf{c}_{re}$,   $\lambda_e \leftarrow +1$
22:    $\ell \leftarrow \|\mathbf{c}_e - \mathbf{c}_s\|$,
23:    $\vartheta \leftarrow \text{angle}(\mathbf{c}_e - \mathbf{c}_s)$,
24:    $\vartheta_2 \leftarrow \cos^{-1}\frac{2R}{\ell}$
25:    $\mathbf{q}_1 \leftarrow \mathcal{R}_z\left(\vartheta + \vartheta_2 - \frac{\pi}{2}\right)\mathbf{e}_1$,
26:    $\mathbf{z}_1 \leftarrow \mathbf{c}_s + R\mathcal{R}_z(\vartheta + \vartheta_2)\mathbf{e}_1$,
27:    $\mathbf{z}_2 \leftarrow \mathbf{c}_e + R\mathcal{R}_z(\vartheta + \vartheta_2 - \pi)\mathbf{e}_1$
28: else if $\arg\min\{L_1, L_2, L_3, L_4\} = 4$ then
29:    $\mathbf{c}_s \leftarrow \mathbf{c}_{ls}$,   $\lambda_s \leftarrow -1$,   $\mathbf{c}_e \leftarrow \mathbf{c}_{le}$,   $\lambda_e \leftarrow -1$
30:    $\mathbf{q}_1 \leftarrow \frac{\mathbf{c}_e - \mathbf{c}_s}{\|\mathbf{c}_e - \mathbf{c}_s\|}$,
31:    $\mathbf{z}_1 \leftarrow \mathbf{c}_s + R\mathcal{R}_z\left(\frac{\pi}{2}\right)\mathbf{q}_1$,
32:    $\mathbf{z}_2 \leftarrow \mathbf{c}_e + R\mathcal{R}_z\left(\frac{\pi}{2}\right)\mathbf{q}_1$
33: end if
34: $\mathbf{z}_3 \leftarrow \mathbf{p}_e$
35: $\mathbf{q}_3 \leftarrow \mathcal{R}_z(\chi_e)\mathbf{e}_1$
```

Chapter Twelve

Path Planning

12.1 POINT-TO-POINT ALGORITHMS

12.1.1 Voronoi Graphs

MODIFIED MATERIAL: If σ is unconstrained, then its optimizing value is

$$\sigma^* = \frac{(\mathbf{v}_1 - \mathbf{p})^\top (\mathbf{v}_1 - \mathbf{v}_2)}{\|\mathbf{v}_1 - \mathbf{v}_2\|^2},$$

and

$$D(\mathbf{v}_1, \mathbf{v}_2, \mathbf{p}) = \sqrt{\|\mathbf{p} - \mathbf{v}_1\|^2 - \frac{((\mathbf{v}_1 - \mathbf{p})^\top (\mathbf{v}_1 - \mathbf{v}_2))^2}{\|\mathbf{v}_1 - \mathbf{v}_2\|^2}}.$$

If we define

$$D'(\mathbf{v}_1, \mathbf{v}_2, \mathbf{p}) \triangleq \begin{cases} D(\mathbf{v}_1, \mathbf{v}_2, \mathbf{p}) & \text{if } \sigma^* \in [0,1] \\ \|\mathbf{p} - \mathbf{v}_1\| & \text{if } \sigma^* < 0 \\ \|\mathbf{p} - \mathbf{v}_2\| & \text{if } \sigma^* > 1, \end{cases}$$

then the distance between the point set $\mathcal{Q}$ and the line segment $\overline{\mathbf{v}_1\mathbf{v}_2}$ is given by

$$D(\mathbf{v}_1, \mathbf{v}_2, \mathcal{Q}) = \min_{\mathbf{p} \in \mathcal{Q}} D'(\mathbf{v}_1, \mathbf{v}_2, \mathbf{p}).$$

12.1.2 Rapidly Exploring Random Trees

12.1.2.1 RRT Waypoint Planning over 3-D Terrain

12.1.2.2 RRT Dubins Path Planning in a 2-D Obstacle Field

12.2 CHAPTER SUMMARY

NOTES AND REFERENCES

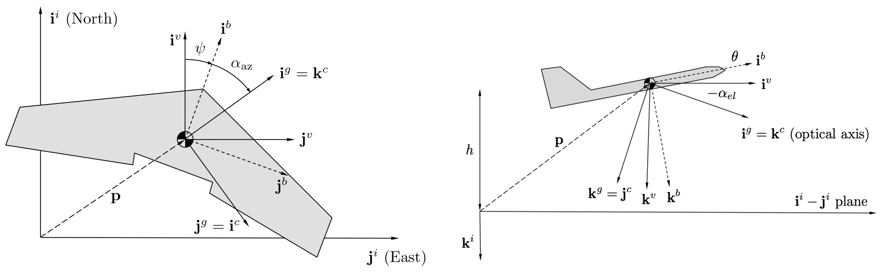
12.3 DESIGN PROJECT

Chapter Thirteen

Vision-guided Navigation

13.1 GIMBAL AND CAMERA FRAMES AND PROJECTIVE GEOMETRY

MODIFIED MATERIAL:



(a) Top view. (b) Side view.

Figure 13.1: A graphic showing the relationship between the gimbal and camera frames and the vehicle and body frames.

13.1.1 Camera Model

13.2 GIMBAL POINTING

MODIFIED MATERIAL: The next step is to determine the desired azimuth and elevation angles that will align the optical axis with $\check{\boldsymbol{\ell}}_d^b$. In the camera frame, the optical axis is given by $(0, 0, 1)^c$. Therefore, the objective is to select the commanded gimbal angles α_{az}^c and α_{el}^c so that

$$\check{\boldsymbol{\ell}}_d^b \triangleq \begin{pmatrix} \check{\ell}_{xd}^b \\ \check{\ell}_{yd}^b \\ \check{\ell}_{zd}^b \end{pmatrix} = \mathcal{R}_g^b(\alpha_{\text{az}}^c, \alpha_{\text{el}}^c)\mathcal{R}_c^g \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{13.1}$$

$$= \begin{pmatrix} \cos\alpha_{\text{el}}^c \cos\alpha_{\text{az}}^c & -\sin\alpha_{\text{az}}^c & -\sin\alpha_{\text{el}}^c \cos\alpha_{\text{az}}^c \\ \cos\alpha_{\text{el}}^c \sin\alpha_{\text{az}}^c & \cos\alpha_{\text{az}}^c & -\sin\alpha_{\text{el}}^c \sin\alpha_{\text{az}}^c \\ \sin\alpha_{\text{el}}^c & 0 & \cos\alpha_{\text{el}}^c \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{13.2}$$

$$= \begin{pmatrix} \cos\alpha_{\text{el}}^c \cos\alpha_{\text{az}}^c \\ \cos\alpha_{\text{el}}^c \sin\alpha_{\text{az}}^c \\ \sin\alpha_{\text{el}}^c \end{pmatrix}. \tag{13.3}$$

13.3 GEOLOCATION

13.3.1 Range to Target Using the Flat-earth Model

13.3.2 Geolocation Using an Extended Kalman Filter

13.4 ESTIMATING TARGET MOTION IN THE IMAGE PLANE

13.4.1 Digital Low-pass Filter and Differentiation

13.4.2 Apparent Motion Due to Rotation

13.5 TIME TO COLLISION

13.5.1 Computing Time to Collision from Target Size

13.5.2 Computing Time to Collision from the Flat-earth Model

13.6 PRECISION LANDING

13.7 CHAPTER SUMMARY

NOTES AND REFERENCES

13.8 DESIGN PROJECT

Appendix A

Nomenclature and Notation

NOMENCLATURE

NOTATION

Appendix B

Quaternions

Appendix C

Simulation Using Object Oriented Programming

C.1 INTRODUCTION

NEW MATERIAL: Object oriented programming is well adapted to the implementation of complex dynamic systems like the simulator project developed in the book. This supplement will explain how the Python programming language can be used to implement the project. We will outline one particular way that the simulator can be constructed, as well as the specific data structures (messages) that are passed between elements of the architecture.

The architecture outlined in the book is shown for convenience in figure C.1. The basic idea beh

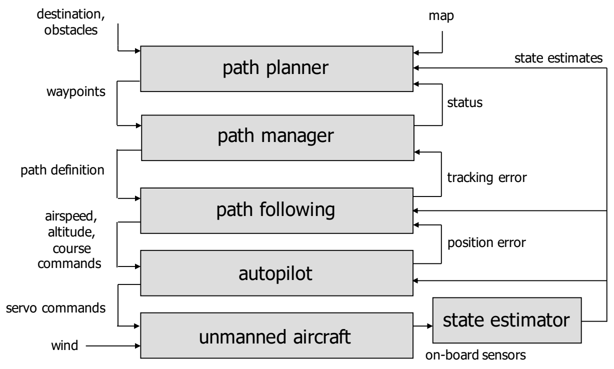


Figure C.1: Architecture outlined in the uavbook.

C.2 NUMERICAL SOLUTIONS TO DIFFERENTIAL EQUATIONS

NEW MATERIAL:

This section provides a brief overview of techniques for numerically solving ordinary differential equations, when the initial value is specified. In particular, we are interested in solving the differential equation

$$\frac{dx}{dt}(t) = f(x(t), u(t)) \tag{C.1}$$

with initial condition $x(t_0) = x_0$, where $x(t) \in \mathbb{R}^n$, and $u(t) \in \mathbb{R}^m$, and where we assume that $f(x, u)$ is sufficiently smooth to ensure that unique solutions to the differential equation exist for every x_0.

Integrating both sides of equation (C.1) from t_0 to t gives

$$x(t) = x(t_0) + \int_{t_0}^{t} f(x(\tau), u(\tau))\, d\tau. \tag{C.2}$$

The difficulty with solving equation (C.2) is that $x(t)$ appears on both sides of the equation, and cannot therefore be solved explicitly for $x(t)$. Numerical solutions techniques for ordinary differential equations attempt to approximate the solution by approximating the integral in equation (C.2). Most techniques break the solution into small time increments, usually equal to the sample rate, which we denote by T_s. Accordingly we write equation (C.2) as

$$\begin{aligned} x(t) &= x(t_0) + \int_{t_0}^{t-T_s} f(x(\tau), u(\tau))\, d\tau + \int_{t-T_s}^{t} f(x(\tau), u(\tau))\, d\tau \\ &= x(t - T_s) + \int_{t-T_s}^{t} f(x(\tau), x(\tau))\, d\tau. \end{aligned} \tag{C.3}$$

The most simple form of numerical approximation for the integral in equation (C.3) is to use the left endpoint method as shown in figure C.2, where

$$\int_{t-T_s}^{t} f(x(\tau), u(\tau))\, d\tau \approx T_s f(x(t - T_s), u(t - T_s)).$$

Using the notation $x_k \stackrel{\triangle}{=} x(t_0 + kT_s)$ we get the numerical approximation to the differential equation in equation (C.1) as

$$x_k = x_{k-1} + T_s f(x_{k-1}, u_{k-1}), \quad x_0 = x(t_0). \tag{C.4}$$

Equation (C.4) is called the Euler method, or the Runge-Kutta first order method (RK1).

While the RK1 method is often effective for numerically solving ODEs, it typically requires a small sample size T_s. The advantage of the RK1 method is that it only requires one evaluation of $f(\cdot, \cdot)$.

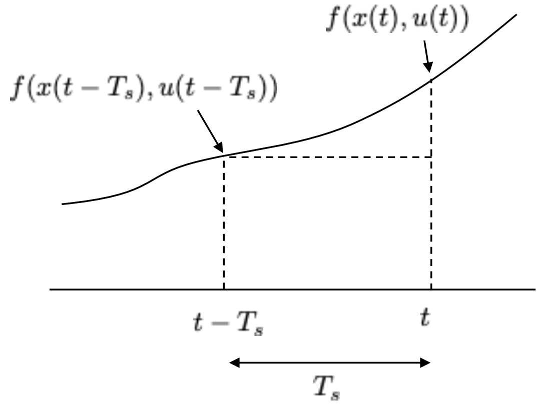


Figure C.2: Approximation of integral using the left endpoint method results in Runge-Kutta first order method (RK1).

To improve the numerical accuracy of the solution, a second order method can be derived by approximating the integral in equation (C.3) using the trapezoidal rule shown in figure **??**, where

$$\int_a^b f(\xi)d\xi \approx (b-a)\left(\frac{f(a)+f(b)}{2}\right).$$

Accordingly,

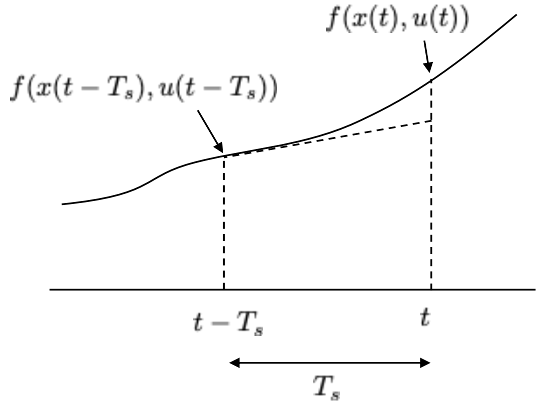


Figure C.3: Approximation of integral using the trapezoidal rule results in Runge-Kutta second order method (RK2).

$$\int_{t-T_s}^{t} f(x(\tau), u(\tau))d\tau \approx \frac{T_s}{2}\left[f(x(t-T_s), u(t-T_s)) + f(x(t), u(t))\right].$$

Since $x(t)$ is unknown, we use the RK1 method to approximate it as

$$x(t) \approx x(t-T_s) + T_s f(x(t-T_s), u(t-T_s)).$$

In addition, we typically assume that $u(t)$ is constant over the interval $[t - T_s, T_s)$ to get

$$\begin{aligned}\int_{t-T_s}^{t} f(x(\tau), u(\tau))d\tau \approx \frac{T_s}{2}\Big[& f(x(t-T_s), u(t-T_s)) \\ &+ f(x(t-T_s) + T_s f(x(t-T_s), u(t-T_s)), u(t-T_s))\Big].\end{aligned}$$

Accordingly, the numerical integration routine can be written as

$$\begin{aligned} x_0 &= x(t_0) \\ X_1 &= f(x_{k-1}, u_{k-1}) \\ X_2 &= f(x_{k-1} + T_s X_1, u_{k-1}) \\ x_k &= x_{k-1} + \frac{T_s}{2}\left(X_1 + X_2\right). \end{aligned} \tag{C.5}$$

Equations (C.5) is the Runge-Kutta second order method or RK2. It requires two function calls to $f(\cdot,\cdot)$ for every time sample.

The most common numerical method for solving ODEs is the Runge-Kutta forth order method RK4. The RK4 method is derived using Simpson's Rule

$$\int_a^b f(\xi)d\xi \approx \left(\frac{b-a}{6}\right)\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right),$$

which is derived by approximating the area under the integral using a parabola, as shown in figure **??**. The standard strategy is to write the approximation as

$$\int_a^b f(\xi)\xi \approx \left(\frac{b-a}{6}\right)\left(f(a) + 2f\left(\frac{a+b}{2}\right) + 2f\left(\frac{a+b}{2}\right) + f(b)\right),$$

and then to use

$$X_1 = f(a) \tag{C.6}$$

$$X_2 = f(a + \frac{T_s}{2}X_1) \approx f\left(\frac{a+b}{2}\right) \tag{C.7}$$

$$X_3 = f(a + \frac{T_s}{2}X_2) \approx f\left(\frac{a+b}{2}\right) \tag{C.8}$$

$$X_4 = f(a + T_s X_3) \approx f(b), \tag{C.9}$$

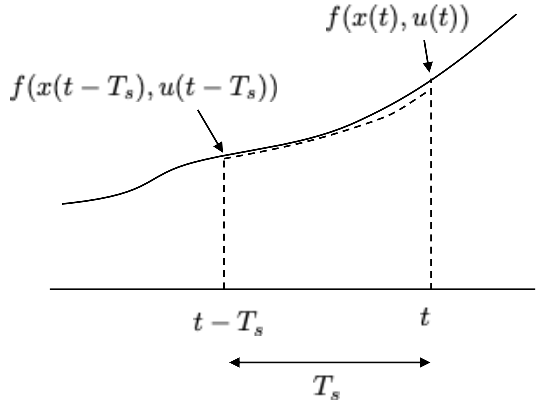


Figure C.4: Approximation of integral using Simpson's rule results in Runge-Kutta fourth order method (RK4).

resulting in the RK4 numerical integration rule

$$\begin{aligned}
x_0 &= x(t_0) \\
X_1 &= f(x_{k-1}, u_{k-1}) \\
X_2 &= f(x_{k-1} + \frac{T_s}{2} X_1, u_{k-1}) \\
X_3 &= f(x_{k-1} + \frac{T_s}{2} X_2, u_{k-1}) \\
X_4 &= f(x_{k-1} + T_s X_3, u_{k-1}) \\
x_k &= x_{k-1} + \frac{T_s}{6} (X_1 + 2X_2 + 2X_3 + X_4) \,.
\end{aligned} \tag{C.10}$$

It can be shown that the RK4 method matches the Taylor series approximation of the integral in equation (C.3) up to the fourth order term. Higher order methods can be derived, but generally do not result in significantly better numerical approximations to the ODE. The step size T_s must still be chosen to be sufficiently small to result in good solutions. It is also possible to develop adaptive step size solvers. For example the ODE45 algorithm in Matlab, solves the ODE using both an RK4 and an RK5 algorithm. The two solutions are then compared and if they are sufficiently different, then the step size is reduced. If the two solutions are close, then the step size can be increased.

A Python implementation of the RK4 method for the dynamics

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ \sin(x_1) + u \end{pmatrix},$$
$$y = x_1$$

with initial conditions $\mathbf{x} = (0,0)^\top$, would be as follows:

```
import numpy as np
import matplotlib.pyplot as plt

class dynamics:
    def __init__(self, Ts):
        self.ts = Ts
        # set initial conditions
        self._state = np.array([[0.0], [0.0]])

    def update(self, u):
        ''' Integrate ODE using Runge-Kutta RK4 algorithm'''
        ts = self.ts
        X1 = self._derivatives(self._state, u)
        X2 = self._derivatives(self._state + ts/2 * X1, u)
        X3 = self._derivatives(self._state + ts/2 * X2, u)
        X4 = self._derivatives(self._state + ts * X3, u)
        self._state += ts/6 * (X1 + 2*X2 + 2*X3 + X4)

    def output(self):
        '''Returns system output'''
        y = self._state.item(0)
        return y

    def _derivatives(self, x, u):
        '''Return xdot for the dynamics xdot = f(x, u)'''
        x1 = x.item(0)
        x2 = x.item(1)
        x1_dot = x2
        x2_dot = np.sin(x1) + u
        x_dot = np.array([[x1_dot], [x2_dot]])
        return x_dot

# initialize the system
Ts = 0.01  # simulation step size
system = dynamics(Ts)

# initialize the simulation time and plot data
sim_time = 0.0
time = [sim_time]
y = [system.output()]

# main simulation loop
while sim_time < 10.0:
    u=np.sin(sim_time)  # set input to u=sin(t)
    system.update(u)  # propagate the system dynamics
    sim_time += Ts   # increment the simulation time

    # update data for plotting
    time.append(sim_time)
    y.append(system.output())

# plot output y
plt.plot(time, y)
plt.xlabel('time (s)')
plt.ylabel('output y')
plt.show()
```

Appendix D

Animation

In the study of aircraft dynamics and control, it is essential to be able to visualize the motion of the airframe. In this appendix we describe how to create 3D animation using Python, Matlab, and Simulink.

D.1 3D ANIMATION USING PYTHON

D.2 3D ANIMATION USING MATLAB

D.3 3D ANIMATION USING SIMULINK

In the study of aircraft dynamics and control, it is essential to be able to visualize the motion of the airframe. In this section we describe how to create animations in Matlab/Simulink.

D.4 HANDLE GRAPHICS IN MATLAB

```
>> plot_handle=plot(t,sin(t))

>> set(plot_handle, 'YData', cos(t))

>> plot_handle1 = plot(t,sin(t))
>> hold on
>> plot_handle2 = plot(t,cos(t))

>> set(plot_handle2, 'YData', cos(2*t))
```

D.5 ANIMATION EXAMPLE: INVERTED PENDULUM

```
function handle = drawBase(y, width, height, gap, handle, mode)
  X = [y-width/2, y+width/2, y+width/2, y-width/2];
  Y = [gap, gap, gap+height, gap+height];
  if isempty(handle),
    handle = fill(X,Y,'m','EraseMode', mode);
  else
```

```
    set(handle,'XData',X,'YData',Y);
  end

function handle = drawRod(y, theta, L, gap, height, handle, mode)
  X = [y, y+L*sin(theta)];
  Y = [gap+height, gap + height + L*cos(theta)];
  if isempty(handle),
    handle = plot(X, Y, 'g', 'EraseMode', mode);
  else
    set(handle, 'XData',X,'YData',Y);
  end

function drawPendulum(u)
    % process inputs to function
    y        = u(1);
    theta    = u(2);
    t        = u(3);

    % drawing parameters
    L = 1;
    gap = 0.01;
    width = 1.0;
    height = 0.1;

    % define persistent variables
    persistent base_handle
    persistent rod_handle

    % first time function is called,
    % initialize plot and persistent vars
    if t==0,
        figure(1), clf
        track_width=3;
        plot([-track_width,track\_width],[0,0],'k');
        hold on
        base_handle = drawBase(y, width, height, gap, [], 'normal');
        rod_handle  = drawRod(y, theta, L, gap, height, [], 'normal');
        axis([-track_width, track_width, -L, 2*track_width-L]);

    % at every other time step, redraw base and rod
    else
        drawBase(y, width, height, gap, base_handle);
        drawRod(y, theta, L, gap, height, rod_handle);
    end
```

D.6 ANIMATION EXAMPLE: SPACECRAFT USING LINES

```
function XYZ=spacecraftPoints
  % define points on the spacecraft in local NED coordinates
  XYZ = [...
          1     1     0;... % point 1
          1    -1     0;... % point 2
         -1    -1     0;... % point 3
         -1     1     0;... % point 4
          1     1     0;... % point 1
          1     1    -2;... % point 5
          1    -1    -2;... % point 6
          1    -1     0;... % point 2
          1    -1    -2;... % point 6
         -1    -1    -2;... % point 7
         -1    -1     0;... % point 3
         -1    -1    -2;... % point 7
         -1     1    -2;... % point 8
         -1     1     0;... % point 4
         -1     1    -2;... % point 8
          1     1    -2;... % point 5
          1     1     0;... % point 1
          1.5   1.5   0;... % point 9
          1.5  -1.5   0;... % point 10
          1    -1     0;... % point 2
          1.5  -1.5   0;... % point 10
         -1.5  -1.5   0;... % point 11
         -1    -1     0;... % point 3
         -1.5  -1.5   0;... % point 11
         -1.5   1.5   0;... % point 12
         -1     1     0;... % point 4
         -1.5   1.5   0;... % point 12
          1.5   1.5   0;... % point 9
          ]';

function XYZ=rotate(XYZ,phi,theta,psi)
  % define rotation matrix
  R_roll = [...
          1, 0, 0;...
          0, cos(phi), -sin(phi);...
          0, sin(phi), cos(phi)];
  R_pitch = [...
          cos(theta), 0, sin(theta);...
          0, 1, 0;...
          -sin(theta), 0, cos(theta)];
  R_yaw = [...
          cos(psi), -sin(psi), 0;...
          sin(psi), cos(psi), 0;...
          0, 0, 1];
  R = R_roll*R_pitch*R_yaw;
  % rotate vertices
  XYZ = R*XYZ;
```

```
        function XYZ = translate(XYZ,pn,pe,pd)
          XYZ = XYZ + repmat([pn;pe;pd],1,size(XYZ,2));
```

Drawing the spacecraft at the desired location is accomplished using the following Matlab code:

```
function handle = drawSpacecraftBody(pn,pe,pd,phi,theta,psi, handle, mode)
  % define points on spacecraft in local NED coordinates
  NED = spacecraftPoints;
  % rotate spacecraft by phi, theta, psi
  NED = rotate(NED,phi,theta,psi);
  % translate spacecraft to [pn; pe; pd]
  NED = translate(NED,pn,pe,pd);
   % transform vertices from NED to XYZ (for matlab rendering)
   R = [...
       0, 1, 0;...
       1, 0, 0;...
       0, 0, -1;...
       ];
   XYZ = R*NED;
  % plot spacecraft
   if isempty(handle),
     handle = plot3(XYZ(1,:),XYZ(2,:),XYZ(3,:),'EraseMode', mode);
   else
         set(handle,'XData',XYZ(1,:),'YData',XYZ(2,:),'ZData',XYZ(3,:));
     drawnow
   end
```

D.7 ANIMATION EXAMPLE: SPACECRAFT USING VERTICES AND FACES

```
function [V, F, patchcolors]=spacecraft
  % Define the vertices (physical location of vertices)
  V = [...
        1     1     0;... % point 1
     1   -1     0;... % point 2
    -1   -1     0;... % point 3
    -1    1     0;... % point 4
        1     1    -2;... % point 5
     1   -1    -2;... % point 6
    -1   -1    -2;... % point 7
    -1    1    -2;... % point 8
        1.5   1.5   0;... % point 9
     1.5  -1.5   0;... % point 10
    -1.5  -1.5   0;... % point 11
    -1.5   1.5   0;... % point 12
  ];
  % define faces as a list of vertices numbered above
  F = [...
    1, 2,  6,  5;...  % front
    4, 3,  7,  8;... % back
    1, 5,  8,  4;... % right
    2, 6,  7,  3;... % left
    5, 6,  7,  8;... % top
    9, 10, 11, 12;... % bottom
   ];
  % define colors for each face
  myred    = [1, 0, 0];
  mygreen  = [0, 1, 0];
  myblue   = [0, 0, 1];
  myyellow = [1, 1, 0];
  mycyan   = [0, 1, 1];
  patchcolors = [...
    myred;...     % front
    mygreen;...   % back
    myblue;...    % right
    myyellow;...  % left
    mycyan;...    % top
    mycyan;...    % bottom
   ];
end
```

The vertices are shown in figure **??** and are defined in Lines 3–16. The faces are defined by listing the indices of the points that define the face. For example, the front face, defined in Line 19, consists of points $1-2-6-5$. Faces can be defined by N-points, where the matrix that defines the faces has N columns, and the number of rows is the number of faces. The color for each face is defined in Lines 32–39. Matlab code that draws the spacecraft body is listed below.

```
function handle = drawSpacecraftBody(pn, pe, pd,
                                     phi, theta, psi,
                                     handle, mode)
  % define points on spacecraft
  [V, F, patchcolors] = spacecraft;
  % rotate and then translate spacecraft
  V = rotate(V', phi, theta, psi)';
  V = translate(V', pn, pe, pd)';
  % transform NED to ENU for rendering
  R = [...
      0, 1, 0;...
      1, 0, 0;...
      0, 0, -1;...
      ];
  V = V*R;
  if isempty(handle)
  handle = patch('Vertices', V, 'Faces', F,...
                 'FaceVertexCData',patchcolors,...
                 'FaceColor', 'flat',...
```

```
                    'EraseMode', mode);
  else
    set(handle, 'Vertices',V, 'Faces', F);
  end
end
```

Appendix E

Airframe Parameters

Table E.1: Physical parameters for the Aerosonde UAV.

Physical Param	**Value**	**Motor Param**	**Value**
m	11.0 kg	V_{max}	44.4 V
J_x	0.824 kg-m^2	D_{prop}	0.5 m
J_y	1.135 kg-m^2	K_V	145 RPM/V
J_z	1.759 kg-m^2	K_Q	0.0659 V-s/rad
J_{xz}	0.120 kg-m^2	R_{motor}	0.042 Ohms
S	0.55 m^2	i_0	1.5 A
b	2.9 m	C_{Q_2}	-0.01664
c	0.19 m	C_{Q_1}	0.004970
ρ	1.2682 kg/m^3	C_{Q_0}	0.005230
e	0.9	C_{T_2}	-0.1079
		C_{T_1}	-0.06044
		C_{T_0}	0.09357

Table E.2: Aerodynamic coefficients for the Aerosonde UAV.

Longitudinal Coef.	**Value**	**Lateral Coef.**	**Value**
C_{L_0}	0.23	C_{Y_0}	0
C_{D_0}	0.043	C_{l_0}	0
C_{m_0}	0.0135	C_{n_0}	0
C_{L_α}	5.61	C_{Y_β}	-0.83
C_{D_α}	0.030	C_{l_β}	-0.13
C_{m_α}	-2.74	C_{n_β}	0.073
C_{L_q}	7.95	C_{Y_p}	0
C_{D_q}	0	C_{l_p}	-0.51
C_{m_q}	-38.21	C_{n_p}	-0.069
$C_{L_{\delta_e}}$	0.13	C_{Y_r}	0
$C_{D_{\delta_e}}$	0.0135	C_{l_r}	0.25
$C_{m_{\delta_e}}$	-0.99	C_{n_r}	-0.095
M	50	$C_{Y_{\delta_a}}$	0.075
α_0	0.47	$C_{l_{\delta_a}}$	0.17
ϵ	0.16	$C_{n_{\delta_a}}$	-0.011
C_{D_p}	0	$C_{Y_{\delta_r}}$	0.19
		$C_{l_{\delta_r}}$	0.0024
		$C_{n_{\delta_r}}$	-0.069

Appendix F

Trim and Linearization

F.1 NUMERICAL COMPUTATION OF THE JACOBIAN

NEW MATERIAL:

The MAV dynamics can be described at a high level by the dynamics

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

where $x \in \mathbb{R}^{12}$ and $u \in \mathbb{R}^4$. Linearization requires the computation of the Jacobians $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial u}$, where

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

where

$$\frac{\partial f}{\partial x_i} = \begin{pmatrix} \frac{\partial f_1}{\partial x_i} \\ \frac{\partial f_2}{\partial x_i} \\ \vdots \\ \frac{\partial f_n}{\partial x_i} \end{pmatrix}$$

is a column vector. By definition we have that

$$\frac{\partial f}{\partial x_i}(x) = \lim_{\epsilon \to 0} \frac{f(x + \epsilon e_i) - f(x)}{\epsilon},$$

where $e_i \in \mathbb{R}^{12}$ is the column vector of all zeros except for a one in the i^{th} row. Therefore, by letting ϵ be a small fixed number, we get that

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

which can be computed numerically. Python code that computes the Jacobian of f at (x, u) is given below.

```
import numpy as np
def df_dx(x, u):
    # take partial of f(x,u) with respect to x
    eps = 0.01  # deviation
    A = np.zeros((12, 12))  # Jacobian of f wrt x
    f_at_x = f(x,u)
    for i in range(0, 12):
        x_eps = np.copy(x)
        x_eps[i][0] += eps # add eps to ith state
        f_at_x_eps = f(x_eps, u)
        df_dxi = (f_at_x_eps - f_at_x) / eps
        A[:,i] = df_dxi[:,0]
    return A
```

NEW MATERIAL:

The simulation developed in the book can be done using either Euler angles or a unit quaternion to represent the attitude of the MAV. The state space and transfer function models are all with respect to Euler angles. Therefore, if unit quaternions are used instead of Euler angles, then we need a technique to compute the Jacobians with respect to Euler angles. Let

$$x_e = (p_n, p_e, p_d, u, v, w, \phi, \theta, \psi, p, q, r)^\top \in \mathbb{R}^{12}$$

represent the state using Euler angles and let

$$x_q = (p_n, p_e, p_d, u, v, w, e_0, e_x, e_y, e_z, p, q, r)^\top \in \mathbb{R}^{13}$$

represent the state using unit quaternions for attitude. Let $T : \mathbb{R}^{13} \to \mathbb{R}^{12}$ be the mapping that converts quaternion representation so that $x_e = T(x_q)$. Specifically,

$$T\begin{pmatrix} p_n \\ p_e \\ p_d \\ u \\ v \\ w \\ e_0 \\ e_x \\ e_y \\ e_z \\ p \\ q \\ r \end{pmatrix} = \begin{pmatrix} p_n \\ p_e \\ p_d \\ u \\ v \\ w \\ \Theta_\phi(e_0, e_x, e_y, e_z) \\ \Theta_\theta(e_0, e_x, e_y, e_z) \\ \Theta_\psi(e_0, e_x, e_y, e_z) \\ p \\ q \\ r \end{pmatrix},$$

where $\Theta(\mathbf{e})$ is given in equation (**??**).

Since $x_e = T(x_q)$ we have

$$\begin{aligned}\dot{x}_e &= \frac{\partial T}{\partial x_q}\dot{x}_q \\ &= \frac{\partial T}{\partial x_q} f(x_q, u) \\ &= \frac{\partial T}{\partial x_q} f(T^{-1}(x_e), u),\end{aligned}$$

where $T^{-1}(x_e)$ is given in equation (**??**), and where

$$\frac{\partial T}{\partial x_q} = \begin{pmatrix} I_{3\times3} & 0_{3\times3} & 0_{3\times4} & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3} & 0_{3\times4} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & \frac{\partial\Theta}{\partial \mathbf{e}} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times4} & I_{3\times3}, \end{pmatrix} \tag{F.1}$$

and where $\frac{\partial\Theta}{\partial \mathbf{e}}$ can be approximated numerically using the technique outlined above.

F.2 TRIM AND LINEARIZATION IN SIMULINK

```
[X,U,Y,DX]=TRIM('SYS',X0,U0,Y0,IX,IU,IY,DX0,IDX),
```

MODIFIED MATERIAL: For our situation we know that

$$\dot{x}^* = ([\text{don't care}], [\text{don't care}], V_a^* \sin\gamma^*, 0, 0, 0, 0, 0, V_a^*/R^* \cos(\gamma^*), 0, 0, 0)^\top,$$

therefore we let

```
  DX = [0; 0; Va*sin(gamma); 0; 0; 0; 0; 0; Va/R*cos(gamma); 0;
0; 0]
  IDX = [3; 4; 5; 6; 7; 8; 9; 10; 11; 12].
```

Similarly, the initial state, inputs, and outputs can be specified as `X0 = [0; 0; 0; Va; 0; 0; 0; gamma; 0; 0; 0; 0]`

```
  IX0 = []
  U0 = [0; 0; 0; 1]
  IU0 = []
  Y0 = [Va; 0; 0]
  IY0 = [1,3].
```

```
[A,B,C,D]=LINMOD('SYS',X,U),
```

where `X` and `U` are the state and input about which the Simulink diagram is to be linearized, and `[A,B,C,D]` is the resulting state-space model. If the `linmod` command is used on the Simulink diagram shown in figure **??**, where there are twelve states and four inputs, the resulting state space equations will include the models given in equations (**??**) and (**??**). To obtain equation (**??**) for example, you could use the following steps:

```
[A,B,C,D]=linmod(filename, x_trim, u_trim)
A_lat = A([5, 10, 12, 7, 9], [5, 10, 12, 7, 9]);
B_lat = B([5, 10, 12, 7, 9], [3, 4]);
```

NEW MATERIAL:

If your Simulink implementation uses quaternions for the state, then `linmod` returns state space equations with respect to x_q, whereas the standard state space equations require the state space equations with respect to x_e. Suppose that the quaternion state space equations are given by

$$\dot{x}_q = f_q(x_q, u),$$

and that the euler state space equations are given by

$$\dot{x}_e = f_e(x_e, u),$$

and recall from the discussion above that $x_e = T(x_q)$. Differentiating with respect to time we get

$$\begin{aligned}\dot{x}_e &= \frac{\partial T}{\partial x_q}\dot{x}_q \\ &= \frac{\partial T}{\partial x_q} f_q(x_q, u) \\ &= \frac{\partial T}{\partial x_q} f_q(T^{-1}(x_e), u) \\ &= f_e(x_e, u).\end{aligned}$$

Therefore the Jacobian with respect to x_e is given by

$$\begin{aligned}\frac{\partial f_e}{\partial x_e} &= \frac{\partial \frac{\partial T}{\partial x_q} f_q(T^{-1}(x_e), u)}{\partial x_e} \\ &= \frac{\partial T}{\partial x_q}\frac{\partial f_q}{\partial x_q}\frac{\partial T^{-1}}{\partial x_e},\end{aligned}$$

where $\frac{\partial T}{\partial x_q}$ given in equation (F.1) and where

$$\frac{\partial T^{-1}}{\partial x_q} = \begin{pmatrix} I_{3\times 3} & 0_{3\times 3} & 0_{3\times 3} & 0_{3\times 3} \\ 0_{3\times 3} & I_{3\times 3} & 0_{3\times 3} & 0_{3\times 3} \\ 0_{4\times 3} & 0_{4\times 3} & \frac{\partial Q}{\partial \boldsymbol{\Theta}} & 0_{4\times 3} \\ 0_{3\times 3} & 0_{3\times 3} & 0_{3\times 3} & I_{3\times 3}, \end{pmatrix} \tag{F.2}$$

and where $Q(\Theta)$ is given by equation (**??**) and where $\frac{\partial Q}{\partial \Theta}$ can be computed numerically.

The Jacobian for the B matrix can be found similarly as

$$\frac{\partial f_e}{\partial u} = \frac{\partial T}{\partial x_q}\frac{\partial f_q}{\partial u}.$$

F.3 TRIM AND LINEARIZATION IN MATLAB

NEW MATERIAL:

In Matlab, trim can be computed using the built in optimization function `fmincon`. In general `fmincon` can be configured to minimize nonlinear functions with general constraints. For example, to minimize the function $J(x) = x\top x - \gamma$ subject to the constraints that $x_1 = x_2$ and $\sin(x_3) \leq -0.5$, the appropriate Matlab code is as follows:

```
x0 = [10; 5; -20; 8];  % initial guess for the optimum
gam = 3;  % parameter passed into objective function
xopt = fmincon(@objective, x0, [], [], [], []
                [], [], @constraints, [], gam);

% objective function to be minimized
function J = objective(x, gam)
    J = x*x' - gam;
end

% nonlinear constraints for trim optimization
function [c, ceq] = constraints(x, gam)
        % inequality constraints c(x)<=0
    c(1) = sin(x(3)) + 0.5;
    equality constraints ceq(x)==0
    ceq(1) = x(1)-x(2);
end
```

F.4 TRIM AND LINEARIZATION IN PYTHON

NEW MATERIAL:

In Python, trim can be computed using the scipy optimize library. For example, to minimize the function $J(x) = x\top x - \gamma$ subject to the constraints that $x_1 = x_2$ and $\sin(x_3) \leq -0.5$, the appropriate Python code is as follows: import numpy as np from scipy.optimize import minimize

```
# initial guess for the optimum
x0 = np.array([[10; 5; -20; 8]]).T;
gam = 3;  # parameter passed into objective function
# first constraint is equality constraint x1==x2
# second constraint is inequality constraint sin(x3)<=-0.5
cons = ({'type': 'eq', 'fun': lambda x: x[0]-x[1],},
        {'type': 'ineq', 'fun': lambda x: np.sin(x[2])+0.5})
# solve the minimization problem
res = minimize(objective, x0,
                           method='SLSQP',
                           args = (gam),
               constraints=cons,
               options={'ftol': 1e-10, 'disp': True})
# extract optimal state
xopt = np.array([res.x]).T

# objective function to be minimized
def objective(x, gam):
        J = np.dot(x.T, x) - gam
    return J
```

Appendix G

Essentials from Probability Theory

Appendix H

Sensor Parameters

H.1 RATE GYROS

H.2 ACCELEROMETERS

H.3 PRESSURE SENSORS

H.4 DIGITAL COMPASS/MAGNETOMETER

H.5 GPS

Appendix I

Useful Formulas and other Information

I.1 CONVERSION FROM KNOTS TO MPH

I.2 DENSITY OF AIR

Bibliography

[1] T. R. Yechout, S. L. Morris, D. E. Bossert, and W. F. Hallgren, *Introduction to Aircraft Flight Mechanics*. AIAA Education Series, American Institute of Aeronautics and Astronautics, 2003.

[2] R. F. Stengel, *Flight Dynamics*. Princeton University Press, 2004.

[3] D. R. Nelson, D. B. Barber, T. W. McLain, and R. W. Beard, "Vector Field Path Following for Miniature Air Vehicles," *IEEE Transactions on Robotics*, vol. 37, pp. 519–529, jun 2007.

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

[5] W. F. Phillips, *Mechanics of Flight*. Wiley, 2004.

[6] B. L. Stevens and F. L. Lewis, *Aircraft Control and Simulation*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2nd ed., 2003.

[7] R. C. Nelson, *Flight Stability and Automatic Control*. Boston, Massachusetts: McGraw-Hill, 2nd ed., 1998.

[8] V. M. Goncalves, L. C. A. Pimenta, C. A. Maia, B. C. O. Durtra, G. A. S. Pereira, B. C. O. Dutra, and G. A. S. Pereira, "Vector Fields for Robot Navigation Along Time-Varying Curves in n-Dimensions," *IEEE Transactions on Robotics*, vol. 26, pp. 647–659, aug 2010.

Index