

Small Unmanned Aircraft

Small Unmanned Aircraft

*Theory and Practice
Supplement*

Randal W. Beard
Timothy W. McLain

PRINCETON UNIVERSITY PRESS
PRINCETON AND OXFORD

Contents

1. Introduction	1
1.1 System Architecture	1
1.2 Design Models	4
1.3 Design Project	6
2. Coordinate Frames	9
2.1 Rotation Matrices	10
2.2 MAV Coordinate Frames	12
2.3 Airspeed, Wind Speed, and Ground Speed	18
2.4 The Wind Triangle	21
2.5 Differentiation of a Vector	25
2.6 Chapter Summary	26
2.7 Design Project	27
3. Kinematics and Dynamics	29
3.1 State Variables	29
3.2 Kinematics	30
3.3 Rigid-body Dynamics	32
3.4 Chapter Summary	37
3.5 Design Project	38
4. Forces and Moments	41
4.1 Gravitational Forces	41
4.2 Aerodynamic Forces and Moments	42
4.3 Propulsion Forces and Moments	55
4.4 Atmospheric Disturbances	60
4.5 Chapter Summary	64
4.6 Design Project	65
5. Linear Design Models	67
5.1 Coordinated Turn	67
5.2 Trim Conditions	69
5.3 Transfer Function Models	72
5.4 Linear State-space Models	75
5.5 Chapter Summary	75
5.6 Design Project	75
6. Autopilot Design	77

6.1 Successive Loop Closure	77
6.2 Total Energy Control	93
6.3 LQR Control	95
6.4 Chapter Summary	97
6.5 Design Project	98
7. Sensors for MAVs	99
7.1 Accelerometers	99
7.2 Rate Gyros	103
7.3 Pressure Sensors	105
7.4 Digital Compasses	110
7.5 Global Positioning System	113
7.6 Chapter Summary	120
7.7 Design Project	120
8. State Estimation	123
8.1 Benchmark Maneuver	123
8.2 Low-pass Filters	124
8.3 State Estimation by Inverting the Sensor Model	125
8.4 Complementary Filter	129
8.5 Dynamic-observer Theory	138
8.6 Derivation of the Continuous-Discrete Kalman Filter	139
8.7 Covariance Update Equations	143
8.8 Measurement Gating	145
8.9 Attitude Estimation	147
8.10 GPS Smoothing	149
8.11 Full State EKF	151
8.12 Chapter Summary	160
8.13 Design Project	161
9. Design Models for Guidance	163
9.1 Autopilot Model	163
9.2 Kinematic Model of Controlled Flight	164
9.3 Kinematic Guidance Models	167
9.4 Dynamic Guidance Model	169
9.5 The Dubins Airplane Model	171
9.6 Chapter Summary	174
9.7 Design Project	174
10. Straight-line and Orbit Following	177
10.1 Straight-line Path Following	178
10.2 Orbit Following	183
10.3 3D Vector-field Path Following	188
10.4 Chapter Summary	194
10.5 Design Project	195
11. Path Manager	197
11.1 Transitions Between Waypoints	197

11.2 Dubins Paths	204
11.3 Chapter Summary	214
11.4 Design Project	215
12. Path Planning	217
12.1 Point-to-Point Algorithms	218
12.2 Coverage Algorithms	233
12.3 Chapter Summary	236
12.4 Design Project	237
13. Vision-guided Navigation	239
13.1 Gimbal and Camera Frames and Projective Geometry	239
13.2 Gimbal Pointing	242
13.3 Geolocation	244
13.4 Estimating Target Motion in the Image Plane	247
13.5 Time to Collision	251
13.6 Precision Landing	253
13.7 Chapter Summary	257
13.8 Design Project	258
A. Nomenclature and Notation	259
B. Quaternions	265
B.1 Another look at complex numbers	265
B.2 Introduction to Quaternions	266
B.3 Quaternion Rotations	266
B.4 Conversion Between Euler Angles and Quaternions	268
B.5 Conversion Between Quaternions and Rotation Matrices	269
B.6 Quaternion Kinematics	270
B.7 Aircraft Kinematic and Dynamic Equations	270
C. Simulation Using Object Oriented Programming	273
C.1 Introduction	273
C.2 Numerical Solutions to Differential Equations	273
D. Animation	281
D.1 3D Animation Using Python	281
D.2 3D Animation Using Matlab	281
D.3 3D Animation Using Simulink	281
E. Airframe Parameters	285
F. Trim and Linearization	287
F.1 Numerical Computation of the Jacobian	287
F.2 Trim and Linearization in Simulink	289
F.3 Trim and Linearization in Matlab	292
F.4 Trim and Linearization in Python	292

Bibliography	295
Index	301

Chapter One

Introduction

1.1 SYSTEM ARCHITECTURE

The objective of this book is to prepare the reader to do research in the exciting and rapidly developing field of autonomous navigation, guidance, and control of unmanned air vehicles. The focus is on the design of the software algorithms required for autonomous and semi-autonomous flight. To work in this area, researchers must be familiar with a wide range of topics, including coordinate transformations, aerodynamics, autopilot design, state estimation, path planning, and computer vision. The aim of this book is to cover these essential topics, focusing in particular on their application to small and miniature air vehicles, which we denote by the acronym MAV.

In the development of the topics, we have in mind the software architecture shown in Figure 1.1. The block labeled *unmanned aircraft* in Figure 1.1

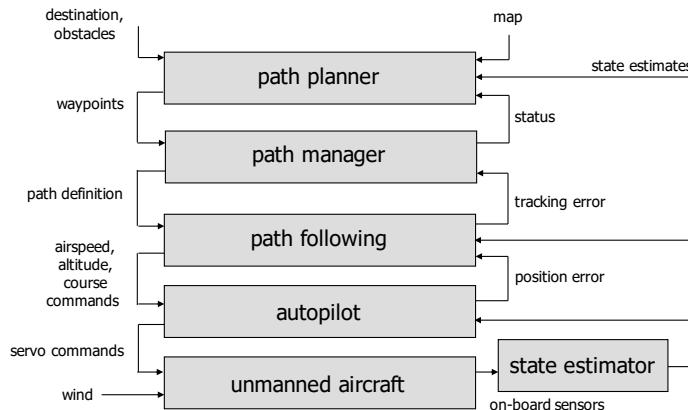


Figure 1.1: The system architecture that will be assumed throughout the book. The path planner produces straight-line or Dubins paths through an obstacle field. The path manager switches between orbit following and straight-line path following to maneuver along the waypoint paths. The path-following block produces commands to the low-level autopilot, which controls the airframe. Each of the blocks relies on estimates of the states produced by filtering the onboard sensors.

is the six-degree-of-freedom (DOF) physical aircraft that responds to servo

command inputs (elevator, aileron, rudder, and throttle) and wind and other disturbances. The mathematical models required to understand fixed-wing flight are complicated and are covered in Chapters 2 to 5 and Chapter 9. In particular, in Chapter 2 we discuss coordinate frames and transformations between frames. A study of coordinate frames is required since most specifications for MAVs are given in the inertial frame (e.g., orbit a specific coordinate), whereas most of the sensor measurements are with respect to the body frame, and the actuators exert forces and torques in the body frame. In Chapter 3 we develop the kinematic and dynamic equations of motion of a rigid body. In Chapter 4 we describe the aerodynamic forces and moments that act on fixed-wing aircraft. Chapter 5 begins by combining the results of Chapters 3 and 4 to obtain a six-DOF, 12-state, nonlinear dynamic model for a MAV. While incorporating the fidelity desired for simulation purposes, the six-DOF model is fairly complicated and cumbersome to work with. The design and analysis of aircraft control approaches are more easily accomplished using lower-order linear models. Linear models that describe small deviations from trim are derived in Chapter 5, including linear transfer function and state-space models.

The block labeled *autopilot* in Figure 1.1 refers to the low-level control algorithms that maintain roll and pitch angles, airspeed, altitude, and course. Chapter 6 introduces the standard technique of successive loop closure to design the autopilot control laws. Nested control loops are closed one at a time, with inner loops maintaining roll and pitch angles and outer loops maintaining airspeed, altitude, and course.

The autopilot and the higher level blocks rely on accurate state estimates obtained by dynamically filtering the onboard sensors which include accelerometers, rate gyros, pressure sensors, magnetometers, and GPS receivers. A description of these sensors and their mathematical models is given in Chapter 7. Because it is not possible to measure all the states of small unmanned aircraft using standard sensors, state estimation plays an important role. Descriptions of several state-estimation techniques that are effective for MAVs are given in Chapter 8.

A complete model of the flight dynamics coupled with the autopilot and state estimation techniques represents a high dimensional, highly complex, nonlinear system of equations. The full model of the system is too complicated to facilitate the development of high level guidance algorithms. Therefore, Chapter 9 develops low-order nonlinear equations that model the closed-loop behavior of the system. These models are used in subsequent chapters to develop guidance algorithms.

One of the primary challenges with MAVs is flight in windy conditions. Since airspeeds in the range of 20 to 40 mph are typical for MAVs, and since wind speeds at several hundred feet above ground level (AGL) almost

always exceed 10 mph, MAVs must be able to maneuver effectively in wind. Traditional trajectory tracking methods used in robotics do not work well for MAVs. The primary difficulty with these methods is the requirement to be in a particular location at a particular time, which cannot properly take into account the variations in ground speed caused by the unknown and changing effects of the wind. Alternatively, path-following methods that simply maintain the vehicle on a desired path have proven to be effective in flight tests. Chapter 10 describes the algorithms and methods used to provide the capabilities of the *path following* block in Figure 1.1. We will focus exclusively on straight-line paths and circular orbits and arcs. Other useful paths can be built up from these straight-line and circular path primitives.

The block labeled *path manager* in Figure 1.1 is a finite-state machine that converts a sequence of waypoint configurations (positions and orientations) into sequences of straight-line paths and circular arcs that can be flown by the MAV. This makes it possible to simplify the path planning problem so that the *path planner* produces either a sequence of straight-line paths that maneuver the MAV through an obstacle field, or a Dubin's path that maneuvers through the obstacle field. Chapter 11 describes the *path manager*, while Chapter 12 describes the *path planner*. For path planning we consider two classes of problems. The first class of problems is point-to-point algorithms, where the objective is to maneuver from a start position to an end position while avoiding a set of obstacles. The second class of problems is search algorithms, where the objective is to cover a region, potentially having no-go regions, with a sensor footprint.

Almost all applications involving MAVs require the use of an onboard electro-optical/infrared (EO/IR) video camera. The typical objective of the camera is to provide visual information to the end user. Since MAV payload capacities are limited, however, it makes sense to also use the video camera for navigation, guidance, and control. Effective use of camera information is currently an active research topic. In Chapter 13 we discuss several potential uses of video cameras on MAVs, including geolocation and vision-based landing. Geolocation uses a sequence of images as well as the onboard sensors to estimate the world coordinates of objects on the ground. Vision-based landing uses video images captured by the MAV to guide it to a target identified in the image plane. We feel that an understanding of these problems will enable further investigations in vision-based guidance of MAVs.

In Chapter 13, we use the software architecture shown in Figure 1.2, where the *path planner* block has been replaced with the block *vision-based guidance*. However, the vision-based guidance laws interact with the architecture in the same manner as the path planner. The modularity of the architecture is one of its most appealing features.

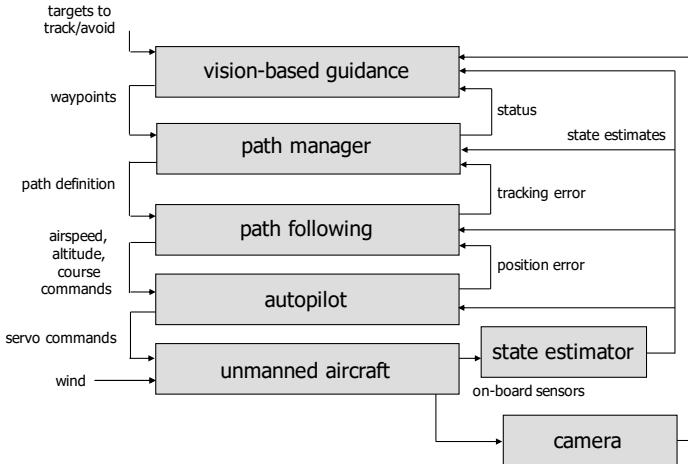


Figure 1.2: System architecture for vision-based navigation, guidance, and control. A video camera is added as an additional sensor and the path planner has been replaced with a vision-based guidance block.

1.2 DESIGN MODELS

The design philosophy that we follow throughout the book is illustrated schematically in Figure 1.3. The unmanned aircraft operating in its environment is depicted in Figure 1.3 as the “Physical System,” and includes the actuators (control flaps and propeller) and the sensors (IMU, GPS, camera, etc.). The first step in the design process is to model the physical system using nonlinear differential equations. While approximations and simplifications will be necessary at this step, the hope is to capture in mathematics all of the important characteristics of the physical system. In this book, the model of the physical system includes rigid body kinematics and dynamics (Chapter 3), aerodynamic forces and moments (Chapter 4), and the onboard sensors (Chapter 7). The resulting model is called the “Simulation Model” in Figure 1.3 and will be used for the high fidelity computer simulation of the physical system. However, we should note that the simulation model is only an approximation of the physical system, and simply because a design is effective on the simulation model, we should not assume that it will function properly on the physical system.

The simulation model is typically nonlinear and high order and is too mathematically complex to be useful for control design. Therefore, to facilitate design, the simulation model is simplified and usually linearized to create lower-order design models. For any physical system, there may be multiple design models that capture certain aspects of the design process.

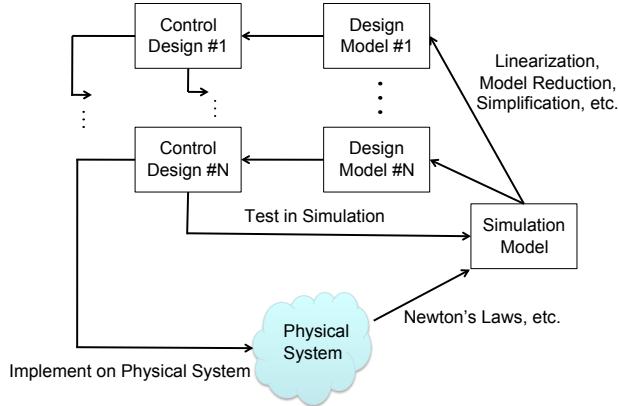


Figure 1.3: The design process. Using principles of physics, the physical system is modeled mathematically, resulting in the simulation model. The simulation model is simplified to create design models that are used for the control design. The control design is then tested and debugged in simulation and finally implemented on the physical system.

For MAVs, we will use a variety of different design models for both low-level control and also for high-level guidance. In Chapter 5, we will decompose the aircraft motion into longitudinal (pitching and climbing) motion and lateral (rolling and heading) motion, and we will have different design models for each type of motion. The linear design models developed in Chapter 5 will be used in Chapter 6 to develop low-level autopilot loops that control the airspeed, altitude, and course angle of the vehicle. In Chapter 8, we show how to estimate the states needed for the autopilot loops using sensors typically found on small and micro air vehicles.

The mathematical equations describing the physics of the system, the low-level autopilot, and the state estimation routines, when considered as a whole, are very complex and are not useful for designing the higher level guidance routines. Therefore, in Chapter 9 we develop nonlinear design models that model the closed-loop behavior of the system, where the input is commanded airspeed, altitude, and course angle, and the outputs are the inertial position and orientation of the aircraft. The design models developed in Chapter 9 are used in Chapters 10 through 13 to develop guidance strategies for the MAV.

As shown in Figure 1.3, the design models are used to design the guidance and control systems. The designs are then tested against the high fidelity simulation model, which sometimes requires that the design models be modified or enhanced if they have not captured the essential features of the system. After the designs have been thoroughly tested against the

simulation model, they are implemented on the physical system and are again tested and debugged, sometimes requiring that the simulation model be modified to more closely match the physical system.

1.3 DESIGN PROJECT

In this textbook we have decided to replace traditional pencil-and-paper homework problems with a complete and rather extensive design project. The design project is an integral part of the book, and we believe that it will play a significant role in helping the reader to internalize the material that is presented.

The design project involves building a MAV flight simulator from the ground up. The flight simulator will be built using Matlab/Simulink, and we have specifically designed the assignments so that additional add-on packages are not required.¹ The website for the book contains a number of different Matlab and Simulink files that will assist you in developing the flight simulator. Our strategy is to provide you with the basic skeleton files that pass the right information between blocks, but to have you write the internal workings of each block. The project builds upon itself and requires the successful completion of each chapter before it is possible to move to the next chapter. To help you know when the design from each chapter is working, we have included graphs and pictures on the website that show the output of our simulator at each stage.

The project assignment in Chapter 2 is to develop an animation of an aircraft and to ensure that you can properly rotate the body of the aircraft on the screen. A tutorial on animating graphics in Matlab is provided in Appendix D. The assignment in Chapter 3 is to drive the animation using a mathematical model of the rigid body equations of motion. In Chapter 4 the force and moments acting on a fixed wing aircraft are added to the simulation. The assignment in Chapter 5 is to use the Simulink commands `trim` and `linmod` to find the trim conditions of the aircraft and to derive linear transfer function and state space models of the system. The assignment in Chapter 6 adds an autopilot block that uses the real states to control the aircraft. In Chapter 7, a model of the sensors is added to the simulator, and in Chapter 8, state estimation schemes are added to estimate the states needed for the autopilot using the available sensors. The result of the project assignment in Chapter 8 is a closed-loop system that controls air-

¹We have also taught the course using the public domain flight simulator Aviones, which is available for download at Sourceforge.net. For those who do not have access to Matlab/Simulink and would prefer to develop the project in C/C++, we encourage the use of *Aviones*.

speed, altitude, and course angle using only available sensor information. The assignment in Chapter 9 is to approximate the closed-loop behavior using simple design models and to tune the parameters of the design model so that it essentially matches the behavior of the closed-loop high-fidelity simulation. The assignment in Chapter 10 is to develop simple guidance algorithms for following straight-lines and circular orbits in the presence of wind. In Chapter 11, straight-line and orbit following are used to synthesize more complicated paths, with emphasis on following Dubins paths. The assignment in Chapter 12 is to implement the RRT path planning scheme to plan Dubins paths through an obstacle field. The project assignment in Chapter 13 is to point a camera at a moving ground target and to estimate the inertial position of the target using camera data and onboard sensors (geolocation).

Chapter Two

Coordinate Frames

In studying unmanned aircraft systems, it is important to understand how different bodies are oriented relative to each other. Most obviously, we need to understand how the aircraft is oriented with respect to the earth. We may also want to know how a sensor (e.g., a camera) is oriented relative to the aircraft or how an antenna is oriented relative to a signal source on the ground. This chapter describes the various coordinate systems used to describe the position and orientation of the aircraft and its sensors, and the transformation between these coordinate systems. It is necessary to use several different coordinate systems for the following reasons:

- Newton's equations of motion are derived relative to a fixed, inertial reference frame. However, motion is most easily described in a body-fixed frame.
- Aerodynamic forces and torques act on the aircraft body and are most easily described in a body-fixed reference frame.
- On-board sensors like accelerometers and rate gyros measure information with respect to the body frame. Alternatively, GPS measures position, ground speed, and course angle with respect to the inertial frame.
- Most mission requirements, like loiter points and flight trajectories, are specified in the inertial frame. In addition, map information is also given in an inertial frame.

One coordinate frame is transformed into another through two basic operations: rotation and translation. Section 2.1 describes rotation matrices and their use in transforming between coordinate frames. Section 2.2 describes the specific coordinate frames used for miniature air vehicle systems. In Section 2.3 we define airspeed, ground speed, and wind speed and the relationship between these quantities. This leads to the more detailed discussion of the wind triangle in Section 2.4. In Section 2.5 we derive an expression for differentiating a vector in a rotating and translating frame.

2.1 ROTATION MATRICES

We begin by considering the two coordinate frames shown in Figure 2.1. The vector \mathbf{p} can be expressed in both the \mathcal{F}^0 frame (specified by $(\mathbf{i}^0, \mathbf{j}^0, \mathbf{k}^0)$)

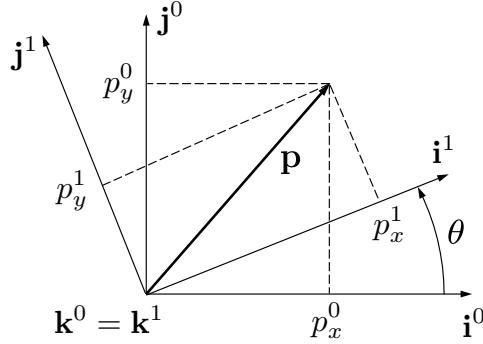


Figure 2.1: Rotation in 2D

and in the \mathcal{F}^1 frame (specified by $(\mathbf{i}^1, \mathbf{j}^1, \mathbf{k}^1)$). In the \mathcal{F}^0 frame we have

$$\mathbf{p} = p_x^0 \mathbf{i}^0 + p_y^0 \mathbf{j}^0 + p_z^0 \mathbf{k}^0.$$

Alternatively in the \mathcal{F}^1 frame we have

$$\mathbf{p} = p_x^1 \mathbf{i}^1 + p_y^1 \mathbf{j}^1 + p_z^1 \mathbf{k}^1.$$

The vector sets $(\mathbf{i}^0, \mathbf{j}^0, \mathbf{k}^0)$ and $(\mathbf{i}^1, \mathbf{j}^1, \mathbf{k}^1)$ are each mutually perpendicular sets of unit basis vectors.

Setting these two expressions equal to each other gives

$$p_x^1 \mathbf{i}^1 + p_y^1 \mathbf{j}^1 + p_z^1 \mathbf{k}^1 = p_x^0 \mathbf{i}^0 + p_y^0 \mathbf{j}^0 + p_z^0 \mathbf{k}^0.$$

Taking the dot product of both sides with \mathbf{i}^1 , \mathbf{j}^1 , and \mathbf{k}^1 respectively, and stacking the result into matrix form gives

$$\mathbf{p}^1 \triangleq \begin{pmatrix} p_x^1 \\ p_y^1 \\ p_z^1 \end{pmatrix} = \begin{pmatrix} \mathbf{i}^1 \cdot \mathbf{i}^0 & \mathbf{i}^1 \cdot \mathbf{j}^0 & \mathbf{i}^1 \cdot \mathbf{k}^0 \\ \mathbf{j}^1 \cdot \mathbf{i}^0 & \mathbf{j}^1 \cdot \mathbf{j}^0 & \mathbf{j}^1 \cdot \mathbf{k}^0 \\ \mathbf{k}^1 \cdot \mathbf{i}^0 & \mathbf{k}^1 \cdot \mathbf{j}^0 & \mathbf{k}^1 \cdot \mathbf{k}^0 \end{pmatrix} \begin{pmatrix} p_x^0 \\ p_y^0 \\ p_z^0 \end{pmatrix}.$$

From the geometry of Figure 2.1 we get

$$\mathbf{p}^1 = \mathcal{R}_0^1 \mathbf{p}^0, \quad (2.1)$$

where

$$\mathcal{R}_0^1 \triangleq \begin{pmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The notation \mathcal{R}_0^1 is used to denote a rotation from coordinate frame \mathcal{F}^0 to coordinate frame \mathcal{F}^1 .

Proceeding in a similar way, a right-handed rotation of the coordinate system about the y -axis gives

$$\mathcal{R}_0^1 \triangleq \begin{pmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{pmatrix},$$

and a right-handed rotation of the coordinate system about the x -axis is

$$\mathcal{R}_0^1 \triangleq \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & \sin \theta \\ 0 & -\sin \theta & \cos \theta \end{pmatrix}.$$

As pointed out in [1], the negative sign on the sine term appears above the line with only ones and zeros.

The matrix \mathcal{R}_0^1 in the above equations is an example of a more general class of *orthonormal* rotation matrices that have the following properties:

P.1. $(\mathcal{R}_a^b)^{-1} = (\mathcal{R}_a^b)^\top = \mathcal{R}_b^a$.

P.2. $\mathcal{R}_b^c \mathcal{R}_a^b = \mathcal{R}_a^c$.

P.3. $\det(\mathcal{R}_a^b) = 1$,

where $\det(\cdot)$ is the determinant of a matrix.

In the derivation of Equation (2.1), note that the vector \mathbf{p} remains constant and the new coordinate frame \mathcal{F}^1 was obtained by rotating \mathcal{F}^0 through a *right-handed* rotation of angle θ . Alternatively, rotation matrices can be used to rotate a vector through a prescribed angle in a fixed reference frame. As an example, consider the *left-handed* rotation of a vector \mathbf{p} in frame \mathcal{F}^0 about the \mathbf{k}^0 -axis by the angle θ , as shown in Figure 2.2.

Assuming \mathbf{p} and \mathbf{q} are confined to the \mathbf{i}^0 - \mathbf{j}^0 plane, we can write the components of \mathbf{p} and \mathbf{q} as

$$\begin{aligned} \mathbf{p} &= \begin{pmatrix} p \cos(\theta + \phi) \\ p \sin(\theta + \phi) \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} p \cos \theta \cos \phi - p \sin \theta \sin \phi \\ p \sin \theta \cos \phi + p \cos \theta \sin \phi \\ 0 \end{pmatrix} \end{aligned} \tag{2.2}$$

and

$$\mathbf{q} = \begin{pmatrix} q \cos \phi \\ q \sin \phi \\ 0 \end{pmatrix}, \tag{2.3}$$

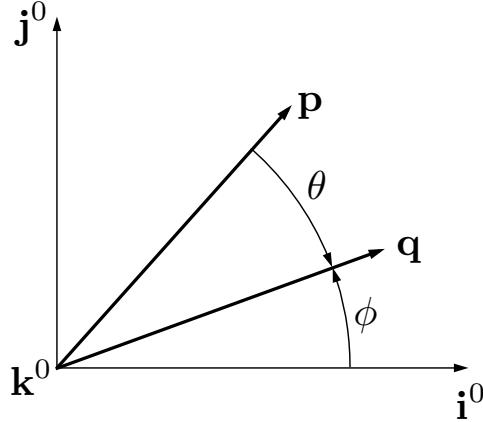


Figure 2.2: Rotation of \mathbf{p} about the k^0 -axis.

where $p \triangleq |\mathbf{p}| = q \triangleq |\mathbf{q}|$. Expressing Equations (2.2) in terms of (2.3) gives

$$\begin{aligned}\mathbf{p} &= \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \mathbf{q} \\ &= (\mathcal{R}_0^1)^\top \mathbf{q}\end{aligned}$$

and

$$\mathbf{q} = \mathcal{R}_0^1 \mathbf{p}.$$

In this case, the rotation matrix \mathcal{R}_0^1 can be interpreted as a left-handed rotation of the vector \mathbf{p} through the angle θ to a new vector \mathbf{q} in the same reference frame. Notice that a right-handed rotation of a vector (in this case from \mathbf{q} to \mathbf{p}) can be obtained by using $(\mathcal{R}_0^1)^\top$. This interpretation contrasts with our original use of the rotation matrix to transform a fixed vector \mathbf{p} from an expression in frame \mathcal{F}^0 to an expression in frame \mathcal{F}^1 where \mathcal{F}^1 has been obtained from \mathcal{F}^0 by a right-handed rotation.

2.2 MAV COORDINATE FRAMES

To derive and understand the dynamic behavior of MAVs, several coordinate systems are of interest. In this section, we will define and describe the following coordinate frames: the inertial frame, the vehicle frame, the vehicle-1 frame, the vehicle-2 frame, the body frame, the stability frame, and the wind frame. The inertial and vehicle frames are related by a transla-

tion, while the remaining frames are related by rotations. The angles defining the relative orientations of the vehicle, vehicle-1, vehicle-2, and body frames are the roll, pitch, and yaw angles that describe the attitude of the aircraft. These angles are commonly known as Euler angles . The rotation angles that define the relative orientation of the body, stability, and wind coordinate frames are the angle of attack and sideslip angles. Throughout the book we assume a flat, non-rotating earth — a valid assumption for MAVs.

2.2.1 The inertial frame \mathcal{F}^i

The inertial coordinate system is an earth-fixed coordinate system with its origin at the defined home location. As shown in Figure 2.3, the unit vector \mathbf{i}^i is directed North, \mathbf{j}^i is directed East, and \mathbf{k}^i is directed toward the center of the earth, or down. This coordinate system is sometimes referred to as a North-East-Down reference frame. It is common for North to be referred to as the inertial x direction, East to be referred to as the inertial y direction, and down to be referred to as the inertial z direction.

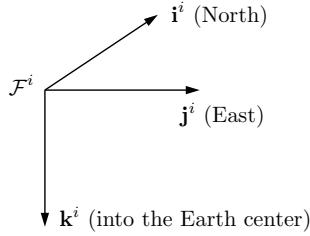


Figure 2.3: The inertial coordinate frame. The \mathbf{i}^i -axis points North, the \mathbf{j}^i -axis points East, and the \mathbf{k}^i -axis points into the earth.

2.2.2 The vehicle frame \mathcal{F}^v

The origin of the vehicle frame is at the center of mass of the MAV. However, the axes of \mathcal{F}^v are aligned with the axis of the inertial frame \mathcal{F}^i . In other words, the unit vector \mathbf{i}^v points North, \mathbf{j}^v points East, and \mathbf{k}^v points toward the center of the earth, as shown in Figure 2.4.

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

The origin of the vehicle-1 frame is identical to the vehicle frame: the center of mass of the aircraft. However, \mathcal{F}^{v1} is rotated in the positive right-handed direction about \mathbf{k}^v by the heading (or yaw) angle ψ . In the absence of additional rotations, \mathbf{i}^{v1} points out the nose of the airframe, \mathbf{j}^{v1} points out

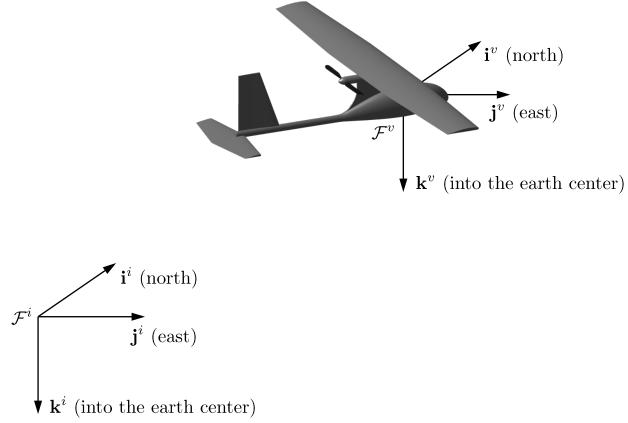


Figure 2.4: The vehicle coordinate frame. The \mathbf{i}^v -axis points North, the \mathbf{j}^v -axis points East, and the \mathbf{k}^v -axis points into the earth.

the right wing, and \mathbf{k}^{v1} is aligned with \mathbf{k}^v and points into the earth. The vehicle-1 frame is shown in Figure 2.5.

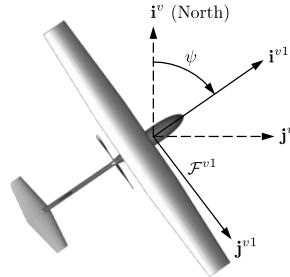


Figure 2.5: The vehicle-1 frame. The \mathbf{i}^{v1} -axis points out the nose of the aircraft, the \mathbf{j}^{v1} -axis points out the right wing, and the \mathbf{k}^{v1} -axis points into the earth.

The transformation from \mathcal{F}^v to \mathcal{F}^{v1} is given by

$$\mathbf{p}^{v1} = \mathcal{R}_v^{v1}(\psi)\mathbf{p}^v,$$

where

$$\mathcal{R}_v^{v1}(\psi) = \begin{pmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

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

The origin of the vehicle-2 frame is again the center of mass of the aircraft and is obtained by rotating the vehicle-1 frame in a right-handed rotation about the j^{v^1} axis by the pitch angle θ . The unit vector i^{v^2} points out the nose of the aircraft, j^{v^2} points out the right wing, and k^{v^2} points out the belly, as shown in Figure 2.6.

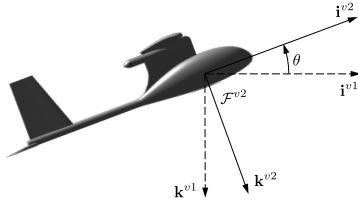


Figure 2.6: The vehicle-2 frame. The i^{v^2} -axis points out the nose of the aircraft, the j^{v^2} -axis points out the right wing, and the k^{v^2} -axis points out the belly.

The transformation from \mathcal{F}^{v^1} to \mathcal{F}^{v^2} is given by

$$\mathbf{p}^{v^2} = \mathcal{R}_{v^1}^{v^2}(\theta) \mathbf{p}^{v^1},$$

where

$$\mathcal{R}_{v^1}^{v^2}(\theta) = \begin{pmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{pmatrix}.$$

2.2.5 The body frame \mathcal{F}^b

The body frame is obtained by rotating the vehicle-2 frame in a right-handed rotation about i^{v^2} by the roll angle ϕ . Therefore, the origin is the center of mass, i^b points out the nose of the airframe, j^b points out the right wing, and k^b points out the belly. The body frame is shown in Figure 2.7. The directions indicated by the i^b , j^b , and k^b unit vectors are sometimes referred to as the body x , the body y , and the body z directions, respectively.

The transformation from \mathcal{F}^{v^2} to \mathcal{F}^b is given by

$$\mathbf{p}^b = \mathcal{R}_{v^2}^b(\phi) \mathbf{p}^{v^2},$$

where

$$\mathcal{R}_{v^2}^b(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{pmatrix}.$$

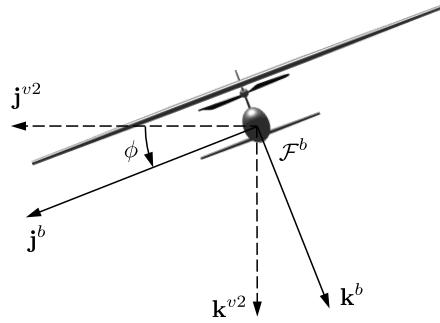


Figure 2.7: The body frame. The i^b -axis points out the nose of the airframe, the j^b -axis points out the right wing, and the k^b -axis points out the belly.

The transformation from the vehicle frame to the body frame is given by

$$\begin{aligned}\mathcal{R}_v^b(\phi, \theta, \psi) &= \mathcal{R}_{v2}^b(\phi)\mathcal{R}_{v1}^{v2}(\theta)\mathcal{R}_v^{v1}(\psi) \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{pmatrix} \begin{pmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{pmatrix} \begin{pmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\phi s_\theta c_\psi - c_\phi s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi c_\theta \\ c_\phi s_\theta c_\psi + s_\phi s_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi & c_\phi c_\theta \end{pmatrix},\end{aligned}\quad (2.4)$$

$$(2.5)$$

where $c_\phi \triangleq \cos \phi$ and $s_\phi \triangleq \sin \phi$. The angles ϕ , θ , and ψ are commonly referred to as Euler angles. Euler angles are commonly used because they provide an intuitive means for representing the orientation of a body in three dimensions. The rotation sequence ψ - θ - ϕ is commonly used for aircraft and is just one of several Euler angle systems in use [2].

The physical interpretation of Euler angles is clear and this contributes to their widespread use. Euler angle representations, however, have a mathematical singularity that can cause computational instabilities. For the ψ - θ - ϕ Euler angle sequence, there is a singularity when the pitch angle θ is ± 90 deg, in which case the yaw angle is not defined. This singularity is commonly referred to as gimbal lock. A common alternative to Euler angles is the quaternion. While the quaternion attitude representation lacks the intuitive appeal of Euler angles, they are free of mathematical singularities and are computationally more efficient. Quaternion attitude representations are discussed in Appendix B.

2.2.6 The stability frame \mathcal{F}^s

Aerodynamic forces are generated as the airframe moves through the air surrounding it. We refer to the velocity of the aircraft relative to the surrounding air as the airspeed vector, denoted \mathbf{V}_a . The magnitude of the airspeed vector is simply referred to as the airspeed, V_a . To generate lift, the wings of the airframe must fly at a positive angle with respect to the airspeed vector. This angle is called the angle of attack and is denoted by α . As shown in figure 2.8, the \mathbf{i}^s axis aligns with the projection of \mathbf{V}_a onto the plane spanned by \mathbf{i}^b and \mathbf{k}^b . The angle of attack is defined as the angle between the \mathbf{i}^s and \mathbf{i}^b axes. The transformation from \mathcal{F}^s to \mathcal{F}^b is given by

$$\mathbf{p}^b = \mathcal{R}_s^b(\alpha) \mathbf{p}^s,$$

where

$$\mathcal{R}_s^b(\alpha) = \begin{pmatrix} \cos \alpha & 0 & -\sin \alpha \\ 0 & 1 & 0 \\ \sin \alpha & 0 & \cos \alpha \end{pmatrix}.$$

The transformation from the body frame to the stability frame is defined as

$$\mathcal{R}_b^s(\alpha) = (\mathcal{R}_s^b)^\top(\alpha) = \begin{pmatrix} \cos \alpha & 0 & \sin \alpha \\ 0 & 1 & 0 \\ -\sin \alpha & 0 & \cos \alpha \end{pmatrix}.$$

2.2.7 The wind frame \mathcal{F}^w

The angle between the airspeed vector and the \mathbf{i}^b - \mathbf{k}^b plane is called the side-slip angle and is denoted by β . As shown in Figure 2.9, the wind frame is obtained by rotating the stability frame by a right-handed rotation of β about \mathbf{k}^s . The unit vector \mathbf{i}^w is aligned with the airspeed vector \mathbf{V}_a .

The transformation from \mathcal{F}^s to \mathcal{F}^w is given by

$$\mathbf{p}^w = \mathcal{R}_s^w(\beta) \mathbf{p}^s,$$

where

$$\mathcal{R}_s^w(\beta) = \begin{pmatrix} \cos \beta & \sin \beta & 0 \\ -\sin \beta & \cos \beta & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

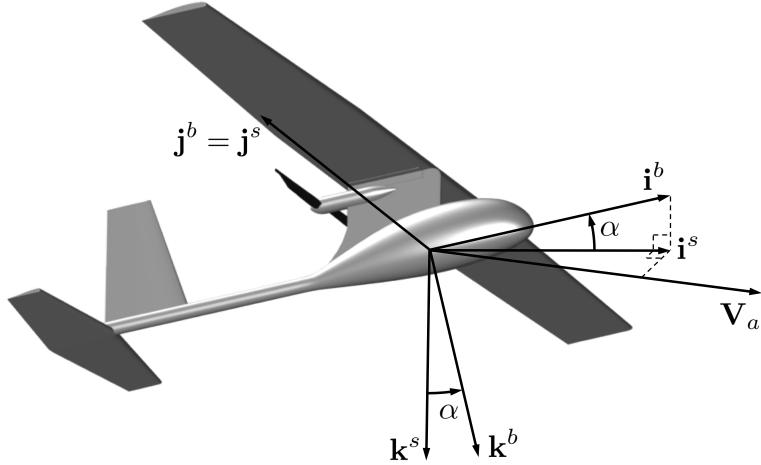


Figure 2.8: The stability frame. The \mathbf{i}^s -axis points along the projection of the airspeed vector onto the $\mathbf{i}^b\text{-}\mathbf{k}^b$ plane of the body frame, the \mathbf{j}^s -axis is identical to the \mathbf{j}^b -axis of the body frame, and the \mathbf{k}^s -axis is constructed to make a right-handed coordinate system. The angle of attack is defined as a *right*-handed rotation about the body \mathbf{j}^s -axis.

The total transformation from the body frame to the wind frame is given by

$$\begin{aligned}\mathcal{R}_b^w(\alpha, \beta) &= \mathcal{R}_s^w(\beta)\mathcal{R}_b^s(\alpha) \\ &= \begin{pmatrix} \cos \beta & \sin \beta & 0 \\ -\sin \beta & \cos \beta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \alpha & 0 & \sin \alpha \\ 0 & 1 & 0 \\ -\sin \alpha & 0 & \cos \alpha \end{pmatrix} \\ &= \begin{pmatrix} \cos \beta \cos \alpha & \sin \beta & \cos \beta \sin \alpha \\ -\sin \beta \cos \alpha & \cos \beta & -\sin \beta \sin \alpha \\ -\sin \alpha & 0 & \cos \alpha \end{pmatrix}.\end{aligned}$$

Alternatively, the transformation from the wind frame to the body frame is

$$\mathcal{R}_w^b(\alpha, \beta) = (\mathcal{R}_b^w)^\top(\alpha, \beta) = \begin{pmatrix} \cos \beta \cos \alpha & -\sin \beta \cos \alpha & -\sin \alpha \\ \sin \beta & \cos \beta & 0 \\ \cos \beta \sin \alpha & -\sin \beta \sin \alpha & \cos \alpha \end{pmatrix}.$$

2.3 AIRSPEED, WIND SPEED, AND GROUND SPEED

When developing the dynamic equations of motion for a MAV, it is important to remember that the inertial forces experienced by the MAV are

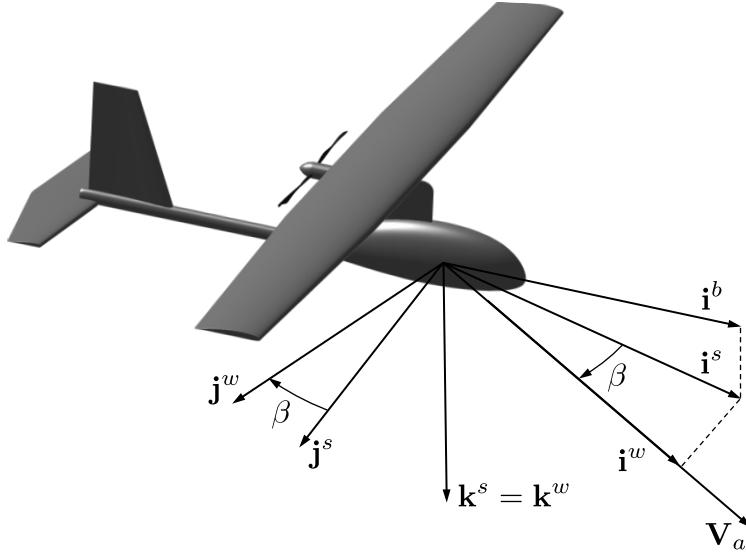


Figure 2.9: The wind frame. The i^w -axis points along the airspeed vector.

dependent on velocities and accelerations relative to a fixed (inertial) reference frame. The aerodynamic forces, however, depend on the velocity of the airframe relative to the surrounding air. When wind is not present, these velocities are the same. However, wind is almost always present with MAVs and we must carefully distinguish between airspeed, represented by the velocity with respect to the surrounding air \mathbf{V}_a , and the ground speed, represented by the velocity with respect to the inertial frame \mathbf{V}_g . These velocities are related by the expression

$$\mathbf{V}_a = \mathbf{V}_g - \mathbf{V}_w, \quad (2.6)$$

where \mathbf{V}_w is the wind velocity relative to the inertial frame.

The MAV velocity \mathbf{V}_g can be expressed in the body frame in terms of components along the i^b , j^b , and k^b axes

$$\mathbf{V}_g^b = \begin{pmatrix} u \\ v \\ w \end{pmatrix},$$

where \mathbf{V}_g^b is the velocity of the MAV *with respect to the inertial frame*, as expressed in the body frame. Similarly, if we define the north, east, and down components of the wind as w_n , w_e , and w_d respectively, we can write

an expression for the wind velocity in the body frame as

$$\mathbf{V}_w^b = \begin{pmatrix} u_w \\ v_w \\ w_w \end{pmatrix} = \mathcal{R}_v^b(\phi, \theta, \psi) \begin{pmatrix} w_n \\ w_e \\ w_d \end{pmatrix}.$$

Keeping in mind that the airspeed vector \mathbf{V}_a is the velocity of the MAV with respect to the wind, it can be expressed in the wind frame as

$$\mathbf{V}_a^w = \begin{pmatrix} V_a \\ 0 \\ 0 \end{pmatrix}.$$

Defining u_r , v_r , and w_r as the body-frame components of the airspeed vector,¹ it can be written in the body frame as

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

When developing a MAV simulation, u_r , v_r , and w_r are used to calculate the aerodynamic forces and moments acting on the MAV. The body-frame velocity components u , v , and w are states of the MAV system and are readily available from the solution of the equations of motion. The wind velocity components u_w , v_w , and w_w typically come from a wind model as inputs to the equations of motion. Combining expressions, we can express the airspeed vector body-frame components in terms of the airspeed magnitude, angle of attack, and sideslip angle as

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

which implies that

$$\begin{pmatrix} u_r \\ v_r \\ w_r \end{pmatrix} = V_a \begin{pmatrix} \cos \alpha \cos \beta \\ \sin \beta \\ \sin \alpha \cos \beta \end{pmatrix}. \quad (2.7)$$

¹Some flight mechanics textbooks define u , v , and w as the body-frame components of the airspeed vector. We define u , v , and w as the body-frame components of the ground speed vector and u_r , v_r , and w_r as the body-frame components of the airspeed vector to clearly distinguish between the two.

Inverting this relationship gives

$$V_a = \sqrt{u_r^2 + v_r^2 + w_r^2}$$

$$\alpha = \tan^{-1} \left(\frac{w_r}{u_r} \right) \quad (2.8)$$

$$\beta = \sin^{-1} \left(\frac{v_r}{\sqrt{u_r^2 + v_r^2 + w_r^2}} \right) \quad (2.9)$$

$$= \tan^{-1} \left(\frac{v_r}{\sqrt{u_r^2 + w_r^2}} \right).$$

Given that aerodynamic forces and moments are commonly expressed in terms of V_a , α , and β , these expressions are essential in formulating the equations of motion for a MAV.

2.4 THE WIND TRIANGLE

For MAVs, the wind speed is often in the range of 20 to 50 percent of the airspeed. The significant effect of wind on MAVs is important to understand, more so than for larger conventional aircraft, where the airspeed is typically much greater than the wind speed. Having introduced the concepts of reference frames, airframe velocity, wind velocity, and the airspeed vector, we can discuss some important definitions relating to the navigation of MAVs.

The direction of the ground speed vector relative to an inertial frame is specified using two angles. These angles are the course angle χ and the (inertial referenced) flight path angle γ . Figure 2.10 shows how these two angles are defined. The flight path angle γ is defined as the angle between the horizontal plane and the ground velocity vector \mathbf{V}_g , while the course χ is the angle between the projection of the ground velocity vector onto the horizontal plane and true North.

The relationship between the ground speed vector, the airspeed vector, and the wind vector, which is given by Equation (2.6) is called the wind triangle. A more detailed depiction of the wind triangle is given in the horizontal plane in Figure 2.11 and in the vertical plane in Figure 2.12. Figure 2.11 shows an air vehicle following a ground track represented by the dashed line. The North direction is indicated by the \mathbf{i}^i vector, and the direction that the vehicle is pointed is shown by the \mathbf{i}^b vector, which is fixed in the direction of the body x -axis. For level flight, the heading (yaw) angle ψ , is the angle between \mathbf{i}^i and \mathbf{i}^b and defines the direction that the vehicle is pointed. The direction the vehicle is traveling with respect to the surrounding air mass is

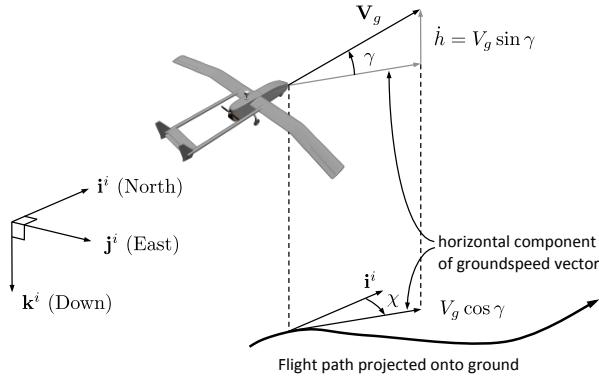


Figure 2.10: The flight path angle γ and the course angle χ .

given by the airspeed vector \mathbf{V}_a . In steady, level flight, \mathbf{V}_a is commonly aligned with \mathbf{i}^b , meaning the sideslip angle β is zero.

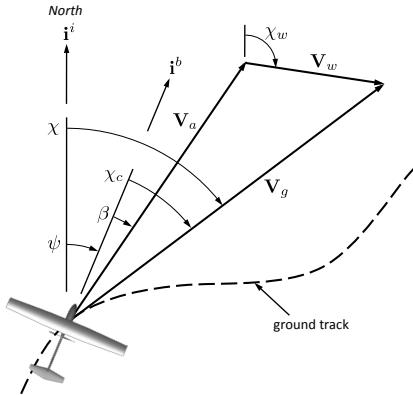


Figure 2.11: Heading is the direction that the MAV is pointed. Course is the direction of travel relative to the earth's surface. The crab angle is the difference between course and heading. In the absence of wind, the crab angle is zero.

The direction the vehicle is traveling with respect to the ground is shown by the velocity vector \mathbf{V}_g . The angle between the inertial North and the inertial velocity vector projected onto the local North-East plane is called the course angle χ . If there is a constant ambient wind, the aircraft will need to crab into the wind in order to follow a ground track that is not aligned with the wind. The *crab angle* χ_c is defined as the difference between the course

and the heading angles as follows:

$$\chi_c \stackrel{\triangle}{=} \chi - \psi.$$

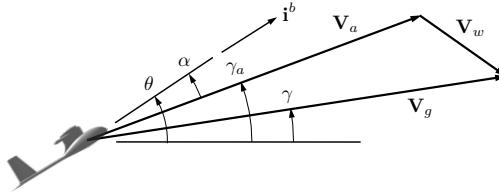


Figure 2.12: The wind triangle projected onto the vertical plane.

Figure 2.12 depicts the vertical component of the wind triangle. When there is a down component of wind, we define the angle from the inertial North-East plane to \mathbf{V}_a as the *air-mass-referenced flight-path angle* and denote it by γ_a . The relationship between the air mass referenced flight path angle, the angle of attack, and the pitch angle is given by

$$\gamma_a = \theta - \alpha.$$

In the absence of wind $\gamma_a = \gamma$.

The ground speed vector in the inertial frame can be expressed as

$$\mathbf{V}_g^i = \begin{pmatrix} \cos \chi & -\sin \chi & 0 \\ \sin \chi & \cos \chi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \gamma & 0 & \sin \gamma \\ 0 & 1 & 0 \\ -\sin \gamma & 0 & \cos \gamma \end{pmatrix} \begin{pmatrix} V_g \\ 0 \\ 0 \end{pmatrix} = V_g \begin{pmatrix} \cos \chi \cos \gamma \\ \sin \chi \cos \gamma \\ -\sin \gamma \end{pmatrix},$$

where $V_g = \|\mathbf{V}_g\|$. Similarly, the airspeed vector in the inertial frame can be expressed as

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

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

$$V_g \begin{pmatrix} \cos \chi \cos \gamma \\ \sin \chi \cos \gamma \\ -\sin \gamma \end{pmatrix} = V_a \begin{pmatrix} \cos \psi \cos \gamma_a \\ \sin \psi \cos \gamma_a \\ -\sin \gamma_a \end{pmatrix} + \begin{pmatrix} w_n \\ w_e \\ w_d \end{pmatrix}. \quad (2.10)$$

Equation (2.10) allows us to derive relationships between V_g , V_a , χ , ψ , γ and γ_a . To find an expression for γ_a , multiply both sides of Equation (2.10) by $(\cos \chi \sin \gamma, \sin \chi \sin \gamma, \cos \gamma)$ to eliminate V_g and obtain

$$V_a (\sin \gamma_a \cos \gamma - \cos \gamma_a \sin \gamma \cos(\chi - \psi)) = \begin{pmatrix} w_n \\ w_e \\ w_d \end{pmatrix}^\top \begin{pmatrix} \cos \chi \sin \gamma \\ \sin \chi \sin \gamma \\ \cos \gamma \end{pmatrix}.$$

If we assume that the crab angle $\chi_c = \chi - \psi$ is less than 30 degrees, then $\cos(\chi - \psi) \approx 1$ and we can solve for γ_a to obtain

$$\gamma_a \approx \gamma + \sin^{-1} \left(\frac{1}{V_a} \begin{pmatrix} w_n \\ w_e \\ w_d \end{pmatrix}^\top \begin{pmatrix} \cos \chi \sin \gamma \\ \sin \chi \sin \gamma \\ \cos \gamma \end{pmatrix} \right). \quad (2.11)$$

To derive an expression for ψ , multiply both sides of Equation (2.10) by $(-\sin \chi, \cos \chi, 0)$ to get the expression

$$0 = V_a \cos \gamma_a (-\sin \chi \cos \psi + \cos \chi \sin \psi) + \begin{pmatrix} w_n \\ w_e \end{pmatrix}^\top \begin{pmatrix} -\sin \chi \\ \cos \chi \end{pmatrix}.$$

Solving for ψ gives

$$\psi = \chi - \sin^{-1} \left(\frac{1}{V_a \cos \gamma_a} \begin{pmatrix} w_n \\ w_e \end{pmatrix}^\top \begin{pmatrix} -\sin \chi \\ \cos \chi \end{pmatrix} \right). \quad (2.12)$$

An expression for groundspeed can be obtained by taking the squared norm of each side of Equation (2.10) to get

$$V_g = \sqrt{V_a^2 + V_w^2 + 2V_a \begin{pmatrix} \cos \psi \cos \gamma_a \\ \sin \psi \cos \gamma_a \\ -\sin \gamma_a \end{pmatrix}^\top \begin{pmatrix} w_n \\ w_e \\ w_d \end{pmatrix}}, \quad (2.13)$$

where $V_w = \|\mathbf{V}_w\| = \sqrt{w_n^2 + w_e^2 + w_d^2}$ is the wind speed.

Because wind typically has a significant impact on the flight behavior of small unmanned aircraft, we have tried to carefully account for it throughout the text. If wind effects are negligible, however, some important simplifications result. For example, when $V_w = 0$, we also have that $V_a = V_g$, $u = u_r$, $v = v_r$, $w = w_r$, $\psi = \chi$ (assuming also that $\beta = 0$), and $\gamma = \gamma_a$.

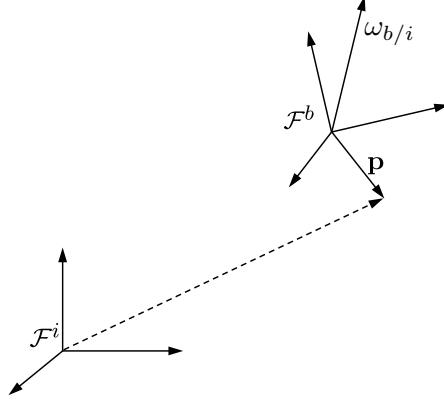


Figure 2.13: A vector in a rotating reference frame.

2.5 DIFFERENTIATION OF A VECTOR

In the process of deriving equations of motion for a MAV, it is necessary to compute derivatives of vectors in reference frames that are moving with respect to one another. Suppose that we are given two coordinate frames, \mathcal{F}^i and \mathcal{F}^b , as shown in Figure 2.13. For example, \mathcal{F}^i might represent the inertial frame and \mathcal{F}^b might represent the body frame of a MAV. Suppose that the vector \mathbf{p} is moving in \mathcal{F}^b and that \mathcal{F}^b is rotating (but not translating) with respect to \mathcal{F}^i . Our objective is to find the time derivative of \mathbf{p} as seen from frame \mathcal{F}^i . To do this, denote the angular velocity of frame \mathcal{F}^b in \mathcal{F}^i as $\omega_{b/i}$ and express the vector \mathbf{p} in terms of its vector components as

$$\mathbf{p} = p_x \mathbf{i}^b + p_y \mathbf{j}^b + p_z \mathbf{k}^b. \quad (2.14)$$

The time derivative of \mathbf{p} with respect to frame \mathcal{F}^i can be found by differentiating Equation (2.14) as

$$\frac{d}{dt_i} \mathbf{p} = \dot{p}_x \mathbf{i}^b + \dot{p}_y \mathbf{j}^b + \dot{p}_z \mathbf{k}^b + p_x \frac{d}{dt_i} \mathbf{i}^b + p_y \frac{d}{dt_i} \mathbf{j}^b + p_z \frac{d}{dt_i} \mathbf{k}^b, \quad (2.15)$$

where d/dt_i represents time differentiation with respect to the inertial frame. The first three terms on the right-hand side of Equation (2.15) represent the change in \mathbf{p} as viewed by an observer in the rotating \mathcal{F}^b frame. Thus, the differentiation is carried out in the moving frame. We denote this local derivative term by

$$\frac{d}{dt_b} \mathbf{p} = \dot{p}_x \mathbf{i}^b + \dot{p}_y \mathbf{j}^b + \dot{p}_z \mathbf{k}^b. \quad (2.16)$$

The next three terms on the right-hand side of Equation (2.15) represent the change in \mathbf{p} due to the rotation of frame \mathcal{F}^b relative to \mathcal{F}^i . Given that \mathbf{i}^b , \mathbf{j}^b , and \mathbf{k}^b are fixed in the \mathcal{F}^b frame, their derivatives can be calculated as shown in [3] as

$$\begin{aligned}\dot{\mathbf{i}}^b &= \boldsymbol{\omega}_{b/i} \times \mathbf{i}^b \\ \dot{\mathbf{j}}^b &= \boldsymbol{\omega}_{b/i} \times \mathbf{j}^b \\ \dot{\mathbf{k}}^b &= \boldsymbol{\omega}_{b/i} \times \mathbf{k}^b.\end{aligned}$$

We can rewrite the last three terms of Equation (2.15) as

$$\begin{aligned}p_x \dot{\mathbf{i}}^b + p_y \dot{\mathbf{j}}^b + p_z \dot{\mathbf{k}}^b &= p_x (\boldsymbol{\omega}_{b/i} \times \mathbf{i}^b) + p_y (\boldsymbol{\omega}_{b/i} \times \mathbf{j}^b) + p_z (\boldsymbol{\omega}_{b/i} \times \mathbf{k}^b) \\ &= \boldsymbol{\omega}_{b/i} \times \mathbf{p}.\end{aligned}\quad (2.17)$$

Combining results from Equations (2.15), (2.16), and (2.17), we obtain the desired relation

$$\frac{d}{dt_i} \mathbf{p} = \frac{d}{dt_b} \mathbf{p} + \boldsymbol{\omega}_{b/i} \times \mathbf{p},\quad (2.18)$$

which expresses the derivative of the vector \mathbf{p} in frame \mathcal{F}^i in terms of its change as observed in frame \mathcal{F}^b and the relative rotation of the two frames. We will use this relation as we derive equations of motion for the MAV in Chapter 3.

2.6 CHAPTER SUMMARY

In this chapter, we have introduced the coordinate frames important to describing the orientation of MAVs. We have described how rotation matrices can be used to transform coordinates in one frame of reference to coordinates in another frame of reference. We have introduced the 3-2-1 Euler angles (ψ , θ , and ϕ) as a means to rotate from the inertial coordinate frame to the body frame. We have also introduced the angle of attack α and the sideslip angle β to describe the relative orientation of the body frame, the stability frame, and the wind frame. An understanding of these orientations is essential to the derivation of equations of motion and the modeling of aerodynamic forces involved in MAV flight. We have introduced the wind triangle and have made the relationships between airspeed, ground speed, wind speed, heading, course, flight path angle, and air mass relative flight path angle explicit. We have also derived the Coriolis formula for the differentiation of a vector.

NOTES AND REFERENCES

There are many references on coordinate frames and rotations matrices. A particularly good overview of rotation matrices is [4]. Overviews of attitude representations are included in [2, 5]. The definition of the different aircraft frames can be found in [?, 6, 1, 7]. A particularly good explanation is given in [8]. Vector differentiation is discussed in most textbooks on mechanics, including [9, 10, ?, 3].

2.7 DESIGN PROJECT

The objective of this assignment is to create a 3D graphic of a MAV that is correctly rotated and translated to the desired configuration. Creating animations in Simulink is described in Appendix D and example files are contained on the textbook website.

Simulink

- 2.1 Read Appendix D and study carefully the spacecraft animation using vertices and faces given at the textbook website.
- 2.2 Create an animation drawing of the aircraft shown in Figure 2.14.
- 2.3 Using a Simulink model like the one given on the website, verify that the aircraft is correctly rotated and translated in the animation.
- 2.4 In the animation file, switch the order of rotation and translation so that the aircraft is first translated and then rotated, and observe the effect.

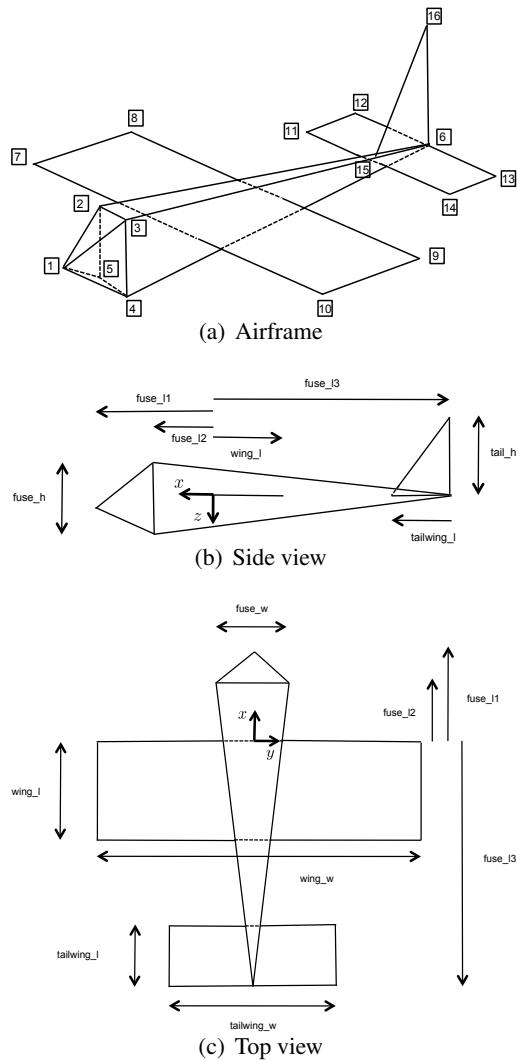


Figure 2.14: Specifications for animation of aircraft for the project.

Chapter Three

Kinematics and Dynamics

The first step in developing navigation, guidance, and control strategies for MAVs is to develop appropriate dynamic models. Deriving the nonlinear equations of motion for a MAV is the focus of Chapters 3 and 4. In Chapter 5, we linearize the equations of motion to create transfer-function and state-space models appropriate for control design.

In this chapter, we derive the expressions for the kinematics and the dynamics of a rigid body. We will apply Newton's laws: for example, $\mathbf{f} = m\dot{\mathbf{v}}$ in the case of the linear motion. In this chapter, we will focus on defining the relations between positions and velocities (the kinematics) and relations between forces and moments and the momentum (dynamics). In Chapter 4, we will concentrate on the definition of the forces and moments involved, particularly the aerodynamic forces and moments. In Chapter 5, we will combine these relations to form the complete nonlinear equations of motion. While the expressions derived in this chapter are general to any rigid body, we will use notation and coordinate frames that are typical in the aeronautics literature. In particular, in Section 3.1 we define the notation that will be used for MAV state variables. In Section 3.2 we derive the kinematics, and in Section 3.3 we derive the dynamics.

3.1 STATE VARIABLES

In developing the equations of motion for a MAV, twelve state variables will be introduced. There are three position states and three velocity states associated with the translational motion of the MAV. Similarly, there are three angular position and three angular velocity states associated with the rotational motion. The state variables are listed in Table 3.1.

The state variables are shown schematically in Figure 3.1. The North-East-Down positions of the MAV (p_n, p_e, p_d) are defined relative to the inertial frame. We will sometimes use $h = -p_d$ to denote the altitude. The linear velocities (u, v, w) and the angular velocities (p, q, r) of the MAV are defined with respect to the body frame. The Euler angles—roll ϕ , pitch θ , and heading (yaw) ψ —are defined with respect to the vehicle-2 frame, the vehicle-1 frame, and the vehicle frame, respectively. Because the Euler an-

Table 3.1: State variables for MAV equations of motion.

Name	Description
p_n	Inertial North position of the MAV along \mathbf{i}^i in \mathcal{F}^i .
p_e	Inertial East position of the MAV along \mathbf{j}^i in \mathcal{F}^i .
p_d	Inertial Down position (negative of altitude) of the aircraft measured along \mathbf{k}^i in \mathcal{F}^i .
u	Body frame velocity measured along \mathbf{i}^b in \mathcal{F}^b .
v	Body frame velocity measured along \mathbf{j}^b in \mathcal{F}^b .
w	Body frame velocity measured along \mathbf{k}^b in \mathcal{F}^b .
ϕ	Roll angle defined with respect to \mathcal{F}^{v^2} .
θ	Pitch angle defined with respect to \mathcal{F}^{v^1} .
ψ	Heading (yaw) angle defined with respect to \mathcal{F}^v .
p	Roll rate measured along \mathbf{i}^b in \mathcal{F}^b .
q	Pitch rate measured along \mathbf{j}^b in \mathcal{F}^b .
r	Yaw rate measured along \mathbf{k}^b in \mathcal{F}^b .

gles are defined relative to intermediate frames of reference, we cannot say that the angular rates (p, q, r) are simply the time derivatives of the attitude angles (ϕ, θ, ψ). As we will show in the following section, $p = \dot{\phi}$, $q = \dot{\theta}$, and $r = \dot{\psi}$ only at the instant that $\phi = \theta = 0$. Generally, the angular rates p , q , and r are functions of the time derivatives of the attitude angles, $\dot{\phi}$, $\dot{\theta}$, and $\dot{\psi}$ and the angles ϕ and θ . The remainder of this chapter is devoted to formulating the equations of motion corresponding to each of the states listed in Table 3.1.

3.2 KINEMATICS

The translational velocity of the MAV is commonly expressed in terms of the velocity components along each of the axes in a body-fixed coordinate frame. The components u , v , and w correspond to the inertial velocity of the vehicle projected onto the \mathbf{i}^b , \mathbf{j}^b , and \mathbf{k}^b axes, respectively. On the other hand, the translational position of the MAV is usually measured and expressed in an inertial reference frame. Relating the translational velocity and position requires differentiation and a rotational transformation

$$\frac{d}{dt} \begin{pmatrix} p_n \\ p_e \\ p_d \end{pmatrix} = \mathcal{R}_b^v \begin{pmatrix} u \\ v \\ w \end{pmatrix} = (\mathcal{R}_v^b)^\top \begin{pmatrix} u \\ v \\ w \end{pmatrix},$$

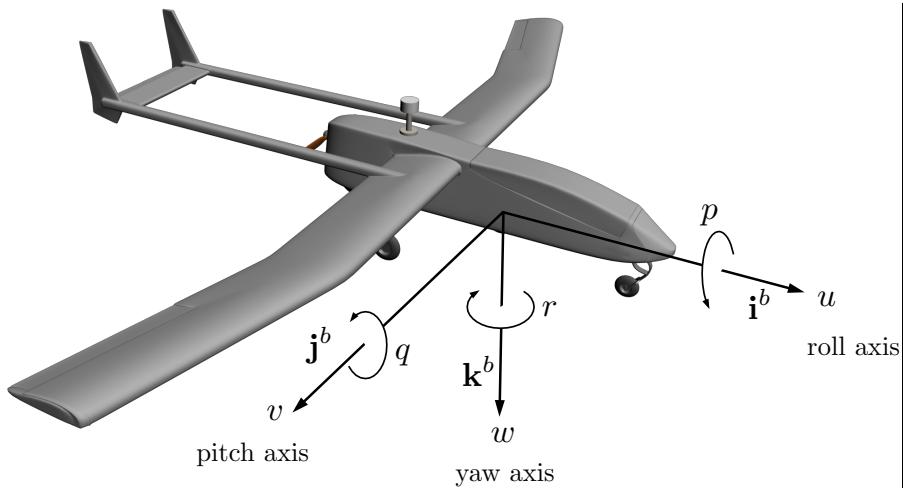


Figure 3.1: Definition of axes of motion.

which, using Equation (2.5) gives

$$\begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{p}_d \end{pmatrix} = \begin{pmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix}, \quad (3.1)$$

where we have used the shorthand notation $c_x \triangleq \cos x$ and $s_x \triangleq \sin x$. This is a kinematic relation in that it relates the derivative of position to velocity: forces or accelerations are not considered.

The relationship between angular positions ϕ , θ , and ψ and the angular rates p , q , and r is also complicated by the fact that these quantities are defined in different coordinate frames. The angular rates are defined in the body frame \mathcal{F}^b . The angular positions (Euler angles) are defined in three different coordinate frames: the roll angle ϕ is a rotation from \mathcal{F}^{v^2} to \mathcal{F}^b about the $i^{v^2} = i^b$ axis; the pitch angle θ is a rotation from \mathcal{F}^{v^1} to \mathcal{F}^{v^2} about the $j^{v^1} = j^{v^2}$ axis; and the yaw angle ψ is a rotation from \mathcal{F}^v to \mathcal{F}^{v^1} about the $k^v = k^{v^1}$ axis.

The body-frame angular rates can be expressed in terms of the derivatives of the Euler angles, provided that the proper rotational transformations are

carried out as follows:

$$\begin{aligned}
 \begin{pmatrix} p \\ q \\ r \end{pmatrix} &= \begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix} + \mathcal{R}_{v2}^b(\phi) \begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix} + \mathcal{R}_{v2}^b(\phi) \mathcal{R}_{v1}^{v2}(\theta) \begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix} \\
 &= \begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{pmatrix} \begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{pmatrix} \begin{pmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix} \\
 &= \begin{pmatrix} 1 & 0 & -\sin \theta \\ 0 & \cos \phi & \sin \phi \cos \theta \\ 0 & -\sin \phi & \cos \phi \cos \theta \end{pmatrix} \begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix}. \tag{3.2}
 \end{aligned}$$

Inverting this expression yields

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & \sin \phi \tan \theta & \cos \phi \tan \theta \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi \sec \theta & \cos \phi \sec \theta \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix}, \tag{3.3}$$

which expresses the derivatives of the three angular position states in terms of the angular positions ϕ and θ and the body rates p , q , and r .

3.3 RIGID-BODY DYNAMICS

To derive the dynamic equations of motion for the MAV, we will apply Newton's second law—first to the translational degrees of freedom and then to the rotational degrees of freedom. Newton's laws hold in inertial reference frames, meaning the motion of the body of interest must be referenced to a fixed (i.e., inertial) frame of reference, which in our case is the ground. We will assume a flat earth model, which is appropriate for small and miniature air vehicles. Even though motion is referenced to a fixed frame, it can be *expressed* using vector components associated with other frames, such as the body frame. We do this with the MAV velocity vector \mathbf{V}_g , which for convenience is most commonly expressed in the body frame as $\mathbf{V}_g^b = (u, v, w)^\top$. \mathbf{V}_g^b is the velocity of the MAV with respect to the ground as expressed in the body frame.

3.3.1 Translational Motion

Newton's second law applied to a body undergoing translational motion can be stated as

$$m \frac{d\mathbf{V}_g}{dt_i} = \mathbf{f}, \tag{3.4}$$

where m is the mass of the MAV,¹ $\frac{d}{dt_i}$ is the time derivative in the inertial frame, and \mathbf{f} is the sum of all external forces acting on the MAV. The external forces include gravity, aerodynamic forces, and propulsion forces.

The derivative of velocity taken in the inertial frame can be written in terms of the derivative in the body frame and the angular velocity according to Equation (2.18) as

$$\frac{d\mathbf{V}_g}{dt_i} = \frac{d\mathbf{V}_g}{dt_b} + \boldsymbol{\omega}_{b/i} \times \mathbf{V}_g, \quad (3.5)$$

where $\boldsymbol{\omega}_{b/i}$ is the angular velocity of the MAV with respect to the inertial frame. Combining (3.4) and (3.5) results in an alternative representation of Newton's second law with differentiation carried out in the body frame:

$$m \left(\frac{d\mathbf{V}_g}{dt_b} + \boldsymbol{\omega}_{b/i} \times \mathbf{V}_g \right) = \mathbf{f}.$$

In the case of a maneuvering aircraft, we can most easily apply Newton's second law by expressing the forces and velocities in the body frame as

$$m \left(\frac{d\mathbf{V}_g^b}{dt_b} + \boldsymbol{\omega}_{b/i}^b \times \mathbf{V}_g^b \right) = \mathbf{f}^b, \quad (3.6)$$

where $\mathbf{V}_g^b = (u, v, w)^\top$ and $\boldsymbol{\omega}_{b/i}^b = (p, q, r)^\top$. The vector \mathbf{f}^b represents the sum of the externally applied forces and is defined in terms of its body-frame components as $\mathbf{f}^b \triangleq (f_x, f_y, f_z)^\top$.

The expression $\frac{d\mathbf{V}_g^b}{dt_b}$ is the rate of change of the velocity expressed in the body frame, as viewed by an observer on the moving body. Since u , v , and w are the instantaneous projections of \mathbf{V}_g^b onto the \mathbf{i}^b , \mathbf{j}^b , and \mathbf{k}^b axes, it follows that

$$\frac{d\mathbf{V}_g^b}{dt_b} = \begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix}.$$

Expanding the cross product in Equation (3.6) and rearranging terms, we get

$$\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} = \begin{pmatrix} rv - qw \\ pw - ru \\ qu - pv \end{pmatrix} + \frac{1}{m} \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix}. \quad (3.7)$$

¹Mass is denoted with a sans serif font m to distinguish it from m , which will be introduced as the sum of moments about the body-fixed \mathbf{j}^b axis.

3.3.2 Rotational Motion

For rotational motion, Newton's second law states that

$$\frac{d\mathbf{h}}{dt_i} = \mathbf{m},$$

where \mathbf{h} is the angular momentum in vector form and \mathbf{m} is the sum of all externally applied moments. This expression is true provided that moments are summed about the center of mass of the MAV. The derivative of angular momentum taken in the inertial frame can be expanded using Equation (2.18) as

$$\frac{d\mathbf{h}}{dt_i} = \frac{d\mathbf{h}}{dt_b} + \boldsymbol{\omega}_{b/i} \times \mathbf{h} = \mathbf{m}.$$

As with translational motion, it is most convenient to express this equation in the body frame, giving

$$\frac{d\mathbf{h}^b}{dt_b} + \boldsymbol{\omega}_{b/i}^b \times \mathbf{h}^b = \mathbf{m}^b. \quad (3.8)$$

For a rigid body, angular momentum is defined as the product of the *inertia matrix* \mathbf{J} and the angular velocity vector: $\mathbf{h}^b \triangleq \mathbf{J}\boldsymbol{\omega}_{b/i}^b$ where \mathbf{J} is given by

$$\begin{aligned} \mathbf{J} &= \begin{pmatrix} \int(y^2 + z^2) dm & -\int xy dm & -\int xz dm \\ -\int xy dm & \int(x^2 + z^2) dm & -\int yz dm \\ -\int xz dm & -\int yz dm & \int(x^2 + y^2) dm \end{pmatrix} \quad (3.9) \\ &\triangleq \begin{pmatrix} J_x & -J_{xy} & -J_{xz} \\ -J_{xy} & J_y & -J_{yz} \\ -J_{xz} & -J_{yz} & J_z \end{pmatrix}. \end{aligned}$$

The diagonal terms of \mathbf{J} are called the *moments of inertia*, while the off-diagonal terms are called the *products of inertia*. The moments of inertia are measures of the aircraft's tendency to oppose acceleration about a specific axis of rotation. For example, J_x can be conceptually thought of as taking the product of the mass of each element composing the aircraft (dm) and the square of the distance of the mass element from the body x axis ($y^2 + z^2$) and adding them up. The larger J_x is in value, the more the aircraft opposes angular acceleration about the x axis. This line of thinking, of course, applies to the moments of inertia J_y and J_z as well. In practice, the inertia matrix is not calculated using Equation (3.9). Instead, it is numerically calculated from mass properties using CAD models or it is measured experimentally using equipment such as a bifilar pendulum [?, 11].

Because the integrals in Equation (3.9) are calculated with respect to the \mathbf{i}^b , \mathbf{j}^b , and \mathbf{k}^b axes fixed in the (rigid) body, \mathbf{J} is constant when viewed from the body frame, hence $\frac{d\mathbf{J}}{dt_b} = 0$. Taking derivatives and substituting into Equation (3.8), we get

$$\mathbf{J} \frac{d\omega_{b/i}^b}{dt_b} + \omega_{b/i}^b \times (\mathbf{J} \omega_{b/i}^b) = \mathbf{m}^b. \quad (3.10)$$

The expression $\frac{d\omega_{b/i}^b}{dt_b}$ is the rate of change of the angular velocity expressed in the body frame, as viewed by an observer on the moving body. Since p , q , and r are the instantaneous projections of $\omega_{b/i}^b$ onto the \mathbf{i}^b , \mathbf{j}^b , and \mathbf{k}^b axes, it follows that

$$\dot{\omega}_{b/i}^b = \frac{d\omega_{b/i}^b}{dt_b} = \begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix}.$$

Rearranging Equation (3.10), we get

$$\dot{\omega}_{b/i}^b = \mathbf{J}^{-1} \left[-\omega_{b/i}^b \times (\mathbf{J} \omega_{b/i}^b) + \mathbf{m}^b \right]. \quad (3.11)$$

Aircraft are often symmetric about the plane spanned by \mathbf{i}^b and \mathbf{k}^b . In that case $J_{xy} = J_{yz} = 0$, which implies that

$$\mathbf{J} = \begin{pmatrix} J_x & 0 & -J_{xz} \\ 0 & J_y & 0 \\ -J_{xz} & 0 & J_z \end{pmatrix}.$$

Under this symmetry assumption, the inverse of \mathbf{J} is given by

$$\begin{aligned} \mathbf{J}^{-1} &= \frac{\text{adj}(\mathbf{J})}{\det(\mathbf{J})} \\ &= \frac{\begin{pmatrix} J_y J_z & 0 & J_y J_{xz} \\ 0 & J_x J_z - J_{xz}^2 & 0 \\ J_{xz} J_y & 0 & J_x J_y \end{pmatrix}}{J_x J_y J_z - J_{xz}^2 J_y} \\ &= \begin{pmatrix} \frac{J_z}{\Gamma} & 0 & \frac{J_{xz}}{\Gamma} \\ 0 & \frac{1}{J_y} & 0 \\ \frac{J_{xz}}{\Gamma} & 0 & \frac{J_x}{\Gamma} \end{pmatrix}, \end{aligned}$$

where $\Gamma \triangleq J_x J_z - J_{xz}^2$.

Defining the components of the externally applied moment about the \mathbf{i}^b , \mathbf{j}^b , and \mathbf{k}^b axes as $\mathbf{m}^b \triangleq (l, m, n)^\top$, we can write Equation (3.11) in component form as

$$\begin{aligned}\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} &= \begin{pmatrix} \frac{J_z}{\Gamma} & 0 & \frac{J_{xz}}{\Gamma} \\ 0 & \frac{1}{J_y} & 0 \\ \frac{J_{xz}}{\Gamma} & 0 & \frac{J_x}{\Gamma} \end{pmatrix} \left[\begin{pmatrix} 0 & r & -q \\ -r & 0 & p \\ q & -p & 0 \end{pmatrix} \begin{pmatrix} J_x & 0 & -J_{xz} \\ 0 & J_y & 0 \\ -J_{xz} & 0 & J_z \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} + \begin{pmatrix} l \\ m \\ n \end{pmatrix} \right] \\ &= \begin{pmatrix} \frac{J_z}{\Gamma} & 0 & \frac{J_{xz}}{\Gamma} \\ 0 & \frac{1}{J_y} & 0 \\ \frac{J_{xz}}{\Gamma} & 0 & \frac{J_x}{\Gamma} \end{pmatrix} \left[\begin{pmatrix} J_{xz}pq + (J_y - J_z)qr \\ J_{xz}(r^2 - p^2) + (J_z - J_x)pr \\ (J_x - J_y)pq - J_{xz}qr \end{pmatrix} + \begin{pmatrix} l \\ m \\ n \end{pmatrix} \right] \\ &= \begin{pmatrix} \Gamma_1 pq - \Gamma_2 qr + \Gamma_3 l + \Gamma_4 n \\ \Gamma_5 pr - \Gamma_6(p^2 - r^2) + \frac{1}{J_y} m \\ \Gamma_7 pq - \Gamma_1 qr + \Gamma_4 l + \Gamma_8 n \end{pmatrix},\end{aligned}\tag{3.12}$$

■

where

$$\begin{aligned}\Gamma_1 &= \frac{J_{xz}(J_x - J_y + J_z)}{\Gamma} \\ \Gamma_2 &= \frac{J_z(J_z - J_y) + J_{xz}^2}{\Gamma} \\ \Gamma_3 &= \frac{J_z}{\Gamma} \\ \Gamma_4 &= \frac{J_{xz}}{\Gamma} \\ \Gamma_5 &= \frac{J_z - J_x}{J_y} \\ \Gamma_6 &= \frac{J_{xz}}{J_y} \\ \Gamma_7 &= \frac{(J_x - J_y)J_x + J_{xz}^2}{\Gamma} \\ \Gamma_8 &= \frac{J_x}{\Gamma}.\end{aligned}\tag{3.13}$$

The six-degree-of-freedom, 12-state model for the MAV kinematics and dynamics are given by Equations (3.1), (3.3), (3.7), and (3.12), and are sum-

marized as follows:

$$\begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{p}_d \end{pmatrix} = \begin{pmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix} \quad (3.14)$$

$$\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} = \begin{pmatrix} rv - qw \\ pw - ru \\ qu - pv \end{pmatrix} + \frac{1}{m} \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix}, \quad (3.15)$$

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & \sin \phi \tan \theta & \cos \phi \tan \theta \\ 0 & \cos \phi & -\sin \phi \\ 0 & \frac{\sin \phi}{\cos \theta} & \frac{\cos \phi}{\cos \theta} \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} \quad (3.16)$$

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} \Gamma_1 pq - \Gamma_2 qr \\ \Gamma_5 pr - \Gamma_6(p^2 - r^2) \\ \Gamma_7 pq - \Gamma_1 qr \end{pmatrix} + \begin{pmatrix} \Gamma_3 l + \Gamma_4 n \\ \frac{1}{J_y} m \\ \Gamma_4 l + \Gamma_8 n \end{pmatrix}. \quad (3.17)$$

Equations (3.14)-(3.17) represent the dynamics of the MAV. They are not complete in that the externally applied forces and moments are not yet defined. Models for forces and moments due to gravity, aerodynamics, and propulsion will be derived in Chapter 4. In Appendix B, an alternative formulation to these equations that uses quaternions to represent the MAV attitude is given.

3.4 CHAPTER SUMMARY

In this chapter, we have derived a six-degree-of-freedom, 12-state dynamic model for a MAV from first principles. This model will be the basis for analysis, simulation, and control design that will be discussed in forthcoming chapters.

NOTES AND REFERENCES

The material in this chapter is standard, and similar discussions can be found in textbooks on mechanics [9, 10, 12], space dynamics [13, 14], flight dynamics [6, 15, 16, 1, 7, 17] and robotics [4, ?].

Equations (3.14) and (3.15) are expressed in terms of inertially referenced velocities u , v , and w . Alternatively, they can be expressed in terms of velocities referenced to the air-mass surrounding the aircraft u_r , v_r , and w_r

as

$$\begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{p}_d \end{pmatrix} = \mathcal{R}_b^v(\phi, \theta, \psi) \begin{pmatrix} u_r \\ v_r \\ w_r \end{pmatrix} + \begin{pmatrix} w_n \\ w_e \\ w_d \end{pmatrix} \quad (3.18)$$

$$\begin{pmatrix} \dot{u}_r \\ \dot{v}_r \\ \dot{w}_r \end{pmatrix} = \begin{pmatrix} rv_r - qw_r \\ pw_r - ru_r \\ qu_r - pv_r \end{pmatrix} + \frac{1}{m} \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} - \mathcal{R}_v^b(\phi, \theta, \psi) \begin{pmatrix} \dot{w}_n \\ \dot{w}_e \\ \dot{w}_d \end{pmatrix}, \quad (3.19)$$

where

$$\mathcal{R}_b^v(\phi, \theta, \psi) = (\mathcal{R}_v^b)^\top(\phi, \theta, \psi) = \begin{pmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{pmatrix}.$$

The choice of which equations to use to express the aircraft kinematics is a matter of personal preference. In Equations (3.14) and (3.15), the velocity states u , v , and w represent the aircraft motion with respect to the ground (inertial frame). In Equations (3.18) and (3.19), the velocity states u_r , v_r , and w_r represent the aircraft motion with respect to the air mass surrounding the aircraft. To correctly represent the motion of the aircraft in the inertial frame using u_r , v_r , and w_r as states, the effect of wind speed and wind acceleration must be taken into account.

3.5 DESIGN PROJECT

MODIFIED MATERIAL:

- 3.1 Implement the MAV equations of motion given in Equations (3.14) through (3.17). 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

The objective of this chapter is to describe the forces and moments that act on a MAV. Following [16], we will assume that the forces and moments are primarily due to three sources, namely, gravity, aerodynamics, and propulsion. Letting \mathbf{f}_g be the force due to gravity, $(\mathbf{f}_a, \mathbf{m}_a)$ be the forces and moments due to aerodynamics, and $(\mathbf{f}_p, \mathbf{m}_p)$ be the forces and moments due to propulsion, we have

$$\begin{aligned}\mathbf{f} &= \mathbf{f}_g + \mathbf{f}_a + \mathbf{f}_p \\ \mathbf{m} &= \mathbf{m}_a + \mathbf{m}_p,\end{aligned}$$

where \mathbf{f} is the total force acting on the airframe and \mathbf{m} is the total moment acting on the airframe.

In this chapter, we derive expressions for each of the forces and moments. Gravitational forces are discussed in Section 4.1, aerodynamic forces and torques are described in Section 4.2, and the forces and torques due to propulsion are described in Section 4.3. Atmospheric disturbances, described in Section 4.4, are modeled as changes in the wind speed and enter the equations of motion through the aerodynamic forces and torques.

4.1 GRAVITATIONAL FORCES

The effect of the earth's gravitational field on a MAV can be modeled as a force proportional to the mass acting at the center of mass. This force acts in the \mathbf{k}^i direction and is proportional to the mass of the MAV by the gravitational constant g . In the vehicle frame \mathcal{F}^v , the gravity force acting on the center of mass is given by

$$\mathbf{f}_g^v = \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix}.$$

When applying Newton's second law in Chapter 3, we summed forces along the axes in the body frame. Therefore, we must transform the gravitational

force into its body-frame components to give

$$\begin{aligned}\mathbf{f}_g^b &= \mathcal{R}_v^b \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} \\ &= \begin{pmatrix} -mg \sin \theta \\ mg \cos \theta \sin \phi \\ mg \cos \theta \cos \phi \end{pmatrix}.\end{aligned}$$

Since the gravity force acts through the center of mass of the MAV, there are no moments produced by gravity.

NEW MATERIAL:

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

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

4.2 AERODYNAMIC FORCES AND MOMENTS

As a MAV passes through the air, a pressure distribution is generated around the MAV body, as depicted in Figure 4.1. The strength and distribution of the pressure acting on the MAV is a function of the airspeed, the air density, and the shape and attitude of the MAV. Accordingly, the dynamic pressure is given by $\frac{1}{2}\rho V_a^2$, where ρ is the air density and V_a is the speed of the MAV through the surrounding air mass.

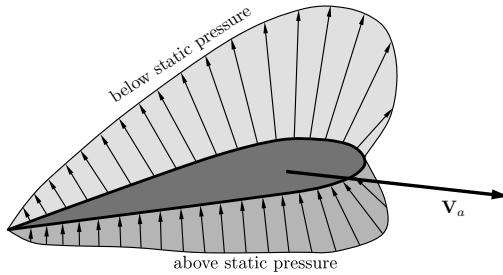


Figure 4.1: Pressure distribution around an airfoil.

Instead of attempting to characterize the pressure distribution around the wing, the common approach is to capture the effect of the pressure with a combination of forces and a moment. For example, if we consider the

longitudinal (\mathbf{i}^b - \mathbf{k}^b) plane, the effect of the pressure acting on the MAV body can be modeled using a lift force, a drag force, and a moment. As shown in Figure 4.2, the lift and drag forces are applied at the quarter-chord point, also known as the aerodynamic center.

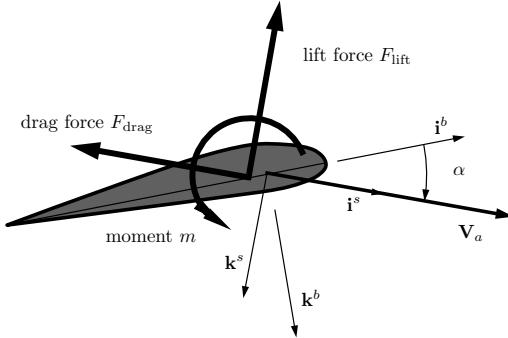


Figure 4.2: Effect of pressure distribution can be modeled using a lift force, a drag force, and a moment.

The lift, drag, and moment are commonly expressed as

$$\begin{aligned} F_{\text{lift}} &= \frac{1}{2}\rho V_a^2 S C_L \\ F_{\text{drag}} &= \frac{1}{2}\rho V_a^2 S C_D \\ m &= \frac{1}{2}\rho V_a^2 S c C_m, \end{aligned} \quad (4.1)$$

where C_L , C_D , and C_m are nondimensional aerodynamic coefficients, S is the planform area of the MAV wing, and c is the mean chord of the MAV wing. For airfoils generally, the lift, drag, and pitching moment coefficients are significantly influenced by the airfoil shape, Reynolds number, Mach number, and the angle of attack. For the range of airspeeds flown by small and miniature aircraft, the Reynolds number and Mach number effects are approximately constant. We will consider the effects of the angles α and β ; the angular rates p , q , and r ; and the deflection of control surfaces on the aerodynamic coefficients.

It is common to decompose the aerodynamic forces and moments into two groups: longitudinal and lateral. The longitudinal forces and moments act in the \mathbf{i}^b - \mathbf{k}^b plane, also called the pitch plane. They include the forces in the \mathbf{i}^b and \mathbf{k}^b directions (caused by lift and drag) and the moment about the \mathbf{j}^b axis. The lateral forces and moments include the force in the \mathbf{j}^b direction and the moments about the \mathbf{i}^b and \mathbf{k}^b axes.

4.2.1 Control Surfaces

Before giving detailed equations that describe the aerodynamic forces and moments due to the lifting surfaces, we need to define the control surfaces that are used to maneuver the aircraft. The control surfaces are used to modify the aerodynamic forces and moments. For standard aircraft configurations, the control surfaces include the elevator, the aileron, and the rudder. Other surfaces, including spoilers, flaps, and canards, will not be discussed in this book but are modeled similarly.

Figure 4.3 shows the standard configuration, where the aileron deflection is denoted by δ_a , the elevator deflection is denoted by δ_e , and the rudder deflection is denoted by δ_r . The positive direction of a control surface deflection can be determined by applying the right-hand rule to the hinge axis of the control surface. For example, the hinge axis of the elevator is aligned with the body j^b axis. Applying the right-hand rule about the j^b axis implies that a positive deflection for the elevator is trailing edge down. Similarly, positive deflection for the rudder is trailing edge left. Finally, positive aileron deflection is trailing edge down on each aileron. The aileron deflection δ_a can be thought of as a composite deflection defined by

$$\delta_a = \frac{1}{2} (\delta_{a\text{-left}} - \delta_{a\text{-right}}).$$

Therefore a positive δ_a is produced when the left aileron is trailing edge down and the right aileron is trailing edge up.

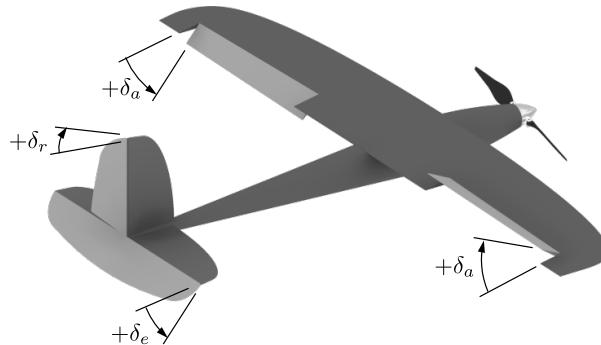


Figure 4.3: Control surfaces for a standard aircraft configuration. The ailerons are used to control the roll angle ϕ . The elevators are used to control the pitch angle θ . The rudder directly effects the yaw angle ψ .

For small aircraft, there are two other standard configurations. The first is the v-tail configuration as shown in Figure 4.4. The control surfaces for a v-tail are called ruddervators. The angular deflection of the right ruddervator is

denoted as δ_{rr} , and the angular deflection of the left ruddervator is denoted as δ_{rl} . Driving the ruddervators differentially has the same effect as a rudder, producing a torque about \mathbf{k}^b . Driving the ruddervators together has the same effect as an elevator, producing torque about \mathbf{j}^b . Mathematically, we can convert between ruddervators and rudder-elevator signals as

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

Using this relation, the mathematical model for forces and torques for v-tail aircraft can be expressed in terms of standard rudder-elevator notation.

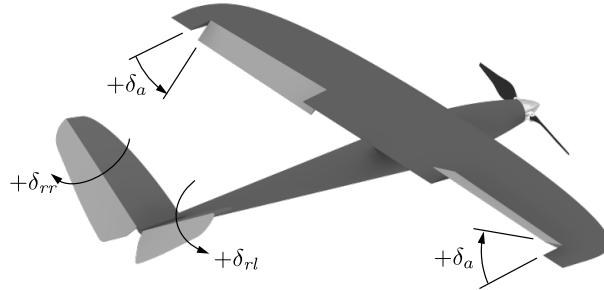


Figure 4.4: Ruddervators are used to control a v-tail aircraft. The ruddervators replace the rudder and the elevator. Driving the ruddervators together has the same effect as an elevator, and driving them differentially has the same effect as a rudder.

The other standard configuration for small aircraft is the flying wing depicted in Figure 4.5. The control surfaces for a flying wing are called elevons. The angular deflection of the right elevon is denoted as δ_{er} , and the angular deflection of the left elevon is denoted as δ_{el} . Driving the elevons differentially has the same effect as ailerons, producing a torque about \mathbf{i}^b , while driving the elevons together has the same effect as an elevator, causing a torque about \mathbf{j}^b . Mathematically, we can convert between elevons and aileron-elevator signals with

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

Therefore, the mathematical model for forces and torques for flying-wing aircraft can be expressed in terms of standard aileron-elevator notation.

4.2.2 Longitudinal Aerodynamics

The longitudinal aerodynamic forces and moments cause motion in the body \mathbf{i}^b - \mathbf{k}^b plane, also known as the pitch plane. They are the aerodynamic forces

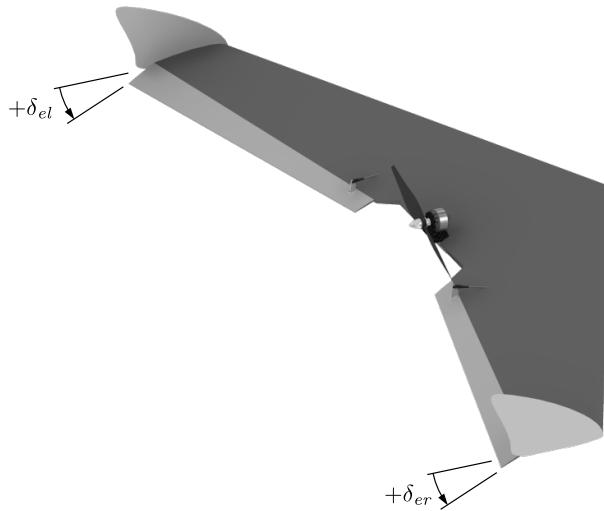


Figure 4.5: Elevons are used to control a flying-wing aircraft. The elevons replace the aileron and the elevator. Driving the elevons together has the same effect as an elevator, and driving them differentially has the same effect as ailerons.

and moment with which we are perhaps most familiar: lift, drag, and pitching moment. By definition, the lift and drag forces are aligned with the axes of the stability frame, as shown in Figure 4.2. When represented as a vector, the pitching moment also aligns with the j^s axis of the stability frame. The lift and drag forces and the pitching moment are heavily influenced by the angle of attack. The pitch rate q and the elevator deflection δ_e also influence the longitudinal forces and moment. Based on this, we can rewrite the equations for lift, drag, and pitching moment to express this functional dependence on α , q and δ_e as

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

In general, these force and moment equations are nonlinear. For small angles of attack, however, the flow over the wing will remain laminar and attached. Under these conditions, the lift, drag, and pitching moment can be modeled with acceptable accuracy using linear approximations. Working with the lift equation as an example, a first-order Taylor series approxima-

tion of the lift force can be written as

$$F_{\text{lift}} = \frac{1}{2} \rho V_a^2 S \left[C_{L_0} + \frac{\partial C_L}{\partial \alpha} \alpha + \frac{\partial C_L}{\partial q} q + \frac{\partial C_L}{\partial \delta_e} \delta_e \right]. \quad (4.2)$$

The coefficient C_{L_0} is the value of the C_L when $\alpha = q = \delta_e = 0$. It is common to nondimensionalize the partial derivatives of this linear approximation. Since C_L and the angles α and δ_e (expressed in radians) are dimensionless, the only partial requiring nondimensionalization is $\partial C_L / \partial q$. Since the units of q are rad/s, a standard factor to use is $c/(2V_a)$. We can then rewrite Equation (4.2) as

$$F_{\text{lift}} = \frac{1}{2} \rho V_a^2 S \left[C_{L_0} + C_{L_\alpha} \alpha + C_{L_q} \frac{c}{2V_a} q + C_{L_{\delta_e}} \delta_e \right], \quad (4.3)$$

where the coefficients C_{L_0} , $C_{L_\alpha} \triangleq \frac{\partial C_L}{\partial \alpha}$, $C_{L_q} \triangleq \frac{\partial C_L}{\partial \frac{q c}{2V_a}}$, and $C_{L_{\delta_e}} \triangleq \frac{\partial C_L}{\partial \delta_e}$ are dimensionless quantities. C_{L_α} and C_{L_q} are commonly referred to as stability derivatives, while $C_{L_{\delta_e}}$ is an example of a control derivative. The label “derivative” comes from the fact that the coefficients originated as partial derivatives in the Taylor series approximation. In a similar manner, we express linear approximations for the aerodynamic drag force and pitching moment as

$$F_{\text{drag}} = \frac{1}{2} \rho V_a^2 S \left[C_{D_0} + C_{D_\alpha} \alpha + C_{D_q} \frac{c}{2V_a} q + C_{D_{\delta_e}} \delta_e \right] \quad (4.4)$$

$$m = \frac{1}{2} \rho V_a^2 S c \left[C_{m_0} + C_{m_\alpha} \alpha + C_{m_q} \frac{c}{2V_a} q + C_{m_{\delta_e}} \delta_e \right]. \quad (4.5)$$

Equations (4.3), (4.4), and (4.5) are commonly used as the basis for the longitudinal aerodynamic model. Under typical, low-angle-of-attack flight conditions, they are a sufficiently accurate representation of the forces and moments produced. The flow over the aircraft body is laminar and attached and the flow field over the aircraft is termed quasi-steady, meaning it only changes slowly with respect to time. The shape of the flow field is predictable and changes in response to changes in the angle of attack, pitch rate, and elevator deflection. The quasi-steady behavior of the flow field results in longitudinal aerodynamic forces and torques that are predictable and fairly straightforward to model, as shown above.

In contrast to the quasi-steady aerodynamics typically experienced by aircraft, unsteady aerodynamics are challenging to model and predict. Unsteady aerodynamics are characterized by nonlinear, three-dimensional, time-varying, separated flows that significantly affect the forces and moments experienced by the aircraft. Two unsteady flow scenarios of possible interest to MAV designers are high-angle-of-attack, high-angular-rate aircraft

maneuvers, such as those performed by fighter aircraft, and flapping-wing flight. In fact, the efficiency and maneuverability demonstrated by insects and birds is in part because of their ability to exploit unsteady aerodynamic flow effects.

Perhaps the most important unsteady flow phenomena for MAV designers and users to understand is stall, which occurs when the angle of attack increases to the point that the flow separates from the wing, resulting in a drastic loss of lift. Under stall conditions, Equations (4.3), (4.4), and (4.5) produce dangerously optimistic estimates of the aerodynamic forces on the aircraft. This wing stall phenomenon is depicted in Figure 4.6. At low or moderate angles of attack, the flow over the wing is laminar and stays attached to the wing as it flows over it. It is this attached flow over the wing that produces the desired lift. When the angle of attack exceeds the critical stall angle, the flow begins to separate from the top surface of the wing causing turbulent flow and an abrupt drop in the lift produced by the wing, which can lead to catastrophic results for the aircraft. The key weakness of the linear aerodynamic model of Equations (4.3) to (4.5) is that it fails to predict this abrupt drop in lift force with increasing angle of attack. Instead, it erroneously predicts that the lift force continues to increase as the angle of attack increases to physically unrealistic flight conditions. Given that the MAV dynamic model presented here could be used to design control laws for real aircraft and to simulate their performance, it is important that the effects of wing stall be incorporated into the longitudinal aerodynamic model.

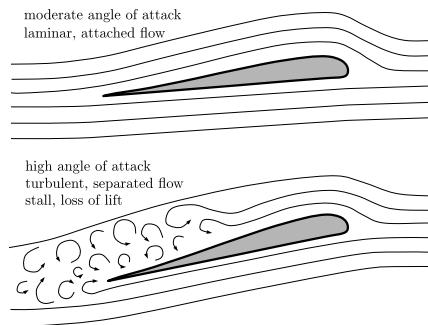


Figure 4.6: The upper drawing depicts a wing under normal flow conditions. The flow is laminar and follows the surface of the wing. The lower drawing shows a wing under stall conditions due to a high angle of attack. In this case, the flow separates from the top surface of the wing, leading to turbulent flow and a significant drop in the lift force produced by the wing.

To incorporate wing stall into our longitudinal aerodynamic model, we

modify Equations (4.3) and (4.4) so that the lift and drag forces are nonlinear in angle of attack. This will allow us to more accurately model lift and drag over wider ranges of α . Lift and drag can be modeled more generally as

$$F_{\text{lift}} = \frac{1}{2} \rho V_a^2 S \left[C_L(\alpha) + C_{Lq} \frac{c}{2V_a} q + C_{L\delta_e} \delta_e \right] \quad (4.6)$$

$$F_{\text{drag}} = \frac{1}{2} \rho V_a^2 S \left[C_D(\alpha) + C_{Dq} \frac{c}{2V_a} q + C_{D\delta_e} \delta_e \right], \quad (4.7)$$

where C_L and C_D are now expressed nonlinear functions of α . For angles of attack that are beyond the onset of stall conditions, the wing acts roughly like a flat plate, whose lift coefficient can be modeled as [17]

$$C_{L,\text{flat plate}} = 2\text{sign}(\alpha) \sin^2 \alpha \cos \alpha. \quad (4.8)$$

To obtain an accurate model of lift versus angle of attack for a specific wing design over a large range of angles of attack requires either wind tunnel testing or a detailed computational study. While for many simulation purposes it may not be necessary to have a high-fidelity lift model specific to the aircraft under consideration, it is desirable to have a lift model that incorporates the effects of stall. A lift model that incorporates the common linear lift behavior and the effects of stall is given by

$$C_L(\alpha) = (1 - \sigma(\alpha)) [C_{L_0} + C_{L\alpha} \alpha] + \sigma(\alpha) [2\text{sign}(\alpha) \sin^2 \alpha \cos \alpha], \quad (4.9)$$

where

$$\sigma(\alpha) = \frac{1 + e^{-M(\alpha - \alpha_0)} + e^{M(\alpha + \alpha_0)}}{(1 + e^{-M(\alpha - \alpha_0)}) (1 + e^{M(\alpha + \alpha_0)})}, \quad (4.10)$$

and M and α_0 are positive constants. The sigmoid function in Equation (4.10) is a blending function with cutoff at $\pm\alpha_0$ and transition rate M . Figure 4.7 shows the lift coefficient in Equation (4.9) as a blended function of the linear term $C_{L_0} + C_{L\alpha} \alpha$ and the flat plate term in Equation (4.8). For small aircraft, the linear lift coefficient can be reasonably approximated as

$$C_{L\alpha} = \frac{\pi AR}{1 + \sqrt{1 + (AR/2)^2}},$$

where $AR \triangleq b^2/S$ is the wing aspect ratio, b is the wingspan, and S is the wing area.

The drag coefficient C_D is also a nonlinear function of the angle of attack. There are two contributions to the drag coefficient, namely induced drag and parasitic drag [17]. The parasitic drag, generated by the shear stress of air

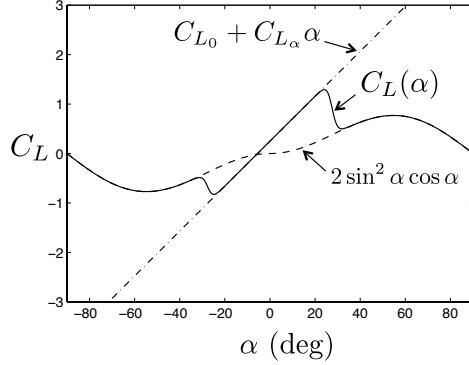


Figure 4.7: The lift coefficient as a function of α (solid) can be approximated by blending a linear function of alpha (dot-dashed), with the lift coefficient of a flat plate (dashed).

moving over the wing and other effects, is roughly constant and is denoted by C_{D_p} .¹ For small angles of attack, the induced drag is proportional to the square of the lift force. Combining the parasitic drag and the induced drag, we have

$$C_D(\alpha) = C_{D_p} + \frac{(C_{L_0} + C_{L_\alpha} \alpha)^2}{\pi e_{os} AR}. \quad (4.11)$$

The parameter e_{os} is the Oswald efficiency factor, which ranges between 0.8 and 1.0 [7].

Figure 4.8 shows typical plots of drag coefficient versus angle of attack for quadratic and linear models. The quadratic model correctly models the drag force as an odd function with respect to α . The drag force is always opposite the forward velocity of the aircraft, independent of the sign of the angle of attack. The linear model incorrectly predicts that the drag force becomes negative (pushing the aircraft forward) when the angle of attack becomes sufficiently negative. The figure clarifies the difference between the parasitic drag, C_{D_p} , also known as the zero-lift drag coefficient, and C_{D_0} , the drag coefficient predicted by the linear model at zero angle of attack. The parameters α^* and C_D^* are the angle of attack and the corresponding drag coefficient at a nominal operating condition $\alpha = \alpha^*$ about which C_D is linearized. While the quadratic model provides a more accurate representation of the influence of the angle of attack over a wider range of α ,

¹The parasitic drag is commonly denoted in the aerodynamics literature as C_{D_0} . To avoid confusion with the constant term of Equation (4.4), we will call it C_{D_p} . See [7, pp. 179–180] for a detailed explanation.

the linear model is sometimes used because of its simplicity and its fidelity under typical flight conditions.

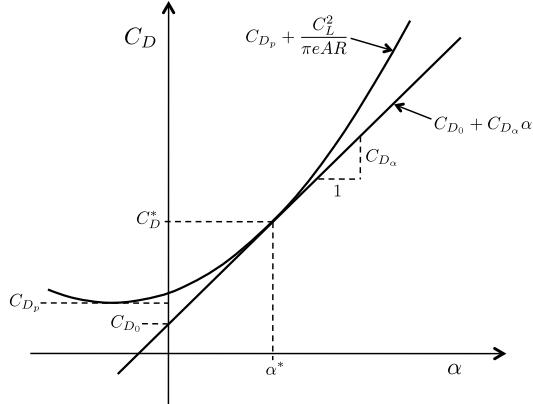


Figure 4.8: The drag coefficient as a function of angle of attack. Linear and quadratic models are represented.

The lift and drag forces expressed in Equations (4.6) and (4.7) are expressed in the stability frame. To express lift and drag in the body frame requires a rotation by the angle of attack:

$$\begin{aligned} \begin{pmatrix} f_x \\ f_z \end{pmatrix} &= \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix} \begin{pmatrix} -F_{\text{drag}} \\ -F_{\text{lift}} \end{pmatrix} \\ &= \frac{1}{2} \rho V_a^2 S \begin{pmatrix} [-C_D(\alpha) \cos \alpha + C_L(\alpha) \sin \alpha] \\ + [-C_{D_q} \cos \alpha + C_{L_q} \sin \alpha] \frac{c}{2V_a} q + [-C_{D_{\delta_e}} \cos \alpha + C_{L_{\delta_e}} \sin \alpha] \delta_e \\ \hline [-C_D(\alpha) \sin \alpha - C_L(\alpha) \cos \alpha] \\ + [-C_{D_q} \sin \alpha - C_{L_q} \cos \alpha] \frac{c}{2V_a} q + [-C_{D_{\delta_e}} \sin \alpha - C_{L_{\delta_e}} \cos \alpha] \delta_e \end{pmatrix}. \end{aligned}$$

The functions $C_L(\alpha)$ and $C_D(\alpha)$ used in the force model above can be nonlinear functions expressed in Equations (4.9) and (4.11), which are valid over a wide range of angles of attack. Alternatively, if simpler models are desired, the linear coefficient models given by

$$C_L(\alpha) = C_{L_0} + C_{L_\alpha} \alpha \quad (4.12)$$

$$C_D(\alpha) = C_{D_0} + C_{D_\alpha} \alpha \quad (4.13)$$

can be used.

The pitching moment of the aircraft is generally a nonlinear function of angle of attack and must be determined by wind tunnel or flight experiments for the specific aircraft of interest. For the purposes of simulation, we will

use the linear model

$$C_m(\alpha) = C_{m_0} + C_{m_\alpha} \alpha,$$

where $C_{m_\alpha} < 0$ implies that the airframe is inherently pitch stable.

4.2.3 Lateral Aerodynamics

The lateral aerodynamic force and moments cause translational motion in the lateral direction along the \mathbf{j}^b axis as well as rotational motions in roll and yaw that will result in directional changes in the flight path of the MAV. The lateral aerodynamics are most significantly influenced by the sideslip angle β . They are also influenced by the roll rate p , the yaw rate r , the deflection of the aileron δ_a , and the deflection of the rudder δ_r . Denoting the lateral force as f_y and the roll and yaw moments as l and n , respectively, we have

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

where C_Y , C_l , and C_n are nondimensional aerodynamic coefficients, and b is the wingspan of the aircraft. As with the longitudinal aerodynamic forces and moments, the coefficients C_Y , C_l , and C_n are nonlinear in their constitutive parameters, in this case β , p , r , δ_a , and δ_r . These nonlinear relationships, however, are difficult to characterize. Further, linear aerodynamic models yield acceptable accuracy in most applications and provide valuable insights into the dynamic stability of the aircraft. We will follow the approach used in Section 4.2.2 to produce the linear longitudinal aerodynamic models: first-order Taylor series approximation and nondimensionalization of the aerodynamic coefficients. Using this approach, linear relationships

for lateral force, roll moment, and yaw moment are given by

$$f_y \approx \frac{1}{2} \rho V_a^2 S \left[C_{Y_0} + C_{Y_\beta} \beta + C_{Y_p} \frac{b}{2V_a} p + C_{Y_r} \frac{b}{2V_a} r + C_{Y_{\delta_a}} \delta_a + C_{Y_{\delta_r}} \delta_r \right] \quad (4.14)$$

$$l \approx \frac{1}{2} \rho V_a^2 S b \left[C_{l_0} + C_{l_\beta} \beta + C_{l_p} \frac{b}{2V_a} p + C_{l_r} \frac{b}{2V_a} r + C_{l_{\delta_a}} \delta_a + C_{l_{\delta_r}} \delta_r \right] \quad (4.15)$$

$$n \approx \frac{1}{2} \rho V_a^2 S b \left[C_{n_0} + C_{n_\beta} \beta + C_{n_p} \frac{b}{2V_a} p + C_{n_r} \frac{b}{2V_a} r + C_{n_{\delta_a}} \delta_a + C_{n_{\delta_r}} \delta_r \right]. \quad (4.16) \blacksquare$$

These forces and moments are aligned with the body axes of the aircraft and do not require a rotational transformation to be implemented in the equations of motion. The coefficient C_{Y_0} is the value of the lateral force coefficient C_Y when $\beta = p = r = \delta_a = \delta_r = 0$. For aircraft that are symmetrical about the $\mathbf{i}^b\text{-}\mathbf{k}^b$ plane, C_{Y_0} is typically zero. The coefficients C_{l_0} and C_{n_0} are defined similarly and are also typically zero for symmetric aircraft.

4.2.4 Aerodynamic Coefficients

The aerodynamic coefficients C_{m_α} , C_{l_β} , C_{n_β} , C_{m_q} , C_{l_p} , and C_{n_r} are referred to as *stability derivatives* because their values determine the static and dynamic stability of the MAV. Static stability deals with the direction of aerodynamic moments as the MAV is perturbed away from its nominal flight condition. If the moments tend to restore the MAV to its nominal flight condition, the MAV is said to be statically stable. Most aircraft are designed to be statically stable. The coefficients C_{m_α} , C_{l_β} , and C_{n_β} determine the static stability of the MAV. They represent the change in the moment coefficients with respect to changes in the direction of the relative airspeed, as represented by α and β .

C_{m_α} is referred to as the longitudinal static stability derivative. For the MAV to be statically stable, C_{m_α} must be less than zero. In this case, an increase in α due to an updraft would cause the MAV to nose down in order to maintain the nominal angle of attack.

C_{l_β} is called the roll static stability derivative and is typically associated with dihedral in the wings. For static stability in roll, C_{l_β} must be negative. A negative value for C_{l_β} will result in rolling moments that roll the MAV away from the direction of sideslip, thereby driving the sideslip angle β to zero.

C_{n_β} is referred to as the yaw static stability derivative and is sometimes called the weathercock stability derivative. If an aircraft is statically stable

in yaw, it will naturally point into the wind like a weathervane (or weather-cock). The value of C_{n_β} is heavily influenced by the design of the tail of the aircraft. The larger the tail and the further the tail is aft of the center of mass of the aircraft, the larger C_{n_β} will be. For the MAV to be stable in yaw, C_{n_β} must be positive. This simply implies that for a positive sideslip angle, a positive yawing moment will be induced. This yawing moment will yaw the MAV into the direction of the relative airspeed, driving the sideslip angle to zero.

Dynamic stability deals with the dynamic behavior of the airframe in response to disturbances. If a disturbance is applied to the MAV, the MAV is said to be dynamically stable if the response of the MAV damps out over time. If we use a second-order mass-spring-damper analogy to analyze the MAV, the stability derivatives C_{m_α} , C_{l_β} , and C_{n_β} behave like torsional springs, while the derivatives C_{m_q} , C_{l_p} , and C_{n_r} behave like torsional dampers. The moments of inertia of the MAV body provide the mass. As we will see in Chapter 5, when we linearize the dynamic equations of motion for the MAV, the signs of the stability derivatives must be consistent in order to ensure that the characteristic roots of the MAV dynamics lie in the left half of the complex plane.

C_{m_q} is referred to as the pitch damping derivative, C_{l_p} is called the roll damping derivative, and C_{n_r} is referred to as the yaw damping derivative. Each of these damping derivatives is usually negative, meaning that a moment is produced that opposes the direction of motion, thus damping the motion.

The aerodynamic coefficients $C_{m_{\delta_e}}$, $C_{l_{\delta_a}}$, and $C_{n_{\delta_r}}$ are associated with the deflection of control surfaces and are referred to as the *primary control derivatives*. They are primary because the moments produced are the intended result of the specific control surface deflection. For example, the intended result of an elevator deflection δ_e is a pitching moment m . $C_{l_{\delta_r}}$ and $C_{n_{\delta_a}}$ are called *cross-control derivatives*. They define the off-axis moments that occur when the control surfaces are deflected. Control derivatives can be thought of as gains. The larger the value of the control derivative, the larger the magnitude of the moment produced for a given deflection of the control surface.

The sign convention described in Section 4.2.1 implies that a positive elevator deflection results in a nose-down pitching moment (negative about \mathbf{j}^b), positive aileron deflection causes a right-wing-down rolling moment (positive about \mathbf{i}^b), and positive rudder deflection causes a nose-left yawing moment (negative about \mathbf{k}^b). We will define the signs of the primary control derivatives so that positive deflections cause positive moments. For this to be the case, $C_{m_{\delta_e}}$ will be negative, $C_{l_{\delta_a}}$ will be positive, and $C_{n_{\delta_r}}$ will be negative.

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 4.9 and 4.10, 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 4.9 and 4.10, 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 4.9 and 4.10. Combining these equations, the thrust

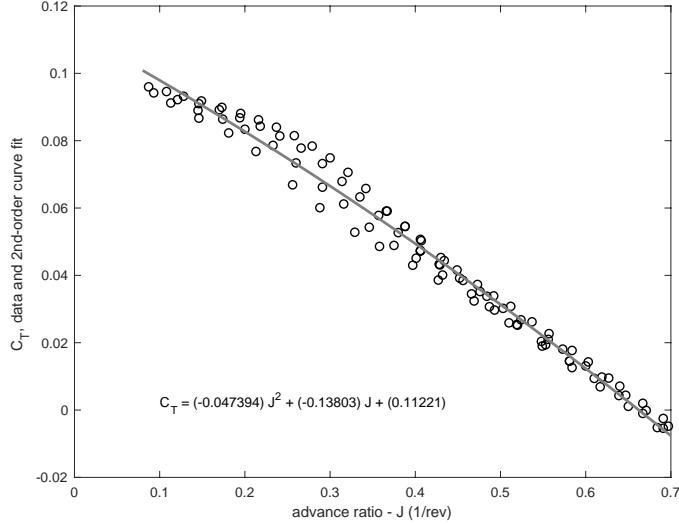


Figure 4.9: Thrust coefficient versus advance ratio for APC Thin Electric 10x5 propeller [?].

and torque produced by the propeller are given by

$$\begin{aligned} T_p(\Omega_p, V_a) &= \frac{\rho D^4}{4\pi^2} \Omega_p^2 (C_{T2} J^2(\Omega_p, V_a) + C_{T1} J(\Omega_p, V_a) + C_{T0}) \\ &= \left(\frac{\rho D^4 C_{T0}}{4\pi^2} \right) \Omega_p^2 + \left(\frac{\rho D^3 C_{T1} V_a}{2\pi} \right) \Omega_p + (\rho D^2 C_{T2} V_a^2) \end{aligned} \quad (4.17)$$

$$\begin{aligned} Q_p(\Omega_p, V_a) &= \frac{\rho D^5}{4\pi^2} \Omega_p^2 (C_{Q2} J^2(\Omega_p, V_a) + C_{Q1} J(\Omega_p, V_a) + C_{Q0}) \\ &= \left(\frac{\rho D^5 C_{Q0}}{4\pi^2} \right) \Omega_p^2 + \left(\frac{\rho D^4 C_{Q1} V_a}{2\pi} \right) \Omega_p + (\rho D^3 C_{Q2} V_a^2). \end{aligned} \quad (4.18)$$

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

$$Q_m = K_Q \left[\frac{1}{R} (V_{in} - K_V \Omega_m) - i_0 \right], \quad (4.19)$$

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

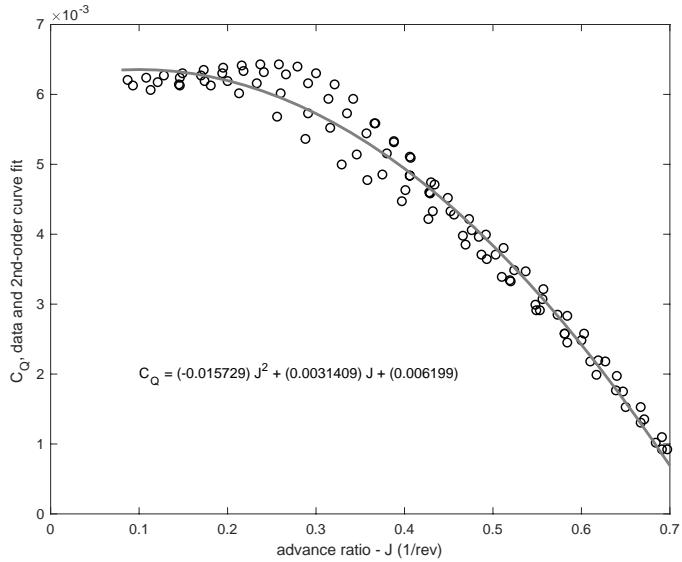


Figure 4.10: Torque coefficient versus advance ratio for APC Thin Electric 10x5 propeller [?].

energy domains. If consistent units (such as SI units) are utilized, K_Q and K_T are identical in value.² 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 (4.18) equal to Equation (4.19) and grouping like terms gives

$$\begin{aligned} & \left(\frac{\rho D^5}{(2\pi)^2} C_{Q0} \right) \Omega_p^2 + \left(\frac{\rho D^4}{2\pi} C_{Q1} V_a + \frac{K_Q^2}{R} \right) \Omega_p \\ & + \left(\rho D^3 C_{Q2} V_a^2 - \frac{K_Q}{R} V_{in} + K_Q i_0 \right) = 0. \quad (4.20) \end{aligned}$$

²We 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

$$\Omega_p(V_{in}, V_a) = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad (4.21)$$

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 4.11 and 4.12. Although the voltages are plotted at discrete intervals, the voltage input is assumed to be continuously variable. Figure 4.11 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,

$$V_{in} = V_{max} \delta_t, \quad (4.22)$$

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,

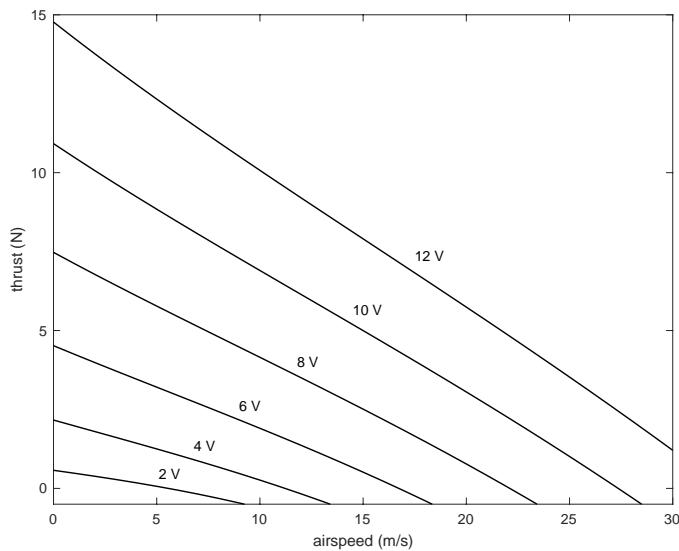


Figure 4.11: Propeller thrust versus airspeed for various motor voltage settings.

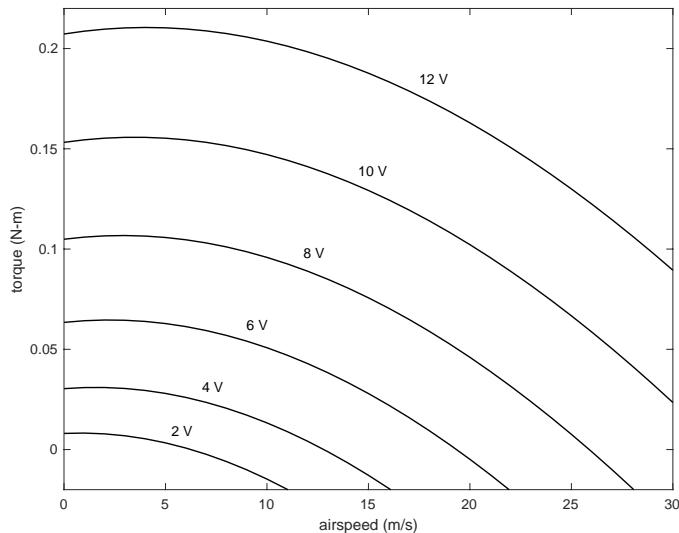


Figure 4.12: Motor torque versus airspeed for various motor voltage settings.

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

Compute V_{in} using Equation (4.22)

Compute Ω_p using Equation (4.21)

Compute T_p using Equation (4.17)

Compute Q_p using Equation (4.18)

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

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

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 (2.6), the relationship between ground velocity, air velocity, and wind velocity is given by

$$\mathbf{V}_g = \mathbf{V}_a + \mathbf{V}_w. \quad (4.23)$$

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

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

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 [17]. 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

$$\begin{aligned} H_u(s) &= \sigma_u \sqrt{\frac{2V_a}{\pi L_u}} \frac{1}{s + \frac{V_a}{L_u}} \\ H_v(s) &= \sigma_v \sqrt{\frac{3V_a}{\pi L_v}} \frac{\left(s + \frac{V_a}{\sqrt{3}L_v}\right)}{\left(s + \frac{V_a}{L_v}\right)^2} \\ H_w(s) &= \sigma_w \sqrt{\frac{3V_a}{\pi L_w}} \frac{\left(s + \frac{V_a}{\sqrt{3}L_w}\right)}{\left(s + \frac{V_a}{L_w}\right)^2}, \end{aligned}$$

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 4.13 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 (2.5). From the components of the wind velocity \mathbf{V}_w^b and the

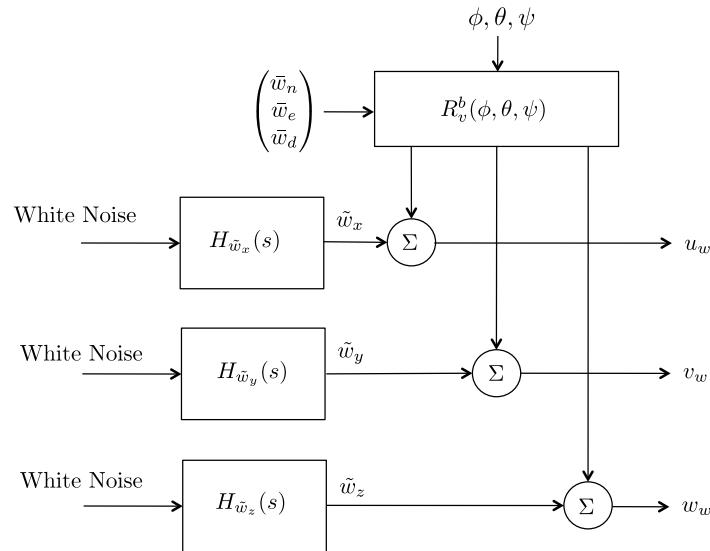


Figure 4.13: The wind is modeled as a constant wind field plus turbulence. The turbulence is generated by filtering white noise with a Dryden model.

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 (2.8) 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

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

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

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

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:

$$\begin{aligned}
 \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} &= \begin{pmatrix} -mg \sin \theta \\ mg \cos \theta \sin \phi \\ mg \cos \theta \cos \phi \end{pmatrix} + \begin{pmatrix} T_p \\ 0 \\ 0 \end{pmatrix} \\
 &\quad + \frac{1}{2} \rho V_a^2 S \begin{pmatrix} C_X(\alpha) + C_{X_q}(\alpha) \frac{c}{2V_a} q \\ C_{Y_0} + C_{Y_\beta} \beta + C_{Y_p} \frac{b}{2V_a} p + C_{Y_r} \frac{b}{2V_a} r \\ C_Z(\alpha) + C_{Z_q}(\alpha) \frac{c}{2V_a} q \end{pmatrix} \\
 &\quad + \frac{1}{2} \rho V_a^2 S \begin{pmatrix} C_{X_{\delta_e}}(\alpha) \delta_e \\ C_{Y_{\delta_a}} \delta_a + C_{Y_{\delta_r}} \delta_r \\ C_{Z_{\delta_e}}(\alpha) \delta_e \end{pmatrix}, \quad (4.24)
 \end{aligned}$$

where

$$\begin{aligned}
 C_X(\alpha) &\stackrel{\Delta}{=} -C_D(\alpha) \cos \alpha + C_L(\alpha) \sin \alpha \\
 C_{X_q}(\alpha) &\stackrel{\Delta}{=} -C_{D_q} \cos \alpha + C_{L_q} \sin \alpha \\
 C_{X_{\delta_e}}(\alpha) &\stackrel{\Delta}{=} -C_{D_{\delta_e}} \cos \alpha + C_{L_{\delta_e}} \sin \alpha \\
 C_Z(\alpha) &\stackrel{\Delta}{=} -C_D(\alpha) \sin \alpha - C_L(\alpha) \cos \alpha \\
 C_{Z_q}(\alpha) &\stackrel{\Delta}{=} -C_{D_q} \sin \alpha - C_{L_q} \cos \alpha \\
 C_{Z_{\delta_e}}(\alpha) &\stackrel{\Delta}{=} -C_{D_{\delta_e}} \sin \alpha - C_{L_{\delta_e}} \cos \alpha,
 \end{aligned} \tag{4.25}$$

and where $C_L(\alpha)$ is given by Equation (4.9) and $C_D(\alpha)$ is given by Equation (4.11). 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:

$$\begin{pmatrix} l \\ m \\ n \end{pmatrix} = \frac{1}{2} \rho V_a^2 S \begin{pmatrix} b \left[C_{l_0} + C_{l_\beta} \beta + C_{l_p} \frac{b}{2V_a} p + C_{l_r} \frac{b}{2V_a} r \right] \\ c \left[C_{m_0} + C_{m_\alpha} \alpha + C_{m_q} \frac{c}{2V_a} q \right] \\ b \left[C_{n_0} + C_{n_\beta} \beta + C_{n_p} \frac{b}{2V_a} p + C_{n_r} \frac{b}{2V_a} r \right] \end{pmatrix} + \frac{1}{2} \rho V_a^2 S \begin{pmatrix} b \left[C_{l_{\delta_a}} \delta_a + C_{l_{\delta_r}} \delta_r \right] \\ c \left[C_{m_{\delta_e}} \delta_e \right] \\ b \left[C_{n_{\delta_a}} \delta_a + C_{n_{\delta_r}} \delta_r \right] \end{pmatrix} + \begin{pmatrix} Q_p \\ 0 \\ 0 \end{pmatrix}. \tag{4.26}$$

NOTES AND REFERENCES

The material in this chapter can be found in most textbooks on flight dynamics, including [7, 17, 6, 15, 16, 1, 18]. Our discussion on lift, drag, and moment coefficients is drawn primarily from [17]. Decomposing the wind vector into a constant and a random term follows [17]. Our discussion of aircraft aerodynamics and dynamics focused on effects of primary significance. A more thorough coverage of flight mechanics, including topics such as ground effect and gyroscopic effects, can be found in [18].

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

As Chapters 3 and 4 have shown, the equations of motion for a MAV are a fairly complicated set of 12 nonlinear, coupled, first-order, ordinary differential equations, which we will present in their entirety in Section ???. Because of their complexity, designing controllers based on them is difficult and requires more straightforward approaches. In this chapter, we will linearize and decouple the equations of motion to produce reduced-order transfer function and state-space models more suitable for control system design. Low-level autopilot control loops for unmanned aircraft will be designed based on these linear design models, which capture the approximate dynamic behavior of the system under specific conditions. The objective of this chapter is to derive the linear design models that will be used in Chapter 6 to design the autopilot.

The dynamics for fixed-wing aircraft can be approximately decomposed into longitudinal motion, which includes airspeed, pitch angle, and altitude, and into lateral motion, which includes roll and heading angles. While there is coupling between longitudinal and lateral motion, for most airframes the dynamic coupling is sufficiently small that its unwanted effects can be mitigated by control algorithms designed for disturbance rejection. In this chapter we will follow the standard convention and decompose the dynamics into lateral and longitudinal motion. Many of the linear models presented in this chapter are derived with respect to an equilibrium condition. In flight dynamics, force and moment equilibrium is called *trim*, which is discussed in Section 5.2. Transfer functions for both the lateral and longitudinal dynamics are derived in Section 5.3. State-space models are derived in Section 5.4.

5.1 COORDINATED TURN

Referring to Equation (5.13), we can see that heading rate is related to the pitch rate, yaw rate, pitch, and roll states of the aircraft. Each of these states is governed by an ordinary differential equation. Physically, we know that heading rate is related to the roll or bank angle of the aircraft, and we seek a simplified relationship to help us develop linear transfer function relation-

ships in coming sections of this chapter. The coordinated-turn condition provides this relationship. The coordinated turn is a sought-after flight condition in manned flight for reasons of passenger comfort. During a coordinated turn, there is no lateral acceleration in the body frame of the aircraft. The aircraft “carves” the turn rather than skidding laterally. From an analysis perspective, the assumption of a coordinated turn allows us to develop a simplified expression that relates course (or heading) rate and bank angle, as shown by Phillips [18]. During a coordinated turn, the bank angle ϕ is set so that there is no net side force acting on the MAV. As shown in the free-body diagram of Figure 5.1, the centrifugal force acting on the MAV is equal and opposite to the horizontal component of the lift force. Summing forces in the horizontal direction gives

$$\begin{aligned} F_{\text{lift}} \sin \phi &= m \frac{v^2}{R} \\ &= mv\omega \\ &= m(V_g \cos \gamma) \dot{\chi}, \end{aligned} \quad (5.1)$$

where F_{lift} is the lift force, γ is the flight path angle, V_g is the ground speed, and $\dot{\chi}$ is the course angle. The centrifugal force is calculated using the

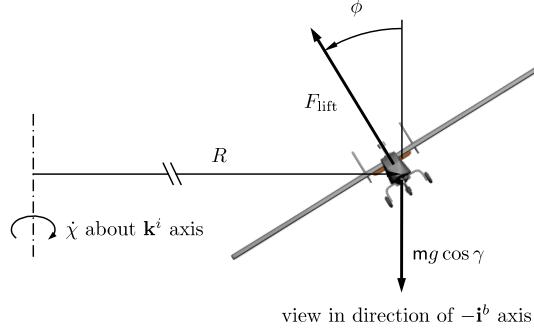


Figure 5.1: Free-body diagram indicating forces on a MAV in a climbing coordinated turn. The MAV is flying at a flight path angle of γ with the nose of the MAV directed out of the page. The forces shown are in the j^b - k^b plane.

angular rate $\dot{\chi}$ about the inertial frame k^i axis and the horizontal component of the groundspeed, $V_g \cos \gamma$. Similarly, the vertical component of the lift force is equal and opposite to the projection of the gravitational force onto the j^b - k^b plane as shown in Figure 5.1. Summing vertical force components gives

$$F_{\text{lift}} \cos \phi = mg \cos \gamma. \quad (5.2)$$

Dividing Equation (5.1) by Equation (5.2) and solving for $\dot{\chi}$ gives

$$\dot{\chi} = \frac{g}{V_g} \tan \phi, \quad (5.3)$$

which is the equation for a coordinated turn. Given that the turning radius is given by $R = V_g \cos \gamma / \dot{\chi}$, we get

$$R = \frac{V_g^2 \cos \gamma}{g \tan \phi}. \quad (5.4)$$

In the absence of wind or sideslip, we have that $V_a = V_g$ and $\psi = \chi$, which leads to the more common expression for the coordinated turn

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

These expressions for the coordinated turn will be used at several points in the text as a means for deriving simplified expressions for the turn dynamics of a MAV. Further discussion on coordinated turns can be found in [18, 19, 20], and we will have more to say about coordinated turns in Section 9.2.

5.2 TRIM CONDITIONS

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 (3.14)-(3.17), we get the

following equations of motion:

$$\dot{p}_n = (\cos \theta \cos \psi)u + (\sin \phi \sin \theta \cos \psi - \cos \phi \sin \psi)v + (\cos \phi \sin \theta \cos \psi + \sin \phi \sin \psi)w \quad (5.5)$$

$$\dot{p}_e = (\cos \theta \sin \psi)u + (\sin \phi \sin \theta \sin \psi + \cos \phi \cos \psi)v + (\cos \phi \sin \theta \sin \psi - \sin \phi \cos \psi)w \quad (5.6)$$

$$\dot{h} = u \sin \theta - v \sin \phi \cos \theta - w \cos \phi \cos \theta \quad (5.7)$$

$$\dot{u} = rv - qw - g \sin \theta + \frac{\rho V_a^2 S}{2m} \left[C_X(\alpha) + C_{X_q}(\alpha) \frac{cq}{2V_a} + C_{X_{\delta_e}}(\alpha) \delta_e \right] + \frac{T_p(\delta_t, V_a)}{m} \quad (5.8)$$

$$\dot{v} = pw - ru + g \cos \theta \sin \phi + \frac{\rho V_a^2 S}{2m} \left[C_{Y_0} + C_{Y_\beta} \beta + C_{Y_p} \frac{bp}{2V_a} + C_{Y_r} \frac{br}{2V_a} + C_{Y_{\delta_a}} \delta_a + C_{Y_{\delta_r}} \delta_r \right] \quad (5.9)$$

$$\dot{w} = qu - pv + g \cos \theta \cos \phi + \frac{\rho V_a^2 S}{2m} \left[C_Z(\alpha) + C_{Z_q}(\alpha) \frac{cq}{2V_a} + C_{Z_{\delta_e}}(\alpha) \delta_e \right] \quad (5.10)$$

$$\dot{\phi} = p + q \sin \phi \tan \theta + r \cos \phi \tan \theta \quad (5.11)$$

$$\dot{\theta} = q \cos \phi - r \sin \phi \quad (5.12)$$

$$\dot{\psi} = q \sin \phi \sec \theta + r \cos \phi \sec \theta \quad (5.13)$$

$$\dot{p} = \Gamma_1 pq - \Gamma_2 qr + \frac{1}{2} \rho V_a^2 S b \left[C_{p_0} + C_{p_\beta} \beta + C_{p_p} \frac{bp}{2V_a} + C_{p_r} \frac{br}{2V_a} + C_{p_{\delta_a}} \delta_a + C_{p_{\delta_r}} \delta_r \right] \quad (5.14)$$

$$\dot{q} = \Gamma_5 pr - \Gamma_6 (p^2 - r^2) + \frac{\rho V_a^2 Sc}{2J_y} \left[C_{m_0} + C_{m_\alpha} \alpha + C_{m_q} \frac{cq}{2V_a} + C_{m_{\delta_e}} \delta_e \right] \quad (5.15)$$

$$\dot{r} = \Gamma_7 pq - \Gamma_1 qr + \frac{1}{2} \rho V_a^2 S b \left[C_{r_0} + C_{r_\beta} \beta + C_{r_p} \frac{bp}{2V_a} + C_{r_r} \frac{br}{2V_a} + C_{r_{\delta_a}} \delta_a + C_{r_{\delta_r}} \delta_r \right], \quad (5.16)$$

where $h = -p_d$ is the altitude and

$$\begin{aligned} C_{p_0} &= \Gamma_3 C_{l_0} + \Gamma_4 C_{n_0} \\ C_{p_\beta} &= \Gamma_3 C_{l_\beta} + \Gamma_4 C_{n_\beta} \\ C_{p_p} &= \Gamma_3 C_{l_p} + \Gamma_4 C_{n_p} \\ C_{p_r} &= \Gamma_3 C_{l_r} + \Gamma_4 C_{n_r} \\ C_{p_{\delta_a}} &= \Gamma_3 C_{l_{\delta_a}} + \Gamma_4 C_{n_{\delta_a}} \\ C_{p_{\delta_r}} &= \Gamma_3 C_{l_{\delta_r}} + \Gamma_4 C_{n_{\delta_r}} \\ C_{r_0} &= \Gamma_4 C_{l_0} + \Gamma_8 C_{n_0} \\ C_{r_\beta} &= \Gamma_4 C_{l_\beta} + \Gamma_8 C_{n_\beta} \\ C_{r_p} &= \Gamma_4 C_{l_p} + \Gamma_8 C_{n_p} \\ C_{r_r} &= \Gamma_4 C_{l_r} + \Gamma_8 C_{n_r} \\ C_{r_{\delta_a}} &= \Gamma_4 C_{l_{\delta_a}} + \Gamma_8 C_{n_{\delta_a}} \\ C_{r_{\delta_r}} &= \Gamma_4 C_{l_{\delta_r}} + \Gamma_8 C_{n_{\delta_r}}. \end{aligned}$$

The inertia parameters specified by $\Gamma_1, \Gamma_2, \dots, \Gamma_8$ are defined in Equation (3.13). As shown in Chapter 4, the aerodynamic force coefficients in the X and Z directions are nonlinear functions of the angle of attack. For completeness, we restate them here as

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

If we incorporate the effects of stall into the lift coefficient, we can model it as

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

where

$$\sigma(\alpha) = \frac{1 + e^{-M(\alpha - \alpha_0)} + e^{M(\alpha + \alpha_0)}}{(1 + e^{-M(\alpha - \alpha_0)}) (1 + e^{M(\alpha + \alpha_0)})},$$

and M and α_0 are positive constants.

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.

If we are interested in modeling MAV flight under low-angle-of-attack conditions, simpler, linear models for the lift and drag coefficients can be used, such as

$$\begin{aligned} C_L(\alpha) &= C_{L_0} + C_{L_\alpha}\alpha \\ C_D(\alpha) &= C_{D_0} + C_{D_\alpha}\alpha. \end{aligned}$$

The equations provided in this section completely describe the dynamic behavior of a MAV in response to inputs from the throttle and the aerodynamic control surfaces (ailerons, elevator, and rudder). These equations are the basis for much of what we do in the remainder of the book and are the core of the MAV simulation environment developed as part of the project exercises at the end of each chapter.

An alternative form of these equations, utilizing quaternions to represent the MAV attitude, is given in Appendix B. The quaternion-based equations are free of the gimbal-lock singularity and are more computationally efficient than the Euler-angle-based equations of motion. For this reason, the quaternion form of the equations of motion are often as the basis for high-fidelity simulations. The quaternion representation of attitude is difficult to interpret physically. For this reason, the Euler-angle representation of attitude is preferred for the reduced-order, linear models that will be developed in this chapter. Furthermore, the gimbal-lock singularity is far removed from the flight conditions that will be considered subsequently, and thus will not cause issues with the models to be developed.

5.3 TRANSFER FUNCTION MODELS

5.3.1 Lateral Transfer Functions

5.3.2 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 (2.7), we get

$$\begin{aligned}\dot{V}_a &= \dot{u} \cos \alpha \cos \beta + \dot{v} \sin \beta + \dot{w} \sin \alpha \cos \beta \\ &= \dot{u} \cos \alpha + \dot{w} \sin \alpha + d_{V_1},\end{aligned}\tag{5.17}$$

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 $d_{V_1} = 0$. Substituting Equations (5.8) and (5.10) in Equation (5.17), we obtain

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

where $T_p(\delta_t, V_a)$ is the thrust produced by the motor. Using Equations (2.7) and the linear approximation $C_D(\alpha) \approx C_{D_0} + C_{D_\alpha} \alpha$, and simplifying, we

get

$$\begin{aligned}
\dot{V}_a &= rV_a \cos \alpha \sin \beta - pV_a \sin \alpha \sin \beta \\
&\quad - g \cos \alpha \sin \theta + g \sin \alpha \cos \theta \cos \phi \\
&\quad + \frac{\rho V_a^2 S}{2m} \left[-C_{D_0} - C_{D_\alpha} \alpha - C_{D_q} \frac{cq}{2V_a} - C_{D_{\delta_e}} \delta_e \right] \\
&\quad + \frac{1}{m} T_p(\delta_t, V_a) \cos \alpha + d_{V_1} \\
&= (rV_a \cos \alpha - pV_a \sin \alpha) \sin \beta \\
&\quad - g \sin(\theta - \alpha) - g \sin \alpha \cos \theta (1 - \cos \phi) \\
&\quad + \frac{\rho V_a^2 S}{2m} \left[-C_{D_0} - C_{D_\alpha} \alpha - C_{D_q} \frac{cq}{2V_a} - C_{D_{\delta_e}} \delta_e \right] \\
&\quad + \frac{1}{m} T_p(\delta_t, V_a) \cos \alpha + d_{V_1} \\
&= -g \sin \gamma + \frac{\rho V_a^2 S}{2m} \left[-C_{D_0} - C_{D_\alpha} \alpha - C_{D_q} \frac{cq}{2V_a} - C_{D_{\delta_e}} \delta_e \right] \\
&\quad + \frac{1}{m} T_p(\delta_t, V_a) + d_{V_2}, \tag{5.18}
\end{aligned}$$

where

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

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 (5.18) 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 5.4.1, we can linearize Equation (5.18) 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 (5.18) can then be linearized around the

wings-level, constant-altitude ($\gamma^* = 0$) trim condition to give

$$\dot{\bar{V}}_a = -g \cos(\theta^* - \alpha^*) \bar{\theta} \quad (5.19)$$

$$\begin{aligned} &+ \left\{ \frac{\rho V_a^* S}{m} [-C_{D_0} - C_{D_\alpha} \alpha^* - C_{D_{\delta_e}} \delta_e^*] + \frac{1}{m} \frac{\partial T_p}{\partial V_a} (\delta_t^*, V_a^*) \right\} \bar{V}_a \\ &+ \frac{1}{m} \frac{\partial T_p}{\partial \delta_t} (\delta_t^*, V_a^*) \bar{\delta}_t + d_V \\ &= -a_{V_1} \bar{V}_a + a_{V_2} \bar{\delta}_t - a_{V_3} \bar{\theta} + d_V, \end{aligned} \quad (5.20)$$

where

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

and d_V includes d_{V_2} as well as the linearization error. In the Laplace domain we have

$$\bar{V}_a(s) = \frac{1}{s + a_{V_1}} (a_{V_2} \bar{\delta}_t(s) - a_{V_3} \bar{\theta}(s) + d_V(s)). \quad (5.21)$$

5.4 LINEAR STATE-SPACE MODELS

5.4.1 Linearization

5.4.2 Lateral State-space Equations

5.4.3 Longitudinal State-space Equations

5.4.4 Reduced-order Modes

5.5 CHAPTER SUMMARY

NOTES AND REFERENCES

5.6 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_{10n} 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_{10n} adequately predict the short period and phugoid modes.

- 5.5 Compute eigenvalues of A_{10t} 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_{10t} adequately predict the roll, spiral-divergence, and dutch-roll modes.

Chapter Six

Autopilot Design

In general terms, an autopilot is a system used to guide an aircraft without the assistance of a pilot. For manned aircraft, the autopilot can be as simple as a single-axis wing-leveling autopilot, or as complicated as a full flight control system that controls position (altitude, latitude, longitude) and attitude (roll, pitch, yaw) during the various phases of flight (e.g., take-off, ascent, level flight, descent, approach, landing). For MAVs, the autopilot is in complete control of the aircraft during all phases of flight. While some control functions may reside in the ground control station, the autopilot portion of the MAV control system resides on board the MAV.

This chapter presents an autopilot design suitable for the sensors and computational resources available on board MAVs. We will utilize a method called successive loop closure to design lateral and longitudinal autopilots. The successive loop closure approach is discussed generally in Section 6.1. Because the lifting surfaces of aircraft have limited range, we discuss actuator saturation and the limit it imposes on performance in Section 6.1.1. Lateral and longitudinal autopilot designs are presented in Sections 6.1.2 and 6.1.3. The chapter concludes with a discussion of discrete-time implementation of proportion-integral-derivative (PID) feedback control laws in Section 6.1.4.

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 than 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.

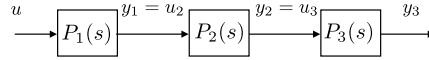


Figure 6.1: Open-loop transfer function modeled as a cascade of three transfer functions.

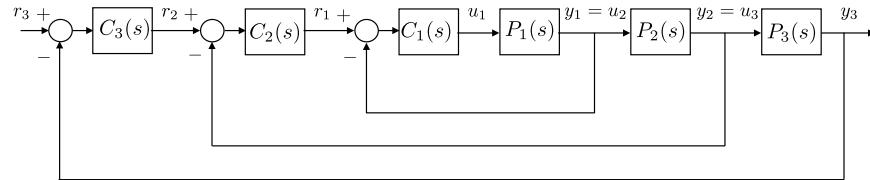


Figure 6.2: Three-stage successive loop closure design.

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 ω_{BW1} . The key assumption 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 $\omega_{BW2} < \frac{1}{W}\omega_{BW1}$ thus ensuring that the unity gain assumption on the inner loop is not violated over the range of frequencies that the middle loop operates.

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

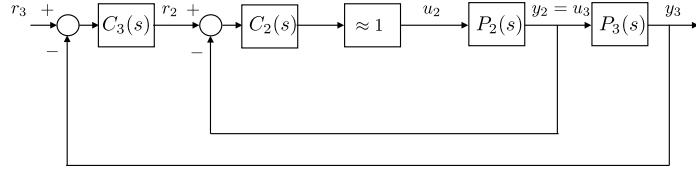


Figure 6.3: Successive loop closure design with inner loop modeled as a unity gain.

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: $\omega_{BW3} < \frac{1}{W_2} \omega_{BW2}$. 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.

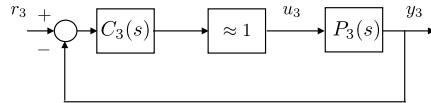


Figure 6.4: Successive-loop-closure design with two inner loops modeled as a unity gain.

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 5.3 and will be used to design the autopilots in this chapter.

6.1.1 Saturation Constraints and Performance

RWB: Remove this section.

6.1.2 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 $k_{d\phi}$ provides roll rate damping for the innermost loop. The roll attitude is regulated with the proportional gain $k_{p\phi}$. The course angle is regulated with the proportional and integral gains k_{p_x} and k_{i_x} . 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.

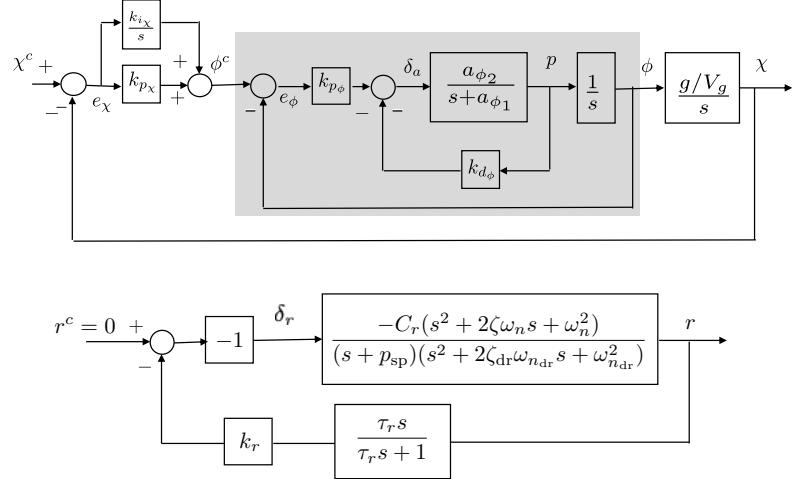


Figure 6.5: Autopilot for lateral control using successive loop closure.

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

6.1.2.1 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.

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

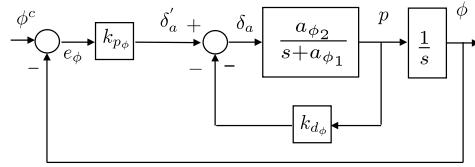


Figure 6.6: Roll attitude hold control loops.

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

$$H_{\phi/\phi^c}^d(s) = \frac{\omega_{n_\phi}^2}{s^2 + 2\zeta_\phi \omega_{n_\phi} s + \omega_{n_\phi}^2},$$

then equating denominator polynomial coefficients, we get

$$\omega_{n_\phi}^2 = k_{p_\phi} a_{\phi_2} \quad (6.1)$$

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

Accordingly, the proportional and derivative gains are given by

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

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

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

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

MODIFIED MATERIAL:

6.1.2.2 Course Hold using Commanded Roll

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.

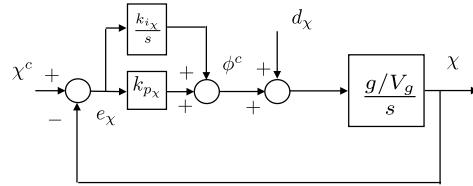


Figure 6.7: Course hold outer feedback loop.

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 $\chi \rightarrow \chi^c$. 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.

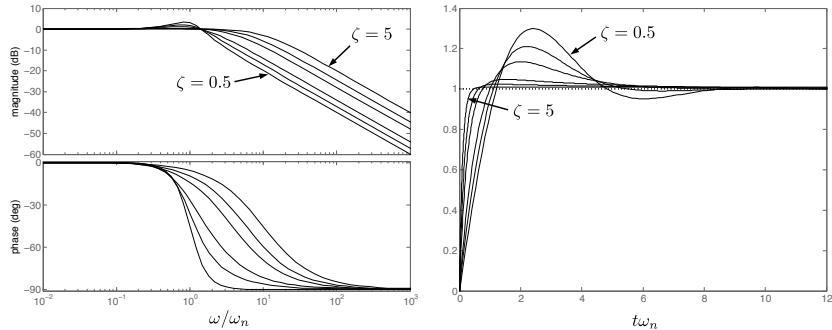


Figure 6.8: Frequency and step response for a second-order system with a transfer function zero for $\zeta = 0.5, 0.7, 1, 2, 3, 5$.

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

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

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

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

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

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

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

NEW MATERIAL:

6.1.2.3 Yaw Damper using the Rudder

Up to this point, we have talked about the lateral-directional control design and in doing so we have confined our attention to the roll-attitude-hold and course-attitude hold loops. In our Chapter 5 discussion, we also looked at the open-loop dynamics of the dutch-roll, spiral, and roll modes. An interesting aspect of the lateral-directional control design is that some aircraft can have their lightly damped dutch-roll mode excited by aileron deflections in the course of banking the aircraft. At best, this degrades the turning performance of the aircraft and at worst it can lead to loss of control.

This unwanted excitation of the dutch-roll mode and its dynamic coupling with the turning behavior of the aircraft can be mitigated through the use of yaw-damping control. The yaw damper utilizes feedback of the yaw rate to control the aircraft rudder in a way that damps the yawing motion of the aircraft induced by the dutch-roll mode.

The transfer function from the rudder angle input to the yaw rate output taking into consideration all of the lateral-directional modes (dutch-roll, spiral, and roll) can be calculated from equation (5.43) as

$$\frac{r(s)}{\delta_r(s)} = \frac{-C_r(s + z_{\text{roll}})(s^2 + 2\zeta\omega_n s + \omega_n^2)}{(s + p_{\text{roll}})(s + p_{\text{sp}})(s^2 + 2\zeta_{\text{dr}}\omega_{n_{\text{dr}}} s + \omega_{n_{\text{dr}}}^2)},$$

where p_r is the pole associated with the roll mode, p_s is the mode associated with the spiral mode, and $(\zeta_{dr}, \omega_{n_{dr}})$ are the damping ratio and natural frequency of the lightly-damped poles associated with the dutch-roll mode. In the numerator is a zero z_{roll} that is in close proximity to the roll pole and a complex pair of zeros that typically are lower in frequency and more highly damped than the dutch-roll poles. Because the roll pole and nearby zero are significantly faster than the other more dominant poles in the system, they effectively cancel one another and their affect on the dynamics from rudder deflection to yaw rate are minimal. Taking this into consideration, we can express a reduced-order form of the rudder to yaw rate transfer function as

$$\frac{r(s)}{\delta_r(s)} = \frac{-C_r(s^2 + 2\zeta\omega_n s + \omega_n^2)}{(s + p_{sp})(s^2 + 2\zeta_{dr}\omega_{n_{dr}}s + \omega_{n_{dr}}^2)}. \quad (6.7)$$

Because the dutch-roll mode can be excited by aileron inputs in a bank-to-turn maneuver, we will use the rudder to dampen the dutch-roll mode using yaw-rate feedback. An inherent weakness of yaw-rate feedback alone is that it will tend to drive the yaw rate to zero always, which is undesirable during a turning maneuver. This turn-limiting behavior can be mitigated using a high-pass washout filter. The feedback structure of the yaw damper with a washout filter is shown in figure 6.9. The washout filter limits the rate feedback's influence at low-frequencies associated with turning the aircraft, but allows the yaw rate feedback to actively damp higher frequency oscillations such as those introduced by the dutch-roll mode. The break frequency of the washout filter $1/\tau_r$ should be chosen to fall well below the dutch-roll natural frequency $\omega_{n_{dr}}$ to enable effective damping of the dutch-roll mode.

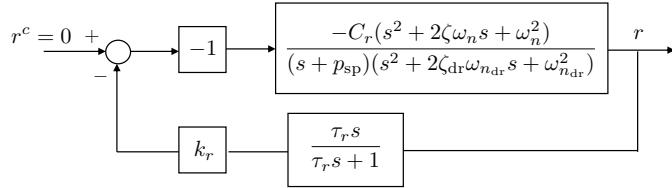


Figure 6.9: Yaw damper with washout filter.

The selection of the yaw-damper gain is straightforward using root-locus techniques. Figure 6.10 shows the root-locus corresponding to the open-loop transfer function of equation (6.7) and the feedback structure of figure 6.9 with proportional gain k_r and yaw damper time constant τ . The open-loop system is based on the lateral directional dynamic model of equation (5.43) and the physical parameters of the Aerosonde aircraft given in Appendix E. The open-loop poles depicted by \times 's correspond to the spiral-mode (near the origin), the dutch-roll mode (lightly-damped complex pair),

and the washout filter ($s = -1 \text{ rad/s}$). The closed-loop poles depicted by triangles show the effectiveness of the yaw damper in damping the dutch-roll mode.

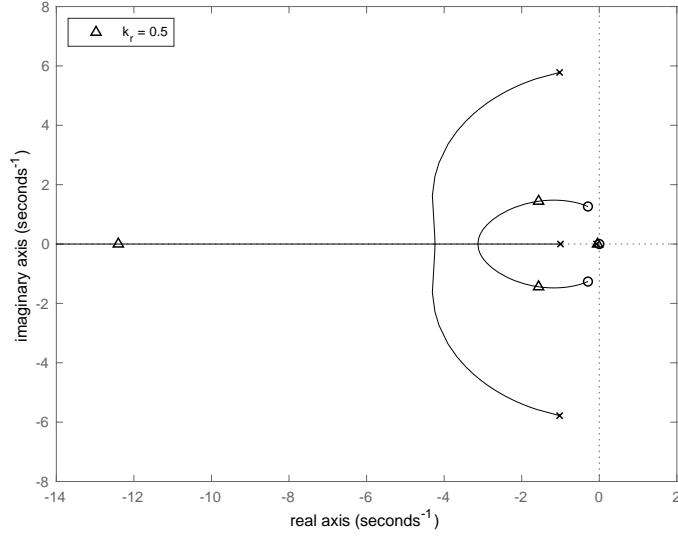


Figure 6.10: Yaw damper root locus plot.

With the yaw damper implemented, the design and implementation of the roll and course loops of the lateral-directional control can go forward as outlined in sections 6.3.1 and 6.3.2. Figure 6.11 shows the lateral-directional performance of the 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 without and with 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 $t = 1 \text{ s}$ 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.

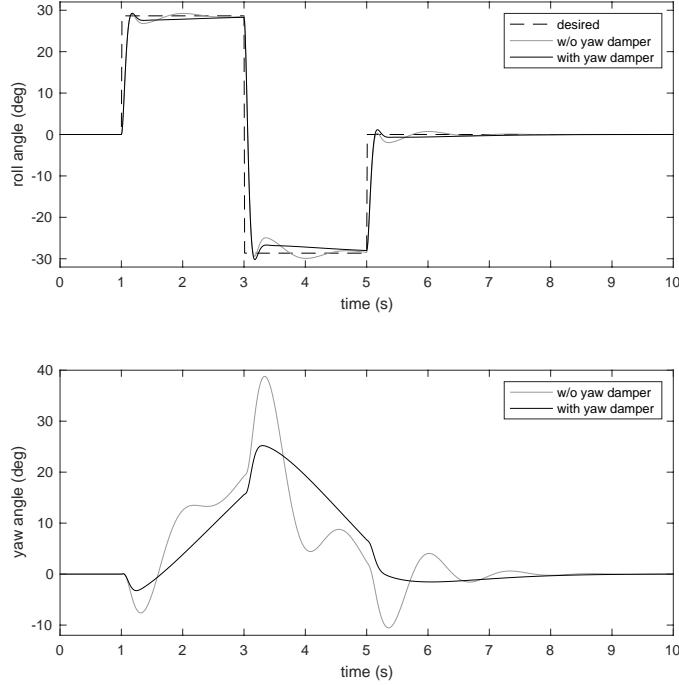


Figure 6.11: Turning performance with and without yaw damper.

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.

RWB: Add section on yaw damper implementation. Show python code.

6.1.3 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 $k_{d\theta}$ provides pitch rate damping for the innermost loop. The pitch attitude is regulated with the proportional gain $k_{p\theta}$. The altitude is regulated with the proportional and integral gains k_{ph} and k_{ih} . The airspeed 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, $k_{d\theta}$ and $k_{p\theta}$ 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.

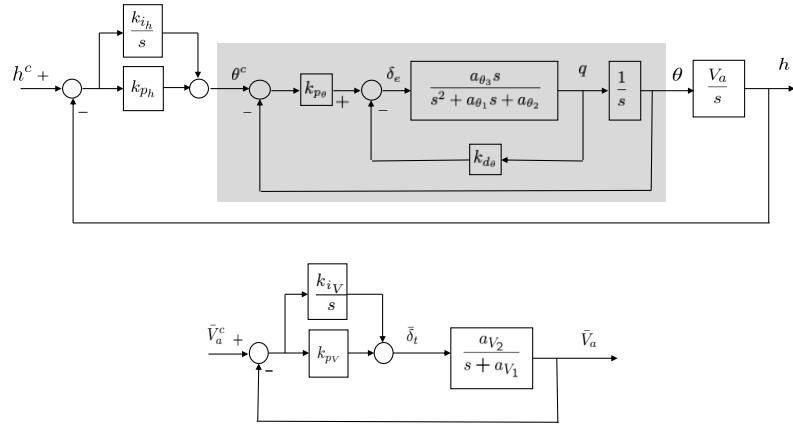


Figure 6.12: Autopilot for longitudinal control using successive loop closure.

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

6.1.3.1 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

$$H_{\theta/\theta^c}(s) = \frac{k_{p\theta}a_{\theta_3}}{s^2 + (a_{\theta_1} + k_{d\theta}a_{\theta_3})s + (a_{\theta_2} + k_{p\theta}a_{\theta_3})}. \quad (6.8)$$

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

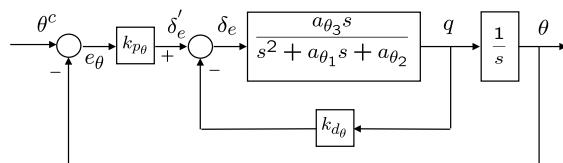


Figure 6.13: Pitch attitude hold feedback loops.

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

$$H_{\theta/\theta^c}^d = \frac{K_{\theta_{DC}} \omega_{n_\theta}^2}{s^2 + 2\zeta_\theta \omega_{n_\theta} s + \omega_{n_\theta}^2},$$

then, equating denominator coefficients, we get

$$\omega_{n_\theta}^2 = a_{\theta_2} + k_{p_\theta} a_{\theta_3} \quad (6.9)$$

$$2\zeta_\theta \omega_{n_\theta} = a_{\theta_1} + k_{d_\theta} a_{\theta_3} \quad (6.10)$$

$$K_{\theta_{DC}} \omega_{n_\theta}^2 = k_{p_\theta} a_{\theta_3}. \quad (6.11)$$

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

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

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

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

6.1.3.2 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 $\theta \approx K_{\theta_{DC}} \theta^c$, the altitude hold loop using the commanded pitch can be approximated by the block diagram shown in Figure 6.14.

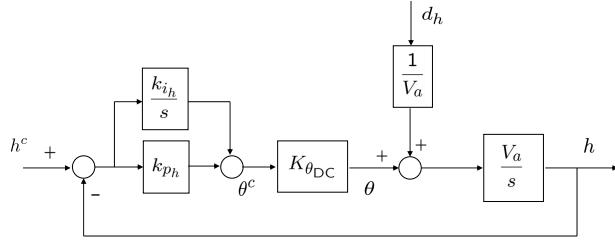


Figure 6.14: The altitude hold loop using the commanded pitch angle.

In the Laplace domain, we have

$$\begin{aligned} h(s) = & \left(\frac{K_{\theta_{DC}} V_a k_{ph} \left(s + \frac{k_{ih}}{k_{ph}} \right)}{s^2 + K_{\theta_{DC}} V_a k_{ph} s + K_{\theta_{DC}} V_a k_{ih}} \right) h^d(s) \\ & + \left(\frac{s}{s^2 + K_{\theta_{DC}} V_a k_{ph} s + K_{\theta_{DC}} V_a k_{ih}} \right) d_h(s), \end{aligned}$$

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_{ph} and k_{ih} 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

$$H_{h/h^c}^d = \frac{\omega_{n_h}^2}{s^2 + 2\zeta_h \omega_{n_h} s + \omega_{n_h}^2},$$

then, equating denominator coefficients, we get

$$\begin{aligned} \omega_{n_h}^2 &= K_{\theta_{DC}} V_a k_{ih} \\ 2\zeta_h \omega_{n_h} &= K_{\theta_{DC}} V_a k_{ph}. \end{aligned}$$

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

$$k_{i_h} = \frac{\omega_{n_h}^2}{K_{\theta_{DC}} V_a} \quad (6.12)$$

$$k_{p_h} = \frac{2\zeta_h \omega_{n_h}}{K_{\theta_{DC}} V_a}. \quad (6.13)$$

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

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

6.1.3.3 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 control, then

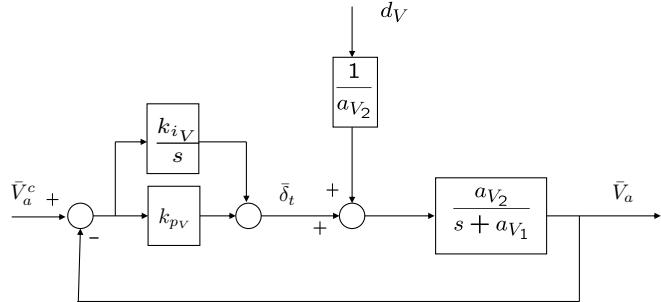


Figure 6.15: Airspeed hold using throttle.

$$\bar{V}_a(s) = \left(\frac{aV_2 k_{p_V}}{s + (aV_1 + aV_2 k_{p_V})} \right) \bar{V}_a^c(s) + \left(\frac{1}{s + (aV_1 + aV_2 k_{p_V})} \right) d_V(s).$$

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

$$\begin{aligned} \bar{V}_a = & \left(\frac{aV_2(k_{p_V}s + k_{i_V})}{s^2 + (aV_1 + aV_2 k_{p_V})s + aV_2 k_{i_V}} \right) \bar{V}_a^c \\ & + \left(\frac{1}{s^2 + (aV_1 + aV_2 k_{p_V})s + aV_2 k_{i_V}} \right) d_V. \end{aligned}$$

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

$$k_{i_V} = \frac{\omega_{n_V}^2}{a_{V_2}} \quad (6.14)$$

$$k_{p_V} = \frac{2\zeta_V \omega_{n_V} - a_{V_1}}{a_{V_2}}. \quad (6.15)$$

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

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

6.1.4 Digital Implementation of PID Loops

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 \
            + (self.Ts / 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 \
            + (self.Ts / 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

```

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 simpler 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 $K = \frac{1}{2}m\|\mathbf{v}\|^2$. 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

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

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

$$U_{\text{error}} = mg(h^c - h). \quad (6.16)$$

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 (6.16) 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

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

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

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

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

$$\begin{aligned} \dot{x} &= Ax + Bu \\ z &= Hx \end{aligned}$$

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

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

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

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

The LQR controller is designed using

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

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

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

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

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

The LQR controller is designed using

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

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 to 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

Critical to the creation and realization of small unmanned air vehicles has been the development of small, lightweight solid-state sensors. Based on microelectromechanical systems (MEMS) technology, small but accurate sensors such as accelerometers, angular rate sensors, and pressure sensors have enabled the development of increasingly smaller and more capable autonomous aircraft. Coupled with the development of small global positioning systems (GPS), computationally capable microcontrollers, and more powerful batteries, the capabilities of MAVs have gone from being purely radio controlled (RC) by pilots on the ground to highly autonomous systems in less than 20 years. The objective of this chapter is to describe the onboard sensors typically used on MAVs and to quantify what they measure. We will focus on sensors used for guidance, navigation, and control of the aircraft. Payload sensors, such as cameras, and their use will be described in Chapter 13.

The following sensors are often found on MAVs:

- Accelerometers
- Rate gyros
- Pressure sensors
- Magnetometers
- GPS

The following sections will discuss each of these sensors, describe their sensing characteristics, and propose models that describe their behavior for analysis and simulation purposes.

7.1 ACCELEROMETERS

Acceleration transducers (accelerometers) typically employ a proof mass held in place by a compliant suspension as shown in Figure 7.1. When the case of the accelerometer experiences an acceleration, the proof mass moves

relative to the case through a distance proportional to the acceleration. The acceleration experienced by the proof mass is converted to a displacement by the springs in the suspension. A simple force balance analysis of the proof mass yields the relationship

$$m\ddot{x} + kx = ky(t),$$

where x is the inertial position of the proof mass and $y(t)$ is the inertial position of the housing – the acceleration of which we want to sense. Given that the deflection of the suspension is $\delta = y(t) - x$, this relation can be expressed as

$$\ddot{x} = \frac{k}{m}\delta.$$

Thus, the acceleration of the proof mass is proportional to the deflection of the suspension. At frequencies below the resonant frequency, the acceleration of the proof mass is the same as the acceleration of the housing. This can be seen by examining the transfer function from the housing position input to the proof mass position output

$$\frac{X(s)}{Y(s)} = \frac{1}{\frac{m}{k}s^2 + 1},$$

or equivalently, the transfer function from the housing acceleration input to the proof mass acceleration output

$$\frac{A_X(s)}{A_Y(s)} = \frac{1}{\frac{m}{k}s^2 + 1}.$$

At frequencies corresponding to $\omega < \sqrt{k/m}$, the transfer function $A_X(s)/A_Y(s) \approx 1$ and the displacement of the proof mass is an accurate indicator of the acceleration of the body to which the accelerometer is attached.

The accelerometer in Figure 7.1 is shown with a capacitive transducer to convert the proof mass displacement into a voltage output as is common in many MEMS devices. Other approaches to convert the displacement to a usable signal include piezoelectric, reductive, and strain-based designs. As with other analog devices, accelerometer measurements are subject to signal bias and random uncertainty. The output of an accelerometer can be modeled as

$$\Upsilon_{\text{accel}} = k_{\text{accel}}A + \beta_{\text{accel}} + \eta'_{\text{accel}},$$

where Υ_{accel} is in volts, k_{accel} is a gain, A is the acceleration in meters per second squared, β_{accel} is a bias term, and η'_{accel} is zero-mean Gaussian noise.

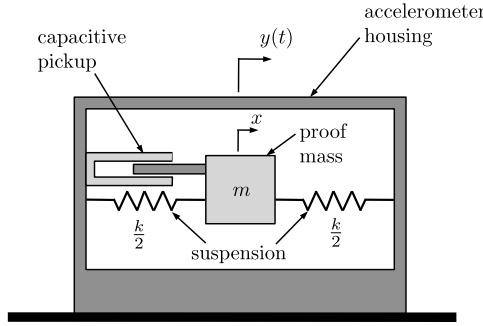


Figure 7.1: Conceptual depiction of MEMS accelerometer.

The gain k_{accel} may be found on the data sheet of the sensor. Due to variations in manufacturing, however, it is imprecisely known. A one-time lab calibration is usually done to accurately determine the calibration constant, or gain, of the sensor. The bias term β_{accel} is dependent on temperature and should be calibrated prior to each flight.

In aircraft applications, three accelerometers are commonly used. The accelerometers are mounted near the center of mass, with the sensitive axis of one accelerometer aligned with each of the body axes. Accelerometers measure the specific force in the body frame of the vehicle. Another interpretation is that they measure the difference between the acceleration of the aircraft and the gravitational acceleration. To understand this phenomena, imagine that the device shown in Figure 7.1 were to be turned ninety degrees and set on a table. The forces acting on the casing will be gravity pulling down, and an equal and opposite normal force pushing up to keep the casing on the table. Therefore, the total acceleration on the casing will be zero. However, since the normal force of the table does not act on the proof mass, it will deflect under the force of gravity and the sensor will measure an acceleration equal to one g . Therefore the measured acceleration is the total acceleration of the casing minus gravity. Mathematically we have

$$\begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} = \frac{d\mathbf{v}}{dt_b} + \boldsymbol{\omega}_{b/i} \times \mathbf{v} - \mathcal{R}_v^b \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix},$$

which can be expressed in component form as

$$\begin{aligned} a_x &= \dot{u} + qw - rv + g \sin \theta \\ a_y &= \dot{v} + ru - pw - g \cos \theta \sin \phi \\ a_z &= \dot{w} + pw - qu - g \cos \theta \cos \phi. \end{aligned}$$

It can be seen that each accelerometer measures elements of linear acceleration, Coriolis acceleration, and gravitational acceleration.

The voltage output of an accelerometer is converted into a number corresponding to the voltage inside the autopilot microcontroller by an analog-to-digital converter at a sample rate T_s . Through calibration, this voltage can be converted to a numerical representation of the acceleration in meters per second squared. Assuming that the biases can be removed through the calibration process, the accelerometer signals inside the autopilot can be modeled as

$$\begin{aligned} y_{\text{accel},x} &= \dot{u} + qw - rv + g \sin \theta + \eta_{\text{accel},x} \\ y_{\text{accel},y} &= \dot{v} + ru - pw - g \cos \theta \sin \phi + \eta_{\text{accel},y} \\ y_{\text{accel},z} &= \dot{w} + pv - qu - g \cos \theta \cos \phi + \eta_{\text{accel},z}, \end{aligned} \quad (7.1)$$

where $\eta_{\text{accel},x}$, $\eta_{\text{accel},y}$, and $\eta_{\text{accel},z}$ are zero mean Gaussian processes with variance $\sigma_{\text{accel},x}^2$, $\sigma_{\text{accel},y}^2$, and $\sigma_{\text{accel},z}^2$ respectively. Because of the calibration, the units of $y_{\text{accel},x}$, $y_{\text{accel},y}$, and $y_{\text{accel},z}$ are in m/s².

Depending on the organization of the simulation software, the terms \dot{u} , \dot{v} , and \dot{w} (state derivatives), may be inconvenient to calculate for inclusion in Equation (7.1). As an alternative, we can substitute from Equations (5.8), (5.9), and (5.10) to obtain

$$\begin{aligned} y_{\text{accel},x} &= \frac{\rho V_a^2 S}{2m} \left[C_X(\alpha) + C_{X_q}(\alpha) \frac{\bar{c}q}{2V_a} + C_{X_{\delta_e}}(\alpha) \delta_e \right] + \frac{\rho S_{\text{prop}} C_{\text{prop}}}{2m} \left[(k_{\text{motor}} \delta_t)^2 - V_a^2 \right] + \eta_{\text{accel},x} \\ y_{\text{accel},y} &= \frac{\rho V_a^2 S}{2m} \left[C_{Y_0} + C_{Y_\beta} \beta + C_{Y_p} \frac{bp}{2V_a} + C_{Y_r} \frac{br}{2V_a} + C_{Y_{\delta_a}} \delta_a + C_{Y_{\delta_r}} \delta_r \right] + \eta_{\text{accel},y} \\ y_{\text{accel},z} &= \frac{\rho V_a^2 S}{2m} \left[C_Z(\alpha) + C_{Z_q}(\alpha) \frac{\bar{c}q}{2V_a} + C_{Z_{\delta_e}}(\alpha) \delta_e \right] + \eta_{\text{accel},z}. \end{aligned} \quad (7.2)$$

However, since the forces are already calculated as part of the dynamics, the best way to organize the simulation files is to use the forces to compute the output of the accelerometers. The resulting equations are

$$\begin{aligned} y_{\text{accel},x} &= \frac{f_x}{m} + g \sin \theta + \eta_{\text{accel},x} \\ y_{\text{accel},y} &= \frac{f_y}{m} - g \cos \theta \sin \phi + \eta_{\text{accel},y} \\ y_{\text{accel},z} &= \frac{f_z}{m} - g \cos \theta \cos \phi + \eta_{\text{accel},z}, \end{aligned} \quad (7.3)$$

where f_x , f_y , and f_z are given in Equation (4.24). With the exception of the noise terms, the terms on the right hand sides of Equation (7.3) represent the

specific force experienced by the aircraft. The acceleration of the aircraft is commonly expressed in units of g , the gravitational constant. To express the acceleration measurements in g 's, Equation (7.3) can be divided by g . The choice of units is up to the preference of the engineer, however, maintaining consistent units reduces the potential for mistakes in implementation.

7.2 RATE GYROS

MEMS rate gyros typically operate based on the principle of the Coriolis acceleration. In the early 19th century, French scientist G.G. de Coriolis discovered that a point translating on a rotating rigid body experiences an acceleration, now called Coriolis acceleration, that is proportional to the velocity of the point and the rate of rotation of the body

$$\mathbf{a}_C = 2\boldsymbol{\Omega} \times \mathbf{v}, \quad (7.4)$$

where $\boldsymbol{\Omega}$ is the angular velocity of the body in an inertial reference frame, and \mathbf{v} is the velocity of the point in the reference frame of the body. In this case, $\boldsymbol{\Omega}$ and \mathbf{v} are both vector quantities and \times represents the vector cross product.

MEMS rate gyros commonly consist of a vibrating proof mass as depicted in Figure 7.2. In this figure, the cantilever and proof mass are actuated at their resonant frequency to cause oscillation in the vertical plane. The cantilever is actuated so that the velocity of the proof mass due to these oscillations is a constant amplitude sinusoid

$$\mathbf{v} = A\omega_n \sin(\omega_n t),$$

where A is the amplitude of the oscillation and ω_n is the natural frequency of the oscillation. If the sensitive axis of the rate gyro is configured to be the longitudinal axis of the undeflected cantilever, then rotation about this axis will result in a Coriolis acceleration in the horizontal plane described by Equation (7.4) and shown in Figure 7.2. Similar to the accelerometer, the Coriolis acceleration of the proof mass results in a lateral deflection of the cantilever. This lateral deflection of the cantilever can be detected in several ways: by capacitive coupling, through a piezoelectrically generated charge, or through a change in piezoresistance of the cantilever. Whatever the transduction method, a voltage proportional to the lateral Coriolis acceleration is produced.

With the sensing axis orthogonal to direction of vibration, the ideal output voltage of the rate gyro is proportional to the amplitude of Coriolis acceler-

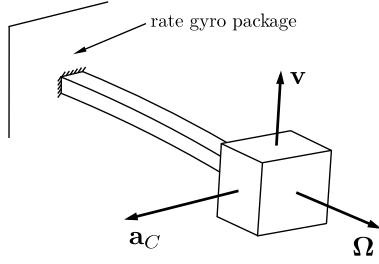


Figure 7.2: Conceptual depiction of proof mass rate gyro. ω is the angular velocity of the sensor package to be measured. v is the actuated vibration velocity of the cantilever. a_C is the Coriolis acceleration that results as the sensor package undergoes an angular velocity.

ation, and is given by

$$\begin{aligned} V_{\text{gyro}} &= k_C |\mathbf{a}_C| \\ &= 2k_C |\boldsymbol{\Omega} \times \mathbf{v}|. \end{aligned}$$

Since $\boldsymbol{\Omega}$, the angular rate of rotation about the sensitive axis of the gyro, and \mathbf{v} are orthogonal

$$|\boldsymbol{\Omega} \times \mathbf{v}| = \Omega |\mathbf{v}|,$$

and

$$\begin{aligned} V_{\text{gyro}} &= 2k_C \Omega |A\omega_n \sin(\omega_n t)| \\ &= 2k_C A\omega_n \Omega \\ &= K_C \Omega, \end{aligned}$$

where K_C is a calibration constant and Ω represents the magnitude and direction (sign) of the angular velocity about the sensitive axis.

The output of a rate gyro can be modeled as

$$\Upsilon_{\text{gyro}} = k_{\text{gyro}} \Omega + \beta_{\text{gyro}} + \eta'_{\text{gyro}},$$

where Υ_{gyro} corresponds to the measured rate of rotation in volts, k_{gyro} is a gain converting the rate in radians per second to Volts, Ω is the angular rate in radians per second, β_{gyro} is a bias term, and η'_{gyro} is zero mean Gaussian noise. An approximate value for the gain k_{gyro} should be given on the spec sheet of the sensor. To ensure accurate measurements, the value of this gain should be determined through experimental calibration. The bias term β_{gyro} is strongly dependent on temperature and should be calibrated prior to each flight. For low-cost MEMS gyros, drift in this bias term can be significant and care must be taken to zero the gyro bias periodically during flight. This

is done by flying a straight and level path ($\Omega = 0$) and resetting the gyro bias so that T_{gyro} averages zero over a period of 100 or so samples.

For simulation purposes, we are interested in modeling the calibrated gyro signals inside the autopilot. The rate gyro signals are converted from analog voltages coming out of the sensor to numerical representations of angular rates (in units of rad/s) inside the autopilot. We assume that the gyros have been calibrated so that in the nominal case 1 rad/s of angular rate experienced by the sensor results in a numerical measurement inside the autopilot of 1 rad/s (i.e., the gain from the physical rate to its numerical representation inside the autopilot is one) and that the biases have been estimated and subtracted from the measurements. It is common to measure the angular rates about each of the body axes using three gyros by aligning the sensitive axis of a gyro along each of the \mathbf{i}^b , \mathbf{j}^b , and \mathbf{k}^b axes of the MAV. These rate gyro measurements of angular body rates p , q , and r can be modeled as

$$\begin{aligned} y_{\text{gyro},x} &= p + \eta_{\text{gyro},x} \\ y_{\text{gyro},y} &= q + \eta_{\text{gyro},y} \\ y_{\text{gyro},z} &= r + \eta_{\text{gyro},z}, \end{aligned} \tag{7.5}$$

where $y_{\text{gyro},x}$, $y_{\text{gyro},y}$, and $y_{\text{gyro},z}$ are angular rate measurements with units of rad/s. The variables $\eta_{\text{gyro},x}$, $\eta_{\text{gyro},y}$, and $\eta_{\text{gyro},z}$ represent zero-mean Gaussian processes with variances $\sigma_{\text{gyro},x}^2$, $\sigma_{\text{gyro},y}^2$, and $\sigma_{\text{gyro},z}^2$, respectively. MEMS gyros are analog devices that are sampled by the autopilot microcontroller. We will assume that the sample rate is given by T_s .

7.3 PRESSURE SENSORS

Pressure, a quantity commonly associated with fluids, is defined as the force per unit area acting on a surface. Pressure acts in a direction normal to the surface of the body to which it is applied. We will use measurements of pressure to provide indications of the altitude of the aircraft and the airspeed of the aircraft. To measure altitude, we will use an absolute pressure sensor. To measure airspeed, we will use a differential pressure sensor.

7.3.1 Altitude Measurement

Measurements of altitude can be inferred from measurements of atmospheric pressure. The basic equation of hydrostatics, given by

$$P_2 - P_1 = \rho g(z_2 - z_1), \tag{7.6}$$

states that for a static fluid, the pressure at a point of interest changes with the depth of the point below the surface of the fluid. This relationship assumes that the density of the fluid is constant between the points of interest. Although the air in the atmosphere is compressible and its density changes significantly over altitudes from sea level to altitudes commonly flown by modern aircraft, the hydrostatic relationship of Equation (7.6) can be useful over small altitude changes where the air density remains essentially constant.

We are typically interested in the altitude or height of the aircraft above a ground station and the corresponding change in pressure between the ground and the altitude of interest. From Equation (7.6), the change in pressure due to a change in altitude is given by

$$\begin{aligned} P - P_{\text{ground}} &= -\rho g(h - h_{\text{ground}}) \\ &= -\rho g h_{\text{AGL}}, \end{aligned} \quad (7.7)$$

where h is the absolute altitude of the aircraft, h_{ground} is the absolute altitude of the ground, $h_{\text{AGL}} = h - h_{\text{ground}}$, and h and h_{ground} are measured with respect to sea level and P is the corresponding absolute pressure measurement. The change in sign between Equations (7.6) and (7.7) comes from the fact that depth, z , is measured positive down while altitude h is measured positive up. A decrease in altitude above the ground results in an increase in measured pressure. In implementation, P_{ground} is the atmospheric pressure measured at ground level prior to take off and ρ is the air density at the flight location.

Equation (7.7) assumes that the air density is constant over the altitude range of interest. In truth, it varies with both weather conditions and altitude. Assuming that weather conditions are invariant over the flight duration, we must consider the effects of changing air density due to changes in pressure and temperature that occur with altitude.

Below altitudes of 11,000 m above sea level, the pressure of the atmosphere can be calculated using the barometric formula [?]. This formula takes into account the change in density and pressure due to decreasing temperature with altitude and is given by

$$P = P_0 \left[\frac{T_0}{T_0 + L_0 h_{\text{ASL}}} \right]^{\frac{gM}{RL_0}}, \quad (7.8)$$

where $P_0 = 101,325 \text{ N/m}^2$ is the standard pressure at sea level, $T_0 = 288.15 \text{ K}$ is the standard temperature at sea level, $L_0 = -0.0065 \text{ K/m}$ is the lapse rate or the rate of temperature decrease in the lower atmosphere, $g = 9.80665 \text{ m/s}^2$ is the gravitational constant, $R = 8.31432 \text{ N-m/(mol-K)}$ is the universal gas

constant for air, and $M = 0.0289644 \text{ kg/mol}$ is the standard molar mass of atmospheric air. The altitude h_{ASL} is referenced to sea level.

The relative significance of the constant-density assumption can be seen by comparing pressures calculated using Equations (7.7) and (7.8) as shown in Figure 7.3. It can be seen that over the full range of altitudes for which the barometric formula is valid (0 to 11,000 m above sea level), the pressure versus altitude relationship is not linear and that the linear approximation of Equation (7.7) is not valid. The plot on the right of Figure 7.3, however, shows that over narrower altitude ranges, such as those common to small unmanned aircraft, a linear approximation can be used with reasonable accuracy. For this particular plot, Equation (7.7) was employed with $h_{\text{ground}} = 0$ and air density calculated at sea level.

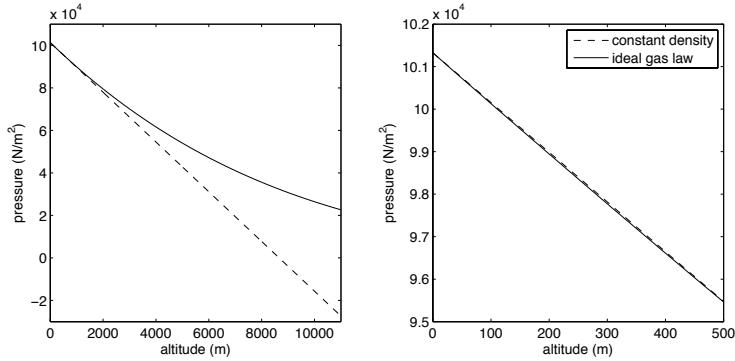


Figure 7.3: Comparison of atmospheric pressure calculations using constant-density and variable-density models.

One key to accurately calculating altitude from pressure using Equation (7.7) is to have an accurate measure of air density at the flight location. This can be determined from the ideal gas formula, with measurements of the local temperature and barometric pressure at flight time according to

$$\rho = \frac{MP}{RT},$$

using values for the universal gas constant and molar mass of air specified above. Notice that in this formula, temperature is expressed in units of Kelvin. The conversion from Fahrenheit to Kelvin is given by

$$T [K] = \frac{5}{9} (T [F] - 32) + 273.15.$$

The atmospheric pressure is expressed in N/m^2 . Typical weather data reports pressure in inches of mercury (Hg). The conversion factor is $1 \text{ N/m}^2 = 3385$ inches of Hg.

In practice, we will utilize the measurement of absolute pressure to give an indication of altitude above ground level of the aircraft. Figure 7.4 shows an example of an absolute pressure sensor in schematic form. The pressure

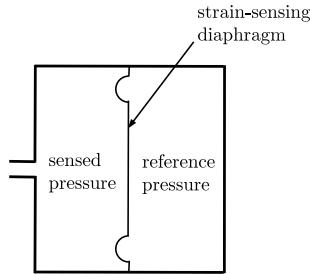


Figure 7.4: Schematic of an absolute pressure sensor.

sensor consists of two volumes separated by a diaphragm. The volume on the right is closed and at a constant reference pressure. The volume at the left is open to the ambient air. Changes in the pressure of the ambient air cause the diaphragm to deflect. These deflections are measured and produce a signal proportional to the sensed pressure.

Following Equation (7.7), the output of the absolute pressure sensor of interest is given by

$$\begin{aligned} y_{\text{abs pres}} &= (P_{\text{ground}} - P) + \beta_{\text{abs pres}} + \eta_{\text{abs pres}} \\ &= \rho g h_{\text{AGL}} + \beta_{\text{abs pres}} + \eta_{\text{abs pres}} \end{aligned} \quad (7.9)$$

where h_{AGL} is the altitude above ground level, $\beta_{\text{abs pres}}$ is a temperature-related bias drift, and $\eta_{\text{abs pres}}$ is zero-mean Gaussian noise with variance $\sigma_{\text{abs pres}}^2$. P_{ground} is the pressure measured at ground level prior to take-off and held in the memory of the autopilot microcontroller. P is the absolute pressure measured by the sensor during flight. The difference between these two measurements is proportional to the altitude of the aircraft above ground level.

7.3.2 Airspeed Sensor

Airspeed can be measured using a pitot-static probe in conjunction with a differential pressure transducer as depicted schematically in Figure 7.5. The pitot-static tube has two ports: one that is exposed to the total pressure and another that is exposed to the static pressure. The total pressure is also known as the stagnation pressure or pitot pressure. It is the pressure at the tip of the probe, which is open to the oncoming flow. The flow is stagnant or stopped at the tip. As a result, pressure is built up so that the pressure at the tip

is higher than that of the surrounding fluid. The static pressure is simply the ambient pressure of the surrounding fluid (or atmosphere). The difference in pressures on each side of the diaphragm in the differential pressure sensor cause the diaphragm to deflect causing a strain in the diaphragm proportional to the pressure difference. This strain is measured, producing a voltage output representing the differential pressure.

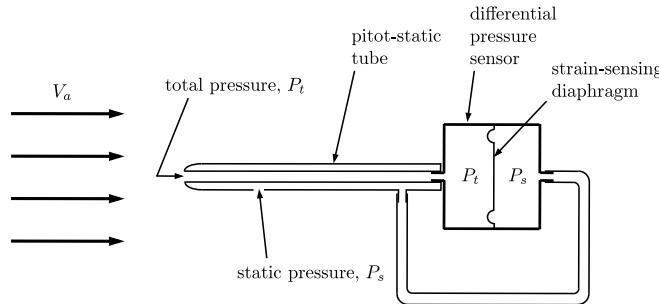


Figure 7.5: Schematic of pitot-static tube and differential pressure sensor. Not to scale.

Bernoulli's equation states that total pressure is the sum of the static pressure and dynamic pressure. In equation form, we can write this as

$$P_t = P_s + \frac{\rho V_a^2}{2},$$

where ρ is the air density and V_a is the airspeed of the MAV. Rearranging, we have

$$\frac{\rho V_a^2}{2} = P_t - P_s,$$

which is the quantity that the differential pressure sensor measures. With proper calibration to convert the sensor output from volts to a number inside the microcontroller representing pressure in units of N/m², we can model the output of the differential pressure sensor as

$$y_{\text{diff pres}} = \frac{\rho V_a^2}{2} + \beta_{\text{diff pres}} + \eta_{\text{diff pres}}, \quad (7.10)$$

where $\beta_{\text{diff pres}}$ is a temperature-related bias drift, $\eta_{\text{diff pres}}$ is zero-mean Gaussian noise with variance $\sigma_{\text{diff pres}}^2$. The absolute and differential pressure sensors are analog devices that are sampled by the onboard processor at the same update rate as the main autopilot control loop.

7.4 DIGITAL COMPASSES

The earth's magnetic field has been used as a navigational aid for centuries. The first magnetic compasses are believed to have originated with the Chinese around the first century A.D. Compasses appeared in Europe around the 11th century A.D. and were used by Christopher Columbus and other world explorers in the late 15th century. The earth's magnetic field continues to provide a means for navigation for a variety of vehicles, including unmanned aircraft.

The magnetic field around the earth behaves similarly to that of a common magnetic dipole with the magnetic field lines running normal to the earth's surface at the poles and parallel to the earth's surface near the equator. Except near the poles, the earth's magnetic field points to magnetic North. A compass measures the direction of the magnetic field locally and provides an indication of heading relative to magnetic North, ψ_m . This is depicted schematically in Figure 7.6. The declination angle δ is the angle between true North and magnetic North.

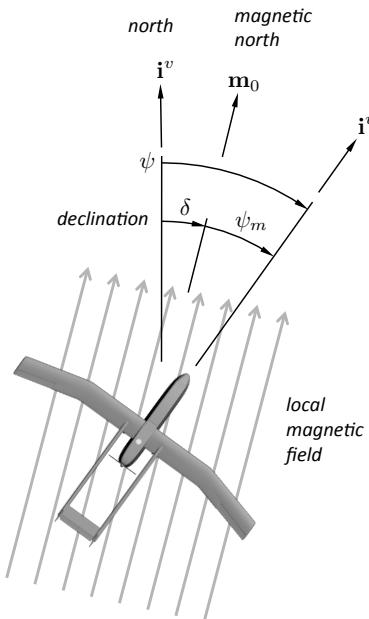


Figure 7.6: Magnetic field and compass measurement.

The earth's magnetic field is three dimensional, with North, East, and Down components that vary with location along the earth's surface. For example, in Provo, Utah, the North component (X) of the magnetic field

is 21,053 nT, the East component (Y) is 4520 nT, and the Down component (Z) is 47,689 nT. The declination angle is 12.12 degrees. Figure 7.7 shows the declination angle over the surface of the earth and illustrates the significant dependence of the magnetic North direction on location. The inclination angle is the angle that the magnetic field makes with the horizontal plane. In Provo, the inclination is 65.7 degrees.

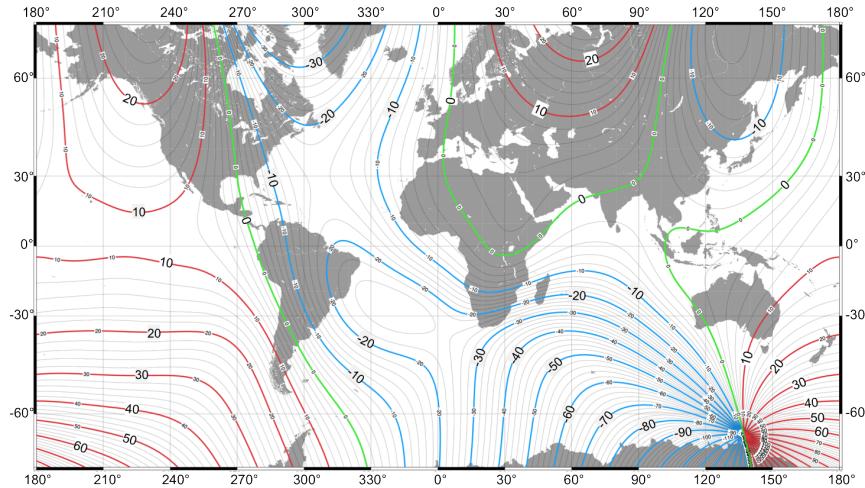


Figure 7.7: Magnetic field declination according to the U.S./U.K. World Magnetic Model. Adapted from [?].

Modern digital compasses use three-axis magnetometers to measure the strength of the magnetic field along three orthogonal axes. In UAV applications, these axes of measurement are usually aligned with the body axes of the aircraft. Although only two sensing axes are required to measure the magnetic heading if the aircraft is in level flight, a third sensing axis is necessary if the aircraft is pitching or rolling out of the horizontal plane.

From Figure 7.6, we can see that the heading angle is the sum of the declination angle and the magnetic heading measurement

$$\psi = \delta + \psi_m. \quad (7.11)$$

The declination angle for a given latitude and longitude can be calculated using models such as the World Magnetic Model (WMM), available from the National Geophysical Data Center (NGDC) [?]. The magnetic heading can be determined from measurements of the magnetic field strength along the body-frame axes. To do so, we project the body-frame measurements of the magnetic field onto the horizontal plane. The angle between the horizontal component of the magnetic field and the i^v axis (the heading) is the

magnetic heading. Mathematically, we can calculate the magnetic heading from the following expressions as

$$\begin{aligned}
 \mathbf{m}_0^{v1} &= \begin{pmatrix} m_{0x}^{v1} \\ m_{0y}^{v1} \\ m_{0z}^{v1} \end{pmatrix} = \mathcal{R}_b^{v1}(\phi, \theta) \mathbf{m}_0^b \\
 &= \mathcal{R}_{v2}^{v1}(\theta) \mathcal{R}_b^{v2}(\phi) \mathbf{m}_0^b \\
 &= \begin{pmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{pmatrix} \mathbf{m}_0^b \\
 \begin{pmatrix} m_{0x}^{v1} \\ m_{0y}^{v1} \\ m_{0z}^{v1} \end{pmatrix} &= \begin{pmatrix} c_\theta & s_\theta s_\phi & s_\theta c_\phi \\ 0 & c_\phi & -s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{pmatrix} \mathbf{m}_0^b,
 \end{aligned} \tag{7.12}$$

and

$$\psi_m = -\text{atan2}(m_{0y}^{v1}, m_{0x}^{v1}). \tag{7.13}$$

In these equations, \mathbf{m}_0^b is a vector containing the body-frame measurements of the magnetic field taken onboard the vehicle. The four-quadrant inverse tangent function $\text{atan2}(y, x)$ returns the arctangent of y/x in the range $[-\pi, \pi]$ using the signs of both arguments to determine the quadrant of the return value. The components m_{0x}^{v1} and m_{0y}^{v1} are the horizontal components of the magnetic field measurement that result when \mathbf{m}_0^b is projected onto the horizontal plane. Notice that in level flight ($\phi = \theta = 0$), $\mathbf{m}_0^{v1} = \mathbf{m}_0^b$.

In practice, the use of magnetometers and digital compasses can be challenging. This is primarily due to the sensitivity of the sensor to electromagnetic interference. Careful placement of the sensor on the aircraft is essential to avoid interference from electric motors, servos, and power wiring. Magnetometers can also be sensitive to interference from power lines and weather systems. Some of the challenges associated with magnetometers have been addressed by manufacturers that package magnetometers, signal conditioning, and a microcontroller into a single-chip solution called a digital compass. These digital compasses vary in their sophistication with full-featured versions incorporating tilt compensation and automatic declination/inclination calculation from latitude and longitude data.

To create a sensor model for simulation purposes, a reasonable approach to modeling a digital compass is to assume that the compass gives a measure of the true heading with a bias error, due to uncertainty in the declination angle, and sensor noise from the magnetometers. Mathematically, this can

be represented as

$$y_{\text{mag}} = \psi + \beta_{\text{mag}} + \eta_{\text{mag}}, \quad (7.14)$$

where η_{mag} is a zero mean Gaussian process with variance σ_{mag}^2 and β_{mag} is a bias error. A digital compass communicates over a serial link with the autopilot at sample rate T_s .

7.5 GLOBAL POSITIONING SYSTEM

The Global Positioning System (GPS) is a satellite-based navigation system that provides 3-D position information for objects on or near the earth's surface. The NAVSTAR GPS system was developed by the U.S. Department of Defense and has been fully operational since 1993. For unmanned aircraft, it would be difficult to overstate the significance of the development and availability of the GPS system. It was, and continues to be, a critical enabling technology for small UAVs. The GPS system and global navigation satellite systems have been described in detail in numerous texts (e.g., [?, ?, ?, ?]). In this section, we will provide a brief overview of GPS position sensing and present a model for GPS position sensing suitable for simulation.

The key component of the GPS system is the constellation of 24 satellites that continuously orbit the earth at an altitude of 20,180 km [?]. The configuration of the satellite orbits is designed so that any point on the earth's surface is observable by at least four satellites at all times. By measuring the times of flight of signals from a minimum of four satellites to a receiver on or near the surface of the earth, the location of the receiver in three dimensions can be determined. The time of flight of the radio wave signal is used to determine the range from each satellite to the receiver. Because synchronization errors exist between the satellite clocks and the receiver clock, the range estimate determined from the time of flight measurement is called *pseudorange* to distinguish it from the true range.

Because of clock synchronization errors between the satellites (whose atomic clocks are almost perfectly synchronized) and the receiver, four independent pseudorange measurements are required to triangulate the position of the receiver as depicted in Figure 7.8. Why are four pseudorange measurements required? We know that with one range measurement from a known location, we can locate a point on a line (1-D). Two range measurements can locate a point on a plane (2-D) and three range measurements can locate a point on a 3-D surface. To resolve position in three dimensions with a receiver clock offset error, at least four measurements are needed. The geometry associated with pseudorange 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 [?].

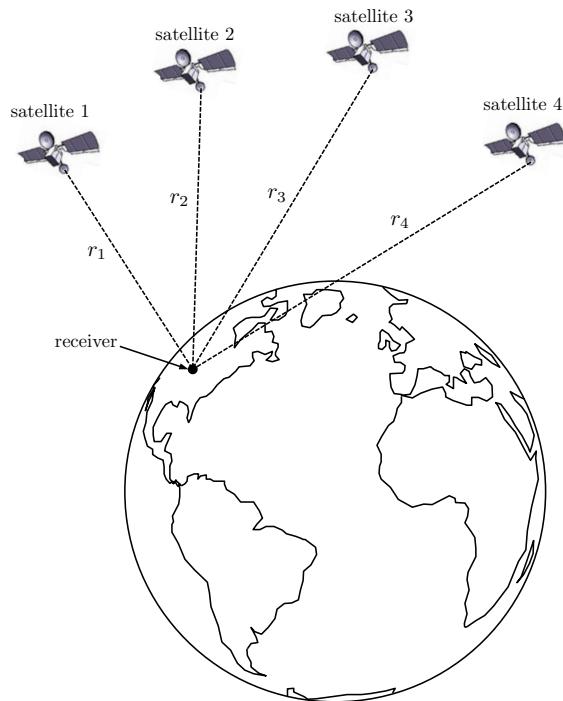


Figure 7.8: Pseudo-range measurements from four satellites used to triangulate the position of a receiver.

7.5.1 GPS Measurement Error

The accuracy of a GPS position measurement is affected by the accuracy of the satellite pseudorange measurements and by the geometry of the satellites from which pseudorange measurements are taken. The effect of satellite geometry is taken into consideration through a factor called dilution of precision (DOP). Pseudorange accuracy is affected by errors in the time of flight measurement for each satellite. Given that the electromagnetic radio signals from the satellites travel at the speed of light, small timing errors can cause significant positioning errors. For example, a timing error of only 10 ns can result in a positioning error of about 3 m. Error sources in the time of flight are discussed briefly in the following paragraphs drawing on information presented in [?, ?].

EPHEMERIS DATA

The satellite ephemeris is a mathematical description of its orbit. Calculation of the receiver location requires the satellite locations to be known. Ephemeris errors in the pseudorange calculation are due to uncertainty in the transmitted location of the satellite. Errors in the range of 1 to 5 m are typical.

SATELLITE CLOCK

GPS satellites use cesium and rubidium atomic clocks, which over the course of one day, drift about 10 ns, introducing an error of about 3.5 m. Given that the clocks are updated every 12 hours, satellite clock errors introduce a positioning error of 1 to 2 m on average.

IONOSPHERE

The ionosphere is the uppermost layer of the earth's atmosphere and is characterized by the presence of free electrons that delay the transmission of GPS signals. Although receivers make corrections for this delay based on information in the GPS message, errors caused by variations in the speed of light through the ionosphere are the largest source of ranging errors in GPS measurements. Errors are typically between 2 and 5 m.

TROPOSPHERE

The troposphere is the lowest layer of the earth's atmosphere, extending from the earth's surface to an altitude of between 7 and 20 km. Most of the mass of the atmosphere is in the troposphere and almost all of the weather activity around the earth occurs in the troposphere. Variations in the temperature, pressure, and humidity in the troposphere affect the speed of light and thus affect time of flight and pseudorange estimates. The uncertainty in the speed of light through the troposphere introduces range errors of about 1 m.

MULTIPATH RECEPTION

Multipath errors are caused when a GPS receiver receives reflected signals that mask the true signal of interest. Multipath is most significant for static receivers located near large reflecting surfaces, such as might be encountered near large buildings or structures. Multipath errors are below 1 m in most circumstances.

Table 7.1: Standard pseudorange error model ($1-\sigma$, in meters) [?].

Error source	Bias	Random	Total
Ephemeris data	2.1	0.0	2.1
Satellite clock	2.0	0.7	2.1
Ionosphere	4.0	0.5	4.0
Troposphere monitoring	0.5	0.5	0.7
Multipath	1.0	1.0	1.4
Receiver measurement	0.5	0.2	0.5
UERE, rms	5.1	1.4	5.3
Filtered UERE, rms	5.1	0.4	5.1

RECEIVER MEASUREMENT

Receiver measurement errors stem from the inherent limits with which the timing of the satellite signal can be resolved. Improvements in signal tracking and processing have resulted in modern receivers that can compute the signal timing with sufficient accuracy to keep ranging errors due to the receiver less than 0.5 m.

Pseudorange errors from the sources described above are treated as statistically uncorrelated and can be added using the root sum of squares. The cumulative effect of each of these error sources on the pseudorange measurement is called the user-equivalent range error (UERE). Parkinson, et al. [?] characterized these errors as a combination of slowly varying biases and random noise. The magnitudes of these errors are tabulated in Table 7.1. Recent publications indicate that measurement accuracies have improved in recent years due to improvements in error modeling and receiver technology with total UERE being estimated as approximately 4.0 m ($1-\sigma$) [?].

The pseudorange error sources described above contribute to the UERE in the range estimates for individual satellites. An additional source of position error in the GPS system comes from the geometric configuration of the satellites used to compute the position of the receiver. This satellite geometry error is expressed in terms of a single factor called the dilution of precision (DOP) [?]. The DOP value describes the increase in positioning error attributed to the positioning of the satellites in the constellation. In general, a GPS position estimate from a group of visible satellites that are positioned close to one another will result in a higher DOP value, while a position estimate from a group of visible satellites that are spread apart will result in a lower DOP value.

There are a variety of DOP terms defined in the literature. The two DOP terms of greatest interest to us are the horizontal DOP (HDOP) and the ver-

tical DOP (VDOP). HDOP describes the influence of the satellite geometry on the GPS position measurement accuracy in the horizontal plane, while VDOP describes the influence of satellite geometry on position measurement accuracy in altitude. Since DOP depends on the number and configuration of visible satellites, it varies continuously with time. In open areas where satellites are readily visible, a nominal HDOP value would be 1.3, while a nominal VDOP value would be 1.8 [?].

The total error in a GPS position measurement takes into account the UERE and DOP. The standard deviation of the rms error in the North-East plane is given by

$$\begin{aligned} E_{n-e,\text{rms}} &= \text{HDOP} \times \text{UERE}_{\text{rms}} \\ &= (1.3)(5.1 \text{ m}) \\ &= 6.6 \text{ m.} \end{aligned} \quad (7.15)$$

Similarly, the standard deviation of the rms altitude error is given by

$$\begin{aligned} E_{h,\text{rms}} &= \text{VDOP} \times \text{UERE}_{\text{rms}} \\ &= (1.8)(5.1 \text{ m}) \\ &= 9.2 \text{ m.} \end{aligned} \quad (7.16)$$

These expressions give an indication of the size of the error that can be anticipated for a single receiver position measurement. As indicated in Table 7.1, these errors consist of statistically independent slowly-varying biases and random noise components. Techniques, such as differential GPS, can be used to reduce the bias error components of GPS position measurements to much smaller values.

7.5.2 Transient Characteristics of GPS Positioning Error

The previous discussion has given us a good sense of the root-mean-square magnitude of the positioning errors involved in GPS measurements. For the purposes of simulation, however, we are not only interested in the size of the error, we are also interested in knowing the dynamic characteristics of the error. Referring to Equation (7.15) and assuming that the horizontal position error is composed of a North position error and East position error that are independent but of similar size, we can calculate the North and East error magnitudes to be about 4.7 m in size. The North, East, and altitude errors are comprised of a slowly changing bias along with random noise. For example, based on a VDOP of 1.8 and the UERE values of Table 7.1, we can approximately model the altitude position error from GPS as having a slowly varying, zero mean bias of 9.2 m and a random noise component of 0.7 m.

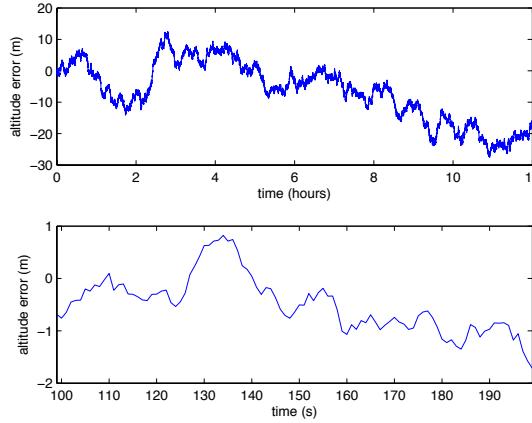
Table 7.2: Gauss-Markov Error Model Parameters.

Direction	Nominal 1- σ error (m)		Model Parameters		
	Bias	Random	Std. Dev. w (m)	$1/k_{\text{GPS}}$ (s)	T_s (s)
North	4.7	0.4	2.1	16,000	1.0
East	4.7	0.4	2.1	16,000	1.0
Altitude	9.2	0.7	4.0	16,000	1.0

To model the transient behavior of the error, we follow the approach of [?] and model the error as a Gauss-Markov process. Gauss-Markov processes are modeled by

$$\nu[n + 1] = e^{-k_{\text{GPS}}T_s}\nu[n] + \eta_{\text{GPS}}[n] \quad (7.17)$$

where $\nu[n]$ is the error being modeled, $\eta_{\text{GPS}}[n]$ is zero-mean Gaussian white noise, $1/k_{\text{GPS}}$ is the time constant of the process, and T_s is the sample time. Figure 7.9 shows results from the Gauss-Markov GPS altitude error model given by Equation (7.17). The error over the 12-hour period shown has a standard deviation of 9.4 m, while the noise component of the error has a standard deviation of 0.69 m. The upper plot shows the error over a 12-hour period, while the lower plot shows the error over a 100-second segment of time. Suitable Gauss-Markov process parameters to model the GPS error are given in Table 7.2.

**Figure 7.9:** Example GPS altitude position error from Gauss-Markov model.

Drawing on the error model in Equation (7.17) and the parameters in Table 7.2, we can create position error models for the North, East, and altitude

measurements from GPS: ν_n , ν_e , and ν_h . Accordingly, a model for GPS measurements that is suitable for simulation purposes is given by

$$y_{\text{GPS},n}[n] = p_n[n] + \nu_n[n] \quad (7.18)$$

$$y_{\text{GPS},e}[n] = p_e[n] + \nu_e[n] \quad (7.19)$$

$$y_{\text{GPS},h}[n] = -p_d[n] + \nu_h[n], \quad (7.20)$$

where p_n , p_e , and h are the actual earth coordinates and altitude above sea level, and n is the sample index. GPS measurements are commonly available from small UAV receivers at 1 Hz. New systems suitable for small UAV implementations provide GPS measurements at 5 Hz updates.

7.5.3 GPS Velocity Measurements

Using carrier phase Doppler measurements from GPS satellite signals, the velocity of the receiver can be calculated to accuracies with standard deviations in the range of 0.01 to 0.05 m/s. Many modern GPS receiver chips provide velocity information as part of their output data packet. In addition, they provide information on horizontal ground speed and course over the ground. Horizontal ground speed and course are calculated from the North and East velocity components from GPS as

$$V_g = \sqrt{V_n^2 + V_e^2} \quad (7.21)$$

$$\chi = \tan^{-1} \left(\frac{V_n}{V_e} \right), \quad (7.22)$$

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

Using basic principles of uncertainty analysis [?], the uncertainty in ground speed and course measurements can be estimated to be

$$\sigma_{V_g} = \sqrt{\frac{V_n^2 \sigma_{V_n}^2 + V_e^2 \sigma_{V_e}^2}{V_n^2 + V_e^2}}$$

$$\sigma_\chi = \sqrt{\frac{V_n^2 \sigma_{V_e}^2 + V_e^2 \sigma_{V_n}^2}{(V_n^2 + V_e^2)^2}}.$$

If the uncertainty in the North and East directions have the same magnitude (i.e., $\sigma_{V_n} = \sigma_{V_e} = \sigma_V$), these expressions simplify to be

$$\sigma_{V_g} = \sigma_V \quad (7.23)$$

$$\sigma_\chi = \frac{\sigma_V}{V_g}. \quad (7.24)$$

Notice that the uncertainty in the course measurement scales with the inverse of the ground speed – for high speeds the error is small and for low speeds the error is large. This is not unexpected since course is undefined for a stationary object. Based on Equations (7.21)–(7.22) and (7.23)–(7.24), we can model the ground speed and course measurements available from GPS as

$$y_{\text{GPS},V_g} = \sqrt{(V_a \cos \psi + w_n)^2 + (V_a \sin \psi + w_e)^2} + \eta_V \quad (7.25)$$

$$y_{\text{GPS},\chi} = \text{atan2}(V_a \sin \psi + w_e, V_a \cos \psi + w_n) + \eta_\chi, \quad (7.26)$$

where η_V and η_χ are zero mean Gaussian processes with variances $\sigma_{V_g}^2$ and σ_χ^2 .

7.6 CHAPTER SUMMARY

In this chapter, we have described the sensors commonly found on small unmanned aircraft and proposed models that describe their function for the purposes of simulation, analysis, and observer design. The simulation models characterize the errors of the sensors and their effective update rates. We have focused on sensors used for guidance, navigation, and control of the aircraft including accelerometers, rate gyros, absolute pressure sensors, differential pressure sensors, magnetometers, and GPS. Camera sensors will be discussed in Chapter 13.

NOTES AND REFERENCES

The accelerometers, rate gyros, and pressure sensors used on small unmanned aircraft are usually based on MEMS technology due to their small size and weight. Several references provide excellent overviews of these devices including [?, ?, ?]. The development of the global positioning system has been described in detail in several texts. Details describing its function and modeling of position errors can be found in [?, ?, ?, ?]. Specific information for sensor models can be found in manufacturer data sheets for the devices of interest.

7.7 DESIGN PROJECT

The objective of this project assignment is to add the sensors to the simulation model of the MAV.

- 7.1 Download the files associated with this chapter from the book website. Note that we have added a block for the sensors which contains two files: `sensors.m` and `gps.m`. The file `sensors.m` will model all of the sensors that update at rate T_s (gyros, accelerometers, pressure sensors), and `gps.m` will model the GPS sensor which is updated at rate $T_{s,\text{GPS}}$.
- 7.2 Using the sensor parameters listed in Appendix ??, modify `sensors.m` to simulate the output of the rate gyros (Eq. (7.5)), the accelerometers (Eq. (7.3)), and the pressure sensors (Eq (7.9) and (7.10)).
- 7.3 Using the sensor parameters listed in Appendix ??, modify `gps.m` to simulate the position measurement output of the GPS sensor (Eq. (7.18)–(7.20)) and the ground speed and course output of the GPS sensor(Eq. (7.25)–(7.26)).
- 7.4 Using a Simulink scope, observe the output of each sensor and verify that its sign and magnitude are approximately correct, and that the shape of the waveform is approximately correct.

Chapter Eight

State Estimation

The autopilot designed in Chapter 6 assumes that states of the system like roll and pitch angles are available for feedback. However, one of the challenges of MAV flight control is that sensors that directly measure roll and pitch are not available. Therefore, the objective of this chapter is to describe techniques for estimating the state of a small or micro air vehicle from the sensor measurements described in Chapter 7. Since rate gyros directly measure roll rates in the body frame, the states p , q , and r can be recovered by low-pass filtering the rate gyros. Therefore, we begin the chapter by discussing digital implementation of low-pass filters in Section 8.2. In Section 8.3 we describe a simple state-estimation scheme that is based on mathematically inverting the sensor models. However, this scheme does not account for the dynamics of the system and therefore does not perform well over the full range of flight conditions. Accordingly, in Section 8.5 we introduce dynamic-observer theory as a precursor to our discussion on the Kalman filter. A mathematical derivation of the Kalman filter is given in Section 8.6. For those with limited exposure to stochastic processes, Appendix ?? provides an overview of basic concepts from probability theory with a focus on Gaussian stochastic processes. The last two sections of the chapter describe applications of the Kalman filter. In Section 8.9, an extended Kalman filter is designed to estimate the roll and pitch attitude of the MAV, and in Section 8.10, an extended Kalman filter is used to estimate the position, ground speed, course, and heading of the MAV, as well as the wind speed and direction.

8.1 BENCHMARK MANEUVER

To illustrate the different estimation schemes presented in this chapter, we will use a maneuver that adequately excites all of the states. Initially the MAV will be in wings-level, trimmed flight at an altitude of 100 m, and an airspeed of 10 m/s. The maneuver is defined by commanding a constant airspeed of 10 m/s and by commanding altitude and heading as shown in Figure 8.1. By using the same benchmark maneuver to estimate the performance of the different estimators developed in this chapter, we can evaluate

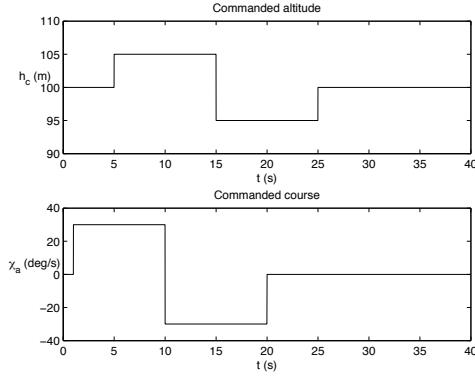


Figure 8.1: The altitude and heading commands that define the benchmark maneuver that will be used to evaluate and tune the state estimation scheme.

their relative performance. The benchmark maneuver commands longitudinal and lateral motions simultaneously, thus exposing any significant sensitivities of estimators to assumptions of decoupled dynamics.

8.2 LOW-PASS FILTERS

Since some of the estimation schemes described in this chapter require low-pass filtering of the sensor signal, this section describes digital implementation of a single pole low-pass filter. The Laplace transform representation of a simple unity DC-gain low-pass filter with cut-off frequency a is given by

$$Y(s) = \frac{a}{s+a} U(s),$$

where $U(s) = \mathcal{L}\{u(t)\}$ and $u(t)$ is the input of the filter, and where $Y(s) = \mathcal{L}\{y(t)\}$ and $y(t)$ is the output. Taking the inverse Laplace transform gives

$$\dot{y} = -ay + au. \quad (8.1)$$

From linear-systems theory, it is well known that the sampled-data solution to (8.1) is given by

$$y(t + T_s) = e^{-aT_s} y(t) + a \int_0^{T_s} e^{-a(T_s-\tau)} u(\tau) d\tau.$$

Assuming that $u(t)$ is constant between sample periods results in the expression

$$\begin{aligned} y[n+1] &= e^{-aT_s}y[n] + a \int_0^{T_s} e^{-a(T_s-\tau)} d\tau u[n] \\ &= e^{-aT_s}y[n] + (1 - e^{-aT_s})u[n]. \end{aligned} \quad (8.2)$$

If we let $\alpha_{LPF} = e^{-aT_s}$ then we get the simple form

$$y[n+1] = \alpha_{LPF}y[n] + (1 - \alpha_{LPF})u[n].$$

Note that this equation has a nice physical interpretation: the new value of y (filtered value) is a weighted average of the old value of y and u (unfiltered value). If u is noisy, then $\alpha_{LPF} \in [0, 1]$ should be close to unity. However, if u is relatively noise free, then α should be close to zero.

We will use the notation $LPF(\cdot)$ to represent the low-pass filter operator. Therefore $\hat{x} = LPF(x)$ is the low-pass filtered version of x .

8.3 STATE ESTIMATION BY INVERTING THE SENSOR MODEL

In this section we will derive the simplest possible state estimation scheme based on inverting the sensor models derived in Chapter 7. While this method is effective for angular rates, altitude, and airspeed, it is not effective for estimating the Euler angles or the position and course of the MAV.

8.3.1 Angular Rates

The angular rates p , q , and r can be estimated by low-pass filtering the rate gyro signals given by Equation (7.5) to obtain

$$\hat{p} = LPF(y_{gyro,x}) \quad (8.3)$$

$$\hat{q} = LPF(y_{gyro,y}) \quad (8.4)$$

$$\hat{r} = LPF(y_{gyro,z}). \quad (8.5)$$

For the benchmark maneuver discussed in Section 8.1, the estimation error for p , q , and r are shown in Figure 8.2. From the figure we see that low-pass filtering the gyro measurements produces acceptable estimates of p , q , and r .

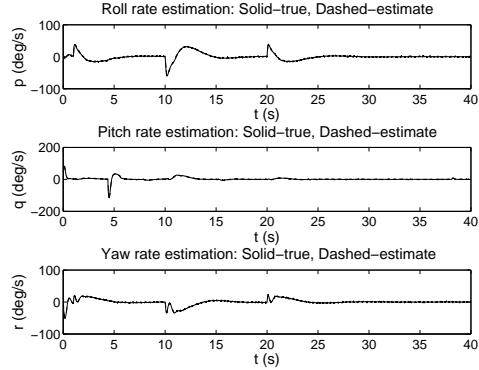


Figure 8.2: Estimation error on the angular rates obtained by low-pass filtering the rate gyros.

8.3.2 Altitude

A estimate of the altitude can be obtained from the absolute pressure sensor. Applying a low-pass filter to Equation (7.9) and dividing by ρg we get

$$\hat{h} = \frac{LPF(y_{\text{static pres}})}{\rho g}. \quad (8.6)$$

8.3.3 Airspeed

The airspeed can be estimated by applying a low-pass filter to the differential pressure sensor represented by Equation (7.10), and inverting to obtain

$$\hat{V}_a = \sqrt{\frac{2}{\rho} LPF(y_{\text{diff pres}})}. \quad (8.7)$$

For the benchmark maneuver discussed in Section 8.1, the estimates of the altitude and airspeed are shown in Figure 8.3, together with truth data. As can be seen from the figure, inverting the sensor model produces a fairly accurate model of the altitude and airspeed.

8.3.4 Roll and Pitch Angles

Roll and pitch angles are the most difficult variables to estimate well on small unmanned aircraft. A simple scheme that works in unaccelerated

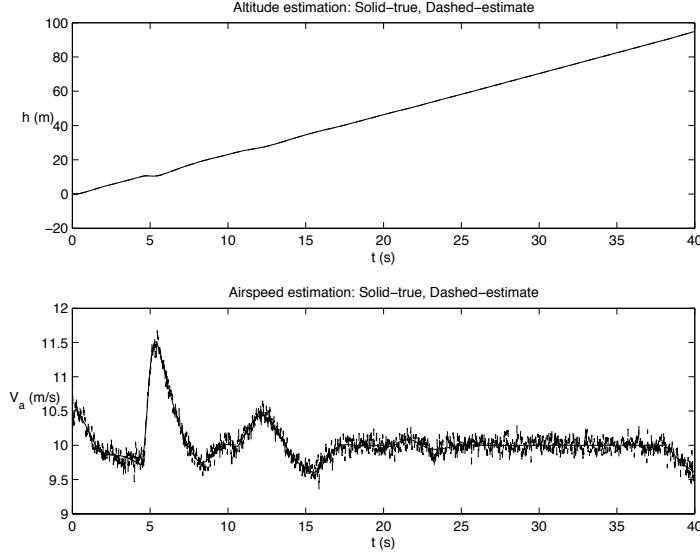


Figure 8.3: Estimation error of the altitude and airspeed obtained by low-pass filtering the pressure sensors and inverting the sensor model. For altitude and airspeed, the accuracy of the simple scheme is adequate.

flight can be derived as follows. Recall from Equation (7.1) that

$$\begin{aligned} y_{\text{accel},x} &= \dot{u} + qw - rv + g \sin \theta + \eta_{\text{accel},x} \\ y_{\text{accel},y} &= \dot{v} + ru - pw - g \cos \theta \sin \phi + \eta_{\text{accel},y} \\ y_{\text{accel},z} &= \dot{w} + pv - qu - g \cos \theta \cos \phi + \eta_{\text{accel},z}. \end{aligned}$$

In unaccelerated flight we have $\dot{u} = \dot{v} = \dot{w} = p = q = r = 0$, which implies that

$$\begin{aligned} LPF(y_{\text{accel},x}) &= g \sin \theta \\ LPF(y_{\text{accel},y}) &= -g \cos \theta \sin \phi \\ LPF(y_{\text{accel},z}) &= -g \cos \theta \cos \phi. \end{aligned}$$

Solving for ϕ and θ we get

$$\hat{\phi}_{\text{accel}} = \tan^{-1} \left(\frac{LPF(y_{\text{accel},y})}{LPF(y_{\text{accel},z})} \right) \quad (8.8)$$

$$\hat{\theta}_{\text{accel}} = \sin^{-1} \left(\frac{LPF(y_{\text{accel},x})}{g} \right). \quad (8.9)$$

The estimation errors of the roll and pitch angles for the benchmark maneuver discussed in Section 8.1 are shown in Figure 8.4, where it is clear that

the estimation error during accelerated flight is unacceptable. In Section 8.9 we will use an extended Kalman filter to provide more accurate estimates of the roll and pitch angles.

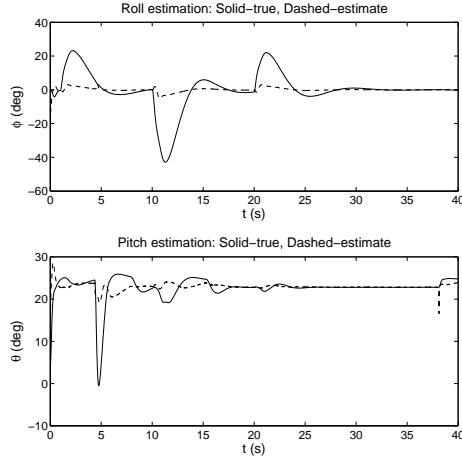


Figure 8.4: Estimation error on the roll and pitch angles obtained by low-pass filtering the accelerometers and inverting the model. Since this scheme assumes unaccelerated flight during maneuvers where acceleration exists, the estimation error can be unacceptably large.

8.3.5 Position, Course, and Groundspeed

The position of the MAV can be estimated by low-pass filtering Equations (7.18) and (7.19). The biases due to multipath, clock, and satellite geometry will not be removed. The estimate of the position variables is therefore given by

$$\hat{p}_n = LPF(y_{GPS,n}) \quad (8.10)$$

$$\hat{p}_e = LPF(y_{GPS,e}). \quad (8.11)$$

Similarly, an estimate of the course angle and ground speed of the MAV can be obtained by low-pass filtering Equations (7.26) and (7.25) to obtain

$$\hat{\chi} = LPF(y_{GPS,\chi}) \quad (8.12)$$

$$\hat{V}_g = LPF(y_{GPS,V_g}). \quad (8.13)$$

The primary downside of low-pass filtering GPS signals is that since the sample rate is slow (usually on the order of 1 Hz), there is significant delay in the estimate. The estimation scheme described in Section 8.10 will resolve this problem.

For the benchmark maneuver discussed in Section 8.1, the estimation error for the North and East position, and for the course and ground speed are shown in Figure 8.5. There is significant error due, in part, to the fact that the GPS sensor is only updated at 1 Hz. Clearly, simply low-pass filtering the GPS data does not produce satisfactory results.

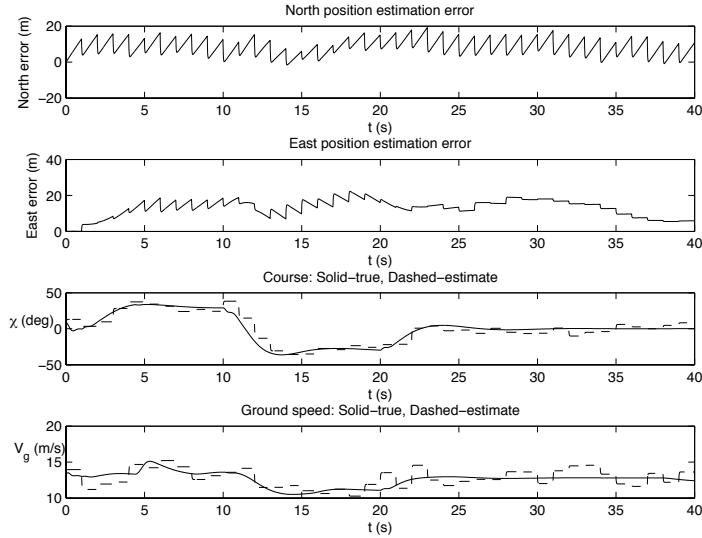


Figure 8.5: Estimation error for the North and East position, course, and ground speed obtained by low-pass filtering the GPS sensors.

We have shown in this section that adequate estimates of the body rates p , q , and r , as well as the altitude and the airspeed can be obtained by low pass filtering the sensors. However, estimating the roll and pitch angles and the position, course, and ground speed will require more sophisticated techniques. In particular, a simple low pass filter does not account for the underlying dynamics of the system. In the following section we will introduce dynamic observer theory. The most commonly used dynamic observer is the Kalman filter which will be derived in Section 8.6. The application of the Kalman filter to attitude estimation is in Section 8.9, and the application of the Kalman filter to position, course, and ground speed estimation is in Section 8.10.

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 $\omega_{b/i}^b = (p, q, r)^\top$ 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

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

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

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

■

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

$$\mathbf{m}^{v1} = R_b^{v1}(\phi, \theta) R_z^\top(\delta) R_y^\top(\iota)(\mathbf{y}_{mag} - \mathbf{b}_{mag}) \quad (8.14)$$

$$= R_b^{v1}(\phi, \theta) R_y(\iota) R_z(\delta) (\mathbf{m}^b + \boldsymbol{\nu}_{mag} + \boldsymbol{\eta}_{mag}) \quad (8.15)$$

$$= \begin{pmatrix} c_\theta & s_\theta s_\phi & s_\theta c_\phi \\ 0 & c_\phi & -s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{pmatrix} \begin{pmatrix} c_\delta & s_\delta & 0 \\ -s_\delta & c_\delta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_\iota & 0 & s_\iota \\ 0 & 1 & 0 \\ -s_\iota & 0 & c_\iota \end{pmatrix} (\mathbf{m}^b + \boldsymbol{\nu}_{mag} + \boldsymbol{\eta}_{mag}), \quad (8.16)$$

■

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

$$\begin{aligned}\hat{\psi}_{\text{mag}}(t) &= -\text{atan}2(m_y^{v1}, m_x^{v1}) \\ &= \psi(t) + \nu_\psi(t) + \eta_\psi.\end{aligned}$$

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 $\hat{\phi}_{\text{gyro}}$ and a low pass version of $\hat{\phi}_{\text{accel}}$ as

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

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

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

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

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

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 8.6.

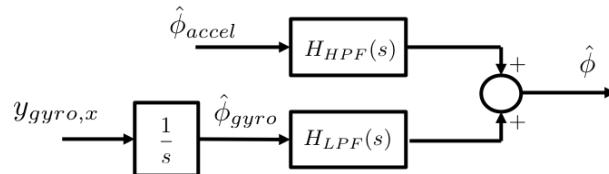


Figure 8.6: Naive complementary filter for roll angle estimation

The implementation of the complementary filter shown in Figure 8.6 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

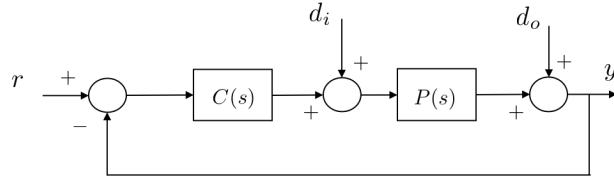


Figure 8.7: Standard feedback loop

strategy is to use a feedback configuration, as explained below. Consider the feedback loop show in Figure 8.7. Following standard block diagram manipulation we get that

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

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 ($\gg 1$) at low frequency and small ($\ll 1$) 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 8.8.

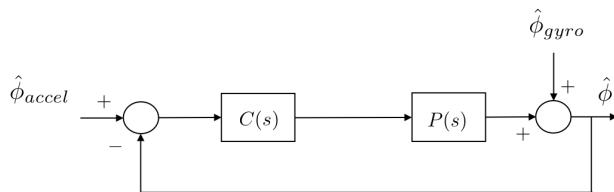


Figure 8.8: Feedback loop implementation of the complementary filter.

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 8.9. Figure 8.9 also

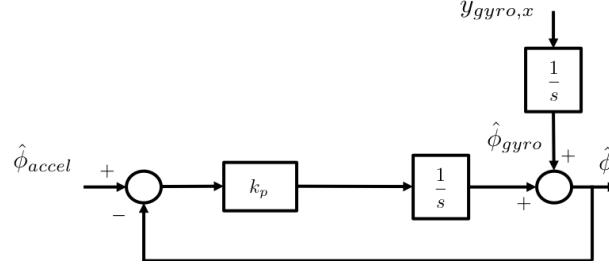


Figure 8.9: Feedback loop implementation of the complementary filter.

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 8.10.

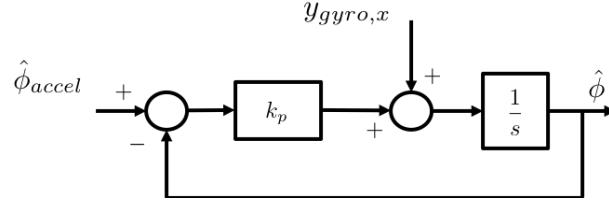


Figure 8.10: Feedback loop implementation of the complementary filter.

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

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

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

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

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 8.11. When $C(s) = k_p + k_i/s$ the steady state response to a constant

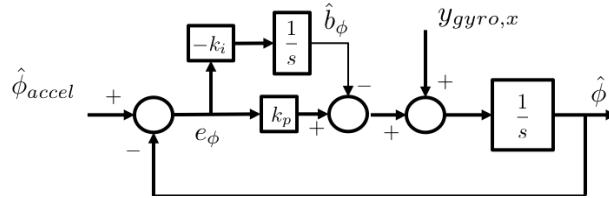


Figure 8.11: Feedback loop implementation of the complementary filter.

bias is

$$\begin{aligned}\hat{\phi}_{ss} &= \lim_{s \rightarrow 0} \frac{\frac{b}{s}}{1 + \frac{k_p + k_i/s}{s}} \\ &= \lim_{s \rightarrow 0} \frac{bs}{s^2 + k_ps + k_i} \\ &= 0.\end{aligned}$$

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

$$\begin{aligned}\dot{\hat{b}}_\phi &= -k_i(\hat{\phi}_{accel} - \hat{\phi}) \\ \dot{\hat{\phi}} &= (y_{gyro,x} - \hat{b}_\phi) + k_p(\hat{\phi}_{accel} - \hat{\phi}).\end{aligned}\tag{8.17}$$

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 (8.17) 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

$$\begin{aligned}\dot{\hat{b}}_\theta &= -k_i(\hat{\theta}_{accel} - \hat{\theta}) \\ \dot{\hat{\theta}} &= (y_{gyro,y} - \hat{b}_\theta) + k_p(\hat{\theta}_{accel} - \hat{\theta}),\end{aligned}\tag{8.18}$$

$$\begin{aligned}\dot{\hat{b}}_\psi &= -k_i(\hat{\psi}_{mag} - \hat{\psi}) \\ \dot{\hat{\psi}} &= (y_{gyro,z} - \hat{b}_\psi) + k_p(\hat{\psi}_{mag} - \hat{\psi}).\end{aligned}\tag{8.19}$$

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:

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

8.5 DYNAMIC-OBSERVER THEORY

The objective of this section is to briefly review observer theory, which serves as a precursor to our discussion on the Kalman filter. Suppose that we have a linear time-invariant system modeled by the equations

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

A continuous-time observer for this system is given by the equation

$$\dot{\hat{x}} = \underbrace{A\hat{x} + Bu}_{\text{copy of the model}} + \underbrace{L(y - C\hat{x})}_{\text{correction due to sensor reading}}, \quad (8.20)$$

where \hat{x} is the estimated value of x . Defining the observation error as $\tilde{x} = x - \hat{x}$ we find that

$$\dot{\tilde{x}} = (A - LC)\tilde{x},$$

which implies that the observation error decays exponentially to zero if L is chosen so that the eigenvalues of $A - LC$ are in the open left half of the complex plane.

In practice, the sensors are usually sampled and processed in digital hardware at sample rate T_s . How do we modify the observer equation shown in Equation (8.20) to account for sampled sensor readings? One approach is to propagate the system model between samples using the equation

$$\dot{\hat{x}} = A\hat{x} + Bu, \quad (8.21)$$

and then to update the estimate when a measurement is received using

$$\hat{x}^+ = \hat{x}^- + L(y(t_n) - C\hat{x}^-), \quad (8.22)$$

where t_n is the instant in time that the measurement is received and \hat{x}^- is the state estimate produced by Equation (8.21) at time t_n . Equation (8.21) is then re-instantiated with initial conditions given by \hat{x}^+ . If the system is nonlinear, then the propagation and update equations become

$$\dot{\hat{x}} = f(\hat{x}, u) \quad (8.23)$$

$$\hat{x}^+ = \hat{x}^- + L(y(t_n) - h(\hat{x}^-)). \quad (8.24)$$

The observation process is shown graphically in Figure 8.12. Note that it is not necessary to have a fixed sample rate. The continuous-discrete observer can be implemented using Algorithm 2.

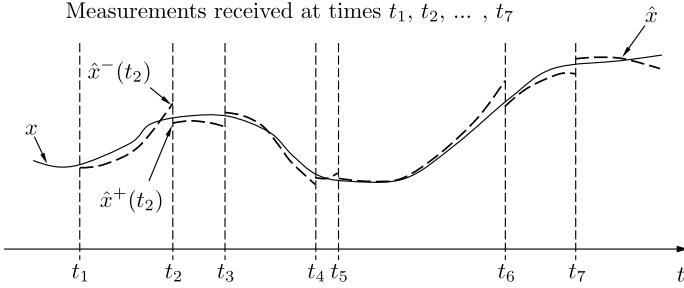


Figure 8.12: Time line for continuous-discrete dynamic observer. The vertical dashed lines indicate sample times at which measurements are received. In between measurements, the state is propagated using Equation (8.23). When a measurement is received, the state is updated using Equation (8.24).

A pseudo-code implementation of the continuous-discrete observer is shown in Algorithm 2. In Line 1 the state estimate is initialized to zero. If additional information is known, then the state can be initialized accordingly. The ODE in Equation (8.23) is propagated between samples with the for-loop in lines 4-6 using and Euler integration method. When a measurement is received, the state is updated using Equation (8.24) in line 8.

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: $\hat{x} = \hat{x} + L_i (y_i - h_i(\hat{x}))$
 - 9: **end if**
-

8.6 DERIVATION OF THE CONTINUOUS-DISCRETE KALMAN FILTER

The key parameter for the dynamic observer discussed in the previous section is the observer gain L . The Kalman filter and extended Kalman filters discussed in the remainder of this chapter are standard techniques for choosing L . If the process and measurement are linear, and the process and

measurement noise are zero mean white Gaussian processes with known covariance matrices, then the Kalman filter gives the optimal gain, where the optimality criteria will be defined later in this section. There are several different forms for the Kalman filter, but the form that is particularly useful for MAV applications is the continuous-propagation, discrete-measurement Kalman filter.

We will assume that the (linear) system dynamics are given by

$$\begin{aligned}\dot{x} &= Ax + Bu + \xi \\ y[n] &= Cx[n] + \eta[n],\end{aligned}\tag{8.25}$$

where $y[n] = y(t_n)$ is the n^{th} sample of y , $x[n] = x(t_n)$ is the n^{th} sample of x , $\eta[n]$ is the measurement noise at time t_n , ξ is a zero-mean Gaussian random process with covariance Q , and $\eta[n]$ is a zero-mean Gaussian random variable with covariance R . The random process ξ is called the process noise and represents modeling error and disturbances on the system. The random variable η is called the measurement noise and represents noise on the sensors. The covariance R can usually be estimated from sensor calibration, but the covariance Q is generally unknown and therefore becomes a system gain that can be tuned to improve the performance of the observer. Note that the sample rate does not need to be fixed.

Similar to Equations (8.21) and (8.22) the continuous-discrete Kalman filter has the form

$$\begin{aligned}\dot{\hat{x}} &= A\hat{x} + Bu \\ \hat{x}^+ &= \hat{x}^- + L(y(t_n) - C\hat{x}^-).\end{aligned}$$

Define the estimation error as $\tilde{x} = x - \hat{x}$. The covariance of the estimation error at time t is given by

$$P(t) \triangleq E\{\tilde{x}(t)\tilde{x}(t)^\top\}.\tag{8.26}$$

Note that $P(t)$ is symmetric and positive semi-definite, therefore, its eigenvalues are real and non-negative. Also, small eigenvalues of $P(t)$ imply small variance, which implies low average estimation error. Therefore, we would like to choose $L(t)$ to minimize the eigenvalues of $P(t)$. Recall that

$$\text{tr}(P) = \sum_{i=1}^n \lambda_i,$$

where $\text{tr}(P)$ is the trace of P , and λ_i are the eigenvalues of P . Therefore, minimizing $\text{tr}(P)$ minimizes the estimation error covariance. The Kalman filter is derived by finding L to minimize $\text{tr}(P)$.

8.6.0.1 Between Measurements:

Differentiating \tilde{x} we get

$$\begin{aligned}\dot{\tilde{x}} &= \dot{x} - \dot{\hat{x}} \\ &= Ax + Bu + \xi - A\hat{x} - Bu \\ &= A\tilde{x} + \xi.\end{aligned}$$

Solving the differential equation with initial conditions \tilde{x}_0 we obtain

$$\tilde{x}(t) = e^{At}\tilde{x}_0 + \int_0^t e^{A(t-\tau)}\xi(\tau) d\tau.$$

We can compute the evolution of the error covariance P as

$$\begin{aligned}\dot{P} &= \frac{d}{dt}E\{\tilde{x}\tilde{x}^\top\} \\ &= E\{\dot{\tilde{x}}\tilde{x}^\top + \tilde{x}\dot{\tilde{x}}^\top\} \\ &= E\left\{A\tilde{x}\tilde{x}^\top + \xi\tilde{x}^\top + \tilde{x}\tilde{x}^\top A^\top + \tilde{x}\xi^\top\right\} \\ &= AP + PA^\top + E\{\xi\tilde{x}^\top\} + E\{\tilde{x}\xi^\top\}.\end{aligned}$$

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

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

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:

At a measurement, we have that

$$\begin{aligned}\tilde{x}^+ &= x - \hat{x}^+ \\ &= x - \hat{x}^- - L(Cx + \eta - C\hat{x}^-) \\ &= \tilde{x}^- - LC\tilde{x}^- - L\eta.\end{aligned}$$

We also have that

$$\begin{aligned}
P^+ &= E\{\tilde{x}^+ \tilde{x}^{+T}\} \\
&= E\left\{(\tilde{x}^- - LC\tilde{x}^- - L\eta)(\tilde{x}^- - LC\tilde{x}^- - L\eta)^T\right\} \\
&= E\left\{\tilde{x}^-\tilde{x}^{-T} - \tilde{x}^-\tilde{x}^{-T}C^T L^T - \tilde{x}^-\eta^T L^T \right. \\
&\quad \left. - LC\tilde{x}^-\tilde{x}^{-T} + LC\tilde{x}^-\tilde{x}^{-T}C^T L^T + LC\tilde{x}^-\eta^T L^T \right. \\
&\quad \left. - L\eta\tilde{x}^{-T} + L\eta\tilde{x}^{-T}C^T L^T + L\eta\eta^T L^T\right\} \\
&= P^- - P^- C^T L^T - LCP^- + LCP^- C^T L^T + LRL^T, \quad (8.27)
\end{aligned}$$

where we have used the fact that since η and \tilde{x}^- are independent, $E\{\tilde{x}^-\eta^T L^T\} = E\{L\eta\tilde{x}^{-T}\} = 0$.

In the derivation that follows, we will need the following matrix relationships:

$$\frac{\partial}{\partial A} \text{tr}(BAD) = B^T D^T \quad (8.28)$$

$$\frac{\partial}{\partial A} \text{tr}(ABA^T) = 2AB, \text{ if } B = B^T. \quad (8.29)$$

Our objective is to pick L to minimize $\text{tr}(P^+)$. A necessary condition is that

$$\begin{aligned}
\frac{\partial}{\partial L} \text{tr}(P^+) &= -P^- C^T - P^- C^T + 2LCP^- C^T + 2LR = 0 \\
&\implies 2L(R + CP^- C^T) = 2P^- C^T \\
&\implies L = P^- C^T (R + CP^- C^T)^{-1}.
\end{aligned}$$

Substituting into Equation (8.27) gives

$$\begin{aligned}
P^+ &= P^- + P^- C^T (R + CP^- C^T)^{-1} CP^- - P^- C^T (R + CP^- C^T)^{-1} CP^- \\
&\quad + P^- C^T (R + CP^- C^T)^{-1} (CP^- C^T + R) (R + CP^- C^T)^{-1} CP^- \\
&= P^- - P^- C^T (R + CP^- C^T)^{-1} CP^- \\
&= (I - P^- C^T (R + CP^- C^T)^{-1} C) P^- \\
&= (I - LC) P^-.
\end{aligned}$$

We can therefore summarize the Kalman filter as follows. In between measurements, propagate the equations

$$\begin{aligned}
\dot{\hat{x}} &= A\hat{x} + Bu \\
\dot{P} &= AP + PA^T + Q,
\end{aligned}$$

where \hat{x} is the estimate of the state, and P is the symmetric covariance matrix of the estimation error. When a measurement from the i^{th} sensor is received, update the state estimate and error covariance according to the equations

$$\begin{aligned} L_i &= P^- C_i^\top (R_i + C_i P^- C_i^\top)^{-1} \\ P^+ &= (I - L_i C_i) P^- \\ \hat{x}^+ &= \hat{x}^- + L_i (y_i(t_n) - C_i \hat{x}^-), \end{aligned}$$

where L_i is called the Kalman gain for sensor i .

We have assumed that the system propagation models and measurement model are linear. However, for many applications, including the applications discussed later in this chapter, the system propagation model and the measurement model are nonlinear. In other words, the model in Equation (8.25) becomes

$$\begin{aligned} \dot{x} &= f(x, u) + \xi \\ y[n] &= h(x[n], u[n]) + \eta[n]. \end{aligned}$$

For this case, the state propagation and update laws use the nonlinear model, but the propagation and update of the error covariance use the Jacobian of f for A , and the Jacobian of h for C . The resulting algorithm is called the Extended Kalman Filter (EKF). Pseudo-code for the EKF is shown in Algorithm 3. The state is initialized in Line 1. The propagation of the ODEs for \hat{x} and P using an Euler integration scheme are given by the for-loop in Lines 4–8. The update equations for the i^{th} sensor are given in Lines 9–14. The application of Algorithm 3 to roll and pitch angle estimation is described in Section 8.9. The application of Algorithm 3 to position, heading, ground speed, course, and wind estimation is described in Section 8.10.

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

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

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:    $\hat{x} = \hat{x} + \left(\frac{T_{out}}{N}\right) f(\hat{x}, u)$ 
6:    $A = \frac{\partial f}{\partial x}(\hat{x}, u)$ 
7:    $P = P + \left(\frac{T_{out}}{N}\right) (AP + PA^\top + Q)$ 
8: end for
9: if Measurement has been received from sensor  $i$  then {Measurement
   Update}
10:   $C_i = \frac{\partial h_i}{\partial x}(\hat{x}, u[n])$ 
11:   $L_i = PC_i^\top(R_i + C_i P C_i^\top)^{-1}$ 
12:   $P = (I - L_i C_i)P$ 
13:   $\hat{x} = \hat{x} + L_i(y_i[n] - h(\hat{x}, u[n]))$ .
14: end if

```

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

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

Using the approximation

$$\begin{aligned} A_d &= e^{AT_s} \approx I + AT_s + A^2 \frac{T_s^2}{2} \\ B_d &= \left(\int_0^{T_s} e^{A\tau}d\tau\right) \approx \int_0^{T_s} Id\tau = T_s I, \end{aligned}$$

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

$$P^+ = (I - LC)P^- \quad (8.30)$$

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 (8.27):

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

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 (8.30), it is more numerically stable to use the so-called Joseph's Stabilized form:

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

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.

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

$$L = P^- C^\top S^{-1}.$$

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

This section describes the application of the EKF to estimate the roll and pitch angles of the MAV. To apply the continuous-discrete extended Kalman filter derived in Section 8.6 to roll and pitch estimation, we use the nonlinear propagation model

$$\begin{aligned}\dot{\phi} &= p + q \sin \phi \tan \theta + r \cos \phi \tan \theta + \xi_{\phi} \\ \dot{\theta} &= q \cos \phi - r \sin \phi + \xi_{\theta},\end{aligned}$$

where we have added the noise terms ξ_{ϕ} and ξ_{θ} to model the noise on p , q , and r , where $\xi_{\phi} \sim \mathcal{N}(0, Q_{\phi})$ and $\xi_{\theta} \sim \mathcal{N}(0, Q_{\theta})$.

We will use the accelerometers as the output equations. From Equation (7.1) we have the accelerometer model

$$y_{\text{accel}} = \begin{pmatrix} \dot{u} + gw - rv + g \sin \theta \\ \dot{v} + ru - pw - g \cos \theta \sin \phi \\ \dot{w} + pv - qu - g \cos \theta \cos \phi \end{pmatrix} + \eta_{\text{accel}}. \quad (8.31)$$

However, we do not have a method for directly measuring \dot{u} , \dot{v} , \dot{w} , u , v , and w . We will assume that $\dot{u} = \dot{v} = \dot{w} \approx 0$. From Equation (2.7) we have

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} \approx V_a \begin{pmatrix} \cos \alpha \cos \beta \\ \sin \beta \\ \sin \alpha \cos \beta \end{pmatrix}.$$

Assuming that $\alpha \approx \theta$ and $\beta \approx 0$ we obtain

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} \approx V_a \begin{pmatrix} \cos \theta \\ 0 \\ \sin \theta \end{pmatrix}.$$

Substituting into Equation (8.31), we get

$$y_{\text{accel}} = \begin{pmatrix} qV_a \sin \theta + g \sin \theta \\ rV_a \cos \theta - pV_a \sin \theta - g \cos \theta \sin \phi \\ -qV_a \cos \theta - g \cos \theta \cos \phi \end{pmatrix} + \eta_{\text{accel}}.$$

Defining $x = (\phi, \theta)^\top$, $u = (p, q, r, V_a)^\top$, $\xi = (\xi_\phi, \xi_\theta)^\top$, and $\eta = (\eta_\phi, \eta_\theta)^\top$, gives

$$\begin{aligned} \dot{x} &= f(x, u) + \xi \\ y &= h(x, u) + \eta, \end{aligned}$$

where

$$\begin{aligned} f(x, u) &= \begin{pmatrix} p + q \sin \phi \tan \theta + r \cos \phi \tan \theta \\ q \cos \phi - r \sin \phi \end{pmatrix} \\ h(x, u) &= \begin{pmatrix} qV_a \sin \theta + g \sin \theta \\ rV_a \cos \theta - pV_a \sin \theta - g \cos \theta \sin \phi \\ -qV_a \cos \theta - g \cos \theta \cos \phi \end{pmatrix}. \end{aligned}$$

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

$$\begin{aligned} \frac{\partial f}{\partial x} &= \begin{pmatrix} q \cos \phi \tan \theta - r \sin \phi \tan \theta & \frac{q \sin \phi - r \cos \phi}{\cos^2 \theta} \\ -q \sin \phi - r \cos \phi & 0 \end{pmatrix} \\ \frac{\partial h}{\partial x} &= \begin{pmatrix} 0 & qV_a \cos \theta + g \cos \theta \\ -g \cos \phi \cos \theta & -rV_a \sin \theta - pV_a \cos \theta + g \sin \phi \sin \theta \\ g \sin \phi \cos \theta & (qV_a + g \cos \phi) \sin \theta \end{pmatrix}. \end{aligned}$$

The extended Kalman filter is implemented using Algorithm 3.

For the benchmark maneuver discussed in Section 8.1, the estimation error of the roll and pitch angles using Algorithm 3 is shown in Figure 8.13. Comparing Figure 8.13 with Figures 8.4 shows that the continuous-discrete extended Kalman filter produces much better results during accelerated flight. ■

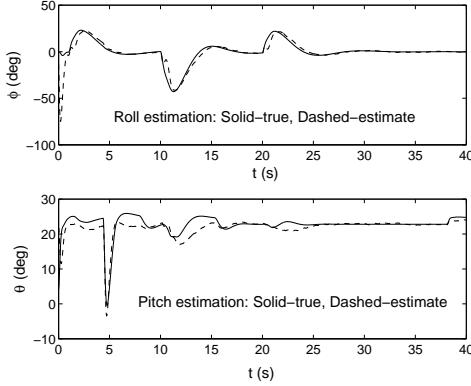


Figure 8.13: Estimation error on the roll and pitch angles using continuous-discrete extended Kalman filter.

8.10 GPS SMOOTHING

In this section we will use GPS measurements to estimate the position, ground speed, course, wind, and heading of the MAV. If we assume that the flight path angle $\gamma = 0$, then the evolution of the position is given by

$$\begin{aligned}\dot{p}_n &= V_g \cos \chi \\ \dot{p}_e &= V_g \sin \chi.\end{aligned}$$

By differentiating Equation (7.21) we get that the evolution of the ground speed is given by

$$\begin{aligned}\dot{V}_g &= \frac{d}{dt} \sqrt{(V_a \cos \psi + w_n)^2 + (V_a \sin \psi + w_e)^2} \\ &= \frac{(V_a \cos \psi + w_n)(\dot{V}_a \cos \psi - V_a \dot{\psi} \sin \psi + \dot{w}_n) + (V_a \sin \psi + w_e)(\dot{V}_a \sin \psi + V_a \dot{\psi} \cos \psi + \dot{w}_e)}{V_g}.\end{aligned}$$

Assuming that wind and airspeed are constant we get

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

From Equation (5.3) the evolution of χ is given by

$$\dot{\chi} = \frac{g}{V_g} \tan \phi.$$

Assuming that wind is constant we have

$$\begin{aligned}\dot{w}_n &= 0 \\ \dot{w}_e &= 0.\end{aligned}$$

From Equation (5.13) the evolution of ψ is given by

$$\dot{\psi} = q \frac{\sin \phi}{\cos \theta} + r \frac{\cos \phi}{\cos \theta}. \quad (8.32)$$

Defining the state as $x = (p_n, p_e, V_g, \chi, w_n, w_e, \psi)^\top$, and the input as $u = (V_a, q, r, \phi, \theta)^\top$, the nonlinear propagation model is given by $\dot{x} = f(x, u)$, where

$$f(x, u) \triangleq \begin{pmatrix} V_g \cos \chi \\ V_g \sin \chi \\ \frac{(V_a \cos \psi + w_n)(-V_a \dot{\psi} \sin \psi) + (V_a \sin \psi + w_e)(V_a \dot{\psi} \cos \psi)}{V_g} \\ \frac{q}{V_g} \tan \phi \\ 0 \\ 0 \\ q \frac{\sin \phi}{\cos \theta} + r \frac{\cos \phi}{\cos \theta} \end{pmatrix}.$$

The Jacobian of f is given by

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

where $\dot{\psi}$ is given in Equation (8.32) and

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

For measurements, we will use the GPS signals for North and East position, ground speed, and course. Since the states are not independent, we will use the wind triangle relationship given in Equation (2.10). Assuming that $\gamma = \gamma_a = 0$ we have that

$$\begin{aligned}V_a \cos \psi + w_n &= V_g \cos \chi \\ V_a \sin \psi + w_e &= V_g \sin \chi.\end{aligned}$$

From these expressions, we define the pseudo measurements

$$\begin{aligned} y_{\text{wind,n}} &= V_a \cos \psi + w_n - V_g \cos \chi \\ y_{\text{wind,e}} &= V_a \sin \psi + w_e - V_g \sin \chi, \end{aligned}$$

where the (pseudo) measurement values are equal to zero. The resulting measurement model is given by

$$y_{\text{GPS}} = h(x, u) + \eta_{\text{GPS}},$$

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

$$h(x, u) = \begin{pmatrix} p_n \\ p_e \\ V_g \\ \chi \\ V_a \cos \psi + w_n - V_g \cos \chi \\ V_a \sin \psi + w_e - V_g \sin \chi \end{pmatrix},$$

and where the Jacobian is given by

$$\frac{\partial h}{\partial x}(\hat{x}, u) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -\cos \chi & V_g \sin \chi & 1 & 0 & -V_a \sin \psi \\ 0 & 0 & -\sin \chi & -V_g \cos \chi & 0 & 1 & V_a \cos \psi \end{pmatrix}.$$

The extended Kalman filter to estimate p_n , p_e , V_g , χ , w_n , w_e , and ψ is implemented using Algorithm 3.

For the benchmark maneuver discussed in Section 8.1, the estimation error for the position and heading using Algorithm 3 is shown in Figure 8.14. Comparing Figure 8.14 with Figure 8.5, shows that the continuous-discrete extended Kalman filter produces much better results than a simple low-pass filter.

8.11 FULL STATE EKF

NEW MATERIAL:

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

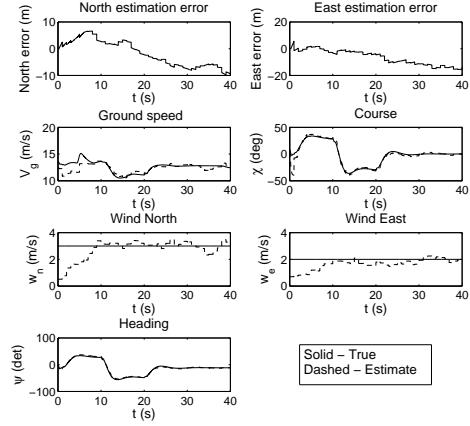


Figure 8.14: Estimation error for position, ground speed, course, wind, and heading using continuous-discrete extended Kalman filter.

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

$$R(\Theta) = \begin{pmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\phi s_\theta c_\psi - c_\phi s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi c_\theta \\ c_\phi s_\theta c_\psi + s_\phi s_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi & c_\phi c_\theta \end{pmatrix}$$

$$S(\Theta) = \begin{pmatrix} 1 & \sin \phi \tan \theta & \cos \phi \tan \theta \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi \sec \theta & \cos \phi \sec \theta \end{pmatrix},$$

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

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

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

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

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 $\mathbf{w} \in \mathbb{R}^3$

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

Therefore

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

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

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

where

$$\begin{aligned} Q_{\text{gyro}} &= \text{diag}([\sigma_{\text{gyro},x}^2, \sigma_{\text{gyro},y}^2, \sigma_{\text{gyro},z}^2]) \\ Q_{\text{accel}} &= \text{diag}([\sigma_{\text{accel},x}^2, \sigma_{\text{accel},y}^2, \sigma_{\text{accel},z}^2]), \end{aligned}$$

and where

$$Q = \text{diag}([\sigma_{p_n}^2, \sigma_{p_e}^2, \sigma_{p_d}^2, \sigma_u^2, \sigma_v^2, \sigma_w^2, \sigma_\phi^2, \sigma_\theta^2, \sigma_\psi^2, \sigma_{b_x}^2, \sigma_{b_y}^2, \sigma_{b_z}^2, \sigma_{w_n}^2, \sigma_{w_e}^2])$$

are tuning parameters.

8.11.3 Sensor Update Equations

8.11.3.1 Static Pressure Sensor

The model for the static pressure sensor is

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

Therefore, the Jacobian is given by

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

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 $V_a^2 = (\mathbf{v} - R^\top(\Theta)\mathbf{w})^\top (\mathbf{v} - R^\top(\Theta)\mathbf{w})$, and where $\mathbf{w} = (w_n, w_e, 0)^\top$. 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

$$C_{\text{diff}}(\mathbf{x}) = \rho \begin{pmatrix} 0_{3 \times 1} \\ \mathbf{v} - R^\top(\Theta)\mathbf{w} \\ -\frac{\partial [R^\top(\Theta)\mathbf{w}]}{\partial \Theta}^\top (\mathbf{v} - R^\top(\Theta)\mathbf{w}) \\ 0_{3 \times 1} \\ - (I_{2 \times 2}, \quad 0_{2 \times 1}) R(\Theta) (\mathbf{v} - R^\top(\Theta)\mathbf{w}) \end{pmatrix}^\top.$$

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

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

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

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

8.11.3.5 GPS ground-speed sensor

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

$$\mathbf{v}_{g\perp} = [I_{2 \times 2}, \quad 0_{2 \times 1}] R(\Theta) \mathbf{v}. \quad (8.33)$$

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 $P = [I_{2 \times 2}, \quad 0_{2 \times 1}]$, 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 (8.33). 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

$$C_{\text{GPS}, \chi}(\mathbf{x}) = \begin{pmatrix} 0_{1 \times 3}, & \frac{\partial \chi(\mathbf{v}, \mathbf{v})}{\partial \mathbf{v}}^\top, & \frac{\partial \chi(\mathbf{v}, \Theta)}{\partial \Theta}^\top, & 0_{1 \times 3}, & 0_{1 \times 2} \end{pmatrix},$$

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

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

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

$$v_r(\mathbf{v}, \Theta, \mathbf{w}) = [0 \ 1 \ 0] (\mathbf{v} - R(\Theta)^\top \mathbf{w}).$$

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

$$C_\beta(\mathbf{x}) = \begin{pmatrix} 0_{1 \times 3}, & \frac{\partial v_r(\mathbf{v}, \Theta, \mathbf{w})}{\partial \mathbf{v}}^\top, & \frac{\partial v_r(\mathbf{v}, \Theta, \mathbf{w})}{\partial \Theta}^\top, & 0_{1 \times 3}, & \frac{\partial v_r(\mathbf{v}, \Theta, \mathbf{w})}{\partial \mathbf{w}}^\top, & \end{pmatrix},$$

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 $T_{\text{GPS}} \gg T_s$.

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

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

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

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

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

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

8.12 CHAPTER SUMMARY

This chapter has shown how to estimate the states that are required for the autopilot discussed in Chapter 6 using the sensors described in Chapter 7. We have shown that the angular rates in the body frame p , q , and r , can be estimated by low-pass filtering the rate gyros. Similarly, the altitude h and airspeed V_a can be estimated by low-pass filtering the absolute and differential pressure sensors and inverting the sensor model. The remaining states must be estimated using extended Kalman filters. In Section 8.9 we showed how a two-state EKF can be used to estimate the roll and pitch angles. In Section 8.10 we showed how the position, ground speed, course, wind, and heading can be estimated using a seven-state EKF based on GPS measurements.

NOTES AND REFERENCES

The Kalman filter was first introduced in [?]. There are many excellent texts on the Kalman filter including [21, 22, 23, 24]. Some of the results in this chapter have been discussed previously in [25, 26]. State estimation using computer vision instead of GPS has been discussed in [27, 28, ?].

8.13 DESIGN PROJECT

- 8.1 Download the simulation files for this chapter from the web. State estimation is performed in the file `estimate_states.m`.
- 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 , ψ , 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 `estimate_states.m` 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 `estimate_states.m` 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 In the Simulink model `mavsim_chap8.mdl` change the switch that directs true states to the autopilot, to direct the estimated states to the autopilot. 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

As described in Chapter 1, when the equations of motion for a system become complex, it is often necessary to develop design models that have significantly less mathematical complexity, but still capture the essential behavior of the system. If we include all the elements discussed in the previous eight chapters, including the six-degree-of-freedom model developed in Chapters 3 and 4, the autopilot developed in Chapter 6, the sensors developed in Chapter 7, and the state-estimation scheme developed in Chapter 8, the resulting model is extremely complex. This chapter approximates the performance of the closed-loop MAV system and develops reduced-order design models that are appropriate for the design of higher-level guidance strategies for MAVs. We will present several different models that are commonly used in the literature. The design models developed in this chapter will be used in Chapters 10 through 13.

9.1 AUTOPILOT MODEL

The guidance models developed in this chapter use a high-level representation of the autopilot loops developed in Chapter 6. The airspeed-hold and roll-hold loops are represented by the first-order models

$$\dot{V}_a = b_{V_a}(V_a^c - V_a) \quad (9.1)$$

$$\dot{\phi} = b_\phi(\phi^c - \phi), \quad (9.2)$$

where b_{V_a} and b_ϕ are positive constants that depend on the implementation of the autopilot and the state estimation scheme. Drawing on the closed-loop transfer functions of Chapter 6, the altitude and course-hold loops are represented by the second-order models

$$\ddot{h} = b_{\dot{h}}(\dot{h}^c - \dot{h}) + b_h(h^c - h) \quad (9.3)$$

$$\ddot{\chi} = b_{\dot{\chi}}(\dot{\chi}^c - \dot{\chi}) + b_\chi(\chi^c - \chi), \quad (9.4)$$

where $b_{\dot{h}}$, b_h , $b_{\dot{\chi}}$, and b_χ are also positive constants that depend on the implementation of the autopilot and the state estimation schemes. As explained

in subsequent sections, some of the guidance models also assume autopilot-hold loops for the flight-path angle γ and the load factor n_{lf} , where load factor is defined as lift divided by weight. The first-order autopilot loops for flight-path angle and load factor are given by

$$\dot{\gamma} = b_\gamma(\gamma^c - \gamma) \quad (9.5)$$

$$\dot{n}_{lf} = b_n(n_{lf}^c - n_{lf}), \quad (9.6)$$

where b_γ and b_n are positive constants that depend on the implementation of the low-level autopilot loops.

9.2 KINEMATIC MODEL OF CONTROLLED FLIGHT

In deriving reduced-order guidance models, the main simplification we make is to eliminate the force- and moment-balance equations of motion (those involving \dot{u} , \dot{v} , \dot{w} , \dot{p} , \dot{q} , \dot{r}), thus eliminating the need to calculate the complex aerodynamic forces acting on the airframe. These general equations are replaced with simpler kinematic equations derived for the specific flight conditions of a coordinated turn and an accelerating climb.

Recall from Figure 2.10 that the velocity vector of the aircraft with respect to the inertial frame can be expressed in terms of the course angle and the (inertially referenced) flight-path angle as

$$\mathbf{V}_g^i = V_g \begin{pmatrix} \cos \chi \cos \gamma \\ \sin \chi \cos \gamma \\ -\sin \gamma \end{pmatrix}.$$

Therefore, the kinematics can be expressed as

$$\begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{h} \end{pmatrix} = V_g \begin{pmatrix} \cos \chi \cos \gamma \\ \sin \chi \cos \gamma \\ \sin \gamma \end{pmatrix}. \quad (9.7)$$

Because it is common to control the heading and airspeed of an aircraft, it is useful to express Equation (9.7) in terms of ψ and V_a . With reference to the wind triangle expression in Equation (2.10), we can write Equation (9.7) as

$$\begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{h} \end{pmatrix} = V_a \begin{pmatrix} \cos \psi \cos \gamma_a \\ \sin \psi \cos \gamma_a \\ \sin \gamma_a \end{pmatrix} + \begin{pmatrix} w_n \\ w_e \\ -w_d \end{pmatrix}. \quad (9.8)$$

If we assume that the aircraft is maintained at a constant altitude and that there is no downward component of wind, then the kinematic expressions

simplify as

$$\begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{h} \end{pmatrix} = V_a \begin{pmatrix} \cos \psi \\ \sin \psi \\ 0 \end{pmatrix} + \begin{pmatrix} w_n \\ w_e \\ 0 \end{pmatrix}, \quad (9.9)$$

which is a model commonly used in the UAV literature.

9.2.1 Coordinated Turn

In Section 5.1 we showed that the coordinated-turn condition is described by

$$\dot{\chi} = \frac{g}{V_g} \tan \phi. \quad (9.10)$$

Even though the coordinated-turn condition is not enforced by the autopilot loops described in Chapter 6, the essential behavior—that the aircraft must bank to turn (as opposed to skid to turn)—is captured by this model.

The coordinated-turn condition can also be expressed in terms of the heading and the airspeed. To obtain the correct expression, we start by differentiating both sides of Equation (2.10) to get

$$\begin{aligned} & \begin{pmatrix} \cos \chi \cos \gamma & -V_g \sin \chi \cos \gamma & -V_g \cos \chi \sin \gamma \\ \sin \chi \cos \gamma & V_g \cos \chi \cos \gamma & -V_g \sin \chi \sin \gamma \\ -\sin \gamma & 0 & -\cos \gamma \end{pmatrix} \begin{pmatrix} \dot{V}_g \\ \dot{\chi} \\ \dot{\gamma} \end{pmatrix} \\ &= \begin{pmatrix} \cos \psi \cos \gamma_a & -V_g \sin \psi \cos \gamma_a & -V_g \cos \psi \sin \gamma_a \\ \sin \psi \cos \gamma_a & V_g \cos \psi \cos \gamma_a & -V_g \sin \psi \sin \gamma_a \\ -\sin \gamma_a & 0 & -\cos \gamma_a \end{pmatrix} \begin{pmatrix} \dot{V}_a \\ \dot{\psi} \\ \dot{\gamma}_a \end{pmatrix}. \end{aligned} \quad (9.11)$$

Under the condition of constant-altitude flight and no down component of wind, where γ , γ_a , $\dot{\gamma}$, $\dot{\gamma}_a$, and w_d are zero, we solve for \dot{V}_g and $\dot{\psi}$ in terms of \dot{V}_a and $\dot{\chi}$ to obtain

$$\dot{V}_g = \frac{\dot{V}_a}{\cos(\chi - \psi)} + V_g \dot{\chi} \tan(\chi - \psi) \quad (9.12)$$

$$\dot{\psi} = \frac{\dot{V}_a}{V_a} \tan(\chi - \psi) + \frac{V_g \dot{\chi}}{V_a \cos(\chi - \psi)}. \quad (9.13)$$

If we assume that the airspeed is constant, then from Equations (9.13) and (9.10), we have

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

Note that if the wind speed is zero ($\chi = \psi$), this reduces to the familiar coordinated-turn expression

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

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 at an angle of ϕ , the

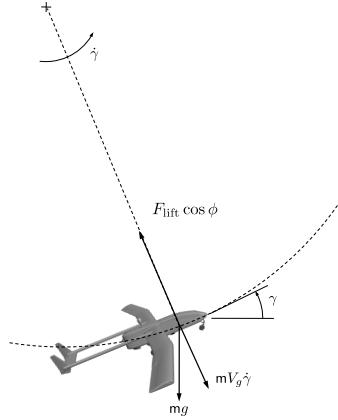


Figure 9.1: Free-body diagram for a pull-up maneuver. The MAV is at a roll angle of ϕ .

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

$$F_{\text{lift}} \cos \phi = mV_g\dot{\gamma} + mg \cos \gamma. \quad (9.14)$$

Solving for $\dot{\gamma}$ gives

$$\dot{\gamma} = \frac{g}{V_g} \left(\frac{F_{\text{lift}}}{mg} \cos \phi - \cos \gamma \right). \quad (9.15)$$

Load factor is defined as the ratio of the lift acting on the aircraft to the weight of the aircraft: $n_{lf} \triangleq F_{\text{lift}}/mg$. In wings-level, horizontal flight

where the roll angle and flight-path angle are zero ($\phi = \gamma = 0$), the load factor is equal to 1. From a control perspective, it is useful to consider the load factor because it represents the force that the aircraft experiences during climbing and turning maneuvers. Although the load factor is a dimensionless number, it is often referred to by the number of “ g ’s” that an aircraft experiences in flight. By controlling the load factor as a state, we can ensure that the aircraft is always given commands that are within its structural capability. Taking into account the definition of load factor, Equation (9.15) becomes

$$\dot{\gamma} = \frac{g}{V_g} (n_{lf} \cos \phi - \cos \gamma). \quad (9.16)$$

We note that in a constant climb, when $\dot{\gamma} = 0$, the load factor can be expressed as

$$n_{lf} = \frac{\cos \gamma}{\cos \phi}. \quad (9.17)$$

This expression will be used in Section 9.4.

9.3 KINEMATIC GUIDANCE MODELS

In this section we summarize several different kinematic guidance models for MAVs. The guidance models that we derive will assume the presence of wind. Wind can be tricky to model correctly using a kinematic model because it introduces aerodynamic forces on the aircraft that are expressed in the dynamic model in terms of the airspeed, angle of attack, and side slip angle, as explained in Chapter 4. The velocity vector can be expressed in terms of the airspeed, heading, and air-mass-referenced flight-path angle, as in Equation (9.8), or in terms of the ground speed, course, and flight-path angle, as in Equation (9.7). However, we typically control the airspeed, the course angle, and the flight-path angle. Therefore, if in the simulation, we directly propagate airspeed, course angle, and air-mass referenced flight path angle in in Equation (9.7), then we will use Equations (2.11) through (2.13) to solve for ground speed, heading, and the flight path angle.

The first guidance model that we will consider assumes that the autopilot controls airspeed, altitude, and heading angle. The corresponding equations

of motion do not include the flight-path angle and are given by

$$\begin{aligned}\dot{p}_n &= V_a \cos \psi + w_n \\ \dot{p}_e &= V_a \sin \psi + w_e \\ \ddot{\chi} &= b_{\dot{\chi}}(\dot{\chi}^c - \dot{\chi}) + b_{\chi}(\chi^c - \chi) \\ \ddot{h} &= b_{\dot{h}}(\dot{h}^c - \dot{h}) + b_h(h^c - h) \\ \dot{V}_a &= b_{V_a}(V_a^c - V_a),\end{aligned}\tag{9.18}$$

where the inputs are the commanded altitude h^c , the commanded airspeed V_a^c , and the commanded course χ^c , and ψ is given by Equation (2.12), with $\gamma_a = 0$.

Alternatively, it is common to consider the roll angle as the input command and to control the heading through roll by using the coordinated-turn condition given in Equation (9.10). In that case, the kinematic equations become

$$\begin{aligned}\dot{p}_n &= V_a \cos \psi + w_n \\ \dot{p}_e &= V_a \sin \psi + w_e \\ \dot{\chi} &= \frac{g}{V_g} \tan \phi \\ \ddot{h} &= b_{\dot{h}}(\dot{h}^c - \dot{h}) + b_h(h^c - h) \\ \dot{V}_a &= b_{V_a}(V_a^c - V_a) \\ \dot{\phi} &= b_\phi(\phi^c - \phi),\end{aligned}\tag{9.19}$$

where ϕ^c is the commanded roll angle, ψ is given by Equation (2.12), and V_g is given by Equation (2.13), with $\gamma_a = 0$.

For the longitudinal motion, altitude is often controlled indirectly through the flight-path angle. In that case, we have

$$\begin{aligned}\dot{p}_n &= V_a \cos \psi \cos \gamma_a + w_n \\ \dot{p}_e &= V_a \sin \psi \cos \gamma_a + w_e \\ \dot{h} &= V_a \sin \gamma_a - w_d \\ \dot{\chi} &= \frac{g}{V_g} \tan \phi \\ \dot{\gamma} &= b_\gamma(\gamma^c - \gamma) \\ \dot{V}_a &= b_{V_a}(V_a^c - V_a) \\ \dot{\phi} &= b_\phi(\phi^c - \phi),\end{aligned}\tag{9.20}$$

where γ^c is the commanded (inertial referenced) flight-path angle, and where γ_a , ψ , and V_g are given by Equations (2.11), (2.12), and (2.13), respectively.

Some autopilots command the load factor instead of the flight-path angle. Using Equation (9.16), a kinematic model that represents this situation is given by

$$\begin{aligned}\dot{p}_n &= V_a \cos \psi \cos \gamma_a + w_n \\ \dot{p}_e &= V_a \sin \psi \cos \gamma_a + w_e \\ \dot{h} &= V_a \sin \gamma_a - w_d \\ \dot{\chi} &= \frac{g}{V_g} \tan \phi \\ \dot{\gamma} &= \frac{g}{V_g} (n_{lf}^c \cos \phi - \cos \gamma) \\ \dot{V}_a &= b_{V_a} (V_a^c - V_a) \\ \dot{\phi} &= b_\phi (\phi^c - \phi) \\ \dot{n}_{lf} &= b_n (n_{lf}^c - n_{lf}),\end{aligned}\tag{9.21}$$

where n_{lf}^c is the commanded load factor, and where γ_a , ψ , and V_g are given by Equations (2.11), (2.12), and (2.13) respectively.

9.4 DYNAMIC GUIDANCE MODEL

The reduced-order guidance models derived in the previous section are based on kinematic relations between positions and velocities. Additionally, they employ first-order differential equations to model the closed-loop response of commanded states. In these equations, we took advantage of the conditions for the coordinated turn to eliminate the lift force from the equations of motion. Furthermore, we assumed that airspeed was a controlled quantity and therefore did not perform a force balance along the body-fixed \mathbf{i}_b axis. In this section, we will derive an alternative set of equations of motion commonly encountered in the literature that utilize relationships drawn from free-body diagrams. Lift, drag, and thrust forces are evident in these dynamic equations.

Figure 9.2 shows a free-body diagram for a MAV climbing at flight-path angle γ and bank angle ϕ . Applying Newton's second law along the \mathbf{i}_b axis and rearranging gives

$$\dot{V}_g = \frac{F_{\text{thrust}}}{m} - \frac{F_{\text{drag}}}{m} - g \sin \gamma,$$

where F_{thrust} is the thrust and F_{drag} is the drag. It is interesting to note that we arrived at this exact equation in the process of deriving transfer functions from the full nonlinear equations of motion in Chapter 5, Equation (5.18).

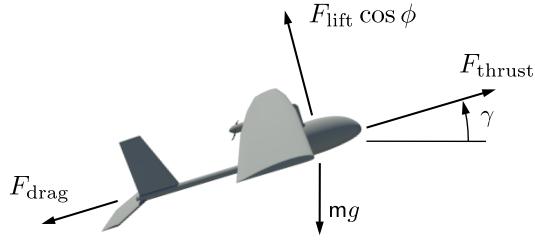


Figure 9.2: Free-body diagram indicating external forces on the UAV along the \mathbf{i}_b axis. The UAV is assumed to be at a roll angle of ϕ .

The course angle can be expressed in terms of lift by combining the coordinated turn equation in (9.10) with expression (9.17) for the load factor to get

$$\dot{\chi} = \frac{g}{V_g} \tan \phi = \frac{g}{V_g} \frac{\sin \phi}{\cos \gamma} n = \frac{F_{\text{lift}}}{m V_g} \frac{\sin \phi}{\cos \gamma}.$$

Similarly, Equation (9.15) expresses the flight-path angle in terms of lift.

Combining these dynamic equations with the kinematic equations relating Cartesian position and velocity gives the following alternative equations of motion:

$$\begin{aligned}\dot{p}_n &= V_g \cos \chi \cos \gamma \\ \dot{p}_e &= V_g \sin \chi \cos \gamma \\ \dot{h} &= V_g \sin \gamma \\ \dot{V}_g &= \frac{F_{\text{thrust}}}{m} - \frac{F_{\text{drag}}}{m} - g \sin \gamma \\ \dot{\chi} &= \frac{F_{\text{lift}}}{m V_g} \frac{\sin \phi}{\cos \gamma} \\ \dot{\gamma} &= \frac{F_{\text{lift}}}{m V_g} \cos \phi - \frac{g}{V_g} \cos \gamma.\end{aligned}\quad (9.22)$$

The control variables are thrust, lift coefficient, and bank angle $[F_{\text{thrust}}, C_L, \phi]^\top$. Lift and drag are given by

$$\begin{aligned}F_{\text{lift}} &= \frac{1}{2} \rho V_a^2 S C_L \\ F_{\text{drag}} &= \frac{1}{2} \rho V_a^2 S C_D,\end{aligned}$$

with $C_D = C_{D_0} + K C_L^2$ [?]. The induced drag factor K can be determined

from the aerodynamic efficiency, which is defined as

$$E_{\max} \triangleq \left(\frac{F_{\text{lift}}}{F_{\text{drag}}} \right)_{\max},$$

and the zero-lift drag coefficient C_{D_0} , using the expression

$$K = \frac{1}{4E_{\max}^2 C_{D_0}}.$$

The popularity of this point-mass model is likely due to the fact that it models aircraft behavior in response to inputs that a pilot commonly controls: engine thrust, lift from the lifting surfaces, and bank angle as observed using the attitude indicator. In the absence of wind, $V_g = V_a$, $\gamma = \gamma_a$ and $\chi = \psi$ so that Equation (9.22) can be expressed as

$$\begin{aligned}\dot{p}_n &= V_a \cos \psi \cos \gamma \\ \dot{p}_e &= V_a \sin \psi \cos \gamma \\ \dot{h} &= V_a \sin \gamma \\ \dot{V}_a &= \frac{F_{\text{thrust}}}{m} - \frac{F_{\text{drag}}}{m} - g \sin \gamma \\ \dot{\psi} &= \frac{F_{\text{lift}} \sin \phi}{m V_a \cos \gamma} \\ \dot{\gamma} &= \frac{F_{\text{lift}}}{m V_a} \cos \phi - \frac{g}{V_a} \cos \gamma.\end{aligned}\tag{9.23}$$

NEW MATERIAL:

9.5 THE DUBINS AIRPLANE MODEL

Unmanned aircraft, particularly smaller systems, fly at relatively low air-speeds 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 [29]. 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 [30].

Figure 9.3 depicts a UAV flying with airspeed V , heading angle ψ and flight-path angle γ . Denoting the inertial position of the UAV as $(r_n, r_e, r_d)^\top$, 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 $V = \|\mathbf{v}\|$.

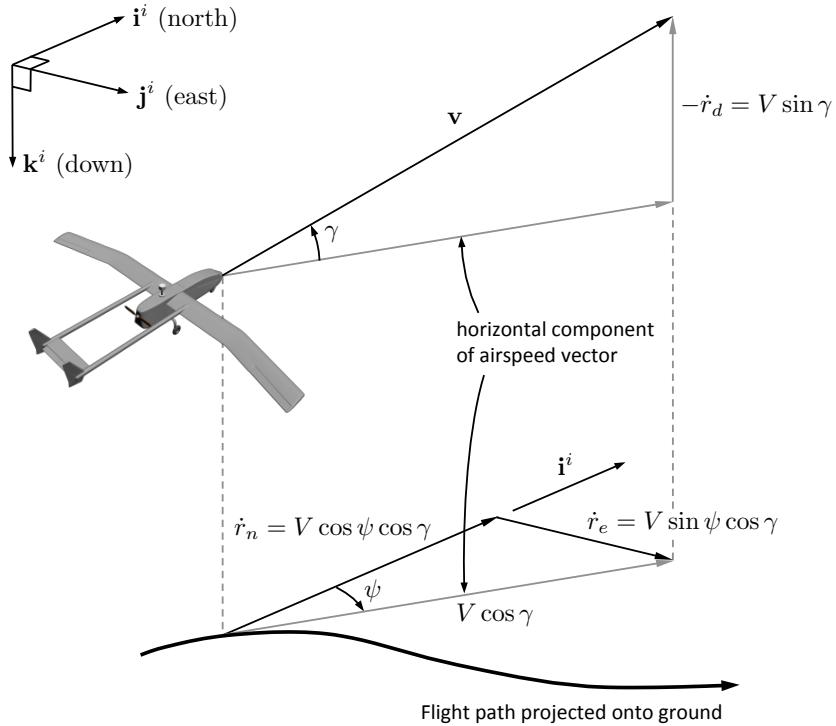


Figure 9.3: Graphical representation of aircraft kinematic model.

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 [30]

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

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

$$\begin{aligned}\dot{r}_n &= V \cos \psi \cos \gamma^c \\ \dot{r}_e &= V \sin \psi \cos \gamma^c \\ \dot{r}_d &= -V \sin \gamma^c \\ \dot{\psi} &= \frac{g}{V} \tan \phi^c.\end{aligned}\tag{9.24}$$

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

$$|\phi^c| \leq \bar{\phi} \tag{9.25}$$

$$|\gamma^c| \leq \bar{\gamma}. \tag{9.26}$$

The kinematic model given by (9.24) with the input constraints (9.25) and (9.26) 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

$$\begin{aligned}\dot{r}_n &= V \cos \psi \\ \dot{r}_e &= V \sin \psi \\ \dot{r}_d &= u_1 \quad |u_1| \leq 1 \\ \dot{\psi} &= u_2 \quad |u_2| \leq 1.\end{aligned}\tag{9.27}$$

Although (9.24) is similar to (9.27), 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 (9.24) 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 high level path planning and path following control design. In-depth discussions of aircraft dynamic models can be found in [18, 1, 6, 7].

9.6 CHAPTER SUMMARY

The objective of this chapter is to present high-level design models for the guidance loops. The guidance models are derived from the six-degree-of-freedom model, kinematic relations, and force balance equations. Kinematic design models are given in Equations (9.18) through (9.21). A dynamic design model often found in the literature is given in Equation (9.23).

NOTES AND REFERENCES

Material supporting the development of the guidance models discussed in this chapter can be found in [15, 18, 17, ?]. The discussion of the air-mass-referenced flight-path angle is from a Boeing circular obtained at www.boeing.com. The derivation of accelerating climb in Section 9.2.2 draws on the discussion in [15, p. 227–228].

9.7 DESIGN PROJECT

The objective of the assignment in this chapter is to estimate the autopilot constants b_* and to develop a reduced-order Simulink model 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 (9.18) and (9.19).

- 9.1 Create a Simulink S-function that implements the model given in Equation (9.18) 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_i , 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. You may want to use the Simulink file `mavsim_chap9.mdl` and the Matlab function `guidance_model.m` located on the website.
- 9.2 Modify your autopilot function so that it uses the commanded roll angle ϕ^c as an input instead of the commanded course χ^c . Create a Simulink S-function that implements the model given in Equation (9.19) and insert it in your MAV simulator. For different inputs ϕ^c , h^c , and V_a^c , compare the output of the two models, and tune the autopilot coefficients b_* to obtain similar behavior. You may need to re-tune the autopilot gains obtained from the previous chapter. Using

the simulation under zero-wind conditions, find the achievable minimum turn radius R_{\min} of the MAV when the commanded roll angle is $\phi^c = 30$ degrees.

Chapter Ten

Straight-line and Orbit Following

This chapter develops guidance laws for tracking straight-line segments and for tracking constant-altitude circular orbits. Chapter 11 will discuss techniques for combining straight-line segments and circular orbits to track more complex paths, and Chapter 12 will describe techniques for path planning through obstacle fields. In the context of the architectures shown in Figures 1.1 and 1.2, this chapter describes algorithms for the path following block. The primary challenge in tracking straight-line segments and circular orbits is wind, which is almost always present. For small unmanned aircraft, wind speeds are commonly 20 to 60 percent of the desired airspeed. Effective path-tracking strategies must overcome the effect of this ever-present disturbance. For most fixed-wing MAVs, the minimum turn radius is in the range of 10 to 50 m. This places a fundamental limit on the spatial frequency of paths that can be tracked. Thus, it is important that the path-tracking algorithms utilize the full capability of the MAV.

Implicit in the notion of trajectory tracking is that the vehicle is commanded to be at a particular location at a specific time and that the location typically varies in time, thus causing the vehicle to move in the desired fashion. With fixed-wing aircraft, the desired position is constantly moving (at the desired ground speed). The approach of tracking a moving point can result in significant problems for MAVs if disturbances, such as those due to wind, are not properly accounted for. If the MAV is flying into a strong wind (relative to its commanded ground speed), the progression of the trajectory point must be slowed accordingly. Similarly, if the MAV is flying downwind, the speed of the tracking point must be increased to keep it from overrunning the desired position. Given that wind disturbances vary and are often not easily predicted, trajectory tracking can be challenging in anything other than calm conditions.

Rather than using a trajectory tracking approach, this chapter focuses on path following, where the objective is to be *on the path* rather than at a certain point at a particular time. With path following, the time dependence of the problem is removed. For this chapter, we will assume that the controlled MAV is modeled by the guidance model given in Equation (9.18). Our objective is to develop a method for accurate path following in the presence of wind. For a given airframe, there is an optimal airspeed for which the air-

frame is the most aerodynamically efficient, and to conserve fuel the MAV should maintain this airspeed. Accordingly, in this chapter we will assume that the MAV is moving with a constant airspeed V_a .

10.1 STRAIGHT-LINE PATH FOLLOWING

A straight-line path is described by two vectors in \mathbb{R}^3 , namely

$$\mathcal{P}_{\text{line}}(\mathbf{r}, \mathbf{q}) = \left\{ \mathbf{x} \in \mathbb{R}^3 : \mathbf{x} = \mathbf{r} + \lambda \mathbf{q}, \lambda \in \mathbb{R} \right\},$$

where $\mathbf{r} \in \mathbb{R}^3$ is the origin of the path, and $\mathbf{q} \in \mathbb{R}^3$ is a unit vector whose direction indicates the desired direction of travel. Figure 10.1 shows a top-down or lateral view of $\mathcal{P}_{\text{line}}(\mathbf{r}, \mathbf{q})$, and Figure 10.2 shows a side or longitudinal view. The course angle of $\mathcal{P}_{\text{line}}(\mathbf{r}, \mathbf{q})$, as measured from north is given by

$$\chi_q \stackrel{\triangle}{=} \text{atan}2 \frac{q_e}{q_n}, \quad (10.1)$$

where $\mathbf{q} = (q_n \ q_e \ q_d)^\top$ expresses the north, east, and down components of the unit direction vector.

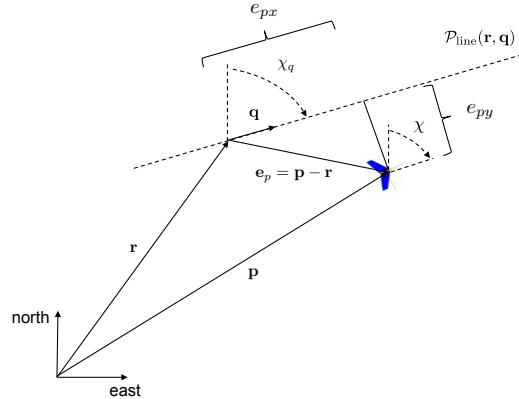


Figure 10.1: The configuration of the MAV indicated by (\mathbf{p}, χ) , and the straight-line path indicated by $\mathcal{P}_{\text{line}}(\mathbf{r}, \mathbf{q})$.

The path-following problem is most easily solved in a frame relative to the straight-line path. Selecting \mathbf{r} as the center of the path-frame, with the x -axis aligned with the projection of \mathbf{q} onto the local north-east plane, the z -axis aligned with the inertial z -axis, and the y -axis selected to create a

right-handed coordinate system, then

$$\mathcal{R}_i^P \triangleq \begin{pmatrix} \cos \chi_q & \sin \chi_q & 0 \\ -\sin \chi_q & \cos \chi_q & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

is the transformation from the inertial frame to the path frame, and

$$\mathbf{e}_p = \begin{pmatrix} e_{px} \\ e_{py} \\ e_{pz} \end{pmatrix} \triangleq \mathcal{R}_i^P (\mathbf{p}^i - \mathbf{r}^i)$$

is the relative path error expressed in the path frame. The relative error dynamics in the north-east inertial plane, expressed in the path frame, are given by

$$\begin{aligned} \begin{pmatrix} \dot{e}_{px} \\ \dot{e}_{py} \end{pmatrix} &= \begin{pmatrix} \cos \chi_q & \sin \chi_q \\ -\sin \chi_q & \cos \chi_q \end{pmatrix} \begin{pmatrix} V_g \cos \chi \\ V_g \sin \chi \end{pmatrix} \\ &= V_g \begin{pmatrix} \cos(\chi - \chi_q) \\ \sin(\chi - \chi_q) \end{pmatrix}. \end{aligned} \quad (10.2)$$

For path following, we desire to regulate the cross-track error e_{py} to zero by commanding the course angle. The relevant dynamics are therefore given by

$$\dot{e}_{py} = V_g \sin(\chi - \chi_q) \quad (10.3)$$

$$\ddot{\chi} = b_{\dot{\chi}}(\dot{\chi}^c - \dot{\chi}) + b_\chi(\chi^c - \chi). \quad (10.4)$$

The lateral straight-line path following problem is to select χ^c so that $e_{py} \rightarrow 0$ when χ_q is known.

The geometry for straight-line path following in the longitudinal direction is shown in Figure 10.2. To calculate the desired altitude, it is necessary to project the relative path error vector onto the vertical plane containing the path direction vector \mathbf{q} as shown in Figure 10.2(a). We denote the projection of \mathbf{e}_p as \mathbf{s} . Referring to the vertical plane containing the path shown in Figure 10.2(b) and using similar triangles, we have the relationship

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

The projection \mathbf{s} of the relative error vector is defined as

$$\begin{aligned} \mathbf{s}^i &= \begin{pmatrix} s_n \\ s_e \\ s_d \end{pmatrix} \\ &= \mathbf{e}_p^i - (\mathbf{e}_p^i \cdot \mathbf{n})\mathbf{n}, \end{aligned}$$

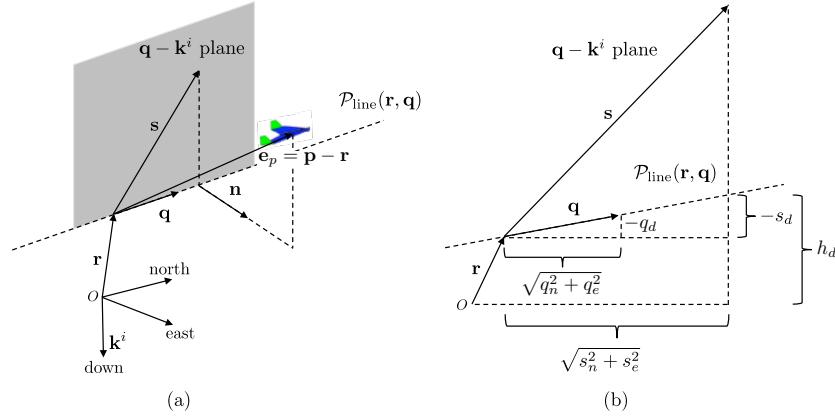


Figure 10.2: Desired altitude calculation for straight-line path following in longitudinal direction.

where

$$\mathbf{e}_p^i = \begin{pmatrix} e_{pn} \\ e_{pe} \\ e_{pd} \end{pmatrix} \triangleq \mathbf{p}^i - \mathbf{r}^i = \begin{pmatrix} p_n - r_n \\ p_e - r_e \\ p_d - r_d \end{pmatrix}$$

and the unit vector normal to the $\mathbf{q} - \mathbf{k}^i$ plane is calculated as

$$\mathbf{n} = \frac{\mathbf{q} \times \mathbf{k}^i}{\|\mathbf{q} \times \mathbf{k}^i\|}.$$

MODIFIED MATERIAL:

From Figure 10.2(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

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

Since the altitude dynamics are given by

$$\ddot{h} = b_h(\dot{h}^c - \dot{h}) + b_h(h^c - h), \quad (10.6)$$

the longitudinal straight-line path following problem is to select h^c so that $h \rightarrow h_d(\mathbf{r}, \mathbf{p}, \mathbf{q})$.

10.1.1 Longitudinal Guidance Strategy for Straight-line Following

In this section we specify the longitudinal guidance law for tracking the altitude portion of the waypoint path. With the desired altitude specified

by Equation (10.5) and the dynamics modeled by Equation (10.6), we will show that letting $h^c = h_d(\mathbf{r}, \mathbf{p}, \mathbf{q})$ and utilizing the altitude state machine of Figure ??, good path-following performance will result, with zero steady-state error in altitude for straight-line paths.

With respect to the altitude state machine, we will assume that the control laws in the climb and descend zones will cause the MAV to climb or descend into the altitude-hold zone. In the altitude-hold zone, pitch attitude is used to control the altitude of the MAV, as shown in Figure ???. Assuming that successive loop closure has been properly implemented, Figure 6.14 shows a simplified representation of the outer-loop dynamics, which has the transfer function

$$\frac{h}{h^c} = \frac{b_h s + b_h}{s^2 + b_h s + b_h}.$$

Defining the altitude error as

$$e_h \triangleq h - h_d(\mathbf{r}, \mathbf{p}, \mathbf{q}) = h - h^c,$$

we get that

$$\begin{aligned} \frac{e_h}{h^c} &= 1 - \frac{h}{h^c} \\ &= \frac{s^2}{s^2 + b_h s + b_h}. \end{aligned}$$

By applying the final value theorem, we find that

$$\begin{aligned} e_{h,ss} &= \lim_{s \rightarrow 0} s \frac{s^2}{s^2 + b_h s + b_h} h^c \\ &= 0, \quad \text{for } h^c = \frac{H_0}{s}, \frac{H_0}{s^2}. \end{aligned}$$

The analysis in Chapter 6 also shows that constant disturbances are rejected. Thus, utilizing the altitude state machine, we can track constant-altitude and inclined straight-line paths with zero steady-state altitude error provided we do not exceed the physical capabilities of the MAV and disturbances (such as vertical components of wind) are zero or constant in magnitude.

10.1.2 Lateral Guidance Strategy for Straight-line Following

The objective in this section is to select the commanded course angle χ^c in Equation (10.4) so that e_{py} in Equation (10.3) is driven to zero asymptotically. The strategy in this section will be to construct a desired course angle

at every spatial point relative to the straight-line path that results in the MAV moving toward the path. The set of desired course angles at every point will be called a vector field because the desired course angle specifies a vector (relative to the straight line) with a magnitude of unity. Figure 10.3 depicts an example vector field for straight-line path following. The objective is to

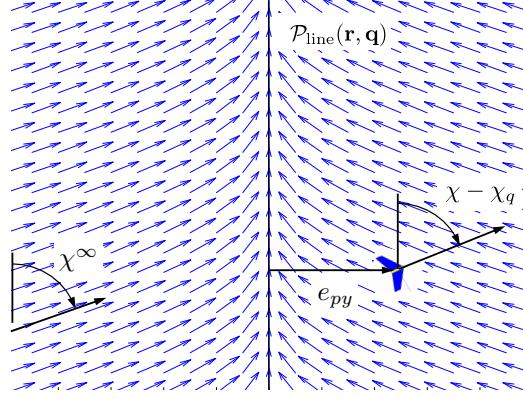


Figure 10.3: Vector field for straight-line path following. Far away from the waypoint path, the vector field is directed with an angle χ^∞ from the perpendicular to the path.

construct the vector field so that when e_{py} is large, the MAV is directed to approach the path with course angle $\chi^\infty \in (0, \frac{\pi}{2}]$, and so that as e_{py} approaches zero, the desired course also approaches zero. Toward that end, we define the desired course of the MAV as

$$\chi_d(e_{py}) = -\chi^\infty \frac{2}{\pi} \tan^{-1}(k_{\text{path}} e_{py}), \quad (10.7)$$

where k_{path} is a positive constant that influences the rate of the transition from χ^∞ to zero. Figure 10.4 shows how the choice of k_{path} affects the rate of transition. Large values of k_{path} result in short, abrupt transitions, while small values of k_{path} cause long, smooth transitions in the desired course.

If χ^∞ is restricted to be in the range $\chi^\infty \in (0, \frac{\pi}{2}]$, then clearly

$$-\frac{\pi}{2} < \chi^\infty \frac{2}{\pi} \tan^{-1}(k_{\text{path}} e_{py}) < \frac{\pi}{2}$$

for all values of e_{py} . Therefore, since $\tan^{-1}(\cdot)$ is an odd function and $\sin(\cdot)$ is odd over $(-\frac{\pi}{2}, \frac{\pi}{2})$, we can use the Lyapunov function $W(e_{py}) = \frac{1}{2}e_{py}^2$ to argue that if $\chi = \chi_q + \chi^d(e_{py})$, then $e_{py} \rightarrow 0$ asymptotically, since

$$\dot{W} = -V_a e_{py} \sin \left(\chi^\infty \frac{2}{\pi} \tan^{-1}(k_{\text{path}} e_{py}) \right)$$

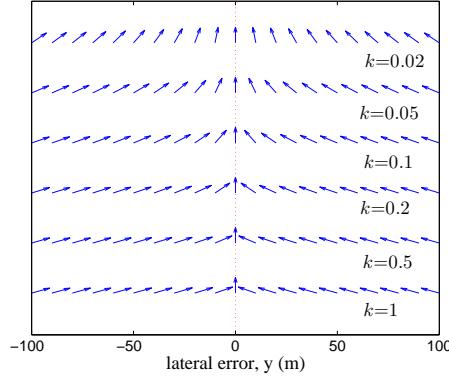


Figure 10.4: Vector fields for various values of k_{path} . Large values of k_{path} yield abrupt transitions from χ^∞ to zero, while small values of k_{path} give smooth transitions.

is less than zero for $e_{py} \neq 0$. The command for lateral path following is therefore given by

$$\chi^c(t) = \chi_q - \chi^\infty \frac{2}{\pi} \tan^{-1}(k_{\text{path}} e_{py}(t)). \quad (10.8)$$

Before moving to orbit following, we note that using Equation (10.8) may result in undesirable behavior if χ_q is computed directly from Equation (10.1), where atan2 returns an angle between $\pm\pi$. As an example, consider the scenario shown in Figure 10.5, where χ_q is a positive number slightly smaller than $+\pi$. Since the current course is negative, Equation (10.8) will cause the MAV to turn right to align with the waypoint path. As an alternative, if χ_q is expressed as a negative angle slightly less than $-\pi$, then the MAV will turn left to align with the waypoint path. To alleviate this problem, χ_q should be computed as

$$\chi_q = \text{atan2}(q_e, q_n) + 2\pi m,$$

where $m \in \mathcal{N}$ is selected so that $-\pi \leq \chi_q - \chi \leq \pi$, and atan2 is a four-quadrant \tan^{-1} function.

10.2 ORBIT FOLLOWING

An orbit path is described by a center $\mathbf{c} \in \mathbb{R}^3$, a radius $\rho \in \mathbb{R}$, and a direction $\lambda \in \{-1, 1\}$, as

$$\mathcal{P}_{\text{orbit}}(\mathbf{c}, \rho, \lambda) = \left\{ \mathbf{r} \in \mathbb{R}^3 : \mathbf{r} = \mathbf{c} + \lambda \rho \begin{pmatrix} \cos \varphi & \sin \varphi & 0 \end{pmatrix}^\top, \varphi \in [0, 2\pi) \right\},$$

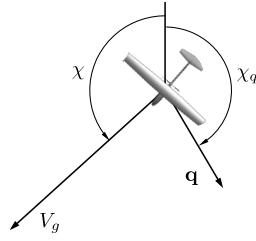


Figure 10.5: The calculation of χ_q needs to account for the current course angle of the MAV. In this scenario, the MAV should turn left to align with the waypoint path, but if χ_q is computed with atan2 , the angle will be a positive number slightly smaller than $+\pi$, which will cause the MAV to turn right to align with the waypoint path.

where $\lambda = 1$ signifies a clockwise orbit and $\lambda = -1$ signifies a counter-clockwise orbit. We assume that the center of the orbit is expressed in inertial coordinates so that $\mathbf{c} = (c_n, c_e, c_d)^\top$, where $-c_d$ represents the desired altitude of the orbit and to maintain altitude we let $h^c = -c_d$

Figure 10.6 shows a top-down view of an orbital path. The guidance

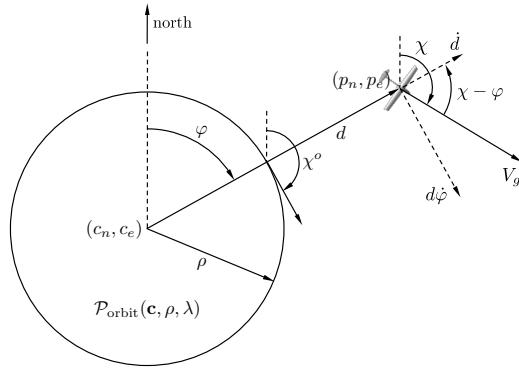


Figure 10.6: Orbital path with center (c_n, c_e) , and radius ρ . The distance from the orbit center to the MAV is d , and the angular position of the MAV relative to the orbit is φ .

strategy for orbit following is best derived in polar coordinates. Let d be the lateral distance from the desired center of the orbit to the MAV, and let φ be the phase angle of the relative position, as shown in Figure 10.6. The constant-altitude MAV dynamics in polar coordinates can be derived by rotating the differential equations that describe the motion of the MAV

in the north and east directions,

$$\begin{pmatrix} \dot{p}_n \\ \dot{p}_e \end{pmatrix} = \begin{pmatrix} V_g \cos \chi \\ V_g \sin \chi \end{pmatrix},$$

through the phase angle φ so that the equations of motion represent the MAV motion in the normal and tangential directions to the orbit:

$$\begin{aligned} \begin{pmatrix} \dot{d} \\ d\dot{\varphi} \end{pmatrix} &= \begin{pmatrix} \cos \varphi & \sin \varphi \\ -\sin \varphi & \cos \varphi \end{pmatrix} \begin{pmatrix} \dot{p}_n \\ \dot{p}_e \end{pmatrix} \\ &= \begin{pmatrix} \cos \varphi & \sin \varphi \\ -\sin \varphi & \cos \varphi \end{pmatrix} \begin{pmatrix} V_g \cos \chi \\ V_g \sin \chi \end{pmatrix} \\ &= \begin{pmatrix} V_g \cos(\chi - \varphi) \\ V_g \sin(\chi - \varphi) \end{pmatrix}. \end{aligned}$$

These expressions can also be derived from the geometry illustrated in Figure 10.6. The MAV dynamics in polar coordinates are therefore given by

$$\dot{d} = V_g \cos(\chi - \varphi) \quad (10.9)$$

$$\dot{\varphi} = \frac{V_g}{d} \sin(\chi - \varphi) \quad (10.10)$$

$$\ddot{\chi} = -b_{\chi}\dot{\chi} + b_{\chi}(\chi^c - \chi). \quad (10.11)$$

As shown in Figure 10.6, for a clockwise orbit, the desired course angle when the MAV is located on the orbit is given by $\chi^o = \varphi + \pi/2$. Similarly, for a counterclockwise orbit, the desired angle is given by $\chi^o = \varphi - \pi/2$. Therefore, in general we have

$$\chi^o = \varphi + \lambda \frac{\pi}{2}.$$

The control objective is to drive $d(t)$ to the orbit radius ρ and to drive the course angle $\chi(t)$ to χ^o in the presence of wind.

Our approach to orbit following is similar to the ideas developed in Section 10.1.2. The strategy is to construct a desired course field that moves the MAV onto the orbit $\mathcal{P}_{\text{orbit}}(\mathbf{c}, \rho, \lambda)$. When the distance between the MAV and the center of the orbit is large, it is desirable for the MAV to fly toward the orbit center. In other words, when $d \gg \rho$, the desired course is

$$\chi_d \approx \chi^o + \lambda \frac{\pi}{2},$$

and when $d = \rho$, the desired course is $\chi_d = \chi^o$. Therefore, a candidate course field is given by

$$\chi_d(d - \rho, \lambda) = \chi^o + \lambda \tan^{-1} \left(k_{\text{orbit}} \left(\frac{d - \rho}{\rho} \right) \right), \quad (10.12)$$

where $k_{\text{orbit}} > 0$ is a constant that specifies the rate of transition from $\lambda\pi/2$ to zero. This expression for χ_d is valid for all values of $d \geq 0$.

We can again use the Lyapunov function $W = \frac{1}{2}(d - \rho)^2$ to argue that if $\chi = \chi_d$, then the tracking objective is satisfied. Differentiating W along the system trajectory gives

$$\dot{W} = -V_g(d - \rho) \sin \left(\tan^{-1} \left(k_{\text{orbit}} \left(\frac{d - \rho}{\rho} \right) \right) \right),$$

which is negative definite since the argument of \sin is in the set $(-\pi/2, \pi/2)$ for all $d > 0$, implying that $d \rightarrow \rho$ asymptotically. The course command for orbit following is therefore given by

$$\chi^c(t) = \varphi + \lambda \left[\frac{\pi}{2} + \tan^{-1} \left(k_{\text{orbit}} \left(\frac{d - \rho}{\rho} \right) \right) \right]. \quad (10.13)$$

Similar to the computation of the path angle χ_q , if the angular position in the orbit φ is computed to be between $\pm\pi$, then there will be a sudden jump of 2π in the commanded course as the MAV transitions from $\varphi = \pi$ to $\varphi = -\pi$. To alleviate this problem, φ should be computed as

$$\varphi = \text{atan2}(p_e - c_e, p_n - c_n) + 2\pi m,$$

where $m \in \mathcal{N}$ is selected so that $-\pi \leq \varphi - \chi \leq \pi$.

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

$$\phi_{ff} = \lambda \tan^{-1} \left(\frac{V_a^2}{gR} \right). \quad (10.14)$$

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

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

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

$$\phi_{ff} = \lambda \tan^{-1} \left(\frac{\left((w_n \cos \chi + w_e \sin \chi) + \sqrt{V_a^2 - (w_n \sin \chi - w_e \cos \chi)^2 - w_d^2} \right)^2}{g R \sqrt{\frac{V_a^2 - (w_n \sin \chi - w_e \cos \chi)^2 - w_d^2}{V_a^2 - w_d^2}}} \right). \quad (10.15)$$

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 (10.15) simplifies to Equation (10.14).

NEW MATERIAL:

10.3 3D VECTOR-FIELD PATH FOLLOWING

This section shows how to develop guidance laws to ensure that the kinematic model (9.24) 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 Methodology

The guidance strategy will use the vector-field methodology proposed in [31], 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 [31] the desired velocity vector $\mathbf{u}' \in \mathbb{R}^3$ can be chosen as

$$\mathbf{u}' = -K_1 \frac{\partial V}{\partial \mathbf{r}} + K_2 \frac{\partial \alpha_1}{\partial \mathbf{r}} \times \frac{\partial \alpha_2}{\partial \mathbf{r}}, \quad (10.16)$$

where K_1 and K_2 are symmetric tuning matrices. It is shown in [31] that the dynamics $\dot{\mathbf{r}} = \mathbf{u}'$ where \mathbf{u}' is given by Equation (10.16), 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 (10.16) 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

$$\mathbf{u} = V \frac{\mathbf{u}'}{\|\mathbf{u}'\|}. \quad (10.17)$$

Fortunately, the stability proof in [31] is still valid when \mathbf{u}' is normalized as in Equation (10.17).

Setting the NED components of the velocity of the Dubins airplane model given in Equation (9.24) 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

$$\begin{aligned} \gamma^c &= -\text{sat}_{\bar{\gamma}} \left[\sin^{-1} \left(\frac{u_3}{V} \right) \right] \\ \psi^d &= \text{atan2}(u_2, u_1), \end{aligned} \tag{10.18}$$

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

$$\phi^c = \text{sat}_{\bar{\phi}} \left[k_\phi (\psi^d - \psi) \right], \tag{10.19}$$

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

$$\mathcal{P}_{\text{line}}(\mathbf{c}_\ell, \psi_\ell, \gamma_\ell) = \{\mathbf{r} \in \mathbb{R}^3 : \mathbf{r} = \mathbf{c}_\ell + \sigma \mathbf{q}_\ell, \sigma \in \mathbb{R}\}. \quad (10.20)$$

A unit vector that is perpendicular to the longitudinal plane defined by \mathbf{q}_ℓ 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}_ℓ 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

$$\alpha_{\text{lon}}(\mathbf{r}) \triangleq \mathbf{n}_{\text{lon}}^\top (\mathbf{r} - \mathbf{c}_\ell) = 0 \quad (10.21)$$

$$\alpha_{\text{lat}}(\mathbf{r}) \triangleq \mathbf{n}_{\text{lat}}^\top (\mathbf{r} - \mathbf{c}_\ell) = 0. \quad (10.22)$$

Figure 10.7 shows \mathbf{q}_ℓ , \mathbf{c}_ℓ , and the surfaces defined by $\alpha_{\text{lon}}(\mathbf{r}) = 0$ and $\alpha_{\text{lat}}(\mathbf{r}) = 0$.

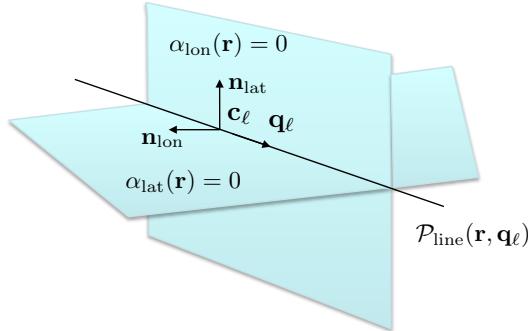


Figure 10.7: This figure shows how the straight-line path $\mathcal{P}_{\text{line}}(\mathbf{c}_\ell, \psi_\ell, \gamma_\ell)$ is defined by the intersection of the two surfaces given by $\alpha_{\text{lon}}(\mathbf{r}) = 0$ and $\alpha_{\text{lat}}(\mathbf{r}) = 0$.

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

$$\mathbf{u}'_{\text{line}} = K_1 \left(\mathbf{n}_{\text{lon}} \mathbf{n}_{\text{lon}}^\top + \mathbf{n}_{\text{lat}} \mathbf{n}_{\text{lat}}^\top \right) (\mathbf{r} - \mathbf{c}_\ell) + K_2 (\mathbf{n}_{\text{lon}} \times \mathbf{n}_{\text{lat}}). \quad (10.23)$$

10.3.3 Helical Paths

A time parameterized helical path is given by

$$\mathbf{r}(t) = \mathbf{c}_h + \begin{pmatrix} R_h \cos(\lambda_h t + \psi_h) \\ R_h \sin(\lambda_h t + \psi_h) \\ -tR_h \tan \gamma_h \end{pmatrix}, \quad (10.24)$$

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 ($N \rightarrow E \rightarrow S \rightarrow W$), and $\lambda_h = -1$ denotes a counter-clockwise helix ($N \rightarrow W \rightarrow S \rightarrow E$), 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 (10.24) needs to be eliminated. Equation (10.24) 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 (10.24) 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

$$\mathcal{P}_{\text{helix}}(\mathbf{c}_h, \psi_h, \lambda_h, R_h, \gamma_h) = \{\mathbf{r} \in \mathbb{R}^3 : \alpha_{\text{cyl}}(\mathbf{r}) = 0 \text{ and } \alpha_{\text{pl}}(\mathbf{r}) = 0\}. \quad (10.25)$$

The two surfaces $\alpha_{\text{cyl}}(\mathbf{r}) = 0$ and $\alpha_{\text{pl}}(\mathbf{r}) = 0$ are shown in Figure 10.8 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.

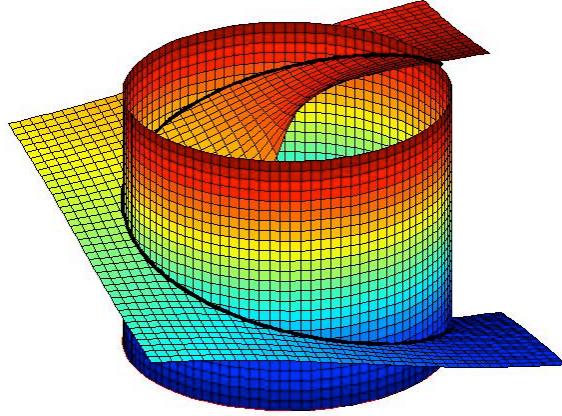


Figure 10.8: A helical path for parameters $\mathbf{c}_h = (0, 0, 0)^\top$, $R_h = 30$ m, $\gamma_h = \frac{15\pi}{180}$ rad, and $\lambda_h = +1$.

The gradients of α_{cyl} and α_{pl} are given by

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

Before normalization, the desired velocity vector is given by

$$\mathbf{u}'_{\text{helix}} = K_1 \left(\alpha_{\text{cyl}} \frac{\partial \alpha_{\text{cyl}}}{\partial \mathbf{r}} + \alpha_{\text{pl}} \frac{\partial \alpha_{\text{pl}}}{\partial \mathbf{r}} \right) + \lambda K_2 \left(\frac{\partial \alpha_{\text{cyl}}}{\partial \mathbf{r}} \times \frac{\partial \alpha_{\text{pl}}}{\partial \mathbf{r}} \right), \quad (10.26)$$

where

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

10.4 CHAPTER SUMMARY

This chapter introduced algorithms for following straight-line paths and circular orbits in the presence of wind. The idea is to construct a heading field that directs the MAV onto the path and is therefore distinctly different from trajectory tracking, where the vehicle would be commanded to follow a time-varying location. The algorithms developed in this chapter are summarized in Algorithms 4 and 5.

Algorithm 4 Straight-line Following: $[h^c, \chi^c] = \text{followStraightLine}(\mathbf{r}, \mathbf{q}, \mathbf{p}, \chi)$

Input: Path definition $\mathbf{r} = (r_n, r_e, r_d)^\top$ and $\mathbf{q} = (q_n, q_e, q_d)^\top$, MAV position $\mathbf{p} = (p_n, p_e, p_d)^\top$, course χ , gains χ_∞ , k_{path} , sample rate T_s .

- 1: Compute commanded altitude using Equation (10.5).
 - 2: $\chi_q \leftarrow \text{atan2}(q_e, q_n)$
 - 3: **while** $\chi_q - \chi < -\pi$ **do**
 - 4: $\chi_q \leftarrow \chi_q + 2\pi$
 - 5: **end while**
 - 6: **while** $\chi_q - \chi > \pi$ **do**
 - 7: $\chi_q \leftarrow \chi_q - 2\pi$
 - 8: **end while**
 - 9: $e_{py} \leftarrow -\sin \chi_q (p_n - r_n) + \cos \chi_q (p_e - r_e)$
 - 10: Compute commanded course angle using Equation (10.8).
 - 11: **return** h^c, χ^c .
-

NOTES AND REFERENCES

The methods described in Sections 10.1 and 10.2 are variations on those described in [29, 32, 33] and are based on the notion of a vector field, which calculates a desired heading based on the distance from the path. A nice extension of [29] is given in [?], which derives general stability conditions for vector-field based methods. The focus is entirely on orbits, but elongated oval orbits and elliptical orbits can be produced. The method in [?], which is based on Lyapunov techniques, could be extended to straight lines.

The notion of vector fields is similar to that of potential fields, which have been widely used as a tool for path planning in the robotics community (see, e.g., [34]). It has also been suggested in [35] that potential fields can be used in UAV navigation for obstacle and collision avoidance applications. The method of [35] provides a way for groups of UAVs to use the gradient

Algorithm 5 Circular Orbit Following: $[h^c, \chi^c] = \text{followOrbit}(\mathbf{c}, \rho, \lambda, \mathbf{p}, \chi)$

Input: Orbit center $\mathbf{c} = (c_n, c_e, c_d)^\top$, radius ρ , and direction λ , MAV position $\mathbf{p} = (p_n, p_e, p_d)^\top$, course χ , gains k_{orbit} , sample rate T_s .

```

1:  $h^c \leftarrow -c_d$ 
2:  $d \leftarrow \sqrt{(p_n - c_n)^2 + (p_e - c_e)^2}$ 
3:  $\varphi \leftarrow \text{atan2}(p_e - c_e, p_n - c_n)$ 
4: while  $\varphi - \chi < -\pi$  do
5:    $\varphi \leftarrow \varphi + 2\pi$ 
6: end while
7: while  $\varphi - \chi > \pi$  do
8:    $\varphi \leftarrow \varphi - 2\pi$ 
9: end while
10: Compute commanded course angle using Equation (10.13).
11: return  $h^c, \chi^c$ 
```

of a potential field to navigate through heavily populated areas safely while still aggressively approaching their targets. Vector fields are different from potential fields in that they do not necessarily represent the gradient of a potential. Rather, the vector field simply indicates a desired direction of travel.

Several approaches have been proposed for UAV trajectory tracking. An approach for tight tracking of curved trajectories is presented in [?]. For straight-line paths, the approach approximates PD control. For curved paths, an additional anticipatory control element that improves the tracking capability is implemented. The approach accommodates the addition of an adaptive element to account for disturbances such as wind. This approach is validated with flight experiments.

Reference [?] describes an integrated approach for developing guidance and control algorithms for autonomous vehicle trajectory tracking. Their approach builds upon the theory of gain scheduling and produces controllers for tracking trajectories that are defined in an inertial reference frame. The approach is illustrated through simulations of a small UAV. Reference [19] presents a path following method for UAVs that provides a constant line of sight between the UAV and an observation target.

10.5 DESIGN PROJECT

The objective of this assignment is to implement Algorithms 4 and 5. Download the sample code for this chapter from the book website and note the addition of two blocks labeled `PathManager` and `PathFollower`. The

output of the path manager is

$$y_{\text{manager}} = \begin{pmatrix} \text{flag} \\ V_a^d \\ \mathbf{r} \\ \mathbf{q} \\ \mathbf{c} \\ \rho \\ \lambda \end{pmatrix},$$

where `flag=1` indicates that $\mathcal{P}_{\text{line}}(\mathbf{r}, \mathbf{q})$ should be followed and `flag=2` indicates that $\mathcal{P}_{\text{orbit}}(\mathbf{c}, \rho, \lambda)$ should be followed, and where V_a^d is the desired airspeed.

- 10.1 Modify `path_follow.m` to implement Algorithms 4 and 5. By modifying `path_manager_chap10.m` test both the straight-line and orbit following algorithms on the guidance model given in Equation (9.18). An example Simulink diagram is given in `mavsim_chap10_model.mdl`. ■
Test your design with significant constant winds (e.g., $w_n = 3$, $w_e = -3$). Tune the gains to get acceptable performance.

Hint: The simulated model of GPS discussed in Section 7.5 contains an unobservable bias. This bias will show up in the simulations. To evaluate and debug your path following algorithm, you may want to turn off the Gauss-Markov bias generator.

- 10.2 Implement the path following algorithms on the full six-DOF simulation of the MAV. An example Simulink diagram is given in `mavsim_chap10.mdl`. ■
Test your design with significant constant winds (e.g., $w_n = 3$, $w_e = -3$). If necessary, tune the gains to get acceptable performance.

Chapter Eleven

Path Manager

In Chapter 10 we developed guidance strategies for following straight-line paths and circular orbits. The objective of this chapter is to describe two simple strategies that combine straight-line paths and orbits to synthesize general classes of paths that are useful for autonomous operation of MAVs. In Section 11.1, we show how the straight-line and orbit guidance strategies can be used to follow a series of waypoints. In Section 11.2, the straight-line and orbit guidance strategies are used to synthesize Dubins paths, which for constant-altitude, constant-velocity vehicles with turning constraints, are time-optimal paths between two configurations. In reference to the architectures shown in Figures 1.1 and 1.2, this chapter describes the path manager.

11.1 TRANSITIONS BETWEEN WAYPOINTS

Define a waypoint path as an ordered sequence of waypoints

$$\mathcal{W} = \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_N\}, \quad (11.1)$$

where $\mathbf{w}_i = (w_{n,i}, w_{e,i}, w_{d,i})^\top \in \mathbb{R}^3$. In this section, we address the problem of switching from one waypoint segment to another. Consider the scenario shown in Figure 11.1 that depicts a MAV tracking the straight-line segment $\overline{\mathbf{w}_{i-1}\mathbf{w}_i}$. Intuitively, when the MAV reaches \mathbf{w}_i , we desire to switch the guidance algorithm so that it will track the straight-line segment $\overline{\mathbf{w}_i\mathbf{w}_{i+1}}$. What is the best method for determining whether the MAV has reached \mathbf{w}_i ? One possible strategy is to switch when the MAV enters a ball around \mathbf{w}_i . 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. However, if there are disturbances like wind or if b is chosen too small or if the segment from \mathbf{w}_{i-1} to \mathbf{w}_i is short and the tracking algorithm has not had time to converge, then the MAV may never enter the b -ball around \mathbf{w}_i .

A better approach, one that is not sensitive to tracking error, is to use a half-plane switching criteria. Given a point $\mathbf{r} \in \mathbb{R}^3$ and a normal vector

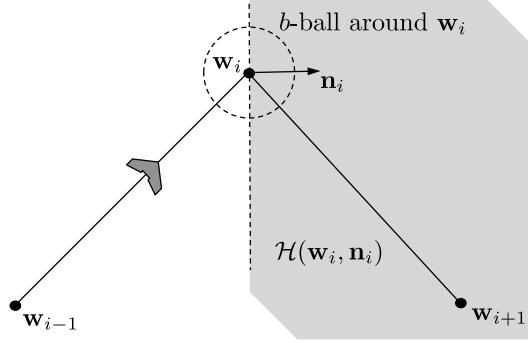


Figure 11.1: When transitioning from one straight-line segment to another, a criterium is needed to indicate when the MAV has completed the first straight-line segment. A possible option is to switch when the MAV enters a b -ball around the transition waypoint. A better option is to switch when the MAV enters the half-plane $\mathcal{H}(w_i, n_i)$.

$\mathbf{n} \in \mathbb{R}^3$, define the half plane

$$\mathcal{H}(\mathbf{r}, \mathbf{n}) \triangleq \left\{ \mathbf{p} \in \mathbb{R}^3 : (\mathbf{p} - \mathbf{r})^\top \mathbf{n} \geq 0 \right\}.$$

Referring to Figure 11.1, we can define the unit vector pointing in the direction of the line $\overline{w_i w_{i+1}}$ as

$$\mathbf{q}_i \triangleq \frac{\mathbf{w}_{i+1} - \mathbf{w}_i}{\|\mathbf{w}_{i+1} - \mathbf{w}_i\|}, \quad (11.2)$$

and accordingly, the unit normal to the 3-D half plane that separates the line $\overline{w_{i-1} w_i}$ from the line $\overline{w_i w_{i+1}}$ is given by

$$\mathbf{n}_i \triangleq \frac{\mathbf{q}_{i-1} + \mathbf{q}_i}{\|\mathbf{q}_{i-1} + \mathbf{q}_i\|}.$$

The MAV tracks the straight-line path from w_{i-1} to w_i until it enters $\mathcal{H}(w_i, n_i)$, at which point it will track the straight-line path from w_i to w_{i+1} . ■

A simple algorithm for following the sequence of waypoints in Equation (11.1) is given in Algorithm 6. The first time that the algorithm is executed, the waypoint pointer is initialized to $i = 2$ Line 2. The MAV will be commanded to follow the straight-line segment $\overline{w_{i-1} w_i}$. The index i is a static variable and retains its value from one execution of the algorithm to the next. Lines 4 and 5 define \mathbf{r} and \mathbf{q} for the current waypoint segment. Line 6 defines the unit vector along the next waypoint path, and Line 7 is a vector that is perpendicular to the half plane that separates $\overline{w_{i-1} w_i}$ from

$\overline{\mathbf{w}_i \mathbf{w}_{i+1}}$. Line 8 checks to see if the half plane defining the next waypoint segment has been reached by the MAV. If it has, then Lines 9 will increment the pointer until reaching the last waypoint segment.

Algorithm 6 Follow Waypoints: $(\mathbf{r}, \mathbf{q}) = \text{followWpp}(\mathcal{W}, \mathbf{p})$

Input: Waypoint path $\mathcal{W} = \{\mathbf{w}_1, \dots, \mathbf{w}_N\}$, MAV position $\mathbf{p} = (p_n, p_e, p_d)^\top$.

Require: $N \geq 3$.

```

1: if New waypoint path  $\mathcal{W}$  is received then
2:   Initialize waypoint index:  $i \leftarrow 2$ .
3: end if
4:  $\mathbf{r} \leftarrow \mathbf{w}_{i-1}$ 
5:  $\mathbf{q}_{i-1} \leftarrow \frac{\mathbf{w}_i - \mathbf{w}_{i-1}}{\|\mathbf{w}_i - \mathbf{w}_{i-1}\|}$ .
6:  $\mathbf{q}_i \leftarrow \frac{\mathbf{w}_{i+1} - \mathbf{w}_i}{\|\mathbf{w}_{i+1} - \mathbf{w}_i\|}$ .
7:  $\mathbf{n}_i \leftarrow \frac{\mathbf{q}_{i-1} + \mathbf{q}_i}{\|\mathbf{q}_{i-1} + \mathbf{q}_i\|}$ 
8: if  $\mathbf{p} \in \mathcal{H}(\mathbf{w}_i, \mathbf{n}_i)$  then
9:   Increment  $i \leftarrow (i + 1)$  until  $i = N - 1$ .
10: end if
11: return  $\mathbf{r}, \mathbf{q} = \mathbf{q}_{i-1}$  at each time step.

```

Algorithm 6 will produce paths like that shown in Figure 11.2. The advantages of Algorithm 6 are that it is extremely simple and that the MAV reaches the waypoint before transitioning to the next straight-line path. However, the paths shown in Figure 11.2 provide neither a smooth nor balanced transition between the straight-line segments. An alternative is to smoothly transition between waypoints by inserting a fillet as shown in Figure 11.3. The disadvantage with the path shown in Figure 11.3 is that the MAV does not directly pass through waypoint \mathbf{w}_i , which may sometimes be desired.

In the remainder of this section we will focus on smoothed paths like those shown in Figure 11.3. The geometry near the transition is shown in Figure 11.4. With the unit vector \mathbf{q}_i aligned with the line between waypoints \mathbf{w}_i and \mathbf{w}_{i+1} defined as in Equation (11.2), the angle between $\overline{\mathbf{w}_{i-1} \mathbf{w}_i}$ and $\overline{\mathbf{w}_i \mathbf{w}_{i+1}}$ is given by

$$\varrho \triangleq \cos^{-1}(-\mathbf{q}_{i-1}^\top \mathbf{q}_i). \quad (11.3)$$

If the radius of the fillet is R , as shown in Figure 11.4, then the distance between the waypoint \mathbf{w}_i and the location where the fillet intersects the line $\overline{\mathbf{w}_i \mathbf{w}_{i+1}}$ is $R/\tan \frac{\varrho}{2}$, and the distance between \mathbf{w}_i and the center of the fillet circle is $R/\sin \frac{\varrho}{2}$. Therefore, the distance between \mathbf{w}_i and the edge of the fillet circle along the bisector of ϱ is given by $R/\sin \frac{\varrho}{2} - R$.

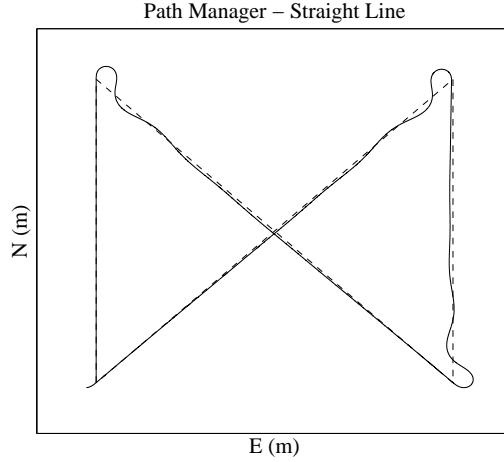


Figure 11.2: Path generated using the path-following approach given in Algorithm 6. The MAV follows the straight-line path until reaching the waypoint, and then maneuvers onto the next straight-line section.

To implement the fillet maneuver using the path-following algorithms described in Chapter 10, we will follow the straight-line segment $\overline{\mathbf{w}_{i-1}\mathbf{w}_i}$ until entering the half plane \mathcal{H}_1 shown in Figure 11.5. The right-handed orbit of radius R is then followed until entering the half plane \mathcal{H}_2 shown in Figure 11.5, at which point the straight-line segment $\overline{\mathbf{w}_i\mathbf{w}_{i+1}}$ is followed.

As shown in Figure 11.5, the center of the fillet is given by

$$\mathbf{c} = \mathbf{w}_i + \left(\frac{R}{\sin \frac{\varrho}{2}} \right) \frac{\mathbf{q}_{i-1} - \mathbf{q}_i}{\|\mathbf{q}_{i-1} - \mathbf{q}_i\|}.$$

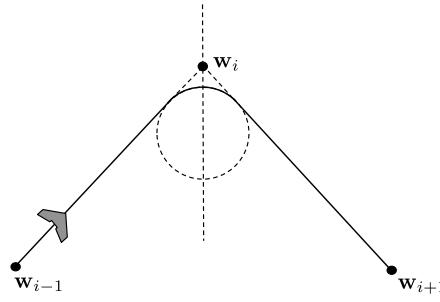


Figure 11.3: The transition from straight-line path $\overline{\mathbf{w}_{i-1}\mathbf{w}_i}$ to $\overline{\mathbf{w}_i\mathbf{w}_{i+1}}$ can be smoothed by inserting a fillet.

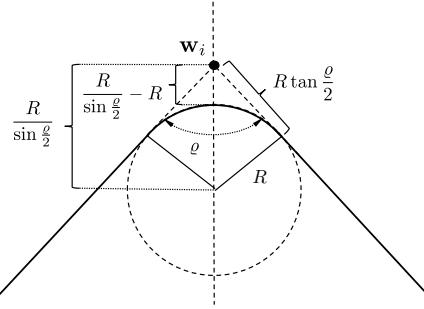


Figure 11.4: The geometry associated with inserting a fillet between waypoint segments.

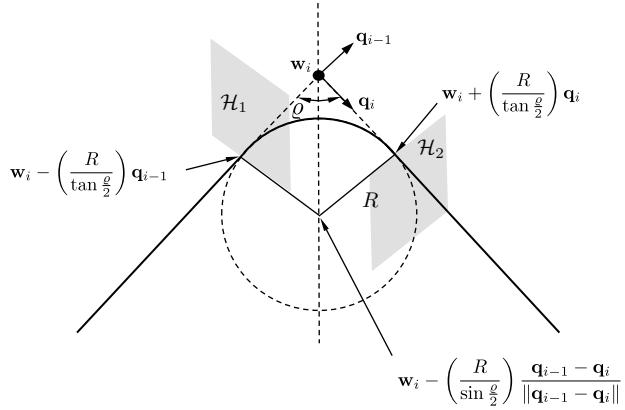


Figure 11.5: Definitions of the half planes associated with following a fillet inserted between waypoint segments.

Similarly, the half plane \mathcal{H}_1 is defined by the location

$$\mathbf{r}_1 = \mathbf{w}_i - \left(\frac{R}{\tan \frac{\varrho}{2}} \right) \mathbf{q}_{i-1},$$

and the normal vector \mathbf{q}_{i-1} . The half plane \mathcal{H}_2 is defined by the location

$$\mathbf{r}_2 = \mathbf{w}_i + \left(\frac{R}{\tan \frac{\varrho}{2}} \right) \mathbf{q}_i$$

and the normal vector \mathbf{q}_i .

The algorithm for maneuvering along the waypoint path \mathcal{W} using fillets to smooth between the straight-line segments is given by Algorithm 7. The `If` statement in Line 1 tests to see if a new waypoint path has been received, including when the algorithm is instantiated. If a new waypoint path has been received by the path manager, then the waypoint pointer and the state

machine are initialized in Line 2. The unit vectors \mathbf{q}_{i-1} and \mathbf{q}_i and the angle ϱ are computed in Lines 4–6.

When the state machine 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 8–10. Lines 11–14 test to see if the MAV has transitioned into the half plane shown as \mathcal{H}_1 in Figure 11.5. If the MAV has transitioned into \mathcal{H}_2 , then the state machine is updated to `state=2`.

When the state machine is in `state=2`, the MAV is commanded to follow the orbit that defines the fillet. The center, radius, and direction of the orbit are assigned in Lines 17–19. In Line 19, $q_{i-1,n}$ and $q_{i-1,e}$ denote the North and East components of \mathbf{q}_{i-1} . Lines 21–24 test to see if the MAV has transitioned into the half plane shown as \mathcal{H}_2 in Figure 11.5. If the MAV has transitioned into \mathcal{H}_2 , then the waypoint pointer is incremented, and the state machine is switched back to `state=1` to follow the segment $\overline{\mathbf{w}_i\mathbf{w}_{i+1}}$. Algorithm 7 produces paths like that shown in Figure 11.6.

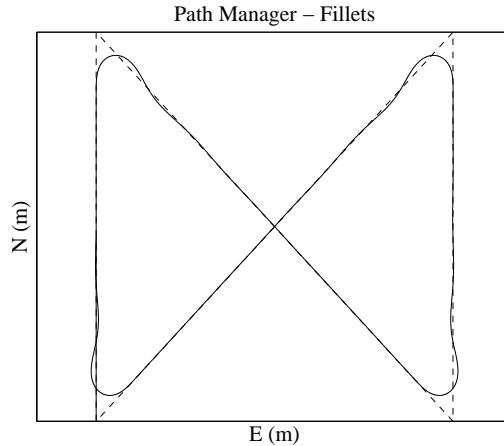


Figure 11.6: An example of the types of flight paths produced by Algorithm 7.

One of the disadvantages of the fillet method as given in Algorithm 7 is that the path length is changed when fillets are inserted. For certain applications, like the cooperative timing problems discussed in [36], it is important to have a high quality estimate of the path length, or the time required to traverse a certain waypoint path. We will conclude this section by deriving an expression for the path length of \mathcal{W} after fillets have been inserted.

Algorithm 7 Follow Waypoints with Fillets: $(\text{flag}, \mathbf{r}, \mathbf{q}, \mathbf{c}, \rho, \lambda) = \text{followWppFillet}(\mathcal{W}, \mathbf{p}, R)$

Input: Waypoint path $\mathcal{W} = \{\mathbf{w}_1, \dots, \mathbf{w}_N\}$, MAV position $\mathbf{p} = (p_n, p_e, p_d)^\top$, fillet radius R .

Require: $N \geq 3$.

```

1: if New waypoint path  $\mathcal{W}$  is received then
2:   Initialize waypoint index:  $i \leftarrow 2$ , and state machine: state  $\leftarrow 1$ .
3: end if
4:  $\mathbf{q}_{i-1} \leftarrow \frac{\mathbf{w}_i - \mathbf{w}_{i-1}}{\|\mathbf{w}_i - \mathbf{w}_{i-1}\|}$ .
5:  $\mathbf{q}_i \leftarrow \frac{\mathbf{w}_{i+1} - \mathbf{w}_i}{\|\mathbf{w}_{i+1} - \mathbf{w}_i\|}$ .
6:  $\varrho \leftarrow \cos^{-1}(-\mathbf{q}_{i-1}^\top \mathbf{q}_i)$ .
7: if state = 1 then
8:   flag  $\leftarrow 1$ 
9:    $\mathbf{r} \leftarrow \mathbf{w}_{i-1}$ 
10:   $\mathbf{q} \leftarrow \mathbf{q}_{i-1}$ 
11:   $\mathbf{z} \leftarrow \mathbf{w}_i - \left(\frac{R}{\tan(\varrho/2)}\right) \mathbf{q}_{i-1}$ 
12:  if  $\mathbf{p} \in \mathcal{H}(\mathbf{z}, \mathbf{q}_{i-1})$  then
13:    state  $\leftarrow 2$ 
14:  end if
15: else if state = 2 then
16:   flag  $\leftarrow 2$ 
17:    $\mathbf{c} \leftarrow \mathbf{w}_i + \left(\frac{R}{\sin(\varrho/2)}\right) \frac{\mathbf{q}_{i-1} - \mathbf{q}_i}{\|\mathbf{q}_{i-1} - \mathbf{q}_i\|}$ 
18:    $\rho \leftarrow R$ 
19:    $\lambda \leftarrow \text{sign}(q_{i-1,n} q_{i,e} - q_{i-1,e} q_{i,n})$ .
20:    $\mathbf{z} \leftarrow \mathbf{w}_i + \left(\frac{R}{\tan(\varrho/2)}\right) \mathbf{q}_i$ 
21:   if  $\mathbf{p} \in \mathcal{H}(\mathbf{z}, \mathbf{q}_i)$  then
22:      $i \leftarrow (i + 1)$  until  $i = N - 1$ .
23:     state  $\leftarrow 1$ 
24:   end if
25: end if
26: return flag,  $\mathbf{r}$ ,  $\mathbf{q}$ ,  $\mathbf{c}$ ,  $\rho$ ,  $\lambda$ .

```

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 7. From Figure 11.4 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,

$$|\mathcal{W}|_F = |\mathcal{W}| + \sum_{i=2}^N \left(R(\pi - \varrho_i) - \frac{2R}{\tan \frac{\varrho_i}{2}} \right), \quad (11.4)$$

where ϱ_i is given in Equation (11.3).

11.2 DUBINS PATHS

11.2.1 Definition of Dubins Path

This section focuses on so-called Dubins paths, where, rather than following a waypoint path, the objective is to transition from one configuration (position and course) to another. It was shown in [?] that for a vehicle with kinematics given by

$$\begin{aligned} \dot{p}_n &= V \cos \vartheta \\ \dot{p}_e &= V \sin \vartheta \\ \dot{\vartheta} &= u, \end{aligned}$$

where V is constant and $u \in [-\bar{u}, \bar{u}]$, the time-optimal path between two different configurations consists of a circular arc, followed by a straight line, and concluding with another circular arc to the final configuration, where the radius of the circular arcs is V/\bar{u} . These turn-straight-turn paths are one of several classes of Dubins paths defined for optimal transitions between configurations. In the context of unmanned aircraft, we will restrict our attention to constant-altitude, constant-goundspeed scenarios.

The radius of the circular arcs that define a Dubins path will be denoted by R , where we assume that R is at least as large as the minimum turn radius of the UAV. Throughout this section, a MAV configuration is defined as (\mathbf{p}, χ) , where \mathbf{p} is inertial position and χ is course angle.

Given a start configuration denoted as (\mathbf{p}_s, χ_s) and an end configuration denoted as (\mathbf{p}_e, χ_e) , a Dubins path consists of an arc of radius R that starts at the initial configuration, followed by a straight line, and concluded by another arc of radius R that ends at the end configuration. As shown in Figure 11.7, for any given start and end configurations, there are four possible paths consisting of an arc, followed by a straight line, followed by an arc. Case I (R-S-R) is a right-handed arc followed by a straight line followed by another right-handed arc. Case II (R-S-L) is a right-handed arc followed by a straight line followed by a left-handed arc. Case III (L-S-R) is a left-handed arc followed by a straight line followed by a right-handed arc. Case IV (L-S-L) is a left-handed arc followed by a straight line followed by another left-handed arc. The Dubins path is defined as case with the shortest path length.

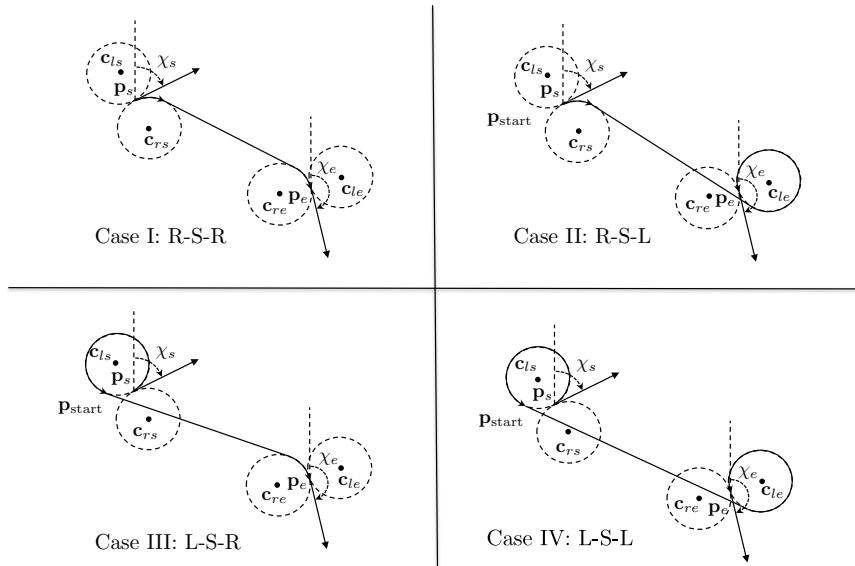


Figure 11.7: Given a start configuration (\mathbf{p}_s, χ_s) , an end configuration (\mathbf{p}_e, χ_e) , and a radius R , there are four possible paths consisting of an arc, a straight line, and an arc. The Dubins path is defined as the case that results in the shortest path length, which for this scenario is Case I.

11.2.2 Path Length Computation

To determine the Dubins path, it is necessary to compute the path length for the four cases shown in Figure 11.7. In this section, we will derive explicit formulas for the path length for each case. Given the position \mathbf{p} , the course

χ , and the radius R , the centers of the right and left turning circles are given by

$$\mathbf{c}_r = \mathbf{p} + R \left(\cos(\chi + \frac{\pi}{2}), \sin(\chi + \frac{\pi}{2}), 0 \right)^\top \quad (11.5)$$

$$\mathbf{c}_l = \mathbf{p} + R \left(\cos(\chi - \frac{\pi}{2}), \sin(\chi - \frac{\pi}{2}), 0 \right)^\top. \quad (11.6)$$

To compute the path length of the different trajectories, we need a general equation for angular distances on a circle. Figure 11.8 shows the geometry for both clockwise (*CW*) and counter clockwise (*CCW*) circles. We will

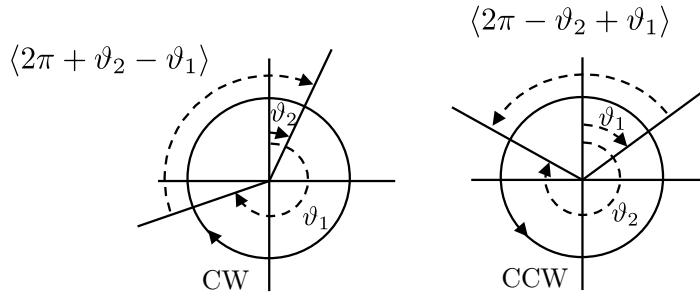


Figure 11.8: The angular distance between angles ϑ_1 and ϑ_2 for clockwise (*CW*) and counter clockwise (*CCW*) circles.

assume that both ϑ_1 and ϑ_2 are between 0 and 2π . For clockwise circles, the angular distance between ϑ_1 and ϑ_2 is given by

$$|\vartheta_2 - \vartheta_1|_{CW} \stackrel{\triangle}{=} \langle 2\pi + \vartheta_2 - \vartheta_1 \rangle, \quad (11.7)$$

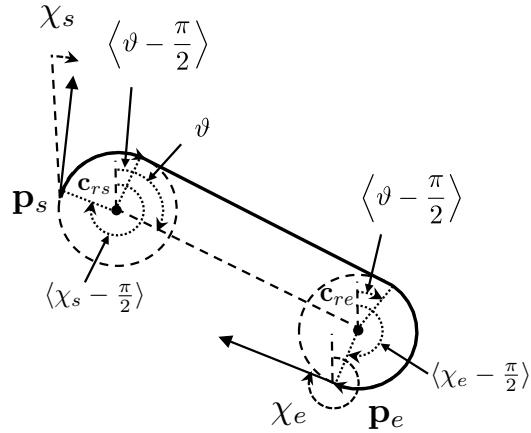
where

$$\langle \varphi \rangle \stackrel{\triangle}{=} \varphi \mod 2\pi.$$

Similarly, for counter clockwise circles, we get

$$|\vartheta_2 - \vartheta_1|_{CCW} \stackrel{\triangle}{=} \langle 2\pi - \vartheta_2 + \vartheta_1 \rangle. \quad (11.8)$$

11.2.2.1 Case I: R-S-R

**Figure 11.9:** Dubins path, Case I.

The geometry for Case I is shown in Figure 11.9, where ϑ is the angle formed by the line between \mathbf{c}_{rs} and \mathbf{c}_{re} . Using Equation (11.7), the angular distance traveled along \mathbf{c}_{rs} is given by

$$R\langle 2\pi + \langle \vartheta - \frac{\pi}{2} \rangle - \langle \chi_s - \frac{\pi}{2} \rangle \rangle.$$

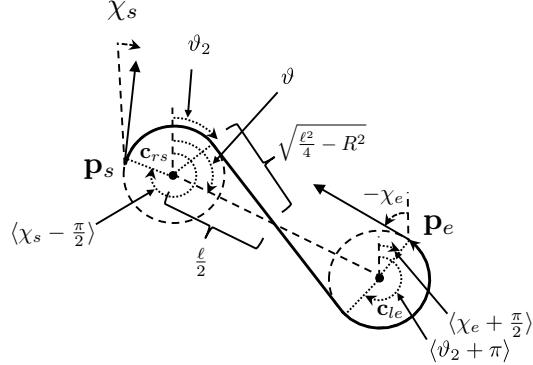
Similarly, using Equation (11.7), the angular distance traveled along \mathbf{c}_{re} is given by

$$R\langle 2\pi + \langle \chi_e - \frac{\pi}{2} \rangle - \langle \vartheta - \frac{\pi}{2} \rangle \rangle.$$

The total path length for Case I is therefore given by

$$L_1 = \|\mathbf{c}_{rs} - \mathbf{c}_{re}\| + R\langle 2\pi + \langle \vartheta - \frac{\pi}{2} \rangle - \langle \chi_s - \frac{\pi}{2} \rangle \rangle + R\langle 2\pi + \langle \chi_e - \frac{\pi}{2} \rangle - \langle \vartheta - \frac{\pi}{2} \rangle \rangle. \blacksquare \quad (11.9)$$

11.2.2.2 Case II: R-S-L

**Figure 11.10:** Dubins path, Case II.

The geometry for Case II is shown in Figure 11.10, where ϑ is the angle formed by the line between \mathbf{c}_{rs} and \mathbf{c}_{le} , $\ell = \|\mathbf{c}_{le} - \mathbf{c}_{rs}\|$, and

$$\vartheta_2 = \vartheta - \frac{\pi}{2} + \sin^{-1} \left(\frac{2R}{\ell} \right).$$

Using Equation (11.7), the angular distance traveled along \mathbf{c}_{rs} is given by

$$R \langle 2\pi + \langle \vartheta_2 \rangle - \langle \chi_s - \frac{\pi}{2} \rangle \rangle.$$

Similarly, using Equation (11.8), the angular distance traveled along \mathbf{c}_{le} is given by

$$R \langle 2\pi + \langle \vartheta_2 + \pi \rangle - \langle \chi_e + \frac{\pi}{2} \rangle \rangle.$$

The total path length for Case II is therefore given by

$$L_2 = \sqrt{\ell^2 - 4R^2} + R \langle 2\pi + \langle \vartheta_2 \rangle - \langle \chi_s - \frac{\pi}{2} \rangle \rangle + R \langle 2\pi + \langle \vartheta_2 + \pi \rangle - \langle \chi_e + \frac{\pi}{2} \rangle \rangle. \quad (11.10)$$

11.2.2.3 Case III: L-S-R

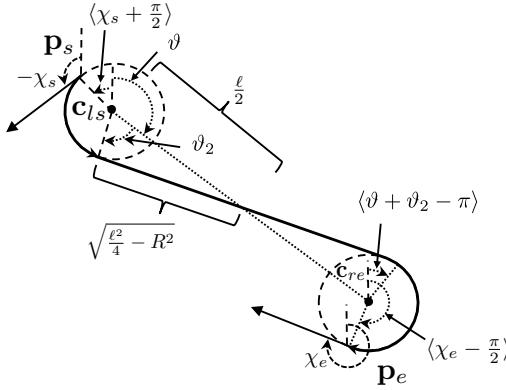


Figure 11.11: Dubins path, Case III.

The geometry for Case III is shown in Figure 11.11, where ϑ is the angle formed by the line between \mathbf{c}_{ls} and \mathbf{c}_{re} , $\ell = \|\mathbf{c}_{re} - \mathbf{c}_{ls}\|$, and

$$\vartheta_2 = \cos^{-1} \frac{2R}{\ell}.$$

Using Equation (11.8), the angular distance traveled along \mathbf{c}_{ls} is given by

$$R\left\langle 2\pi + \left\langle \chi_s + \frac{\pi}{2} \right\rangle - \left\langle \vartheta + \vartheta_2 \right\rangle \right\rangle.$$

Similarly, using Equation (11.7), the angular distance traveled along \mathbf{c}_{re} is given by

$$R\left\langle 2\pi + \left\langle \chi_e - \frac{\pi}{2} \right\rangle - \left\langle \vartheta + \vartheta_2 - \pi \right\rangle \right\rangle.$$

The total path length for Case III is therefore given by

$$L_3 = \sqrt{\ell^2 - 4R^2} + R\left\langle 2\pi + \left\langle \chi_s + \frac{\pi}{2} \right\rangle - \left\langle \vartheta + \vartheta_2 \right\rangle \right\rangle + R\left\langle 2\pi + \left\langle \chi_e - \frac{\pi}{2} \right\rangle - \left\langle \vartheta + \vartheta_2 - \pi \right\rangle \right\rangle. \blacksquare \quad (11.11)$$

11.2.2.4 Case IV: L-S-L

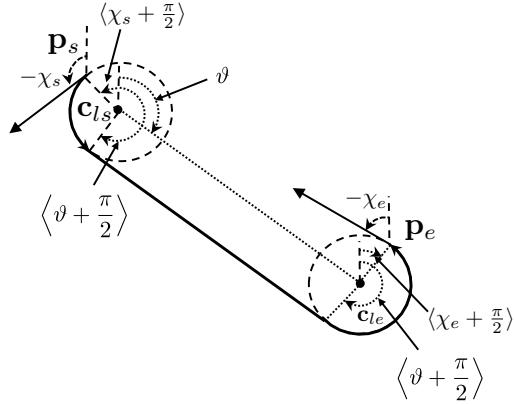


Figure 11.12: Dubins path, Case IV.

The geometry for Case IV is shown in Figure 11.12, where ϑ is the angle formed by the line between \mathbf{c}_{ls} and \mathbf{c}_{le} . Using Equation (11.8), the angular distance traveled along \mathbf{c}_{ls} is given by

$$R\langle 2\pi + \langle \chi_s + \frac{\pi}{2} \rangle - \langle \vartheta + \frac{\pi}{2} \rangle \rangle.$$

Similarly, using Equation (11.8), the angular distance traveled along \mathbf{c}_{le} is given by

$$R\langle 2\pi + \langle \vartheta + \frac{\pi}{2} \rangle - \langle \chi_e + \frac{\pi}{2} \rangle \rangle.$$

The total path length for Case IV is therefore given by

$$L_4 = \|\mathbf{c}_{ls} - \mathbf{c}_{le}\| + R\langle 2\pi + \langle \chi_s + \frac{\pi}{2} \rangle - \langle \vartheta + \frac{\pi}{2} \rangle \rangle + R\langle 2\pi + \langle \vartheta + \frac{\pi}{2} \rangle - \langle \chi_e + \frac{\pi}{2} \rangle \rangle. \quad (11.12)$$

11.2.3 Algorithm for Tracking Dubins Paths

The guidance algorithm for tracking a Dubins path is shown graphically in Figure 11.13 for Case III. The algorithm is initialized in a left-handed orbit about \mathbf{c}_{ls} and continues in that orbit until the MAV enters the half plane denoted as \mathcal{H}_1 . After entering \mathcal{H}_1 , a straight-line guidance strategy is used until the MAV enters the half plane denoted as \mathcal{H}_2 . A right-handed orbit around \mathbf{c}_{re} is then followed until the MAV enters the half plane denoted as \mathcal{H}_3 , which defines the completion of the Dubins path.

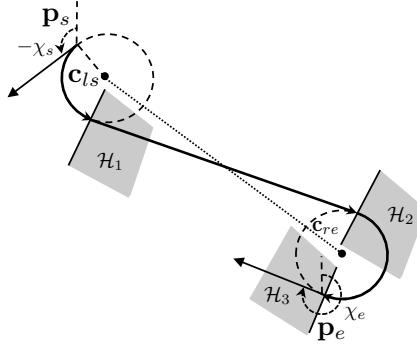


Figure 11.13: Definition of switching half planes for Dubins paths. The algorithm begins in a circular orbit and switches to straight-line tracking when \mathcal{H}_1 is entered. Orbit tracking is again initialized upon entering \mathcal{H}_2 . The half plane \mathcal{H}_3 defines the end of the Dubins path.

It follows that a Dubins path can be parameterized by the start circle \mathbf{c}_s , the direction of the start circle λ_s , the end circle \mathbf{c}_e , the direction of the end circle λ_e , the parameters of the half plane \mathcal{H}_1 denoted as \mathbf{z}_1 and \mathbf{q}_1 , the parameters of the half plane \mathcal{H}_2 denoted as \mathbf{z}_2 and $\mathbf{q}_2 = \mathbf{q}_1$, and the parameters of the half plane \mathcal{H}_3 denoted as \mathbf{z}_3 and \mathbf{q}_3 . The parameters of the Dubins path associated with the start configuration (\mathbf{p}_s, χ_s) , the end configuration (\mathbf{p}_e, χ_e) , and the radius R are computed in Algorithm 8. The length of the Dubins path L is also computed. The notation $\mathcal{R}_z(\vartheta)$ denotes the rotation matrix for a right-handed rotation of ϑ about the z -axis and $\vec{\mathbf{e}}_1 = (1, 0, 0)^\top$.

If we define the sequence of configurations

$$\mathcal{P} = \{(\mathbf{w}_1, \chi_1), (\mathbf{w}_2, \chi_2), \dots, (\mathbf{w}_N, \chi_N)\}, \quad (11.13)$$

then a guidance algorithm that follows Dubins paths between the configurations is given in Algorithm 9. In Line 4 the Dubins parameters are found for the current waypoint segment using Algorithm 8. Since the initial configuration may be in the far side of the circle that is already in \mathcal{H}_1 , the start circle is followed in `state=1` until crossing into the part of circle opposite \mathcal{H}_1 , as shown in Lines 7–9. The start circle is then followed in `state=2` until the MAV has crossed half plane \mathcal{H}_1 , as shown in Lines 10–13. After crossing into \mathcal{H}_1 , the straight-line segment of the Dubins path is followed in `state=3` as shown in Lines 14–18. Line 16 tests to see if the MAV has crossed half plane \mathcal{H}_2 . When it has, the end circle is followed in `state=4` and `state=5`. Two states are again needed since the MAV may already be in \mathcal{H}_3 at the instant that it enters \mathcal{H}_2 . If so, it follows the end circle until entering \mathcal{H}_3 as shown in Lines 19–23. After the MAV has entered into \mathcal{H}_3 ,

Algorithm 8 Find Dubins Parameters:
$$(L, \mathbf{c}_s, \lambda_s, \mathbf{c}_e, \lambda_e, \mathbf{z}_1, \mathbf{q}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{q}_3) = \text{findDubinsParameters}(\mathbf{p}_s, \chi_s, \mathbf{p}_e, \chi_e, R)$$

Input: Start configuration (\mathbf{p}_s, χ_s) , End configuration (\mathbf{p}_e, χ_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_4 using Equations (11.9) through (11.12).
 - 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: $\mathbf{c}_s \leftarrow \mathbf{c}_{rs}, \quad \lambda_s \leftarrow +1, \quad \mathbf{c}_e \leftarrow \mathbf{c}_{re}, \quad \lambda_e \leftarrow +1$
 - 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: $\mathbf{c}_s \leftarrow \mathbf{c}_{rs}, \quad \lambda_s \leftarrow +1, \quad \mathbf{c}_e \leftarrow \mathbf{c}_{le}, \quad \lambda_e \leftarrow -1$
 - 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: $\mathbf{c}_s \leftarrow \mathbf{c}_{ls}, \quad \lambda_s \leftarrow -1, \quad \mathbf{c}_e \leftarrow \mathbf{c}_{re}, \quad \lambda_e \leftarrow +1$
 - 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: $\mathbf{c}_s \leftarrow \mathbf{c}_{ls}, \quad \lambda_s \leftarrow -1, \quad \mathbf{c}_e \leftarrow \mathbf{c}_{le}, \quad \lambda_e \leftarrow -1$
 - 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: $\mathbf{z}_2 \leftarrow \mathbf{c}_e + R\mathcal{R}_z\left(\frac{\pi}{2}\right)\mathbf{q}_2$
 - 33: **end if**
 - 34: $\mathbf{z}_3 \leftarrow \mathbf{p}_e$
 - 35: $\mathbf{q}_3 \leftarrow \mathcal{R}_z(\chi_e)\mathbf{e}_1$
-

Algorithm 9 Follow Waypoints with Dubins: $(\text{flag}, \mathbf{r}, \mathbf{q}, \mathbf{c}, \rho, \lambda) = \text{followWppDubins}(\mathcal{P}, \mathbf{p}, R)$

Input: Configuration path $\mathcal{P} = \{(\mathbf{w}_1, \chi_1), \dots, (\mathbf{w}_N, \chi_N)\}$, MAV position $\mathbf{p} = (p_n, p_e, p_d)^\top$, fillet radius R .

Require: $N \geq 3$.

```

1: if New configuration path  $\mathcal{P}$  is received then
2:   Initialize waypoint pointer:  $i \leftarrow 2$ , and state machine: state  $\leftarrow 1$ .
3: end if
4:  $(L, \mathbf{c}_s, \lambda_s, \mathbf{c}_e, \lambda_e, \mathbf{z}_1, \mathbf{q}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{q}_3)$  ←
   findDubinsParameters( $\mathbf{w}_{i-1}, \chi_{i-1}, \mathbf{w}_i, \chi_i, R$ )
5: if state = 1 then
6:   flag  $\leftarrow 2$ ,  $\mathbf{c} \leftarrow \mathbf{c}_s$ ,  $\rho \leftarrow R$ ,  $\lambda \leftarrow \lambda_s$ 
7:   if  $\mathbf{p} \in \mathcal{H}(\mathbf{z}_1, -\mathbf{q}_1)$  then
8:     state  $\leftarrow 2$ 
9:   end if
10:  else if state = 2 then
11:    if  $\mathbf{p} \in \mathcal{H}(\mathbf{z}_1, \mathbf{q}_1)$  then
12:      state  $\leftarrow 3$ 
13:    end if
14:  else if state = 3 then
15:    flag  $\leftarrow 1$ ,  $\mathbf{r} \leftarrow \mathbf{z}_1$ ,  $\mathbf{q} \leftarrow \mathbf{q}_1$ 
16:    if  $\mathbf{p} \in \mathcal{H}(\mathbf{z}_2, \mathbf{q}_1)$  then
17:      state  $\leftarrow 4$ 
18:    end if
19:  else if state = 4 then
20:    flag  $\leftarrow 2$ ,  $\mathbf{c} \leftarrow \mathbf{c}_e$ ,  $\rho \leftarrow R$ ,  $\lambda \leftarrow \lambda_e$ 
21:    if  $\mathbf{p} \in \mathcal{H}(\mathbf{z}_3, -\mathbf{q}_3)$  then
22:      state  $\leftarrow 5$ 
23:    end if
24:  else if state = 5 then
25:    if  $\mathbf{p} \in \mathcal{H}(\mathbf{z}_3, \mathbf{q}_3)$  then
26:      state  $\leftarrow 1$ 
27:       $i \leftarrow (i + 1)$  until  $i = N$ .
28:       $(L, \mathbf{c}_s, \lambda_s, \mathbf{c}_e, \lambda_e, \mathbf{z}_1, \mathbf{q}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{q}_3)$  ←
         findDubinsParameters( $\mathbf{w}_{i-1}, \chi_{i-1}, \mathbf{w}_i, \chi_i, R$ )
29:    end if
30:  end if
31: return flag,  $\mathbf{r}$ ,  $\mathbf{q}$ ,  $\mathbf{c}$ ,  $\rho$ ,  $\lambda$ .

```

as detected in Line 25, the waypoints are cycled and new Dubins parameters are found in Lines 27–28. Figure 11.14 shows an example of a path generated using Algorithm 9.

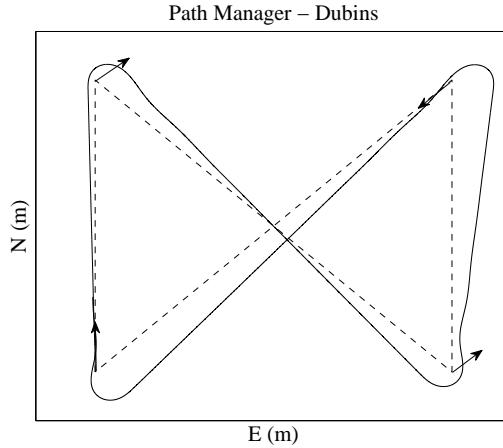


Figure 11.14: An example of the types of flight paths produced by Algorithm 9.

11.3 CHAPTER SUMMARY

This chapter introduced several schemes for transitioning between waypoint configurations using the straight-line and orbit-following algorithms described in Chapter 10. Section 11.1 discussed transitioning between waypoint segments using a half plane and by inserting a fillet between the waypoint segments. Section 11.2 introduced Dubins paths and showed how to construct Dubins paths between waypoint configurations. In the next chapter, we will describe several path-planning algorithms that find waypoint paths and waypoint configurations in order to maneuver through an obstacle field.

NOTES AND REFERENCES

Section 11.1 is based largely on [37]. Dubins paths were introduced in [?]. In certain degenerate cases, the Dubins path may not contain one of the three elements. For example, if the start and end configurations are on a straight line, then the beginning and end arcs will not be necessary. Or if the start and end configurations lie on a circle of radius R , then the straight line

and end arc will not be necessary. In this chapter, we have ignored these degenerate cases. Reference [38] builds upon Dubins's ideas to generate feasible trajectories for UAVs given kinematic and path constraints by algorithmically finding the optimal location of Dubins circles and straight-line paths. In [39], Dubins circles are superimposed as fillets at the junction of straight-line waypoint paths produced from a Voronoi diagram. In some applications, like the cooperative-timing problem described in [36], it may be desirable to transition between waypoints in a way that preserves the path length. A path manager for this scenario is described in [37].

11.4 DESIGN PROJECT

The objective of this assignment is to implement Algorithms 6 and 7 for following a set of waypoints denoted as \mathcal{W} , and Algorithm 9 for following a set of configurations denoted as \mathcal{P} . The input to the path manager is either \mathcal{W} or \mathcal{P} , and the output is the path definition

$$y_{\text{manager}} = \begin{pmatrix} \text{flag} \\ V_a^d \\ \mathbf{r} \\ \vec{\mathbf{q}} \\ \mathbf{c} \\ \rho \\ \lambda \end{pmatrix}.$$

Skeleton code for this chapter is given on the website.

- 11.1 Modify `path_manager_line.m` to implement Algorithm 6 to follow the waypoint path defined in `path_planner_chap11.m`. Test and debug the algorithm on the guidance model given in Equation (9.18). ■ When the algorithm is working well on the guidance model, verify that it performs adequately for the full six-DOF model.
- 11.2 Modify `path_manager_fillet.m` and implement Algorithm 7 to follow the waypoint path defined in `path_planner_chap11.m`. Test and debug the algorithm on the guidance model given in Equation (9.18). When the algorithm is working well on the guidance model, verify that it performs adequately for the full six-DOF model.
- 11.3 Modify `path_manager_dubins.m` and implement Algorithm 9 to follow the path configuration defined in `path_planner_chap11.m`. ■

Test and debug the algorithm on the guidance model given in Equation (9.18). When the algorithm is working well on the guidance model, verify that it performs adequately for the full six-DOF model.

Chapter Twelve

Path Planning

In the robotics literature, there are essentially two different approaches to motion planning: *deliberative* motion planning, where explicit paths and trajectories are computed based on global world knowledge [?, ?, 40], and *reactive* motion planning, which uses behavioral methods to react to local sensor information [41, 42]. In general, deliberative motion planning is useful when the environment is known a priori, but can become computationally intensive in highly dynamic environments. Reactive motion planning, on the other hand, is well suited for dynamic environments, particularly collision avoidance, where information is incomplete and uncertain, but it lacks the ability to specify and direct motion plans.

This chapter focuses on deliberative path planning techniques that we have found to be effective and efficient for miniature air vehicles. In deliberative approaches, the MAV's trajectories are planned explicitly. The drawback of deliberative approaches is that they are strongly dependent upon the models used to describe the state of the world and the motion of the vehicle. Unfortunately, precise modeling of the atmosphere and the vehicle dynamics is not possible. To compensate for this inherent uncertainty, the path planning algorithms need to be executed on a regular basis in an outer feedback loop. It is essential, therefore, that the path planning algorithms be computationally efficient. To reduce the computational demand, we will use simple low-order navigation models for the vehicle and constant-wind models for the atmosphere. We assume that a terrain elevation map is available to the path planning algorithms. Obstacles that are known a priori are represented on the elevation map.

This chapter describes several simple and efficient path planning algorithms that are suitable for miniature air vehicles. The methods that we present are by no means exhaustive and might not be the best possible methods. However, we feel that they provide an accessible introduction to path planning. In reference to the architecture shown in Figure 1.1, this chapter describes the design of the path planner. We will describe path planning algorithms for two types of problems. In Section 12.1 we will address point-to-point problems, where the objective is to plan a waypoint path from one point to another through an obstacle field. In Section 12.2 we will address coverage problems, where the objective is to plan a waypoint path so that

the MAV covers all of the area in a certain region. The output of the path planning algorithms developed in this chapter will be a sequence of either waypoints or configurations (waypoint plus orientation) and will therefore interface with the path management algorithms developed in Chapter 11.

12.1 POINT-TO-POINT ALGORITHMS

12.1.1 Voronoi Graphs

The Voronoi graph is particularly well suited to applications that require the MAV to maneuver through a congested airspace with obstacles that are small relative to the turning radius of the vehicle. The relative size allows the obstacles to be modeled as points with zero area. The Voronoi method is essentially restricted to 2½-D (or constant predefined altitude) path planning, where the altitude at each node is fixed in the map.

Given a finite set \mathcal{Q} of points in \mathbb{R}^2 , the Voronoi graph divides \mathbb{R}^2 into Q convex cells, each containing exactly one point in \mathcal{Q} . The Voronoi graph is constructed so that the interior of each convex cell is closer to its associated point than to any other point in \mathcal{Q} . An example of a Voronoi graph is shown in Figure 12.1.

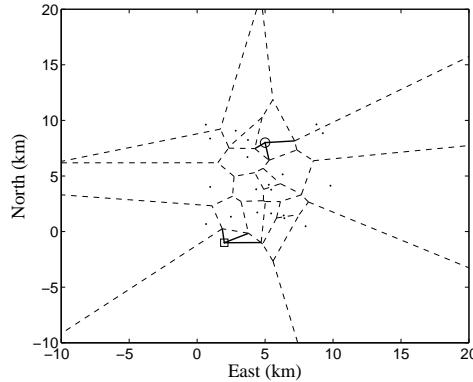


Figure 12.1: An example of a Voronoi graph with $Q = 20$ point obstacles.

The key feature of the Voronoi graph that makes it useful for MAV path planning is that the edges of the graph are perpendicular bisectors between the points in \mathcal{Q} . Therefore, following the edges of the Voronoi graph potentially produces paths that avoid the points in \mathcal{Q} . However, Figure 12.1 illustrates several potential pitfalls of using the Voronoi graph. First, graph edges that extend to infinity are obviously not good potential waypoint paths. Second, even for Voronoi cells with finite area, following the edges of the

Voronoi graph may lead to unnecessarily long excursions. Finally, note that for the two points in the lower right-hand corner of Figure 12.1, the Voronoi graph produces an edge between the two points; however, since the edge is so close to the points, the corresponding waypoint path may not be desirable.

There are well-established and widely available algorithms for generating Voronoi graphs. For example, Matlab has a built-in Voronoi function, and C++ implementations are publicly available on the Internet. Given the availability of Voronoi code, we will not discuss implementation of the algorithm. For additional discussions, see [43, ?, 44].

To use the Voronoi graph for point-to-point path planning, let $G = (V, E)$ be a graph produced by implementing the Voronoi algorithm on the set \mathcal{Q} . The node set V is augmented with the desired start and end locations as

$$V^+ = V \cup \{\mathbf{p}_s, \mathbf{p}_e\},$$

where \mathbf{p}_s is the start position and \mathbf{p}_e is the end position. The edge set E is then augmented with edges that connect the start and end nodes to the three closest nodes in V . The associated graph is shown in Figure 12.2.

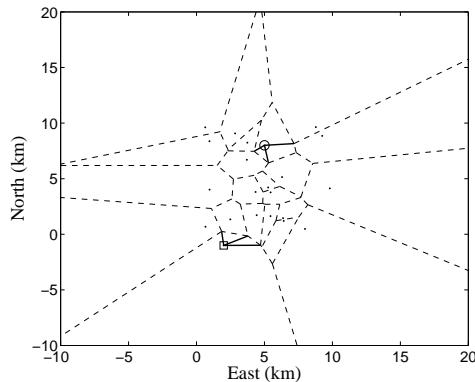


Figure 12.2: The Voronoi graph of \mathcal{Q} is augmented with start and end nodes and with edges that connect the start and end notes to \mathcal{Q} .

The next step is to assign a cost to each edge in the Voronoi graph. Edge costs can be assigned in a variety of ways. For illustrative purposes, we will assume that the cost of traversing each path is a function of the path length and the distance from the path to points in \mathcal{Q} . The geometry for deriving the metric is shown in Figure 12.3. Let the nodes of the graph edge be denoted by \mathbf{v}_1 and \mathbf{v}_2 . The length of the edge is given by $\|\mathbf{v}_1 - \mathbf{v}_2\|$. Any point on

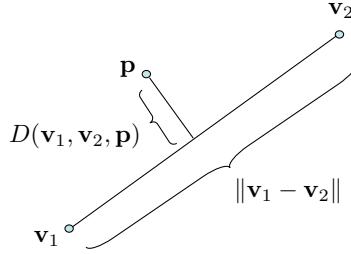


Figure 12.3: The cost penalty assigned to each edge of the Voronoi graph is proportional to the path length $\|v_1 - v_2\|$ and the reciprocal of the minimum distance from the path to a point in \mathcal{Q} .

the line segment can be written as

$$\mathbf{w}(\sigma) = (1 - \sigma)\mathbf{v}_1 + \sigma\mathbf{v}_2,$$

where $\sigma \in [0, 1]$. The minimum distance between \mathbf{p} and the graph edge can be expressed as

$$\begin{aligned} D(\mathbf{v}_1, \mathbf{v}_2, \mathbf{p}) &\stackrel{\Delta}{=} \min_{\sigma \in [0, 1]} \|\mathbf{p} - \mathbf{w}(\sigma)\| \\ &= \min_{\sigma \in [0, 1]} \sqrt{(\mathbf{p} - \mathbf{w}(\sigma))^\top (\mathbf{p} - \mathbf{w}(\sigma))} \\ &= \min_{\sigma \in [0, 1]} \sqrt{\mathbf{p}^\top \mathbf{p} - 2(1 - \sigma)\sigma \mathbf{p}^\top \mathbf{v}_1 - \sigma \mathbf{p}^\top \mathbf{v}_2 + (1 - \sigma)^2 \mathbf{v}_1^\top \mathbf{v}_1 + 2(1 - \sigma)\sigma \mathbf{v}_1^\top \mathbf{v}_2 + \sigma^2 \mathbf{v}_2^\top \mathbf{v}_2} \\ &= \min_{\sigma \in [0, 1]} \sqrt{\|\mathbf{p} - \mathbf{v}_1\|^2 + 2\sigma(\mathbf{p} - \mathbf{v}_1)^\top (\mathbf{v}_1 - \mathbf{v}_2) + \sigma^2 \|\mathbf{v}_1 - \mathbf{v}_2\|^2}. \end{aligned}$$

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

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

If we define

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

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

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

The cost for the edge defined by $(\mathbf{v}_1, \mathbf{v}_2)$ is assigned as

$$J(\mathbf{v}_1, \mathbf{v}_2) = k_1 \|\mathbf{v}_1 - \mathbf{v}_2\| + \frac{k_2}{D(\mathbf{v}_1, \mathbf{v}_2, \mathcal{Q})}, \quad (12.1)$$

where k_1 and k_2 are positive weights. The first term in Eq. (12.1) is the length of the edge, and the second term is the reciprocal of the distance from the edge to the closest point in \mathcal{Q} .

The final step is to search the Voronoi graph to determine the lowest-cost path from the start node to the end node. There are numerous existing graph search techniques that might be appropriate to accomplish this task [44]. A well-known algorithm with readily available code is Dijkstra's algorithm [?], which has a computational complexity equal to $\mathcal{O}(|V|)$. An example of a path found by Dijkstra's algorithm with $k_1 = 0.1$ and $k_2 = 0.9$ is shown in Figure 12.4.

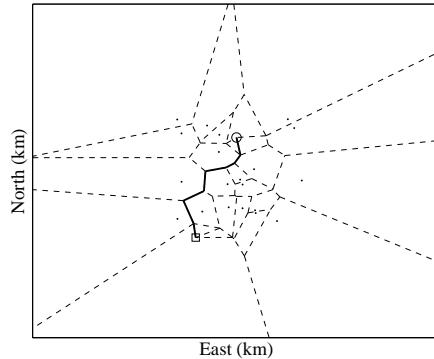


Figure 12.4: Optimal path through the Voronoi graph.

Pseudo-code for the Voronoi path planning method is listed in Algorithm 10. If there are not a sufficient number of obstacle points in \mathcal{Q} , the resulting Voronoi graph will be sparse and could potentially have many edges extending to infinity. To avoid that situation, Algorithm 10 requires that \mathcal{Q} has at least 10 points. That number is, of course, arbitrary. In Line 1 the Voronoi graph is constructed using a standard algorithm. In Line 2 the start and end points are added to the Voronoi graph, and the edges between the start and end points and the closest nodes in \mathcal{Q} are added in Lines 3-4.

Edge costs are assigned in Lines 5–7 according to Equation (12.1), and the waypoint path is determined via a Dijkstra search in Line 8.

Algorithm 10 Plan Voronoi Path: $\mathcal{W} = \text{planVoronoi}(\mathcal{Q}, \mathbf{p}_s, \mathbf{p}_e)$

Input: Obstacle points \mathcal{Q} , start position \mathbf{p}_s , end position \mathbf{p}_e .

Require: $|\mathcal{Q}| \geq 10$. Randomly add points if necessary.

- 1: $(V, E) = \text{constructVoronoiGraph}(\mathcal{Q})$.
 - 2: $V^+ = V \cup \{\mathbf{p}_s\} \cup \{\mathbf{p}_e\}$.
 - 3: Find $\{\mathbf{v}_{1s}, \mathbf{v}_{2s}, \mathbf{v}_{3s}\}$, the three closest points in V to \mathbf{p}_s , and $\{\mathbf{v}_{1e}, \mathbf{v}_{2e}, \mathbf{v}_{3e}\}$, the three closest points in V to \mathbf{p}_e .
 - 4: $E^+ = E \cup_{i=1,2,3} (\mathbf{v}_{is}, \mathbf{p}_s) \cup_{i=1,2,3} (\mathbf{v}_{ie}, \mathbf{p}_e)$.
 - 5: **for** Each element $(\mathbf{v}_a, \mathbf{v}_b) \in E$ **do**
 - 6: Assign edge cost $\mathbf{J}_{ab} = J(\mathbf{v}_a, \mathbf{v}_b)$ according to Equation (12.1).
 - 7: **end for**
 - 8: $\mathcal{W} = \text{DijkstraSearch}(V^+, E^+, \mathbf{J})$.
 - 9: **return** \mathcal{W} .
-

One of the disadvantages of the Voronoi method described in Algorithm 10 is that it is limited to point obstacles. However, there are straightforward modifications for non-point obstacles. For example, consider the obstacle field shown in Figure 12.5(a). A Voronoi graph can be constructed by first adding points around the perimeter of the obstacles that exceed a certain size, as shown in Figure 12.5(b). The associated Voronoi graph, including connections to start and end nodes, is shown in Figure 12.5(c). However, it is obvious from Figure 12.5(c) that the Voronoi graph includes many infeasible links that are contained inside an obstacle or terminate on the obstacle. The final step is to remove the infeasible links, as shown in Figure 12.5(d), which also displays the resulting optimal path.

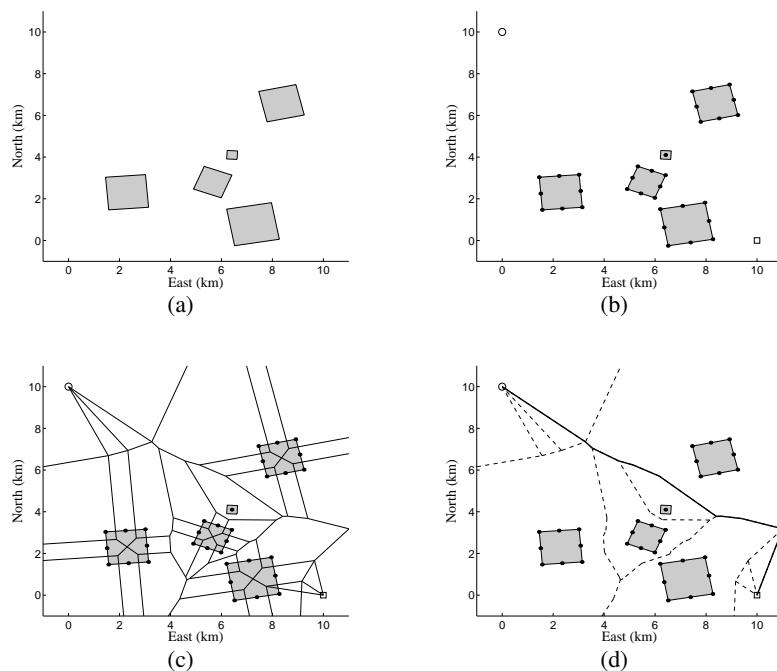


Figure 12.5: (a) An obstacle field with non-point obstacles. (b) The first step in using the Voronoi method to construct a path through the obstacle field is to insert points around the perimeter of the obstacles. (c) The resulting Voronoi graph includes many infeasible links that are contained inside the obstacles or that terminate on the boundary of the obstacle. (d) When infeasible links are removed, the resulting graph can be used to plan paths through the obstacle field.

12.1.2 Rapidly Exploring Random Trees

Another method for planning paths through an obstacle field from a start node to an end node is the Rapidly Exploring Random Tree (RRT) method. The RRT scheme is a random exploration algorithm that uniformly, but randomly, explores the search space. It has the advantage that it can be extended to vehicles with complicated nonlinear dynamics. We assume throughout this section that obstacles are represented in a terrain map that can be queried to detect possible collisions.

The RRT algorithm is implemented using a data structure called a *tree*. A tree is a special case of a directed graph. Figure 12.6 is a graphical depiction of a tree. Edges in trees are directed from a child node to its parent. In a tree, every node has exactly one parent, except the root, which does not have any parents. In the RRT framework, the nodes represent physical states, or configurations, and the edges represent feasible paths between the states. The cost associated with each edge, c_{ij} , is the cost associated with traversing the feasible path between states represented by the nodes.

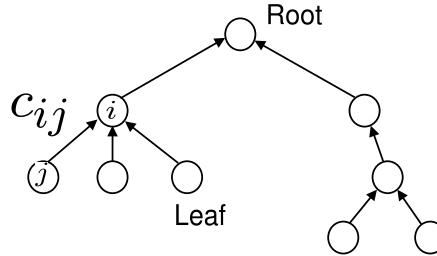


Figure 12.6: A tree is a special graph where every node, except the root, has exactly one parent.

The basic idea of the RRT algorithm is to build a tree that uniformly explores the search space. The uniformity is achieved by randomly sampling from a uniform probability distribution. To illustrate the basic idea, let the nodes represent north-east locations at a constant altitude, and let the cost c_{ij} of the edges between nodes be the length of the straight-line path between the nodes.

Figure 12.7 depicts the basic RRT algorithm. As shown in Figure 12.7(a), the input to the RRT algorithm is a start configuration \mathbf{p}_s , an end configuration \mathbf{p}_e , and the terrain map. The first step of the algorithm is to randomly select a configuration \mathbf{p} in the workspace. As shown in Figure 12.7(b), a new configuration \mathbf{v}_1 is selected a fixed distance D from \mathbf{p}_s along the line $\overline{\mathbf{p}\mathbf{p}_s}$, and inserted into the tree. At each subsequent step, a random configuration \mathbf{p} is generated in the workspace, and the tree is searched to find the node that is closest to \mathbf{p} . As shown in Figure 12.7(c), a new configuration

is generated that is a distance D from the closest node in the tree, along the line connecting \mathbf{p} to the closest node. Before a path segment is added to the tree, it needs to be checked for collisions with the terrain. If a collision is detected, as shown in Figure 12.7(d), then the segment is deleted and the process is repeated. When a new node is added, its distance from the end node \mathbf{p}_e is checked. If it is less than D , then a path segment from \mathbf{p}_e is added to the tree, as shown in Figure 12.7(f), indicating that a complete path through the terrain has been found.

Let \mathcal{T} be the terrain map, and let \mathbf{p}_s and \mathbf{p}_e be the start and end configurations in the map. Algorithm 11 gives the basic RRT algorithm. In Line 1, the RRT graph G is initialized to contain only the start node. The while loop in Lines 2–14 adds nodes to the RRT graph until the end node is included in the graph, indicating that a path from \mathbf{p}_s to \mathbf{p}_e has been found. In Line 3 a random configuration is drawn from the terrain according to a uniform distribution over \mathcal{T} . Line 4 finds the closest node $\mathbf{v}^* \in G$ to the randomly selected point \mathbf{p} . Since the distance between \mathbf{p} and \mathbf{v}^* may be large, Line 5 plans a path of fixed length D from \mathbf{v}^* in the direction of \mathbf{p} . The resulting configuration is denoted as \mathbf{v}^+ . If the resulting path is feasible, as checked in Line 6, then \mathbf{v}^+ is added to G in Line 7 and the cost matrix is updated in Line 8. The *if* statement in Line 10 checks to see if the new node \mathbf{v}^+ can be connected directly to the end node \mathbf{p}_e . If so, \mathbf{p}_e is added to G in Line 11–12, and the algorithm ends in Line 15 by returning the shortest waypoint path in G .

The result of implementing Algorithm 11 for four different randomly generated obstacle fields and randomly generated start and end nodes is displayed with a dashed line in Figure 12.8. Note that the paths generated by Algorithm 11 sometimes wander needlessly and that eliminating some nodes may result in a more efficient path. Algorithm 12 gives a simple scheme for smoothing the paths generated by Algorithm 11. The basic idea is to remove intermediate nodes if a feasible path still exists. The result of applying Algorithm 12 is shown with a solid line in Figure 12.8.

There are numerous extensions to the basic RRT algorithm. A common extension, which is discussed in [45], is to extend the tree from both the start and the end nodes and, at the end of each extension, to attempt to connect the two trees. In the next two subsections, we will give two simple extensions that are useful for MAV applications: waypoint planning over 3-D terrain and using Dubins paths to plan kinematically feasible paths in complex 2-D terrain.

Algorithm 11 Plan RRT Path: $\mathcal{W} = \text{planRRT}(\mathcal{T}, \mathbf{p}_s, \mathbf{p}_e)$

Input: Terrain map \mathcal{T} , start configuration \mathbf{p}_s , end configuration \mathbf{p}_e .

```

1: Initialize RRT graph  $G = (V, E)$  as  $V = \{\mathbf{p}_s\}$ ,  $E = \emptyset$ .
2: while The end node  $\mathbf{p}_e$  is not connected to  $G$ , i.e.,  $\mathbf{p}_e \notin V$  do
3:    $\mathbf{p} \leftarrow \text{generateRandomConfiguration}(\mathcal{T})$ 
4:    $\mathbf{v}^* \leftarrow \text{findClosestConfiguration}(\mathbf{p}, V)$ 
5:    $\mathbf{v}^+ \leftarrow \text{planPath}(\mathbf{v}^*, \mathbf{p}, D)$ 
6:   if  $\text{existFeasiblePath}(\mathcal{T}, \mathbf{v}^*, \mathbf{v}^+)$  then
7:     Update graph  $G = (V, E)$  as  $V \leftarrow V \cup \{\mathbf{v}^+\}$ ,  $E \leftarrow E \cup \{(\mathbf{v}^*, \mathbf{v}^+)\}$ 
8:     Update edge costs as  $C[(\mathbf{v}^*, \mathbf{v}^+)] \leftarrow \text{pathLength}(\mathbf{v}^*, \mathbf{v}^+)$ 
9:   end if
10:  if  $\text{existFeasiblePath}(\mathcal{T}, \mathbf{v}^+, \mathbf{p}_e)$  then
11:    Update graph  $G = (V, E)$  as  $V \leftarrow V \cup \{\mathbf{p}_e\}$   $E \leftarrow E \cup \{(\mathbf{v}^*, \mathbf{p}_e)\}$ 
12:    Update edge costs as  $C[(\mathbf{v}^*, \mathbf{p}_e)] \leftarrow \text{pathLength}(\mathbf{v}^*, \mathbf{p}_e)$ 
13:  end if
14: end while
15:  $\mathcal{W} = \text{findShortestPath}(G, C)$ .
16: return  $\mathcal{W}$ .
```

Algorithm 12 Smooth RRT Path: $(\mathcal{W}_s, C_s) = \text{smoothRRT}(\mathcal{T}, \mathcal{W}, C)$

Input: Terrain map \mathcal{T} , waypoint path $\mathcal{W} = \{\mathbf{w}_1, \dots, \mathbf{w}_N\}$, cost matrix C .

```

1: Initialized smoothed path  $\mathcal{W}_s \leftarrow \{\mathbf{w}_1\}$ 
2: Initialize pointer to current node in  $\mathcal{W}_s$ :  $i \leftarrow 1$ .
3: Initialize pointer to next node in  $\mathcal{W}$ :  $j \leftarrow 2$ 
4: while  $j < N$  do
5:    $\mathbf{w}_s \leftarrow \text{getNode}(\mathcal{W}_s, i)$ 
6:    $\mathbf{w}^+ \leftarrow \text{getNode}(\mathcal{W}, j + 1)$ 
7:   if  $\text{existFeasiblePath}(\mathcal{T}, \mathbf{w}_s, \mathbf{w}^+) = \text{FALSE}$  then
8:     Get last node:  $\mathbf{w} \leftarrow \text{getNode}(\mathcal{W}, j)$ 
9:     Add deconflicted node to smoothed path:  $\mathcal{W}_s \leftarrow \mathcal{W}_s \cup \{\mathbf{w}\}$ 
10:    Update smoothed cost:  $C_s[(\mathbf{w}_s, \mathbf{w})] \leftarrow \text{pathLength}(\mathbf{w}_s, \mathbf{w})$ 
11:     $i \leftarrow i + 1$ 
12:  end if
13:   $j \leftarrow j + 1$ 
14: end while
15: Add last node from  $\mathcal{W}$ :  $\mathcal{W}_s \leftarrow \mathcal{W}_s \cup \{\mathbf{w}_N\}$ 
16: Update smoothed cost:  $C_s[(\mathbf{w}_i, \mathbf{w}_N)] \leftarrow \text{pathLength}(\mathbf{w}_i, \mathbf{w}_N)$ 
17: return  $\mathcal{W}_s$ .
```

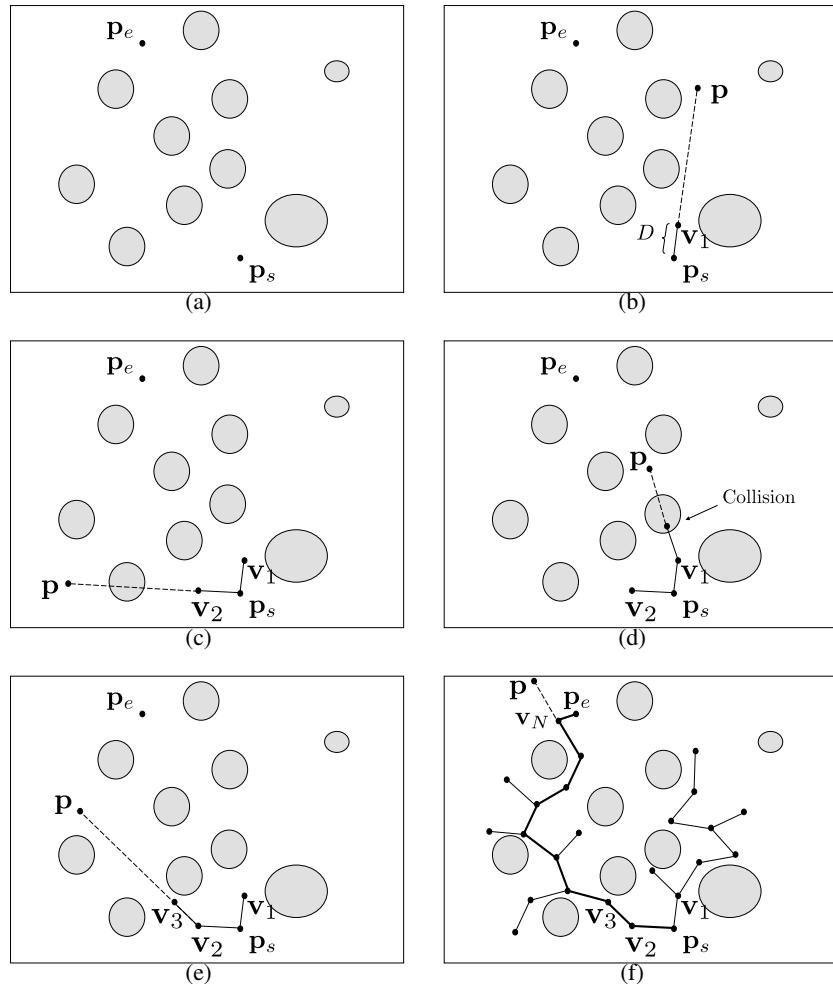


Figure 12.7: (a) The RRT algorithm is initialized with a terrain map and a start node and an end node. (b) and (c) The RRT graph is extended by randomly generating a point p in the terrain and planning a path of length D in the direction of p . (d) If the resulting configuration is not feasible, then it is not added to the RRT graph, and the process continues as shown in (e). (f) The RRT algorithm completes when the end node is added to the the RRT graph.

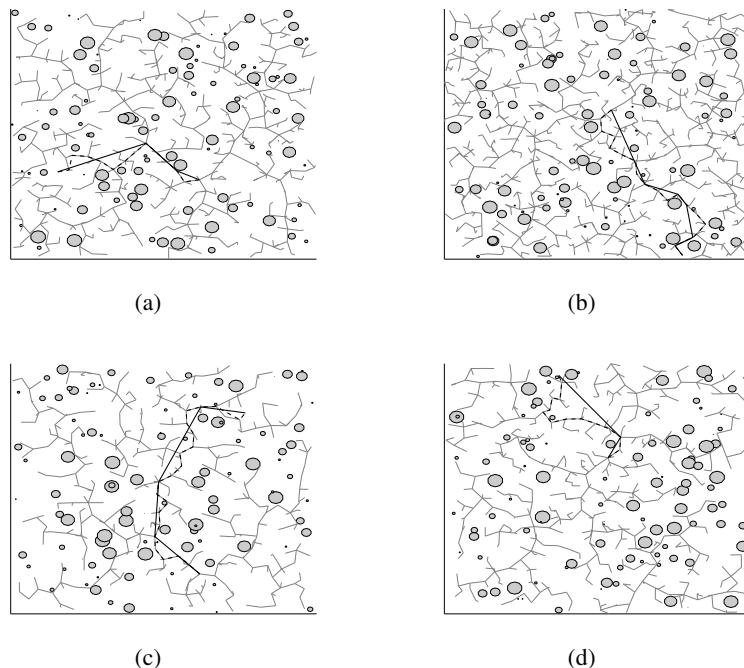


Figure 12.8: The results of Algorithm 11 for four randomly generated obstacle fields and randomly generated start and end nodes are indicated by dashed lines. The smoothed paths generated by Algorithm 12 are indicated by the solid lines.

12.1.2.1 RRT Waypoint Planning over 3-D Terrain

In this section we will consider the extension of the basic RRT algorithm to planning waypoint paths over 3-D terrain. We will assume that the terrain \mathcal{T} can be queried for the altitude of the terrain at any north-east position. The primary question that must be answered to extend the basic RRT algorithm to 3-D is how to generate the altitude at random nodes. For example, one option is to randomly select the altitude as a uniform distribution of height-above-ground, up to a maximum limit. Another option is to pre-select several altitude levels and then randomly select one of these levels.

In this section, we will select the altitude as a fixed height-above-ground h_{AGL} . Therefore, the (unsmoothed) RRT graph will, in essence, be a 2-D graph that follows the contour of the terrain. The output of Algorithm 11 will be a path that follows the terrain at a fixed altitude h_{AGL} . However, the smoothing step represented by Algorithm 12 will result in paths with much less altitude variation. For 3-D terrain, the climb rate and descent rates of the MAV are usually constrained to be within certain limits. The function `existFeasiblePath` in Algorithms 11 and 12 can be modified to ensure that the climb and descent rates are satisfied and that terrain collisions are avoided.

The results of the 3-D RRT algorithm are shown in Figures 12.9 and 12.10, where the thin lines represent the RRT tree, the thick dashed line is the RRT path generated by Algorithm 11, and the thick solid line is the smoothed path generated by Algorithm 12.

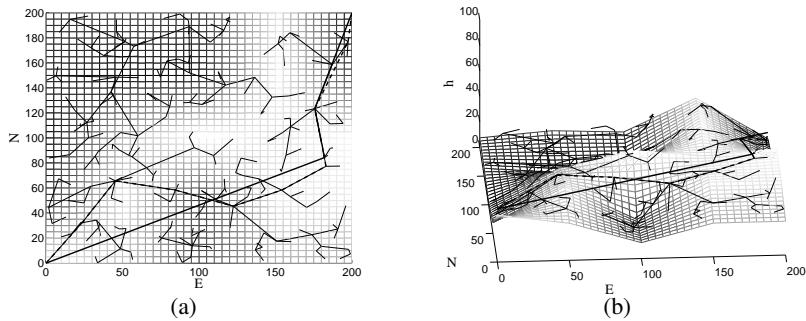


Figure 12.9: (a) Overhead view of the results of the 3-D RRT waypoint path planning algorithm. (b) Side view of the results of the 3-D RRT waypoint path planning algorithm. The thin lines are the RRT graph, the thick dotted line is the RRT path returned by Algorithm 11, and the thick solid line is the smoothed path. The RRT graph is generated at a fixed height above the terrain.

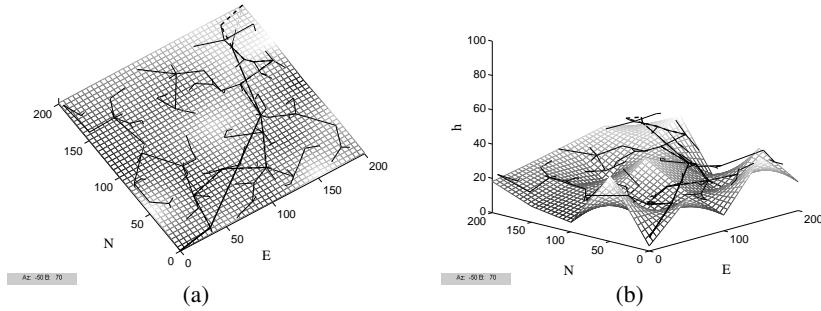


Figure 12.10: (a) Overhead view of the results of the 3-D RRT waypoint path planning algorithm. (b) Side view of the results of the 3-D RRT waypoint path planning algorithm. The thin lines are the RRT graph, the thick dashed line is the RRT path returned by Algorithm 11, and the thick solid line is the smoothed path. The RRT graph is generated at a fixed height above the terrain.

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

In this section, we will consider the extension of the basic RRT algorithm to planning paths subject to turning constraints. Assuming that the vehicle moves at constant velocity, optimal paths between configurations are given by Dubins paths, as discussed in Section 11.2. Dubins paths are planned between two different configurations, where a configuration is given by three numbers representing the north and east positions and the course angle at that position. To apply the RRT algorithm to this scenario, we need to have a technique for generating random configurations.

We will generate a random configuration as follows:

1. Generate a random north-east position in the environment.
2. Find the closest node in the RRT graph to the new point.
3. Select a position of distance L from the closest RRT node, and use that position as the north-east coordinates of the new configuration.
4. Select the course angle for the configuration as the angle of the line that connects the new configuration to the RRT tree.

The RRT algorithm is then implemented as described in Algorithm 11, where the function `pathLength` returns the length of the Dubins path between configuration.

The results of the RRT Dubins algorithm are shown in Figures 12.11 and 12.12, where the thin lines represent the RRT tree, the thick dashed

line is the RRT path generated by Algorithm 11, and the thick solid line is the smoothed path generated by Algorithm 12.

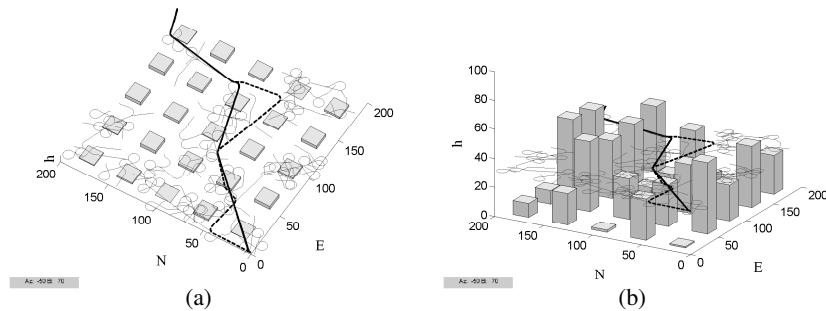


Figure 12.11: (a) Overhead view of the results of the RRT Dubins path planning algorithm. (b) Side view of the results of the RRT Dubins path planning algorithm. The thin lines are the RRT graph, the thick dashed line is the RRT path returned by Algorithm 11, and the thick solid line is the smoothed path. The RRT graph is generated at a fixed height above the terrain.

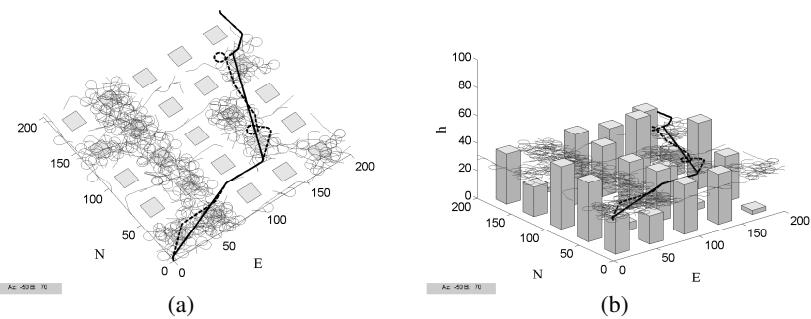


Figure 12.12: (a) Overhead view of the results of the 3-D RRT waypoint path planning algorithm. (b) Side view of the results of the 3-D RRT waypoint path planning algorithm. The thin lines are the RRT graph, the thick dashed line is the RRT path returned by Algorithm 11, and the thick solid line is the smoothed path. The RRT graph is generated at a fixed height above the terrain.

12.2 COVERAGE ALGORITHMS

In this section we will briefly discuss coverage algorithms, where the objective is not to transition from a start configuration to an end configuration, but rather to cover as much area as possible. Coverage algorithms are used, for example, in search problems, where the air vehicle is searching for objects of interest within a given region. Since the location of the object may be unknown, the region must be searched as uniformly as possible. The particular algorithm that we present in this chapter allows for prior information about possible locations of objects to be included.

The basic idea is to maintain two maps in memory: the terrain map and the return map. The terrain map is used to detect possible collisions with the environment. We will model the benefit of being at a particular location in the terrain by a return value Υ_i , where i indexes the location in the terrain. To ensure uniform coverage of an area, the return map is initialized so that all locations in the terrain have the same initial return value. As locations are visited, the return value is decremented by a fixed amount according to

$$\Upsilon_i[k] = \Upsilon_i[k - 1] - c, \quad (12.2)$$

where c is a positive constant. The path planner searches for paths that provide the largest possible return over a finite look-ahead window.

The basic coverage algorithm is listed in Algorithm 13. At each step of the algorithm, a look-ahead tree is generated from the current MAV configuration and used to search for regions of the terrain with a large return value. In Line 1 the look-ahead tree is initialized to the start configuration p_s , and in Line 8 the look-ahead tree is reset to the current configuration. More advanced algorithms could be designed to retain portions of the look-ahead tree that have already been explored. The return map is initialized in Line 2. We initialize the return map to be a large constant number plus additive noise. The additive noise facilitates choices in the initial stages of the algorithm, where all regions of the terrain need to be searched and, therefore, produce equal return values. After the MAV has moved into a new region of the terrain, the return map Υ is updated in Line 9 according to Equation 12.2. In Line 5 the look-ahead tree is generated, starting from the current configuration. The look-ahead tree is generated in a way that avoids obstacles in the terrain. In Line 6 the look-ahead tree G is searched for the path that produces the largest return value.

There are several techniques that can be used to generate the look-ahead tree in Line 5 of Algorithm 13; we will briefly describe two possible methods. In the first method, the path is given by waypoints, and the look-ahead tree is generated by taking finite steps of length L and, at the end of each step, by allowing the MAV to move straight or change heading by $\pm\vartheta$ at

Algorithm 13 Plan Cover Path: $\text{planCover}(\mathcal{T}, \Upsilon, \mathbf{p})$

Input: Terrain map \mathcal{T} , return map Υ , initial configuration \mathbf{p}_s .

- 1: Initialize look-ahead tree $G = (V, E)$ as $V = \{\mathbf{p}_s\}$, $E = \emptyset$
- 2: Initialize return map $\Upsilon = \{\Upsilon_i : i \text{ indexes the terrain}\}$
- 3: $\mathbf{p} = \mathbf{p}_s$
- 4: **for** Each planning cycle **do**
- 5: $G = \text{generateTree}(\mathbf{p}, \mathcal{T}, \Upsilon)$
- 6: $\mathcal{W} = \text{highestReturnPath}(G)$
- 7: Update \mathbf{p} by moving along the first segment of \mathcal{W}
- 8: Reset $G = (V, E)$ as $V = \{\mathbf{p}\}$, $E = \emptyset$
- 9: $\Upsilon = \text{updateReturnMap}(\Upsilon, \mathbf{p})$
- 10: **end for**

each configuration. A three-step look-ahead tree where $\vartheta = \pi/6$ is shown in Figure 12.13.

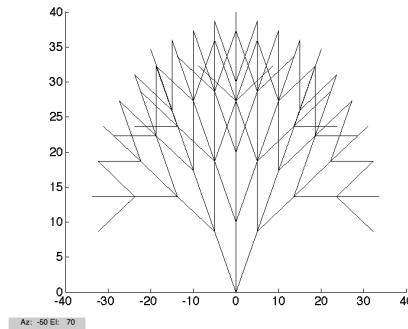


Figure 12.13: A look-ahead tree of depth three is generated from the position $(0, 0)$, where $L = 10$, and $\vartheta = \pi/6$.

The results of using a waypoint look-ahead tree in Algorithm 13 are shown in Figure 12.14. Figure 12.14 (a) shows paths through an obstacle field where the look-ahead length is $L = 5$, the allowed heading change at each step is $\vartheta = \pi/6$, and the depth of the look-ahead tree is three. The associated return map after 200 iterations of the algorithm is shown in Figure 12.14 (b). Figure 12.14 (c) shows paths through an obstacle field where the look-ahead length is $L = 5$, the allowed heading change at each step is $\vartheta = \pi/3$, and the depth of the look-ahead tree is three. The associated return map after 200 iterations of the algorithm is shown in Figure 12.14 (d). Note that the area is approximately uniformly covered, but that paths through regions are often repeated, especially through tight regions.

Another method that can be used to generate the look-ahead tree in Line 5

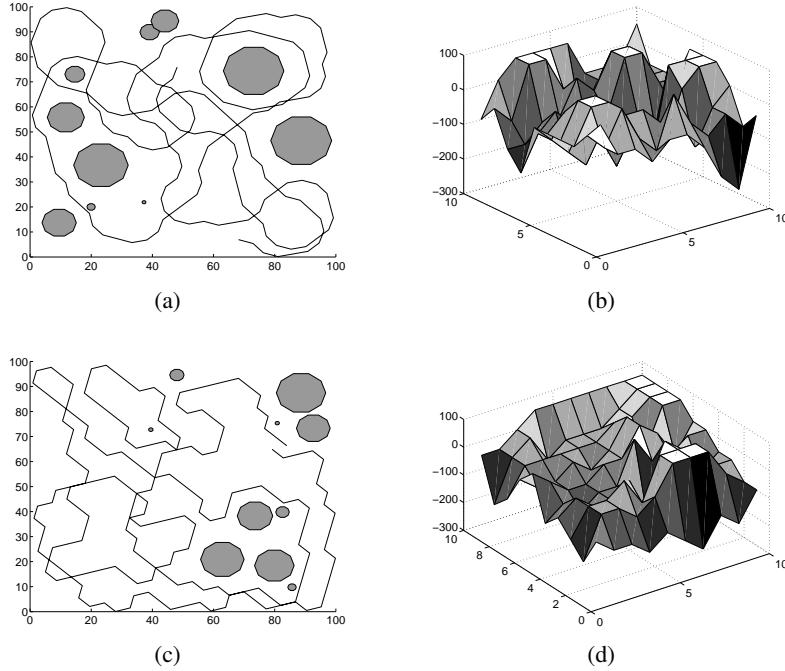


Figure 12.14: Plots (a) and (c) provide an overhead view of the results of the coverage algorithm using a three-angle expansion tree. The associated return maps after 200 planning cycles are shown in plots (b) and (d). In (a) and (b), the allowed heading change is $\vartheta = \pi/6$, and in (c) and (d), the allowed heading change is $\vartheta = \pi/3$.

of Algorithm 13 is an RRT algorithm using Dubins paths. Given the current configuration, N steps of the RRT tree expansion algorithm are used to generate the look-ahead tree. Results using this algorithm in a 3-D urban environment are shown in Figure 12.15. Figures 12.15(a) and 12.15(d) show overhead views of two different instances of the algorithm. The altitude of the MAV is fixed at a certain height, so buildings prevent certain regions from being searched. A side view of the results are shown in Figures 12.15(b) and 12.15(e) to provide the 3-D perspective. The associated return maps are shown in Figures 12.15(c) and 12.15(f). The results again show that the area is covered fairly uniformly, but it also highlights that the coverage algorithm is not particularly efficient.

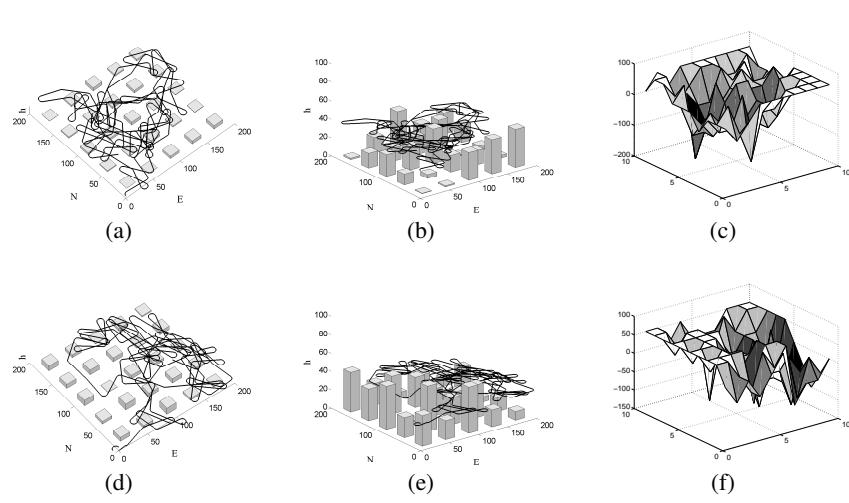


Figure 12.15: Plots (a) and (d) provide an overhead view of the results of the coverage algorithm using the RRT planner to find new regions using Dubins paths between configurations. Plots (b) and (e) show side views of the results. Plots (c) and (f) show the return map after 100 iterations of the search algorithm.

12.3 CHAPTER SUMMARY

This chapter has provided a brief introduction to path planning methods for small unmanned aircraft. The algorithms presented in this chapter are not intended to be complete or even the best algorithms for MAVs. Rather we have selected algorithms that are easy to understand and implement and that provide a good springboard for further research. We have presented two classes of algorithms: point-to-point algorithms for planning paths between two configurations, and coverage algorithms for planning paths that uniformly cover a region, given the constraints of the obstacle field. Our primary focus has been the use of the rapidly exploring random tree (RRT) algorithm using Dubins paths between nodes.

NOTES AND REFERENCES

There is extensive literature on path planning methods, including the textbooks by Latombe [46], Choset, et al. [?], and LaValle [47], which contain thorough reviews of related path planning research. An introduction to the Voronoi graph is contained in [?] and early applications of Voronoi techniques to UAV path planning are described in [39, 48, ?, ?]. Incremental construction of Voronoi graphs based on sensor measurements is described

in [?, 49]. An effective search technique for Voronoi diagrams providing multiple path options is the Eppstein's k -shortest paths algorithm [?].

The RRT algorithm was first introduced in [?] and applied to nonholonomic robotic vehicles in [50, 45]. There are numerous applications of RRTs reported in the literature, as well as extensions to the basic algorithms [51]. A recent extension to the RRT algorithm that converges with probability one to the optimal path is described in [52]. The RRT algorithm is closely related to the probabilistic roadmap technique described in [53], which is applied to UAVs in [?].

There are several coverage algorithms discussed in the literature. Reference [54] describes a coverage algorithm that plans paths such that the robot passes over all points in its free space and also includes a nice survey of other coverage algorithms reported in the literature. A coverage algorithm in the presence of moving obstacles is described in [55]. Multiple vehicle coverage algorithms are discussed in [56], and coverage algorithms in the context of mobile sensor networks are described in [57, 58].

12.4 DESIGN PROJECT

The objective of this assignment is to implement several of the path planning algorithms described in this chapter. Skeleton code for this chapter is given on the website. The file `createWorld.m` creates a map similar to those shown in Figures 12.11 and 12.12. The file `drawEnvironment.m` draws the map, the waypoint path, and the current straight-line or orbit that is being followed. The file `path_planner.m` contains a switch statement for manually choosing between different path planning algorithms. The sample code contains the file `planRRT.m` for planning straight line paths through a synthetic urban terrain.

- 12.1 Using `planRRT.m` as a template, create `planRRTDubins.m` and modify `path_planner.m` so that it calls `planRRTDubins.m` to plan Dubins paths through the map. Modify `planRRTDubins.m` to implement the RRT algorithm based on Dubins paths and the associated smoothing algorithm. Test and debug the algorithm on the guidance model given in Equation (9.18). When the algorithm is working well on the guidance model, verify that it performs adequately for the full six-DOF model.
- 12.2 Using `planCover.m` as a template, create `planCoverRRTDubins.m` and modify `path_planner.m` so that it calls the function `planRRTCoverDubins.m`. Modify `planCoverRRTDubins.m` to implement the coverage algorithm described in Algorithm 13 where the motion of the vehicle is

based on Dubins paths. Test and debug the algorithm on the guidance model given in Equation (9.18). When the algorithm is working well on the guidance model, verify that it performs adequately for the full six-DOF model.

Chapter Thirteen

Vision-guided Navigation

One of the primary reasons for the current interest in small unmanned aircraft is that they offer an inexpensive platform to carry electro-optical (EO) and infrared (IR) cameras. Almost all small and miniature air vehicles that are currently deployed carry either an EO or IR camera. While the camera's primary use is to relay information to a user, it makes sense to attempt to also use the camera for the purpose of navigation, guidance, and control. Further motivation comes from the fact that birds and flying insects use vision as their primary guidance sensor [59].

This chapter briefly introduces some of the issues that arise in vision-based guidance and control of MAVs. In Section 13.1 we revisit coordinate frame geometry and expand upon the discussion in Chapter 2 by introducing the gimbal and camera frames. We also discuss the image plane and the projective geometry that relates the position of 3-D objects to their 2-D projection on the image plane. In Section 13.2 we give a simple algorithm for pointing a pan-tilt gimbal at a known world coordinate. In Section 13.3 we describe a geolocation algorithm that estimates the position of a ground-based target based on the location and motion of the target in the video sequence. In this chapter we will assume that an algorithm exists for tracking the features of a target in the video sequence. The motion of the target on the image plane is influenced by both the target motion and by the translational and rotational motion of the aircraft. In Section 13.4 we describe a method that compensates for the apparent target motion that is induced by gimbal movement and angular rates of the air platform. As a final application of vision-based guidance, Section 13.6 describes an algorithm that uses vision to land accurately at a user-specified location on the ground.

13.1 GIMBAL AND CAMERA FRAMES AND PROJECTIVE GEOMETRY

In this section we will assume that the origins of the gimbal and camera frames are located at the center of mass of the vehicle. For more general geometry, see [60]. Figure 13.1 shows the relationship between the vehicle and body frames of the MAV and the gimbal and camera frames. There are three frames of interest: the gimbal-1 frame denoted by $\mathcal{F}^{g1} = (\mathbf{i}^{g1}, \mathbf{j}^{g1}, \mathbf{k}^{g1})$, the

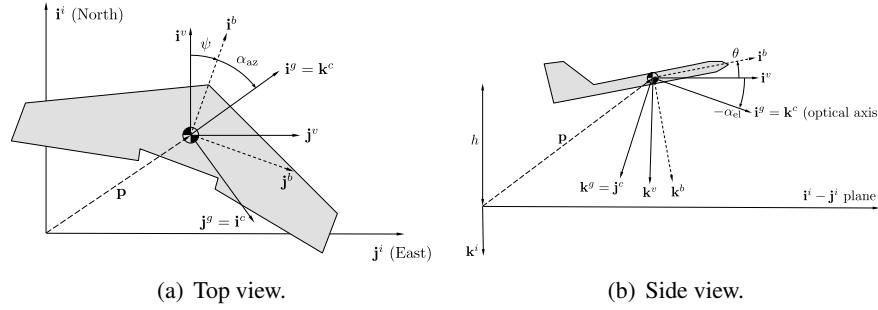


Figure 13.1: A graphic showing the relationship between the gimbal and camera frames and the vehicle and body frames.

gimbal frame denoted by $\mathcal{F}^g = (\mathbf{i}^g, \mathbf{j}^g, \mathbf{k}^g)$, and the camera frame denoted by $\mathcal{F}^c = (\mathbf{i}^c, \mathbf{j}^c, \mathbf{k}^c)$. The gimbal-1 frame is obtained by rotating the body frame about the \mathbf{k}^b axis by an angle of α_{az} , which is called the gimbal azimuth angle. The rotation from the body to the gimbal-1 frame is given by

$$\mathcal{R}_b^{g1}(\alpha_{az}) \triangleq \begin{pmatrix} \cos \alpha_{az} & \sin \alpha_{az} & 0 \\ -\sin \alpha_{az} & \cos \alpha_{az} & 0 \\ 0 & 0 & 1 \end{pmatrix}. \quad (13.1)$$

The gimbal frame is obtained by rotating the gimbal-1 frame about the \mathbf{j}^{g1} axis by an angle of α_{el} , which is called the gimbal elevation angle. Note that a negative elevation angle points the camera toward the ground. The rotation from the gimbal-1 frame to the gimbal frame is given by

$$\mathcal{R}_{g1}^g(\alpha_{el}) \triangleq \begin{pmatrix} \cos \alpha_{el} & 0 & -\sin \alpha_{el} \\ 0 & 1 & 0 \\ \sin \alpha_{el} & 0 & \cos \alpha_{el} \end{pmatrix}. \quad (13.2)$$

The rotation from the body to the gimbal frame is, therefore, given by

$$\mathcal{R}_b^g = \mathcal{R}_{g1}^g \mathcal{R}_b^{g1} = \begin{pmatrix} \cos \alpha_{el} \cos \alpha_{az} & \cos \alpha_{el} \sin \alpha_{az} & -\sin \alpha_{el} \\ -\sin \alpha_{az} & \cos \alpha_{az} & 0 \\ \sin \alpha_{el} \cos \alpha_{az} & \sin \alpha_{el} \sin \alpha_{az} & \cos \alpha_{el} \end{pmatrix}. \quad (13.3)$$

The literature in computer vision and image processing traditionally aligns the coordinate axis of the camera such that \mathbf{i}^c points to the right in the image, \mathbf{j}^c points down in the image, and \mathbf{k}^c points along the optical axis. It follows that the transformation from the gimbal frame to the camera frame is given

by

$$\mathcal{R}_g^c = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}. \quad (13.4)$$

13.1.1 Camera Model

The geometry in the camera frame is shown in Figure 13.2, where f is the focal length in units of pixels and P converts pixels to meters. To simplify the discussion, we will assume that the pixels and the pixel array are square. If the width of the square pixel array in units of pixels is M and the field-of-view of the camera v is known, then the focal length f can be expressed as

$$f = \frac{M}{2 \tan\left(\frac{v}{2}\right)}. \quad (13.5)$$

The location of the projection of the object is expressed in the camera frame as $(P\epsilon_x, P\epsilon_y, Pf)$, where ϵ_x and ϵ_y are the pixel location (in units of pixels) of the object. The distance from the origin of the camera frame to the pixel location (ϵ_x, ϵ_y) , as shown in Figure 13.2, is F where

$$F = \sqrt{f^2 + \epsilon_x^2 + \epsilon_y^2}. \quad (13.6)$$

Using similar triangles in Figure 13.2, we get

$$\frac{\ell_x^c}{\mathbb{L}} = \frac{P\epsilon_x}{PF} = \frac{\epsilon_x}{F}. \quad (13.7)$$

Similarly, we get that $\ell_y^c/\mathbb{L} = \epsilon_y/F$ and $\ell_z^c/\mathbb{L} = f/F$. Combining, we have that

$$\ell^c = \frac{\mathbb{L}}{F} \begin{pmatrix} \epsilon_x \\ \epsilon_y \\ f \end{pmatrix}, \quad (13.8)$$

where ℓ is the vector to the object of interest and $\mathbb{L} = \|\ell\|$.

Note that ℓ^c cannot be determined strictly from camera data since \mathbb{L} is unknown. However, we can determine the unit direction vector to the target as

$$\frac{\ell^c}{\mathbb{L}} = \frac{1}{F} \begin{pmatrix} \epsilon_x \\ \epsilon_y \\ f \end{pmatrix} = \frac{1}{\sqrt{\epsilon_x^2 + \epsilon_y^2 + f^2}} \begin{pmatrix} \epsilon_x \\ \epsilon_y \\ f \end{pmatrix}. \quad (13.9)$$

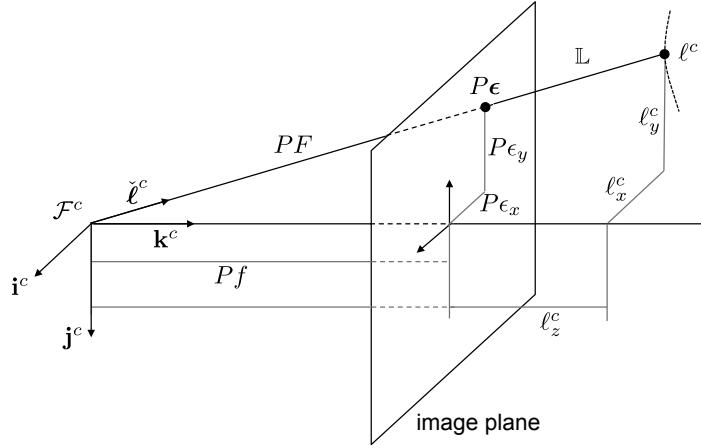


Figure 13.2: The camera frame. The target in the camera frame is represented by ℓ^c . The projection of the target onto the image plane is represented by ϵ . The pixel location $(0,0)$ corresponds to the center of the image, which is assumed to be aligned with the optical axis. The distance to the target is given by \mathbb{L} . ϵ and f are in units of pixels. ℓ is in units of meters.

Since the unit vector ℓ^c/\mathbb{L} plays a major role throughout this chapter, we will use the notation

$$\check{\ell} \triangleq \begin{pmatrix} \check{\ell}_x \\ \check{\ell}_y \\ \check{\ell}_z \end{pmatrix} \triangleq \frac{\ell}{\mathbb{L}}$$

to denote the normalized version of ℓ .

13.2 GIMBAL POINTING

Small and miniature air vehicles are used primarily for intelligence, surveillance, and reconnaissance (ISR) tasks. If the MAV is equipped with a gimbal, this involves maneuvering the gimbal so that the camera points at certain objects. The objective of this section is to describe a simple gimbal-pointing algorithm. We assume a pan-tilt gimbal and that the equations of motion for the gimbal are given by

$$\begin{aligned}\dot{\alpha}_{az} &= u_{az} \\ \dot{\alpha}_{el} &= u_{el},\end{aligned}$$

where u_{az} and u_{el} are control variables for the gimbal's azimuth and elevation angles, respectively.

We will consider two pointing scenarios. In the first scenario, the objective is to point the gimbal at a given world coordinate. In the second scenario, the objective is to point the gimbal so that the optical axis aligns with a certain point in the image plane. For the second scenario, we envision a user watching a video stream from the MAV and using a mouse to click on a location in the image plane. The gimbal is then maneuvered to push that location to the center of the image plane.

For the first scenario, let $\mathbf{p}_{\text{obj}}^i$ be the known location of an object in the inertial frame. The objective is to align the optical axis of the camera with the desired relative position vector

$$\boldsymbol{\ell}_d^i \triangleq \mathbf{p}_{\text{obj}}^i - \mathbf{p}_{\text{MAV}}^i,$$

where $\mathbf{p}_{\text{MAV}}^i = (p_n, p_e, p_d)^\top$ is the inertial position of the MAV and where the subscript d indicates a desired quantity. The body-frame unit vector that points in the desired direction of the object is given by

$$\check{\boldsymbol{\ell}}_d^b = \frac{1}{\|\boldsymbol{\ell}_d^i\|} \mathcal{R}_i^b \boldsymbol{\ell}_d^i.$$

For the second scenario, suppose that we desire to maneuver the gimbal so that the pixel location ϵ is pushed to the center of the image. Using Equation (13.9), the desired direction of the optical axis in the camera frame is given by

$$\check{\boldsymbol{\ell}}_d^c = \frac{1}{\sqrt{f^2 + \epsilon_x^2 + \epsilon_y^2}} \begin{pmatrix} \epsilon_x \\ \epsilon_y \\ f \end{pmatrix}.$$

In the body frame, the desired direction of the optical axis is

$$\check{\boldsymbol{\ell}}_d^b = \mathcal{R}_g^b \mathcal{R}_c^g \check{\boldsymbol{\ell}}_d^c.$$

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

$$\check{\ell}_d^b \triangleq \begin{pmatrix} \check{\ell}_{xd}^b \\ \check{\ell}_{yd}^b \\ \check{\ell}_{zd}^b \end{pmatrix} = \mathcal{R}_g^b(\alpha_{az}^c, \alpha_{el}^c) \mathcal{R}_c^g \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad (13.10)$$

$$= \begin{pmatrix} \cos \alpha_{el}^c \cos \alpha_{az}^c & -\sin \alpha_{az}^c & -\sin \alpha_{el}^c \cos \alpha_{az}^c \\ \cos \alpha_{el}^c \sin \alpha_{az}^c & \cos \alpha_{az}^c & -\sin \alpha_{el}^c \sin \alpha_{az}^c \\ \sin \alpha_{el}^c & 0 & \cos \alpha_{el}^c \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad (13.11)$$

$$= \begin{pmatrix} \cos \alpha_{el}^c \cos \alpha_{az}^c \\ \cos \alpha_{el}^c \sin \alpha_{az}^c \\ \sin \alpha_{el}^c \end{pmatrix}. \quad (13.12)$$

Solving for α_{el}^c and α_{az}^c gives the desired azimuth and elevation angles as

$$\alpha_{az}^c = \tan^{-1} \left(\frac{\check{\ell}_{yd}^b}{\check{\ell}_{xd}^b} \right) \quad (13.13)$$

$$\alpha_{el}^c = \sin^{-1} \left(\frac{\check{\ell}_{zd}^b}{\check{\ell}_{xd}^b} \right). \quad (13.14)$$

The gimbal servo commands can be selected as

$$u_{az} = k_{az}(\alpha_{az}^c - \alpha_{az}) \quad (13.15)$$

$$u_{el} = k_{el}(\alpha_{el}^c - \alpha_{el}),$$

where k_{az} and k_{el} are positive control gains.

13.3 GEOLOCATION

This section presents a method for determining the location of objects in world/inertial coordinates using a gimbaled EO/IR camera on board a fixed-wing MAV. We assume that the MAV can measure its own world coordinates using, for example, a GPS receiver, and that other MAV state variables are also available.

Following Section 13.1, let $\ell = \mathbf{p}_{obj} - \mathbf{p}_{MAV}$ be the relative position vector between the target of interest and the MAV, and define $\mathbb{L} = \|\ell\|$ and $\check{\ell} = \ell/\mathbb{L}$. From geometry, we have the relationship

$$\begin{aligned} \mathbf{p}_{obj}^i &= \mathbf{p}_{MAV}^i + \mathcal{R}_b^i \mathcal{R}_g^b \mathcal{R}_c^g \ell^c \\ &= \mathbf{p}_{MAV}^i + \mathbb{L} \left(\mathcal{R}_b^i \mathcal{R}_g^b \mathcal{R}_c^g \check{\ell}^c \right), \end{aligned} \quad (13.16)$$

where $\mathbf{p}_{\text{MAV}}^i = (p_n, p_e, p_d)^\top$, $\mathcal{R}_b^i = \mathcal{R}_b^i(\phi, \theta, \psi)$, and $\mathcal{R}_g^b = \mathcal{R}_g^b(\alpha_{\text{az}}, \alpha_{\text{el}})$. The only element on the right-hand side of Equation (13.16) that is unknown is \mathbb{L} . Therefore, solving the geolocation problem reduces to the problem of estimating the range to the target \mathbb{L} .

13.3.1 Range to Target Using the Flat-earth Model

If the MAV is able to measure height-above-ground, then a simple strategy for estimating \mathbb{L} is to assume a flat-earth model [60]. Figure 13.3 shows the geometry of the situation, where $h = -p_d$ is the height-above-ground, and φ is the angle between ℓ and the \mathbf{k}^i axis. It is clear from Figure 13.3 that

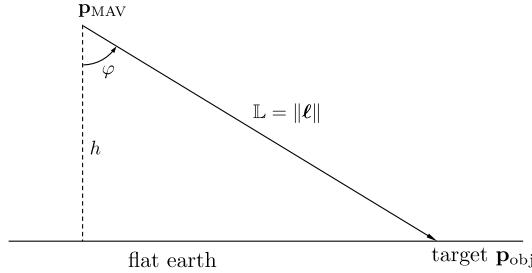


Figure 13.3: Range estimation using the flat-earth assumption.

$$\mathbb{L} = \frac{h}{\cos \varphi},$$

where

$$\begin{aligned}\cos \varphi &= \mathbf{k}^i \cdot \check{\ell}^i \\ &= \mathbf{k}^i \cdot \mathcal{R}_b^i \mathcal{R}_g^b \mathcal{R}_c^g \check{\ell}^c.\end{aligned}$$

Therefore, the range estimate using the flat-earth model is given by

$$\mathbb{L} = \frac{h}{\mathbf{k}^i \cdot \mathcal{R}_b^i \mathcal{R}_g^b \mathcal{R}_c^g \check{\ell}^c}. \quad (13.17)$$

The geolocation estimate is given by combining Equations (13.16) and (13.17) to obtain

$$\mathbf{p}_{\text{obj}}^i = \begin{pmatrix} p_n \\ p_e \\ p_d \end{pmatrix} + h \frac{\mathcal{R}_b^i \mathcal{R}_g^b \mathcal{R}_c^g \check{\ell}^c}{\mathbf{k}^i \cdot \mathcal{R}_b^i \mathcal{R}_g^b \mathcal{R}_c^g \check{\ell}^c}. \quad (13.18)$$

13.3.2 Geolocation Using an Extended Kalman Filter

The geolocation estimate in Equation (13.18) provides a one-shot estimate of the target location. Unfortunately, this equation is highly sensitive to measurement errors, especially attitude estimation errors of the airframe. In this section we will describe the use of the extended Kalman filter (EKF) to solve the geolocation problem.

Rearranging Equation (13.16), we get

$$\mathbf{p}_{\text{MAV}}^i = \mathbf{p}_{\text{obj}}^i - \mathbb{L} \left(\mathcal{R}_b^i \mathcal{R}_g^b \mathcal{R}_c^g \check{\ell}^c \right), \quad (13.19)$$

which, since $\mathbf{p}_{\text{MAV}}^i$ is measured by GPS, will be used as the measurement equation, assuming that GPS noise is zero-mean Gaussian. However, since GPS measurement error contains a constant bias, the geolocation error will also contain a bias. If we assume that the object is stationary, we have

$$\dot{\mathbf{p}}_{\text{obj}}^i = 0.$$

Since $\mathbb{L} = \left\| \mathbf{p}_{\text{obj}}^i - \mathbf{p}_{\text{MAV}}^i \right\|$, we have

$$\begin{aligned} \dot{\mathbb{L}} &= \frac{d}{dt} \sqrt{(\mathbf{p}_{\text{obj}}^i - \mathbf{p}_{\text{MAV}}^i)^\top (\mathbf{p}_{\text{obj}}^i - \mathbf{p}_{\text{MAV}}^i)} \\ &= \frac{(\mathbf{p}_{\text{obj}}^i - \mathbf{p}_{\text{MAV}}^i)^\top (\dot{\mathbf{p}}_{\text{obj}}^i - \dot{\mathbf{p}}_{\text{MAV}}^i)}{\mathbb{L}} \\ &= -\frac{(\mathbf{p}_{\text{obj}}^i - \mathbf{p}_{\text{MAV}}^i)^\top \dot{\mathbf{p}}_{\text{MAV}}^i}{\mathbb{L}}, \end{aligned}$$

where for constant-altitude flight, $\dot{\mathbf{p}}_{\text{MAV}}^i$ can be approximated as

$$\dot{\mathbf{p}}_{\text{MAV}}^i = \begin{pmatrix} \hat{V}_g \cos \hat{\chi} \\ \hat{V}_g \sin \hat{\chi} \\ 0 \end{pmatrix},$$

and where \hat{V}_g and $\hat{\chi}$ are estimated using the EKF discussed in Section 8.10.

A block diagram of the geolocation algorithm is shown in Figure 13.4. The input to the geolocation algorithm is the position and the velocity of the MAV in the inertial frame as estimated by the GPS smoother described in Section 8.10, the estimate of the normalized line-of-sight vector as given in Equation (13.9), and the attitude as estimated by the scheme described in Section 8.9.

The geolocation algorithm is an extended Kalman filter with state $\hat{x} =$

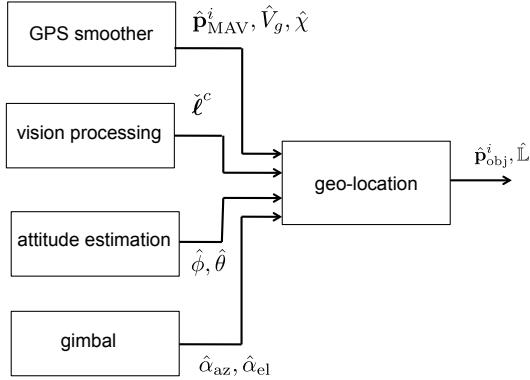


Figure 13.4: The geolocation algorithm uses the output of the GPS smoother, the normalized line-of-sight vector from the vision algorithm, and the attitude to estimate the position of the object in the inertial frame and the distance to the object.

$(\hat{p}_\text{obj}^{i\top} \hat{\mathbb{L}})^\top$ and prediction equations given by

$$\begin{pmatrix} \dot{\hat{p}}_\text{obj}^i \\ \dot{\hat{\mathbb{L}}} \end{pmatrix} = \begin{pmatrix} 0 \\ -\frac{(\hat{p}_\text{obj}^i - \hat{p}_\text{MAV}^i)^\top \dot{\hat{p}}_\text{MAV}^i}{\hat{\mathbb{L}}} \end{pmatrix}.$$

The Jacobian is therefore given by

$$\frac{\partial f}{\partial x} = \begin{pmatrix} 0 & 0 \\ -\frac{\dot{\hat{p}}_\text{MAV}^{iT}}{\hat{\mathbb{L}}} & \frac{(\hat{p}_\text{obj}^i - \hat{p}_\text{MAV}^i)^\top \dot{\hat{p}}_\text{MAV}^i}{\hat{\mathbb{L}}^2} \end{pmatrix}.$$

The output equation is given by Equation (13.19), where the Jacobian of the output equation is

$$\frac{\partial h}{\partial x} = (I \quad \mathcal{R}_b^i \mathcal{R}_g^b \mathcal{R}_c^g \check{\ell}^c).$$

13.4 ESTIMATING TARGET MOTION IN THE IMAGE PLANE

We assume in this chapter that a computer vision algorithm is used to track the pixel location of the target. Since the video stream often contains noise and the tracking algorithms are imperfect, the pixel location returned by these algorithms is noisy. The guidance algorithm described in the next section needs both the pixel location of the target and the pixel velocity of the target. In Section 13.4.1 we show how to construct a simple low-pass filter that returns a filtered version of both the pixel location and the pixel velocity.

The pixel velocity is influenced by both the relative (translational) motion between the target and the MAV, and the rotational motion of the MAV-gimbal combination. In Section 13.4.2 we derive an explicit expression for pixel velocity and show how to compensate for the apparent motion induced by the rotational rates of the MAV and gimbal.

13.4.1 Digital Low-pass Filter and Differentiation

Let $\bar{\epsilon} = (\bar{\epsilon}_x, \bar{\epsilon}_y)^\top$ denote the raw pixel measurements, $\epsilon = (\epsilon_x, \epsilon_y)^\top$ denote the filtered pixel location, and $\dot{\epsilon} = (\dot{\epsilon}_x, \dot{\epsilon}_y)^\top$ denote the filtered pixel velocity. The basic idea is to low-pass filter the raw pixel measurements as

$$\epsilon(s) = \frac{1}{\tau s + 1} \bar{\epsilon}, \quad (13.20)$$

and to differentiate the raw pixel measurements as

$$\dot{\epsilon}(s) = \frac{s}{\tau s + 1} \bar{\epsilon}. \quad (13.21)$$

Using the Tustin approximation [61],

$$s \mapsto \frac{2}{T_s} \frac{z - 1}{z + 1},$$

to convert to the z -domain gives

$$\begin{aligned} \epsilon[z] &= \frac{1}{\frac{2\tau}{T_s} \frac{z-1}{z+1} + 1} \bar{\epsilon} = \frac{\frac{T_s}{2\tau+T_s} (z+1)}{z - \frac{2\tau-T_s}{2\tau+T_s}} \bar{\epsilon} \\ \dot{\epsilon}[z] &= \frac{\frac{2}{T_s} \frac{z-1}{z+1}}{\frac{2\tau}{T_s} \frac{z-1}{z+1} + 1} \bar{\epsilon} = \frac{\frac{2}{2\tau+T_s} (z-1)}{z - \frac{2\tau-T_s}{2\tau+T_s}} \bar{\epsilon}. \end{aligned}$$

Taking the inverse z -transform gives the difference equations

$$\begin{aligned} \epsilon[n] &= \left(\frac{2\tau - T_s}{2\tau + T_s} \right) \epsilon[n-1] + \left(\frac{T_s}{2\tau + T_s} \right) (\bar{\epsilon}[n] + \bar{\epsilon}[n-1]) \\ \dot{\epsilon}[n] &= \left(\frac{2\tau - T_s}{2\tau + T_s} \right) \dot{\epsilon}[n-1] + \left(\frac{2}{2\tau + T_s} \right) (\bar{\epsilon}[n] - \bar{\epsilon}[n-1]), \end{aligned}$$

where $\epsilon[0] = \bar{\epsilon}[0]$ and $\dot{\epsilon}[0] = 0$.

13.4.2 Apparent Motion Due to Rotation

Motion of the target on the image plane is induced by both relative translational motion of the target with respect to the MAV, as well as rotational

motion of the MAV and gimbal platform. For most guidance tasks, we are primarily interested in the relative translational motion and desire to remove the apparent motion due to the rotation of the MAV and gimbal platforms. Following the notation introduced in Section 13.1, let $\check{\ell} \triangleq \ell/\mathbb{L} = (\mathbf{p}_{\text{obj}} - \mathbf{p}_{\text{MAV}})/\|\mathbf{p}_{\text{obj}} - \mathbf{p}_{\text{MAV}}\|$ be the normalized relative position vector between the target and the MAV. Using the Coriolis formula in Equation (2.18), we get

$$\frac{d\check{\ell}}{dt_i} = \frac{d\check{\ell}}{dt_c} + \boldsymbol{\omega}_{c/i}^c \times \check{\ell}. \quad (13.22)$$

The expression on the left-hand side of Equation (13.22) is the true relative translation motion between the target and the MAV. The first expression on the right-hand side of Equation (13.22) is the motion of the target on the image plane which can be computed from camera information. The second expression on the right-hand side of Equation (13.22) is the apparent motion due to the rotation of the MAV and gimbal platform. Equation (13.22) can be expressed in the camera frame as

$$\frac{d\check{\ell}^c}{dt_i} = \frac{d\check{\ell}^c}{dt_c} + \boldsymbol{\omega}_{c/i}^c \times \check{\ell}^c. \quad (13.23)$$

The first expression on the right-hand side of Equation (13.23) can be computed as

$$\begin{aligned} \frac{d\check{\ell}^c}{dt_c} &= \frac{d}{dt_c} \frac{\begin{pmatrix} \epsilon_x \\ \epsilon_y \\ f \end{pmatrix}}{F} = \frac{F \begin{pmatrix} \dot{\epsilon}_x \\ \dot{\epsilon}_y \\ 0 \end{pmatrix} - \dot{F} \begin{pmatrix} \epsilon_x \\ \epsilon_y \\ f \end{pmatrix}}{F^2} = \frac{F \begin{pmatrix} \dot{\epsilon}_x \\ \dot{\epsilon}_y \\ 0 \end{pmatrix} - \frac{\epsilon_x \dot{\epsilon}_x + \epsilon_y \dot{\epsilon}_y}{F} \begin{pmatrix} \epsilon_x \\ \epsilon_y \\ f \end{pmatrix}}{F^2} \\ &= \frac{1}{F^3} \begin{pmatrix} F^2 - \epsilon_x^2 & -\epsilon_x \epsilon_y & \\ -\epsilon_x \epsilon_y & F^2 - \epsilon_y^2 & \\ -\epsilon_x f & -\epsilon_y f & \end{pmatrix} \begin{pmatrix} \dot{\epsilon}_x \\ \dot{\epsilon}_y \\ \end{pmatrix} = \frac{1}{F^3} \begin{pmatrix} \epsilon_y^2 + f^2 & -\epsilon_x \epsilon_y \\ -\epsilon_x \epsilon_y & \epsilon_x^2 + f^2 \\ -\epsilon_x f & -\epsilon_y f \end{pmatrix} \dot{\epsilon} \\ &= Z(\boldsymbol{\epsilon}) \dot{\epsilon}, \end{aligned} \quad (13.24)$$

where

$$Z(\boldsymbol{\epsilon}) \triangleq \frac{1}{F^3} \begin{pmatrix} \epsilon_y^2 + f^2 & -\epsilon_x \epsilon_y \\ -\epsilon_x \epsilon_y & \epsilon_x^2 + f^2 \\ -\epsilon_x f & -\epsilon_y f \end{pmatrix} \dot{\epsilon}.$$

To compute the second term on the right-hand side of Equation (13.23), we need an expression for $\boldsymbol{\omega}_{c/i}^c$, which can be decomposed as

$$\boldsymbol{\omega}_{c/i}^c = \boldsymbol{\omega}_{c/g}^c + \boldsymbol{\omega}_{g/b}^c + \boldsymbol{\omega}_{b/i}^c. \quad (13.25)$$

Since the camera is fixed in the gimbal frame, we have that $\omega_{c/g}^c = 0$. Letting p , q , and r denote the angular body rates of the platform, as measured by onboard rate gyros that are aligned with the body frame axes, gives $\omega_{b/i}^b = (p, q, r)^\top$. Expressing $\omega_{b/i}$ in the camera frame gives

$$\omega_{b/i}^c = \mathcal{R}_g^c \mathcal{R}_b^g \omega_{b/i}^b = \mathcal{R}_g^c \mathcal{R}_b^g \begin{pmatrix} p \\ q \\ r \end{pmatrix}. \quad (13.26)$$

To derive an expression for $\omega_{g/b}$ in terms of the measured gimbal angle rates $\dot{\alpha}_{\text{el}}$ and $\dot{\alpha}_{\text{az}}$, recall that the azimuth angle α_{az} is defined with respect to the body frame, and that the elevation angle α_{el} is defined with respect to the gimbal-1 frame. The gimbal frame is obtained by rotating the gimbal-1 frame about its y -axis by α_{el} . Therefore, $\dot{\alpha}_{\text{az}}$ is defined with respect to the gimbal-1 frame, and $\dot{\alpha}_{\text{el}}$ is defined with respect to the gimbal frame. This implies that

$$\omega_{g/b}^b = \mathcal{R}_{g1}^b(\alpha_{\text{az}}) \mathcal{R}_g^{g1}(\alpha_{\text{el}}) \begin{pmatrix} 0 \\ \dot{\alpha}_{\text{el}} \\ 0 \end{pmatrix} + \mathcal{R}_{g1}^b(\alpha_{\text{az}}) \begin{pmatrix} 0 \\ 0 \\ \dot{\alpha}_{\text{az}} \end{pmatrix}^{g1}.$$

Noting that \mathcal{R}_g^{g1} is a y -axis rotation, we get

$$\omega_{g/b}^b = \mathcal{R}_{g1}^b(\alpha_{\text{az}}) \begin{pmatrix} 0 \\ \dot{\alpha}_{\text{el}} \\ \dot{\alpha}_{\text{az}} \end{pmatrix} = \begin{pmatrix} \cos \alpha_{\text{az}} & -\sin \alpha_{\text{az}} & 0 \\ \sin \alpha_{\text{az}} & \cos \alpha_{\text{az}} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ \dot{\alpha}_{\text{el}} \\ \dot{\alpha}_{\text{az}} \end{pmatrix} = \begin{pmatrix} -\sin(\alpha_{\text{az}})\dot{\alpha}_{\text{el}} \\ \cos(\alpha_{\text{az}})\dot{\alpha}_{\text{el}} \\ \dot{\alpha}_{\text{az}} \end{pmatrix}, \blacksquare$$

and it follows that

$$\omega_{g/b}^c = \mathcal{R}_g^c \mathcal{R}_b^g \omega_{g/b}^b = \mathcal{R}_g^c \mathcal{R}_b^g \begin{pmatrix} -\sin(\alpha_{\text{az}})\dot{\alpha}_{\text{el}} \\ \cos(\alpha_{\text{az}})\dot{\alpha}_{\text{el}} \\ \dot{\alpha}_{\text{az}} \end{pmatrix}. \quad (13.27)$$

Drawing on Equations (13.24) and (13.27), Equation (13.23) can be expressed as

$$\frac{d\check{\ell}^c}{dt_i} = Z(\epsilon)\dot{\epsilon} + \dot{\ell}_{\text{app}}^c, \quad (13.28)$$

where

$$\dot{\ell}_{\text{app}}^c \triangleq \frac{1}{F} \left[\mathcal{R}_g^c \mathcal{R}_b^g \begin{pmatrix} p - \sin(\alpha_{\text{az}})\dot{\alpha}_{\text{el}} \\ q + \cos(\alpha_{\text{az}})\dot{\alpha}_{\text{el}} \\ r + \dot{\alpha}_{\text{az}} \end{pmatrix} \right] \times \begin{pmatrix} \epsilon_x \\ \epsilon_y \\ f \end{pmatrix} \quad (13.29)$$

is the apparent motion of the normalized line-of-sight vector in the camera frame due to the rotation of the gimbal and the aircraft.

13.5 TIME TO COLLISION

For collision avoidance algorithms and for the precision landing algorithm described in Section 13.6, it is important to estimate the time to collision for objects in the camera field of view. If \mathbb{L} is the length of the line-of-sight vector between the MAV and an object, then the time to collision is given by

$$t_c \triangleq \frac{\mathbb{L}}{\dot{\mathbb{L}}}.$$

It is not possible to accurately calculate time to collision using only a monocular camera because of scale ambiguity. However, if additional side information is known, then t_c can be estimated. In Section 13.5.1 we assume that the target size in the image plane can be computed and then use that information to estimate t_c . Alternatively, in Section 13.5.2 we assume that the target is stationary on a flat earth and then use that information to estimate t_c .

13.5.1 Computing Time to Collision from Target Size

In this section we assume that the computer vision algorithm can estimate the size of the target in the image frame. Consider the geometry shown in Figure 13.5. Using similar triangles, we obtain the relationship

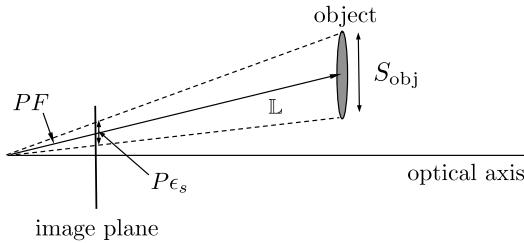


Figure 13.5: The size and growth of the target in the image frame can be used to estimate the time to collision.

$$\frac{S_{\text{obj}}}{\mathbb{L}} = \frac{P\epsilon_s}{PF} = \frac{\epsilon_s}{F}, \quad (13.30)$$

where the size of the target in meters is S_{obj} , and the size of the target in pixels is given by ϵ_s . We assume that the size of the target S_{obj} is not changing

in time. Differentiating Equation (13.30) and solving for $\dot{\mathbb{L}}/\mathbb{L}$, we obtain

$$\begin{aligned}\frac{\dot{\mathbb{L}}}{\mathbb{L}} &= \frac{\mathbb{L}}{S_{\text{obj}}} \left[\frac{\epsilon_s \dot{F}}{F} - \frac{\dot{\epsilon}_s}{F} \right] \\ &= \frac{F}{\epsilon_s} \left[\frac{\epsilon_s \dot{F}}{F} - \frac{\dot{\epsilon}_s}{F} \right] \\ &= \frac{\dot{F}}{F} - \frac{\dot{\epsilon}_s}{\epsilon_s} \\ &= \frac{\epsilon_x \dot{\epsilon}_x + \epsilon_y \dot{\epsilon}_y}{F} - \frac{\dot{\epsilon}_s}{\epsilon_s},\end{aligned}\tag{13.31}$$

the inverse of which is the time to collision t_c .

13.5.2 Computing Time to Collision from the Flat-earth Model

A popular computer vision algorithm for target tracking is to track features on the target [62]. If a feature tracking algorithm is used, then target size information may not be available and the method described in the previous section cannot be applied. In this section we describe an alternative method for computing t_c , where we assume a flat-earth model. Referring to Figure 13.3, we have

$$\mathbb{L} = \frac{h}{\cos \varphi},$$

where $h = -p_d$ is the altitude. Differentiating in the inertial frame gives

$$\begin{aligned}\frac{\dot{\mathbb{L}}}{\mathbb{L}} &= \frac{1}{\mathbb{L}} \left(\frac{\cos \varphi \dot{h} + h \dot{\varphi} \sin \varphi}{\cos^2 \varphi} \right) \\ &= \frac{\cos \varphi}{h} \left(\frac{\cos \varphi \dot{h} + h \dot{\varphi} \sin \varphi}{\cos^2 \varphi} \right) \\ &= \frac{\dot{h}}{h} + \dot{\varphi} \tan \varphi.\end{aligned}\tag{13.32}$$

In the inertial frame, we have that

$$\cos \varphi = \check{\ell}^i \cdot \mathbf{k}^i,\tag{13.33}$$

where $\mathbf{k}^i = (0, 0, 1)^\top$, and therefore

$$\cos \varphi = \check{\ell}_z^i.\tag{13.34}$$

Differentiating Equation (13.34) and solving for $\dot{\phi}$ gives

$$\dot{\phi} = -\frac{1}{\sin \varphi} \frac{d}{dt_i} \check{\ell}_z^i. \quad (13.35)$$

Therefore,

$$\dot{\phi} \tan \varphi = -\frac{1}{\cos \varphi} \frac{d}{dt_i} \check{\ell}_z^i = -\frac{1}{\check{\ell}_z^i} \frac{d}{dt_i} \check{\ell}_z^i, \quad (13.36)$$

where $d\check{\ell}_z^i/dt_i$ can be determined by rotating the right-hand side of Equation (13.28) into the inertial frame.

13.6 PRECISION LANDING

Our objective in this section is to use the camera to guide the MAV to land precisely on a visually distinct target. The problem of guiding an aerial vehicle to intercept a moving target has been well studied. Proportional navigation (PN), in particular, has been an effective guidance strategy against maneuvering targets [63]. In this section, we present a method for implementing a 3-D pure PN guidance law using only vision information provided by a two-dimensional array of camera pixels.

PN generates acceleration commands that are proportional to the (pursuer-evader) line-of-sight (LOS) rates multiplied by the closing velocity. PN is often implemented as two 2-D algorithms implemented in the horizontal and vertical planes. The LOS rate is computed in the plane of interest and PN produces a commanded acceleration in that plane. While this approach works well for roll-stabilized skid-to-turn missiles, it is not appropriate for MAV dynamics. In this section we develop 3-D algorithms, and we show how to map the commanded body-frame accelerations to roll angle and pitch rate commands.

To derive the precision-landing algorithm, we will use the six-state navigation model given by

$$\dot{p}_n = V_g \cos \chi \cos \gamma \quad (13.37)$$

$$\dot{p}_e = V_g \sin \chi \cos \gamma \quad (13.38)$$

$$\dot{p}_d = -V_g \sin \gamma \quad (13.39)$$

$$\dot{\chi} = \frac{g}{V_g} \tan \phi \quad (13.40)$$

$$\dot{\phi} = u_1 \quad (13.41)$$

$$\dot{\gamma} = u_2, \quad (13.42)$$

where (p_n, p_e, p_d) are the North-East-down position of the MAV, V_g is the ground speed (assumed constant), χ is the course angle, γ is the flight path angle, ϕ is the roll angle, g is the gravitational constant, and u_1 and u_2 are control variables. Although Equations (13.37)-(13.42) are valid for non-zero wind conditions, we will assume zero-wind conditions in the foregoing discussion of precision landing.

The objective of this section is to design a vision-based guidance law that causes a MAV to intercept a ground-based target that may be moving. The position of the target is given by \mathbf{p}_{obj} . Similarly, let \mathbf{p}_{MAV} , \mathbf{v}_{MAV} , \mathbf{a}_{MAV} denote the position, velocity, and acceleration of the MAV, respectively.

In the inertial frame, the position and velocity of the MAV can be expressed as

$$\mathbf{p}_{\text{MAV}}^i = (p_n, \ p_e, \ p_d)^\top$$

and

$$\mathbf{v}_{\text{MAV}}^i = (V_g \cos \chi \cos \gamma, \ V_g \sin \chi \cos \gamma, \ -V_g \sin \gamma)^\top.$$

In the vehicle-2 frame, however, the velocity vector of the MAV is given by

$$\mathbf{v}_{\text{MAV}}^{v_2} = (V_g, \ 0, \ 0)^\top.$$

Define $\ell = \mathbf{p}_{\text{obj}} - \mathbf{p}_{\text{MAV}}$ and $\dot{\ell} = \mathbf{v}_{\text{obj}} - \mathbf{v}_{\text{MAV}}$, and let $\mathbb{L} = \|\ell\|$. The geometry associated with the precision landing problem is shown in Figure 13.6. The proportional navigation strategy is to maneuver the MAV so

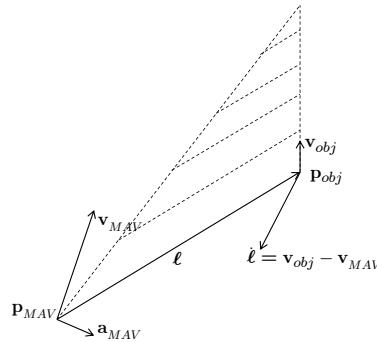


Figure 13.6: The geometry associated with precision landing.

that the line-of-sight rate $\dot{\ell}$ is aligned with the negative line-of-sight vector $-\ell$. Since $\ell \times \dot{\ell}$ is zero when $\dot{\ell}$ and ℓ are aligned, the acceleration will be

proportional to the cross product. However, since ℓ and $\dot{\ell}$ cannot be directly computed from vision data, we normalize both quantities and define

$$\Omega_{\perp} = \check{\ell} \times \frac{\dot{\ell}}{\|\dot{\ell}\|}. \quad (13.43)$$

With reference to Figure 13.6, note that Ω_{\perp} is directed into the page. Since the ground speed is not directly controllable, we will require that the commanded acceleration is perpendicular to the velocity vector of the MAV. Accordingly, let the commanded acceleration of the MAV be given by [64]

$$\mathbf{a}_{\text{MAV}} = N\Omega_{\perp} \times \mathbf{v}_{\text{MAV}}, \quad (13.44)$$

where $N > 0$ is a tunable gain and is called the navigation constant.

The acceleration commands must be converted to control inputs u_1 and u_2 , where the commanded acceleration is produced in the unrolled body frame, or the vehicle-2 frame. Therefore, the commanded acceleration \mathbf{a}_{MAV} must be resolved in the vehicle-2 frame as

$$\begin{aligned} \mathbf{a}_{\text{MAV}}^{v2} &= \mu N \Omega_{\perp}^{v2} \times \mathbf{v}_{\text{MAV}}^{v2} \\ &= \mu N \begin{pmatrix} \Omega_{\perp,x}^{v2} \\ \Omega_{\perp,y}^{v2} \\ \Omega_{\perp,z}^{v2} \end{pmatrix} \times \begin{pmatrix} V_g \\ 0 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ \mu N V_g \Omega_{\perp,z}^{v2} \\ -\mu N V_g \Omega_{\perp,y}^{v2} \end{pmatrix}. \end{aligned} \quad (13.45)$$

It is important to note that the commanded acceleration is perpendicular (by design) to the direction of motion, which is consistent with a constant-airspeed model.

The critical quantity $\Omega_{\perp} = \check{\ell} \times \frac{\dot{\ell}}{\|\dot{\ell}\|}$ must be estimated from video camera data. Our basic approach will be to estimate Ω_{\perp} in the camera frame, and then transform to the vehicle-2 frame using the expression

$$\Omega_{\perp}^{v2} = \mathcal{R}_b^v \mathcal{R}_g^b \mathcal{R}_c^g \Omega_{\perp}^c. \quad (13.46)$$

The normalized line-of-sight vector $\check{\ell}^c$ can be estimated directly from camera data using Equation (13.9). Differentiating $\ell^c / \|\ell^c\|$ gives

$$\frac{d}{dt_i} \frac{\ell^c}{\|\ell^c\|} = \frac{\mathbb{L} \dot{\ell}^c - \dot{\mathbb{L}} \ell^c}{\mathbb{L}^2} = \frac{\dot{\ell}^c}{\mathbb{L}} - \frac{\dot{\mathbb{L}}}{\mathbb{L}} \ell^c, \quad (13.47)$$

which, when combined with Equation (13.28), results in the expression

$$\frac{\dot{\ell}^c}{\mathbb{L}} = \frac{\dot{\mathbb{L}}}{\mathbb{L}} \check{\ell}^c + Z(\epsilon) \dot{\epsilon} + \dot{\ell}_{\text{app}}^c, \quad (13.48)$$

where the inverse of time to collision $\dot{\mathbb{L}}/\mathbb{L}$ can be estimated using one of the techniques discussed in Section 13.5.

Equation (13.45) gives an acceleration command in the vehicle-2 frame. In this section, we will describe how the acceleration command is converted into a roll command and a pitch rate command. The standard approach is to use polar control logic [65], which is shown in Figure 13.7. From

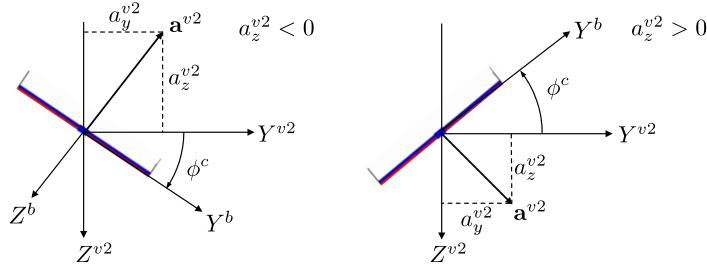


Figure 13.7: Polar converting logic that transforms an acceleration command \mathbf{a}^{v2} to a commanded roll angle ϕ^c and a commanded normal acceleration $V_g \dot{\gamma}^c$.

From Figure 13.7 it is clear that for $a_z^{v2} < 0$, we have

$$\begin{aligned}\phi^c &= \tan^{-1} \left(\frac{a_y^{v2}}{-a_z^{v2}} \right) \\ V_g \dot{\gamma}^c &= \sqrt{(a_y^{v2})^2 + (a_z^{v2})^2}.\end{aligned}$$

Similarly, when $a_z^{v2} > 0$, we have

$$\begin{aligned}\phi^c &= \tan^{-1} \left(\frac{a_y^{v2}}{a_z^{v2}} \right) \\ V_g \dot{\gamma}^c &= -\sqrt{(a_y^{v2})^2 + (a_z^{v2})^2}.\end{aligned}$$

Therefore, the general rule is

$$\phi^c = \tan^{-1} \left(\frac{a_y^{v2}}{|a_z^{v2}|} \right) \quad (13.49)$$

$$\dot{\gamma}^c = -\text{sign}(a_z^{v2}) \frac{1}{V_g} \sqrt{(a_y^{v2})^2 + (a_z^{v2})^2}. \quad (13.50)$$

Unfortunately, Equation (13.49) has a discontinuity at $(a_y^{v2}, a_z^{v2}) = (0, 0)$. For example, when $a_z^{v2} = 0$, the commanded roll angle is $\phi^c = \pi/2$ when $a_y^{v2} > 0$ and $\phi^c = -\pi/2$ when $a_y^{v2} < 0$. The discontinuity can be removed

by multiplying Equation (13.49) by the signed-sigmoidal function

$$\sigma(a_y^{v2}) = \text{sign}(a_y^{v2}) \frac{1 - e^{-ka_y^{v2}}}{1 + e^{-ka_y^{v2}}}, \quad (13.51)$$

where k is a positive control gain. The gain k adjusts the rate of the transition.

13.7 CHAPTER SUMMARY

This chapter has provided a brief introduction to the rich topic of vision-based guidance of MAVs. We have focused on three basic scenarios: gimbal pointing, geolocation, and precision landing.

NOTES AND REFERENCES

Vision-based guidance and control of MAVs is currently an active research topic (see, for example [?, ?, ?, ?, 66, 67, 68, 60, 69, 70, 71, 72]). The gimbal-pointing algorithm described in this chapter was presented in [?]. Geolocation algorithms using small MAV are described in [60, 73, ?, ?, 67, 72]. The removal of apparent motion, or egomotion, in the image plane is discussed in [74, 75, 66]. Time to collision can be estimated using structure from motion [76], ground-plane methods [77, 78], flow divergence [79], and insect-inspired methods [80]. Section 13.6 is taken primarily from [81]. Proportional navigation has been extensively analyzed in the literature. It has been shown to be optimal under certain conditions [?] and to produce zero-miss distances for a constant target acceleration[82]. If rich information regarding the time to go is available, augmented proportional navigation [83] improves the performance by adding terms that account for target and pursuer accelerations. A three-dimensional expression of PN can be found in [64, ?].

13.8 DESIGN PROJECT

- 13.1 Implement the gimbal pointing algorithm described in Section 13.2. Download the files from the website that are associated with this chapter. Modify `param.m` so that the buildings have a maximum height of one meter, and modify the path planner so that the MAV moves between two fixed locations. Use the Simulink model `mavsim_chap13_gimbal.mdl` and modify the file `point_gimbal.m` to implement the gimbal pointing algorithm given by Equations (13.13) and (13.14).
- 13.2 Implement the geolocation algorithms described in Section 13.3. Use the gimbal-pointing routine developed in the previous problem to point the gimbal at the target. Use the Simulink model `mavsim_chap13_geolocation.mdl` and modify the file `geolocation.m` to implement the geolocation algorithm described in Section 13.3.

Appendix A

Nomenclature and Notation

NOMENCLATURE

- Unit vectors along the x , y , and z axes are denoted as \mathbf{i} , \mathbf{j} , and \mathbf{k} , respectively.
- A coordinate frame is denoted by \mathcal{F} , and a superscript denotes the label of the frame. For example, \mathcal{F}^i is the inertial coordinate frame.
- Given a vector $\mathbf{p} \in \mathbb{R}^3$, the expression of \mathbf{p} in the coordinates of frame \mathcal{F}^a is denoted as \mathbf{p}^a .
- Given a vector $\mathbf{p} \in \mathbb{R}^3$, the first, second, and third components of \mathbf{p} expressed with respect to the coordinates of \mathcal{F}^a are denoted as p_x^a , p_y^a , and p_z^a , respectively.
- A rotation matrix from frame \mathcal{F}^a to \mathcal{F}^b is denoted as \mathcal{R}_a^b .
- The transpose of a matrix M is denoted as M^\top .
- Differentiation of a scalar with respect to time is denoted by a “dot” (e.g., \dot{x}). Differentiation of a vector with respect to frame \mathcal{F}^a is denoted by $\frac{d}{dt_a}$.
- Trim conditions are denoted with a star superscript. For example, x^* is the trim state. Deviations from trim are denoted by an overbar (e.g., $\bar{x} = x - x^*$).
- Commanded signals will be denoted by a superscript ‘c’. For example, the commanded course angle is χ^c and the commanded altitude is h^c .
- Zero-mean white Gaussian noise on the sensors is denoted by $\eta(t)$. The standard deviation is denoted by σ .
- A hat over a variable represents an estimate of the variable. For example, \hat{x} could be the estimate of x from an extended Kalman filter.
- Tracking error signals are denoted by \mathbf{e}_* .

- In Chapter 11 we use the notation $\overline{\mathbf{w}_a \mathbf{w}_b}$ to denote the line in \mathbb{R}^3 between waypoints \mathbf{w}_a and \mathbf{w}_b .

NOTATION

The notation is listed in alphabetical order, where we have used the romanized names of Greek letters. For example, ω is listed as if it were spelled “omega.” A “*” is used as a wild card when the same symbol is used for multiple quantities with different superscripts or subscripts. For example, a_{β_*} is used to denote a_{β_1} and a_{β_2} .

a_{β_*}	Constants for transfer function associated with side slip dynamics. (Chapter 5)
a_{ϕ_*}	Constants for transfer function associated with roll dynamics. (Chapter 5)
a_{θ_*}	Constants for transfer function associated with pitch dynamics. (Chapter 5)
a_{V_*}	Constants for transfer function associated with airspeed dynamics. (Chapter 5)
α	Angle of attack. (Chapter 2)
α_{az}	Gimbal azimuth angle. (Chapter 13)
α_{el}	Gimbal elevation angle. (Chapter 13)
b	Wing span. (Chapter 4)
b_*	Coefficients for reduced order model of the autopilot. (Chapter 9)
β	Side slip angle. (Chapter 2)
c	Mean aerodynamic chord of the wing. (Chapter 4)
C_D	Aerodynamic drag coefficient. (Chapter 4)
C_{ℓ_*}	Aerodynamic moment coefficient along the body frame x -axis. (Chapter 4)
C_L	Aerodynamic lift coefficient. (Chapter 4)
C_m	Aerodynamic pitching moment coefficient. (Chapter 4)
C_{n_*}	Aerodynamic moment coefficient along the body frame z -axis. (Chapter 4)
C_{p_*}	Aerodynamic moment coefficient along the body frame x -axis. (Chapter 5)
C_{prop}	Aerodynamic coefficient for the propeller. (Chapter 4)
C_{q_*}	Aerodynamic moment coefficient along the body frame y -axis. (Chapter 5)
C_{r_*}	Aerodynamic moment coefficient along the body frame z -axis. (Chapters 4, 5)
C_{X_*}	Aerodynamic force coefficient along the body frame x -axis. (Chapters 4, 5)
C_{Y_*}	Aerodynamic force coefficient along the body frame y -axis. (Chapter 4)
C_{Z_*}	Aerodynamic force coefficient along the body frame z -axis. (Chapters 4, 5)
χ	Course angle. (Chapter 2)
χ_c	Crab angle: $\chi_c = \chi - \psi$. (Chapter 2)
$\chi_d(e_{py})$	Desired course to track straight line path. (Chapter 10)
χ^∞	Desired approach angle for tracking a straight line path. (Chapter 10)
χ^o	Course angle of the orbit path $\mathcal{P}_{\text{orbit}}$. (Chapter 10)
χ_q	Course angle of the straight line path $\mathcal{P}_{\text{line}}$. (Chapter 10)
d	Distance between center of orbit and the MAV. (Chapter 10)

d_β	Disturbance signal associated with reduced side slip model. (Chapter 5)
d_χ	Disturbance signal associated with reduced course model. (Chapter 5)
d_h	Disturbance signal associated with reduced altitude model. (Chapter 5)
d_{ϕ_*}	Disturbance signals associated with reduced roll model. (Chapter 5)
d_{θ_*}	Disturbance signals associated with reduced pitch model. (Chapter 5)
d_{V_*}	Disturbance signals associated with reduced airspeed model. (Chapter 5)
δ_a	Control signal denoting the aileron deflection. (Chapter 4)
δ_e	Control signal denoting the elevator deflection. (Chapter 4)
δ_r	Control signal denoting the rudder deflection. (Chapter 4)
δ_t	Control signal denoting the throttle deflection. (Chapter 4)
e_p	Path error for straight line path following. (Chapter 10)
ϵ_s	Pixel size. (Chapter 13)
ϵ_x	Pixel location along the camera x -axis. (Chapter 13)
ϵ_y	Pixel location along the camera y -axis. (Chapter 13)
η_*	Zero-mean Gaussian sensor noise. (Chapter 7)
f	Camera focal length. (Chapter 13)
\mathbf{f}	External force applied to the airframe. Body frame components are denoted as f_x , f_y , and f_z . (Chapter 3, 4)
F	$= \sqrt{f^2 + \epsilon_x^2 + \epsilon_y^2}$, the distance to pixel location (ϵ_x, ϵ_y) in pixel. (Chapter 13)
F_{drag}	Force due to aerodynamic drag. (Chapter 4, 9)
F_{lift}	Force due to aerodynamic lift. (Chapter 4, 9)
F_{thrust}	Force due to thrust. (Chapter 9)
\mathcal{F}^b	Body coordinate frame. (Chapter 2)
\mathcal{F}^i	Inertial coordinate frame. (Chapter 2)
\mathcal{F}^s	Stability coordinate frame. (Chapter 2)
\mathcal{F}^v	Vehicle coordinate frame. (Chapter 2)
\mathcal{F}^w	Wind coordinate frame. (Chapter 2)
\mathcal{F}^{v1}	Vehicle-1 frame. (Chapter 2)
\mathcal{F}^{v2}	Vehicle-2 frame. (Chapter 2)
g	Gravitational acceleration (9.81 m/s^2). (Chapter 4)
γ	Inertial-referenced flight path angle. (Chapter 2)
γ_a	Air-mass-referenced flight path angle: $\gamma_a = \theta - \alpha$. (Chapter 2)
Γ_*	Products of the inertia matrix. See Equation (3.13). (Chapter 3)
h	Altitude: $h = -p_d$. (Chapter 5)
h_{AGL}	Altitude above ground level. (Chapter 7)
$\mathcal{H}(\mathbf{r}, \mathbf{n})$	Half plane defined at position \mathbf{w} , with normal vector \mathbf{n} . (Chapter 11)
$(\mathbf{i}^b, \mathbf{j}^b, \mathbf{k}^b)$	Unit vectors defining the body frame. \mathbf{i}^b points out the nose of the airframe, \mathbf{j}^b points out the right wing, and \mathbf{k}^b points through the bottom of the airframe. (Chapter 2)
$(\mathbf{i}^i, \mathbf{j}^i, \mathbf{k}^i)$	Unit vectors defining the inertial frame. \mathbf{i}^i points North, \mathbf{j}^i points east, and \mathbf{k}^i points down. (Chapter 2)

$(\mathbf{i}^v, \mathbf{j}^v, \mathbf{k}^v)$	Unit vectors defining the vehicle frame. \mathbf{i}^v points north, \mathbf{j}^v points east, and \mathbf{k}^v points down. (Chapter 2)
\mathbf{J}	The inertia matrix. Elements of the inertia matrix are denoted as J_x , J_y , J_z , and J_{xz} . (Chapter 3)
k_{d*}	PID derivative gain. (Chapter 6)
k_{GPS}	Inverse of the time constant for GPS bias. (Chapter 7)
k_{i*}	PID integral gain. (Chapter 6)
k_{motor}	Constant that specifies the efficiency of the motor. (Chapter 4)
k_{orbit}	Control gain for tracking orbital path. (Chapter 10)
k_{p*}	PID proportional gain. (Chapter 6)
k_{path}	Control gain for tracking straight line path. (Chapter 10)
$K_{\theta_{DC}}$	DC gain of the transfer function from the elevator to the pitch angle. (Chapter 6)
ℓ	External moment applied to the airframe about the body frame x -axis. (Chapter 3)
ℓ	Line-of-sight vector from the MAV to a target location. $\ell = (\ell_x, \ell_y, \ell_z)^\top$. (Chapter 13)
$\check{\ell}$	Unit vector in the direction of the line of sight: $\check{\ell} = \ell / \mathbb{L}$. (Chapter 13)
L_*	State-space coefficients associated with lateral dynamics. (Chapter 5)
\mathbb{L}	Length of the line of sight vector: $\mathbb{L} = \ \ell\ $. (Chapter 13)
$LPF(x)$	Low-pass filtered version of x .
λ	Direction of orbital path. $\lambda = +1$ specifies a clockwise orbit; $\lambda = -1$ specifies a counter clockwise orbit. (Chapter 10)
$\lambda_{\text{dutch roll}}$	Poles of the Dutch roll mode. (Chapter 5)
λ_{phugoid}	Poles of the phugoid mode. (Chapter 5)
λ_{rolling}	Pole of the rolling mode. (Chapter 5)
λ_{short}	Pole of the short period mode. (Chapter 5)
λ_{spiral}	Pole of the spiral mode. (Chapter 5)
m	Mass of the airframe. (Chapter 3)
m	External moment applied to the airframe about the body frame y -axis. (Chapter 3)
\mathbf{m}	External moments applied to the airframe. The body frame components are denoted as ℓ , m , and n . (Chapters 3, 4)
M	Width of the camera pixel array. (Chapter 13)
M_*	State-space coefficients associated with longitudinal dynamics. (Chapter 5)
n	External moment applied to the airframe about the body frame z -axis. (Chapter 3)
n_{lf}	Load factor. (Chapter 9)
N_*	State-space coefficients associated with lateral dynamics. (Chapter 5)
ν_*	Gauss-Markov process that models GPS bias. (Chapter 7)
ω_{n*}	Natural frequency. (Chapter 6)
$\boldsymbol{\omega}_{b/i}$	Angular velocity of the body frame with respect to the inertial frame. (Chapter 2)
p	Roll rate of the MAV along the body frame x -axis. (Chapter 3)
p_d	Inertial down position of the MAV. (Chapter 3)

p_e	Inertial east position of the MAV. (Chapter 3)
$\mathcal{P}_{\text{line}}$	Set defining a straight line. (Chapter 10)
\mathbf{p}_{MAV}	Position of the MAV. (Chapter 13)
p_n	Inertial north position of the MAV. (Chapter 3)
\mathbf{p}_{obj}	Position of the object of interest. (Chapter 13)
P	Covariance of the estimation error associated with the Kalman filter. (Chapter 8)
$\mathcal{P}_{\text{orbit}}$	Set defining an orbit (Chapter 10)
ϕ	Roll angle. (Chapter 2, 3)
φ	Angle of the MAV relative to a desired orbit. (Chapter 10)
ψ	Heading angle. (Chapters 2, 3)
q	Pitch rate of the MAV along the body frame y -axis. (Chapter 3)
Q_*	Process covariance noise. Typically used to tune a Kalman filter. (Chapter 8)
r	Yaw rate of the MAV along the body frame z -axis. (Chapter 3)
ρ	Density of air. (Chapter 4)
ϱ	Angle between waypoint path segments. (Chapter 11)
R	Turning radius. (Chapter 5)
R_*	Covariance matrix for sensor measurement noise. (Chapter 8)
\mathcal{R}_a^b	Rotation matrix from frame a to frame b . (Chapters 2, 13)
S	Surface area of the wing. (Chapter 4)
S_{prop}	Area of the propeller. (Chapter 4)
σ_*	Standard deviation of zero-mean white Gaussian noise. (Chapter 7)
t_c	Time to collision: $t_c = \mathbb{L}/\dot{\mathbb{L}}$. (Chapter 13)
T_s	Sample rate of the autopilot. (Chapters 6, 7, 8)
τ	Bandwidth of dirty differentiator. (Chapter 13)
\mathcal{T}	Terrain map. (Chapter 12)
θ	Pitch angle. (Chapter 2, 3)
u	Inertial velocity of the airframe projected onto \mathbf{i}^b , the body frame x -axis. (Chapter 2, 3)
u_{lat}	Input vector associated with lateral dynamics: $u_{\text{lat}} = (\delta_a, \delta_r)^\top$. (Chapter 5)
u_{lon}	Input vector associated with longitudinal dynamics: $u_{\text{lon}} = (\delta_e, \delta_t)^\top$. (Chapter 5)
u_r	Airspeed vector projected onto the body frame x -axis: $u_r = u - u_w$. (Chapters 2, 4)
u_w	Inertial wind velocity projected onto \mathbf{i}^b , the body frame x -axis. (Chapters 2, 4)
v	Camera field of view. (Chapter 13)
Υ	Return map used for path planning. The return map at position i is given by Υ_i . (Chapter 12)
v	Inertial velocity of the airframe projected onto \mathbf{j}^b , the body frame y -axis. (Chapters 2, 3)
v_r	Airspeed vector projected onto the body frame y -axis: $v_r = v - v_w$. (Chapters 2, 4)
v_w	Inertial wind velocity projected onto \mathbf{j}^b , the body frame y -axis. (Chapters 2, 4)
\mathbf{V}_a	Airspeed vector defined as the velocity of the airframe with respect to the air mass. (Chapter 2)

V_a	Airspeed where $V_a = \ \mathbf{V}_a\ $. (Chapter 2)
\mathbf{V}_g	Ground speed vector defined as the velocity of the airframe with respect to the inertial frame. (Chapter 2)
V_g	Ground speed where $V_g = \ \mathbf{V}_g\ $. (Chapter 2)
\mathbf{V}_w	Wind speed vector defined as the velocity of the wind with respect to the inertial frame. (Chapter 2)
V_w	Wind speed where $V_w = \ \mathbf{V}_w\ $. (Chapter 2)
w	Inertial velocity of the airframe projected onto \mathbf{k}^b , the body frame z -axis. (Chapters 2, 3)
w_d	Component of the wind in the down directions. (Chapter 2)
w_e	Component of the wind in the east directions. (Chapter 2)
w_n	Component of the wind in the north directions. (Chapter 2)
\mathbf{w}_i	Waypoint in \mathbb{R}^3 . (Chapter 11)
w_r	Airspeed vector projected onto the body frame z -axis: $w_r = w - w_w$. (Chapters 2, 4)
w_w	Inertial wind velocity projected onto \mathbf{k}^b , the body frame z -axis. (Chapters 2, 4)
W_*	Bandwidth separation. (Chapter 6)
\mathcal{W}	Set of waypoints. (Chapter 11)
x	State variables. (Chapter 5)
x_{lat}	State variables associated with lateral dynamics: $x_{\text{lat}} = (v, p, r, \phi, \psi)^\top$. (Chapter 5)
x_{lon}	State variables associated with longitudinal dynamics: $x_{\text{lon}} = (u, w, q, \theta, h)^\top$. (Chapter 5)
X_*	State-space coefficients associated with longitudinal dynamics. (Chapter 5)
$y_{\text{abs pres}}$	Absolute pressure measurement signal. (Chapter 7)
$y_{\text{accel,*}}$	Accelerometer measurement signal. (Chapter 7)
$y_{\text{diff pres}}$	Differential pressure measurement signal. (Chapter 7)
$y_{\text{GPS,*}}$	GPS measurement signal. GPS measurements are available for North, east, altitude, course, and groundspeed. (Chapter 7)
$y_{\text{gyro,*}}$	Rate gyro measurement signal. (Chapter 7)
y_{mag}	Magnetometer measurement signal. (Chapter 7)
Y_*	State-space coefficients associated with lateral dynamics. (Chapter 5)
Z_*	State-space coefficients associated with longitudinal dynamics. (Chapter 5)
$Z(\epsilon)$	Transformation from pixel motion to motion of the line of sight vector in the camera frame. (Chapter 13)
ζ_*	Damping coefficient. (Chapter 6)

Appendix B

Quaternions

B.1 ANOTHER LOOK AT COMPLEX NUMBERS

Let $\mathbf{z} = z_r + jz_i$ and $\mathbf{w} = w_r + jw_i$ be two complex numbers. Since $j^2 = -1$, multiplication gives

$$\begin{aligned}\mathbf{z}\mathbf{w} &= (z_r + jz_i)(w_r + jw_i) \\ &= z_r w_r + j^2 z_i w_i + j(z_i w_r + w_i z_r) \\ &= (z_r w_r - z_i w_i) + j(z_i w_r + w_i z_r).\end{aligned}$$

In an alternate universe, instead of imaginary numbers, mathematicians might have work solely with vectors in \mathbb{R}^2 , if they had defined vector multiplication appropriately. Suppose now that $\mathbf{z} = (z_r, z_i)^\top \in \mathbb{R}^2$ and $\mathbf{w} = (w_r, w_i)^\top \in \mathbb{R}^2$, and that vector multiplication in \mathbb{R}^2 were defined as follows:

$$\begin{aligned}\begin{pmatrix} z_r \\ z_i \end{pmatrix} \begin{pmatrix} w_r \\ w_i \end{pmatrix} &= \begin{pmatrix} z_r & -z_i \\ z_i & z_r \end{pmatrix} \begin{pmatrix} w_r \\ w_i \end{pmatrix} \\ &= \begin{pmatrix} z_r w_r - z_i w_i \\ z_i w_r + w_i z_r \end{pmatrix} \\ &= M(\mathbf{z})\mathbf{w},\end{aligned}$$

where

$$M(\mathbf{z}) = \begin{pmatrix} z_r & -z_i \\ z_i & z_r \end{pmatrix}.$$

Note that defining vector multiplication in this way is identical to multiplying two complex numbers. Note also, that if the complex number is on the unit circle, i.e., $\mathbf{z} = e^{j\theta} = \cos \theta + j \sin \theta$ that the associated matrix is

$$M(\mathbf{z}) = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix},$$

i.e., a complex number on the unit circle, represents a rotation of vector in 2D.

B.2 INTRODUCTION TO QUATERNIONS

Quaternions are an extension of complex numbers to four dimensions. A quaternion can be represented as

$$\mathbf{e} = e_0 + ie_x + je_y + ke_z,$$

where i, j, k satisfy

$$i^2 = j^2 = k^2 = ijk = -1.$$

These relationships also imply that $ij = k$, $jk = i$, and $ki = j$. Using these expressions it can be shown that

$$\begin{aligned}\mathbf{q}\mathbf{e} &= (q_0e_0 - q_xe_x - q_ye_y - q_ze_z) + i(q_xe_0 + q_0e_x + q_ze_y - q_ye_x) \\ &\quad + j(q_ye_0 - q_ze_x + q_0e_y + q_xe_z) + k(q_ze_0 + q_ye_x - q_xe_y + q_0e_z).\end{aligned}$$

If instead, we represent quaternions as elements of \mathbb{R}^4 , then quaternion multiplication can be defined as

$$\mathbf{q}\mathbf{e} = M(\mathbf{q})\mathbf{e},$$

where

$$M(\mathbf{q}) = \begin{pmatrix} q_0 & -q_x & -q_y & -q_z \\ q_x & q_0 & q_z & -q_y \\ q_y & -q_y & q_0 & q_x \\ q_z & q_y & -q_x & q_0 \end{pmatrix}.$$

B.3 QUATERNION ROTATIONS

As complex numbers on the unit circle, represent rotations in 2D, similarly, unit quaternions, i.e., quaternions such that $\|\mathbf{e}\| = 1$ can be used to represent rotations in 3D.

Quaternions provide an alternative way to represent the attitude of an aircraft. While it could be argued that it is more difficult to visualize the angular motion of a vehicle specified by quaternions instead of Euler angles, there are mathematical advantages to the quaternion representation that make it the method of choice for many aircraft simulations. Most significantly, the Euler angle representation has a singularity when the pitch angle θ is ± 90 deg. Physically, when the pitch angle is 90 deg, the roll and yaw angles are indistinguishable. Mathematically, the attitude kinematics specified by Equation (3.3) are indeterminate since $\cos \theta = 0$ when $\theta = 90$ deg. The quaternion representation of attitude has no such singularity. While

this singularity is not an issue for the vast majority of flight conditions, it is an issue for simulating aerobatic flight and other extreme maneuvers, some of which may not be intentional. The other advantage that the quaternion formulation provides is that it is more computationally efficient. The Euler angle formulation of the aircraft kinematics involves nonlinear trigonometric functions, whereas the quaternion formulation results in much simpler linear and algebraic equations. A thorough introduction to quaternions and rotation sequences is given by Kuipers [?]. An in-depth treatment to the use of quaternions specific to aircraft applications is given by Phillips [18].

In its most general form, a quaternion is an ordered list of four real numbers. We can represent the quaternion e as a vector in \mathcal{R}^4 as

$$e = \begin{pmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{pmatrix},$$

where e_0, e_x, e_y , and e_z are scalars. When a quaternion is used to represent a rotation, we require that it be a *unit quaternion*, or in other words, $\|e\| = 1$.

It is common to refer e_0 as the scalar part of the unit quaternion and the vector defined by

$$\mathbf{e} = (e_x, e_y, e_z)^\top$$

as the vector part. The unit quaternion can be interpreted as a single rotation about an axis in three-dimensional space. For a rotation through the angle Θ about the axis specified by the unit vector \mathbf{v} , the scalar part of the unit quaternion is related to the magnitude of the rotation by

$$e_0 = \cos\left(\frac{\Theta}{2}\right).$$

The vector part of the unit quaternion is related to the axis of rotation by

$$\mathbf{v} \sin\left(\frac{\Theta}{2}\right) = \begin{pmatrix} e_x \\ e_y \\ e_z \end{pmatrix}.$$

With this brief description of the quaternion, we can see how the attitude of a MAV can be represented with a unit quaternion. The rotation from the inertial frame to the body frame is simply specified as a single rotation about a specified axis, instead of a sequence of three rotations as required by the Euler angle representation.

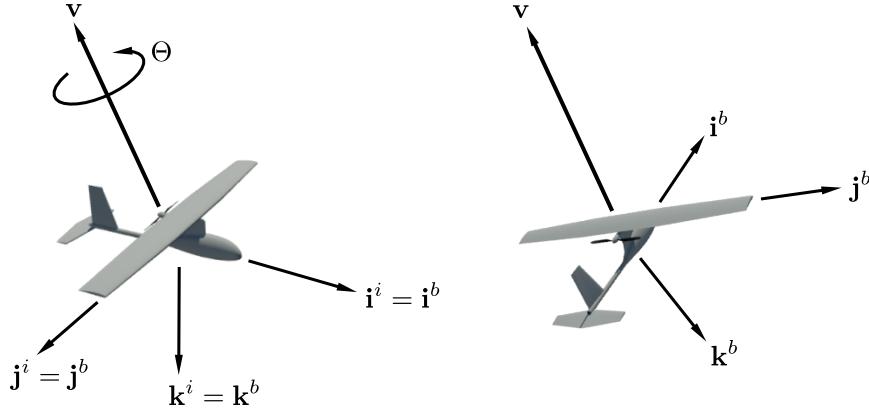


Figure B.1: Rotation represented by a unit quaternion. The aircraft on the left is shown with the body axes aligned with the inertial frame axes. The aircraft on the left has been rotated about the vector \mathbf{v} by $\Theta = 86$ deg. This particular rotation corresponds to the Euler sequence $\psi = -90$ deg, $\theta = 15$ deg, $\phi = -30$ deg.

B.4 CONVERSION BETWEEN EULER ANGLES AND QUATERNIONS

There are simple formulas for converting Euler angles to quaternions, and a quaternion to the associated 3-2-1 Euler angles. Suppose that the quaternion representing a rotation from the body axes to inertial axes is given by

$$\mathbf{e}_b^i = (e_0, e_x, e_y, e_z)^\top,$$

then the associated 3-2-1 Euler angles are

$$\begin{aligned} \phi &= \text{atan2}(2(e_0 e_x + e_y e_z), (e_0^2 + e_z^2 - e_x^2 - e_y^2)) \\ \theta &= \text{asin}(2(e_0 e_y - e_x e_z)) \\ \psi &= \text{atan2}(2(e_0 e_z + e_x e_y), (e_0^2 + e_x^2 - e_y^2 - e_z^2)), \end{aligned} \quad (\text{B.1})$$

where $\text{atan2}(y, x)$ is the two-argument arctangent operator that returns the arctangent of y/x in the range $[-\pi, \pi]$ using the signs of both arguments to determine the quadrant of the return value. Only a single argument is required for the asin operator since the pitch angle is only defined in the range $[\pi/2, \pi/2]$.

Alternatively, given the 3-2-1 Euler angles (ψ, ϕ, θ) , the corresponding quaternion representing a passive rotation from the body axes to the inertial

axes is given by

$$\mathbf{e}_b^i = \begin{pmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{pmatrix} = \begin{pmatrix} \cos \frac{\psi}{2} \cos \frac{\theta}{2} \cos \frac{\phi}{2} + \sin \frac{\psi}{2} \sin \frac{\theta}{2} \sin \frac{\phi}{2} \\ \cos \frac{\psi}{2} \cos \frac{\theta}{2} \sin \frac{\phi}{2} - \sin \frac{\psi}{2} \sin \frac{\theta}{2} \cos \frac{\phi}{2} \\ \cos \frac{\psi}{2} \sin \frac{\theta}{2} \cos \frac{\phi}{2} + \sin \frac{\psi}{2} \cos \frac{\theta}{2} \sin \frac{\phi}{2} \\ \sin \frac{\psi}{2} \cos \frac{\theta}{2} \cos \frac{\phi}{2} - \cos \frac{\psi}{2} \sin \frac{\theta}{2} \sin \frac{\phi}{2} \end{pmatrix}. \quad (\text{B.2})$$

B.5 CONVERSION BETWEEN QUATERNIONS AND ROTATION MATRICES

Given a quaternion \mathbf{e}_b^i that represents a passive rotation from the body axes to the inertial axes, the associated rotation matrix is given by

$$R_b^i = \begin{pmatrix} e_0^2 + e_x^2 - e_y^2 - e_z^2 & 2(e_x e_y - e_0 e_z) & 2(e_x e_z + e_0 e_y) \\ 2(e_x e_y + e_0 e_z) & e_0^2 - e_x^2 + e_y^2 - e_z^2 & 2(e_y e_z - e_0 e_x) \\ 2(e_x e_z - e_0 e_y) & 2(e_y e_z + e_0 e_x) & e_0^2 - e_x^2 - e_y^2 + e_z^2 \end{pmatrix}.$$

Alternatively, suppose that

$$R_b^i = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix},$$

then the associated quaternion $\mathbf{e}_b^i = (e_0, e_x, e_y, e_z)^\top$ can be computed as follows [?]:

$$\begin{aligned} e_0 &= \begin{cases} \frac{1}{2}\sqrt{1+r_{11}+r_{22}+r_{33}}, & \text{if } r_{11}+r_{22}+r_{33} > 0 \\ \frac{1}{2}\sqrt{\frac{(r_{12}-r_{21})^2+(r_{13}-r_{31})^2+(r_{23}-r_{32})^2}{3-r_{11}-r_{22}-r_{33}}}, & \text{otherwise} \end{cases} \\ e_x &= \begin{cases} \frac{1}{2}\sqrt{1+r_{11}-r_{22}-r_{33}}, & \text{if } r_{11}-r_{22}-r_{33} > 0 \\ \frac{1}{2}\sqrt{\frac{(r_{12}+r_{21})^2+(r_{13}+r_{31})^2+(r_{23}-r_{32})^2}{3-r_{11}+r_{22}+r_{33}}}, & \text{otherwise} \end{cases} \\ e_y &= \begin{cases} \frac{1}{2}\sqrt{1-r_{11}+r_{22}-r_{33}}, & \text{if } -r_{11}+r_{22}-r_{33} > 0 \\ \frac{1}{2}\sqrt{\frac{(r_{12}+r_{21})^2+(r_{13}-r_{31})^2+(r_{23}+r_{32})^2}{3+r_{11}-r_{22}+r_{33}}}, & \text{otherwise} \end{cases} \\ e_z &= \begin{cases} \frac{1}{2}\sqrt{1-r_{11}-r_{22}+r_{33}}, & \text{if } -r_{11}-r_{22}+r_{33} > 0 \\ \frac{1}{2}\sqrt{\frac{(r_{12}-r_{21})^2+(r_{13}+r_{31})^2+(r_{23}+r_{32})^2}{3+r_{11}+r_{22}-r_{33}}}, & \text{otherwise} \end{cases}. \end{aligned}$$

B.6 QUATERNION KINEMATICS

Suppose that \mathbf{q} represents a small rotation, i.e.,

$$\mathbf{q}_\Delta = \begin{pmatrix} \cos(\Delta\Theta/2) \\ \sin(\Delta\Theta/2)u_x \\ \sin(\Delta\Theta/2)u_y \\ \sin(\Delta\Theta/2)u_z \end{pmatrix} \approx \begin{pmatrix} 1 \\ \frac{\omega_x\Delta t}{2} \\ \frac{\omega_y\Delta t}{2} \\ \frac{\omega_z\Delta t}{2} \end{pmatrix}.$$

Then

$$\mathbf{e}(t + \Delta t) = M(\mathbf{q}_\Delta)\mathbf{e}(t).$$

Therefore

$$\begin{aligned} \dot{\mathbf{e}} &= \lim_{\Delta t \rightarrow 0} \frac{\mathbf{e}(t + \Delta t) - \mathbf{e}(t)}{\Delta t} \\ &= \lim_{\Delta t \rightarrow 0} \frac{M(\mathbf{q}_\Delta)\mathbf{e}(t) - \mathbf{e}(t)}{\Delta t} \\ &= \lim_{\Delta t \rightarrow 0} \frac{(M(\mathbf{q}_\Delta) - I)\mathbf{e}(t)}{\Delta t} \\ &= \lim_{\Delta t \rightarrow 0} \frac{1}{2\Delta t} \begin{pmatrix} 0 & -\omega_x\Delta t & -\omega_y\Delta t & -\omega_z\Delta t \\ \omega_x\Delta t & 0 & \omega_z\Delta t & -\omega_y\Delta t \\ \omega_y\Delta t & -\omega_z\Delta t & 0 & \omega_x\Delta t \\ \omega_z\Delta t & \omega_y\Delta t & -\omega_x\Delta t & 0 \end{pmatrix} \mathbf{e}(t) \\ &= \frac{1}{2}\Omega(\boldsymbol{\omega})\mathbf{e}(t), \end{aligned}$$

where

$$\Omega(\boldsymbol{\omega}) = \begin{pmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{pmatrix}$$

B.7 AIRCRAFT KINEMATIC AND DYNAMIC EQUATIONS

Using a unit quaternion to represent the aircraft attitude, Equations (3.14) through (3.17), which describe the MAV kinematics and dynamics, can be

reformulated as

$$\begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{p}_d \end{pmatrix} = \begin{pmatrix} e_x^2 + e_0^2 - e_y^2 - e_z^2 & 2(e_x e_y - e_z e_0) & 2(e_x e_z + e_y e_0) \\ 2(e_x e_y + e_z e_0) & e_y^2 + e_0^2 - e_x^2 - e_z^2 & 2(e_y e_z - e_x e_0) \\ 2(e_x e_z - e_y e_0) & 2(e_y e_z + e_x e_0) & e_z^2 + e_0^2 - e_x^2 - e_y^2 \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix} \quad (\text{B.3})$$

$$\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} = \begin{pmatrix} rv - qw \\ pw - ru \\ qu - pv \end{pmatrix} + \frac{1}{m} \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix}, \quad (\text{B.4})$$

$$\begin{pmatrix} \dot{e}_0 \\ \dot{e}_x \\ \dot{e}_y \\ \dot{e}_z \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{pmatrix} \begin{pmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{pmatrix} \quad (\text{B.5})$$

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} \Gamma_1 pq - \Gamma_2 qr \\ \Gamma_5 pr - \Gamma_6(p^2 - r^2) \\ \Gamma_7 pq - \Gamma_1 qr \end{pmatrix} + \begin{pmatrix} \Gamma_3 l + \Gamma_4 n \\ \frac{1}{J_y} m \\ \Gamma_4 l + \Gamma_8 n \end{pmatrix}. \quad (\text{B.6})$$

■

Note that the dynamic equations given by Equations (B.4) and (B.6) are unchanged from Equations (3.15) and (3.17) presented in the summary of Chapter 3. However, care must be taken when propagating Equation (B.5) to ensure that e remains a unit quaternion. If the dynamics are implemented using a Simulink s-function, then one possibility for maintaining $\|e\| = 1$ is to modify Equation (B.5) so that in addition to the normal dynamics, there is also a term that seeks to minimize the cost function $J = \frac{1}{8}(1 - \|e\|^2)^2$. Since J is quadratic, we can use gradient descent to minimize J , and Equation (B.5) becomes

$$\begin{pmatrix} \dot{e}_0 \\ \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{pmatrix} \begin{pmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{pmatrix} - \lambda \frac{\partial J}{\partial e}$$

$$= \frac{1}{2} \begin{pmatrix} \lambda(1 - \|e\|^2) & -p & -q & -r \\ p & \lambda(1 - \|e\|^2) & r & -q \\ q & -r & \lambda(1 - \|e\|^2) & p \\ r & q & -p & \lambda(1 - \|e\|^2) \end{pmatrix} \begin{pmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{pmatrix},$$

■

where $\lambda > 0$ is a positive gain that specifies the strength of the gradient descent. In our experience, a value of $\lambda = 1000$ seems to work well, but in Simulink, a stiff solver like ODE15s must be used. This method for maintaining the orthogonality of the quaternion during integration is called

Corbett-Wright orthogonality control and was first introduced in the 1950's for use with analog computers [18, ?].

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

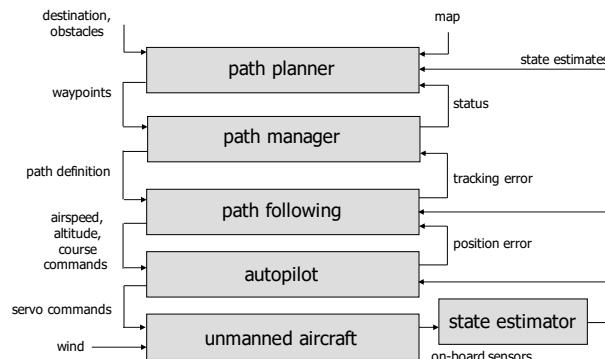


Figure C.1: Architecture outlined in the uavbook.

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

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

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

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

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

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

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

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

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 $f(\cdot, \cdot)$.

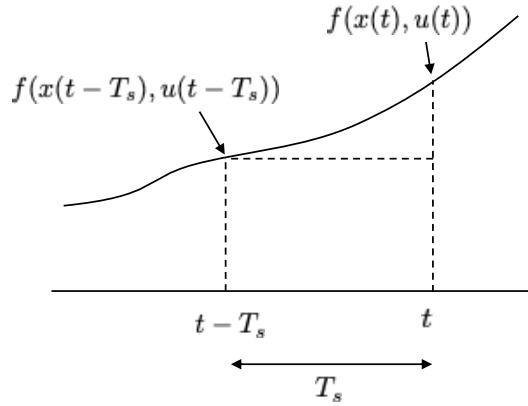


Figure C.2: Approximation of integral using the left endpoint method results in Runge-Kutta first order method (RK1).

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,

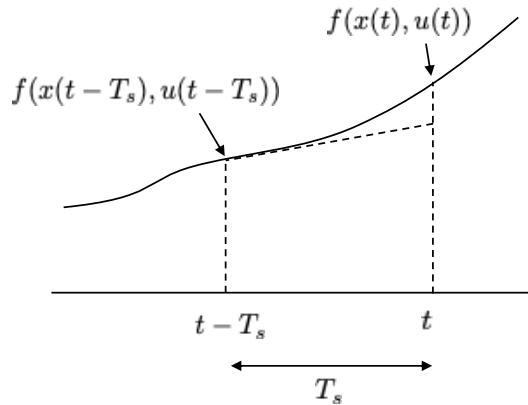


Figure C.3: Approximation of integral using the trapezoidal rule results in Runge-Kutta second order method (RK2).

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

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

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

Accordingly, the numerical integration routine can be written as

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

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

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

and then to use

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

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

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

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

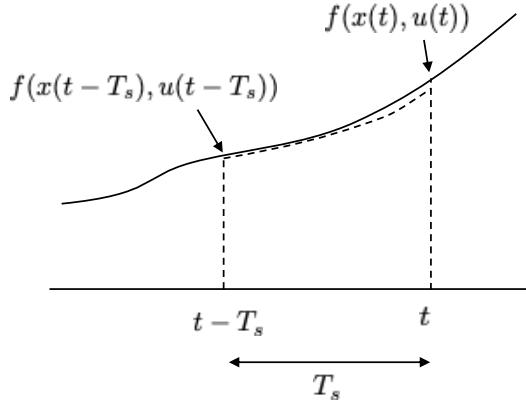


Figure C.4: Approximation of integral using Simpson's rule results in Runge-Kutta fourth order method (RK4).

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

The stick-figure drawing shown in Figure ?? can be improved visually by using the vertex-face structure in Matlab. Instead of using the `plot3` command to draw a continuous line, we will use the `patch` command to draw faces defined by vertices and colors. The vertices, faces, and colors for the spacecraft are defined in the Matlab code listed below.

```
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
```

The transposes in Lines 3 and 4 are used because the physical positions in the vertices matrix V are along the rows instead of the columns. A rendering of the spacecraft using vertices and faces is given in Figure D.1. Additional

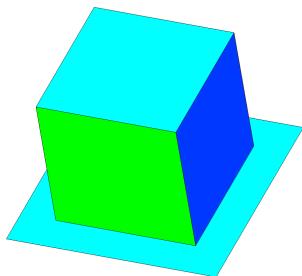


Figure D.1: Rendering of the spacecraft using vertices and faces.

examples using the vertex-face format can be found at the book website.

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

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

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

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

```

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 $T : \mathbb{R}^{13} \rightarrow \mathbb{R}^{12}$ 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 (B.1).

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 (B.2), and where

$$\frac{\partial T}{\partial x_q} = \begin{pmatrix} I_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 4} & 0_{3 \times 3} \\ 0_{3 \times 3} & I_{3 \times 3} & 0_{3 \times 4} & 0_{3 \times 3} \\ 0_{3 \times 3} & 0_{3 \times 3} & \frac{\partial \Theta}{\partial e} & 0_{3 \times 3} \\ 0_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 4} & I_{3 \times 3} \end{pmatrix} \quad (\text{F.1})$$

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

F.2 TRIM AND LINEARIZATION IN SIMULINK

Simulink provides a built-in routine for computing trim conditions for general Simulink diagrams. Useful instructions for using this command can be obtained by typing `help trim` at the Matlab prompt. As described in Section 5.2, given the parameters V_a^* , γ^* , and R^* , the objective is to find x^* and u^* such that $\dot{x}^* = f(x^*, u^*)$ where x and u are defined in Equations (??) and (??), \dot{x}^* is given by Equation (??), and where $f(x, u)$ is defined by the right-hand side of Equations (5.5)–(5.16).

The format for the Simulink `trim` command is

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

where X is the computed trim state x^* , U is the computed trim input u^* , Y is the computed trim output y^* , and DX is the computed derivative of the state \dot{x}^* . The system is specified by the Simulink model `SYS.mdl`, where the state of the model is defined by the union of all of the states in the subsystems of `SYS.mdl` and the inputs and outputs are defined by Simulink `Imports` and `Outports` respectively. Figure F.1 shows a Simulink model that could be used to compute aircraft trim. The inputs to the system as specified by the four `Imports` are the servo commands `delta_e`, `delta_a`, `delta_r`, and `delta_t`. The states of this block are the states of the Simulink model, which in our case are

$$\xi = (p_n, p_e, p_d, u, v, w, \phi, \theta, \psi, p, q, r)^\top,$$

and the outputs are specified by the three Outports as the airspeed V_a , the angle of attack α , and the sideslip angle β . Our purpose in specifying V_a , α , and β as outputs is that we wish to force the Simulink `trim` command to maintain $V_a = V_a^*$ and α^* is often a quantity of interest. If we have access to a rudder, then we can enforce a coordinated turn to forcing the trim command to maintain $\beta^* = 0$. If a rudder is not available, then β will not necessarily be zero in a turn.

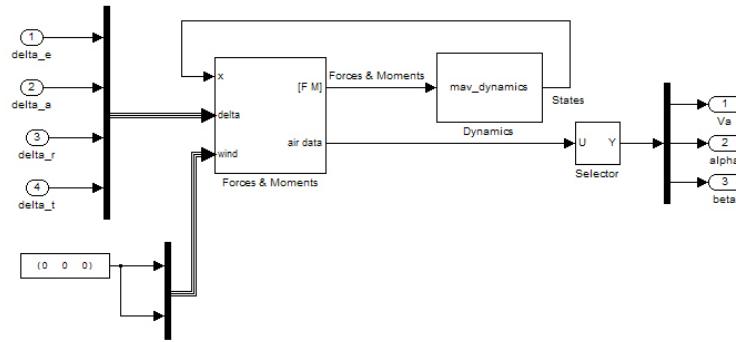


Figure F.1: Simulink diagram used to compute trim and linear state space models.

Since the trim calculation problem reduces to solving a system of nonlinear algebraic equations, which may have many solutions, the Simulink `trim` command requires that initial guesses for the state X_0 , input U_0 , output Y_0 , and derivative of the state DX_0 be specified. If we know from the outset, that some of the states, inputs, outputs, or derivatives of states are fixed and specified by their initial conditions, then those constraints are indicated by the index vectors IX , IU , HY , and IDX .

For our situation we know that

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

therefore we let

```

DX = [0; 0; Va*sin(gamma); 0; 0; 0; 0; 0; Va/R; 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; gamma; 0]
HY0 = [1,3].

```

Simulink also provides a built-in routine for computing a linear state-space model for a general Simulink diagram. Helpful instruction can be obtained by typing `help linmod` at the Matlab prompt. The format for the `linmod` command is

```
[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 F.1, 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]);
```

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

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

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

$$\frac{\partial T^{-1}}{\partial x_q} = \begin{pmatrix} I_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} \\ 0_{3 \times 3} & I_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} \\ 0_{4 \times 3} & 0_{4 \times 3} & \frac{\partial Q}{\partial \Theta} & 0_{4 \times 3} \\ 0_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} & I_{3 \times 3} \end{pmatrix} \quad (\text{F.2})$$

and where $Q(\Theta)$ is given by Equation (B.2) 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

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

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
```

Bibliography

- [1] B. L. Stevens and F. L. Lewis, *Aircraft Control and Simulation*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2nd ed., 2003.
- [2] D. T. Greenwood, *Principles of Dynamics*. Englewood Cliffs, NJ: Prentice Hall, 2nd ed., 1988.
- [3] T. R. Kane and D. A. Levinson, *Dynamics: Theory and Applications*. McGraw Hill, 1985.
- [4] M. W. Spong and M. Vidyasagar, *Robot Dynamics and Control*. John Wiley & Sons, Inc., 1989.
- [5] M. D. Shuster, “A Survey of Attitude Representations,” *The Journal of the Astronautical Sciences*, vol. 41, no. 4, pp. 439–517, 1993.
- [6] R. C. Nelson, *Flight Stability and Automatic Control*. Boston, Massachusetts: McGraw-Hill, 2nd ed., 1998.
- [7] 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.
- [8] M. Rauw, “{FDC} 1.2 - A {SIMULINK} Toolbox for Flight Dynamics and Control Analysis,” feb 1998.
- [9] H. Goldstein, *Classical Mechanics*. Addison-Wesley, 1951.
- [10] A. V. Rao, *Dynamics of Particles and Rigid Bodies: A Systematic Approach*. Cambridge University Press, 2006.
- [11] M. R. Jardin and E. R. Mueller, “Optimized Measurements of UAV Mass Moment of Inertia with a Bifilar Pendulum,” in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, no. AIAA-2007-6822, (Hilton Head, South Carolina), AIAA, aug 2007.
- [12] J. B. Marion, *Classical Dynamics of Particles and Systems*. Academic Press, second edi ed., 1970.
- [13] W. E. Wiesel, *Spaceflight Dynamics*. McGraw Hill, second edi ed., 1997.
- [14] J. R. Wertz, *Spacecraft Attitude Determination and Control*. Kluwer Academic Publishers, 1978.
- [15] J. Roskam, *Airplane Flight Dynamics and Automatic Flight Controls, Parts I {&} II*. Lawrence, Kansas: DARcorporation, 1998.
- [16] M. V. Cook, *Flight Dynamics Principles*. New York: John Wiley & Sons, 1997.
- [17] R. F. Stengel, *Flight Dynamics*. Princeton University Press, 2004.
- [18] W. F. Phillips, *Mechanics of Flight*. Wiley, 2004.

- [19] R. Rysdyk, “{UAV} Path Following for Constant Line-of-Sight,” in *Proceedings of the {AIAA} 2nd Unmanned Unlimited Conference*, AIAA, sep 2003.
- [20] J. Osborne and R. Rysdyk, “Waypoint Guidance for Small UAVs in Wind,” in *Proceedings of the AIAA Infotech@Aerospace Conference*, sep 2005.
- [21] F. L. Lewis, *Optimal Estimation: With an Introduction to Stochastic Control Theory*. New York, New York: John Wiley & Sons, 1986.
- [22] A. Gelb, "Applied Optimal Estimation". The M.I.T. Press, 1974.
- [23] B. D. O. Anderson and J. B. Moore, *Linear Optimal Control*. Englewood Cliffs, New Jersey: Prentice Hall, 1971.
- [24] R. G. Brown, *Introduction to Random Signal Analysis and Kalman Filtering*. John Wiley & Sons, Inc, 1983.
- [25] A. M. Eldredge, *Improved State Estimation for Miniature Air Vehicles*. PhD thesis, Brigham Young University, 2006.
- [26] R. W. Beard, “State Estimation for Micro Air Vehicles,” in *Innovations in Intelligent Machines I* (A. M. M. S.-I. Jovaan S. Chahl Lakhmi C. Jain, ed.), pp. 173–199, Springer Verlag, 2007.
- [27] A. D. Wu, E. N. Johnson, and A. A. Proctor, “Vision-Aided Inertial Navigation for Flight Control,” *Journal of Aerospace Computing, Information, and Communication*, vol. 2, pp. 348–360, sep 2005.
- [28] T. P. Webb, R. J. Prazenica, A. J. Kurdila, and R. Lind, “Vision-Based State Estimation for Autonomous Micro Air Vehicles,” *AIAA Journal of Guidance, Control, and Dynamics*, vol. 30, pp. 816–826, may 2007.
- [29] 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.
- [30] R. W. Beard and T. W. McLain, *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, 2012.
- [31] 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.
- [32] D. R. Nelson, D. B. Barber, T. W. McLain, and R. W. Beard, “Vector Field Path Following for Small Unmanned Air Vehicles,” in *American Control Conference*, (Minneapolis, Minnesota), pp. 5788–5794, jun 2006.
- [33] R. W. Beard, “Embedded {UAS} Autopilot and Sensor Systems,” in *Encyclopedia of Aerospace Engineering* (R. Blockley and W. Shyy, eds.), pp. 4799–4814, Chichester, UK: John Wiley & Sons, Ltd, 2010.
- [34] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” in *Proceedings of the {IEEE} International Conference on Robotics and Automation*, vol. 2, pp. 500–505, 1985.
- [35] K. Sigurd and J. P. How, “{UAV} Trajectory Design Using Total Field Collision Avoidance,” in *Proceedings of the {AIAA} Guidance, Navigation and Control Conference*, aug 2003.

- [36] T. W. McLain and R. W. Beard, "Coordination Variables, Coordination Functions, and Cooperative Timing Missions," *{AIAA} Journal of Guidance, Control and Dynamics*, vol. 28, pp. 150–161, jan 2005.
- [37] E. P. Anderson, R. W. Beard, and T. W. McLain, "Real Time Dynamic Trajectory Smoothing for Uninhabited Aerial Vehicles," *{IEEE} Transactions on Control Systems Technology*, vol. 13, pp. 471–477, may 2005.
- [38] G. Yang and V. Kapila, "Optimal Path Planning for Unmanned Air Vehicles with Kinematic and Tactical Constraints," in *Proceedings of the {IEEE} Conference on Decision and Control*, (Las Vegas, NV), pp. 1301–1306, 2002.
- [39] P. R. Chandler, S. Rasumussen, and M. Pachter, "{UAV} Cooperative Path Planning," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, (Denver, CO), aug 2000.
- [40] R. M. Murray and S. S. Sastry, "Nonholonomic Motion Planning: Steering Using Sinusoids," *{IEEE} Transactions on Automatic Control*, vol. 38, pp. 700–716, may 1993.
- [41] T. Balch and R. C. Arkin, "Behavior-Based Formation Control for Multirobot Teams," *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 926–939, dec 1998.
- [42] R. C. Arkin, *Behavior-based Robotics*. MIT Press, 1998.
- [43] R. Sedgewick, *Algorithms*. Addison-Wesley, 2nd ed., 1988.
- [44] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. McGraw-Hill, 1993.
- [45] S. M. LaValle and J. J. Kuffner, "Randomized Kinodynamic Planning," *International Journal of Robotic Research*, vol. 20, pp. 378–400, may 2001.
- [46] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.
- [47] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.
- [48] T. McLain and R. Beard, "Cooperative Rendezvous of Multiple Unmanned Air Vehicles," in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, (Denver, CO), aug 2000.
- [49] H. Choset, S. Walker, K. Eiamsa-Ard, and J. Burdick, "Sensor-Based Exploration: Incremental Construction of the Hierarchical Generalized {V}oronoi Graph," *The International Journal of Robotics Research*, vol. 19, pp. 126–148, feb 2000.
- [50] J. J. Kuffner and S. M. LaValle, "{RRT}-Connect: An Efficient Approach to Single-Query Path Planning," in *Proceedings of the {IEEE} International Conference on Robotics and Automation*, (San Francisco, CA), pp. 995–1001, apr 2000.
- [51] M. Zucker, J. Kuffner, and M. Branicky, "Multipartite {RRT}s for Rapid Replanning in Dynamic Environments," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Roma, Italy), apr 2007.
- [52] S. Karaman and E. Frazzoli, "Incremental Sampling-based Algorithms for Optimal Motion Planning," *International Journal of Robotic Research*, 2010.
- [53] A. Ladd and L. E. Kavraki, "Generalizing the Analysis of {PRM}, " in *Proceedings of the {IEEE} International Conference on Robotics and Automation*, (Washington DC), pp. 2120–2125, may 2002.
- [54] E. U. Acar, H. Choset, and J. Y. Lee, "Sensor-Based Coverage With Extended Range Detectors," *IEEE Transactions on Robotics*, vol. 22, pp. 189–198, feb 2006.

- [55] C. Luo, S. X. Yang, D. A. Stacey, and J. C. Jofriet, “A Solution to Vicinity Problem of Obstacles in Complete Coverage Path Planning,” in *Proceedings of the {IEEE} International Conference on Robotics and Automation*, (Washington DC), pp. 612–617, may 2002.
- [56] Z. J. Butler, A. A. Rizzi, and R. L. Hollis, “Cooperative Coverage of Rectilinear Environments,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, (San Francisco, CA), pp. 2722–2727, apr 2000.
- [57] J. Cort, S. Mart, J. Cortes, S. Martinez, T. Karatas, and F. Bullo, “Coverage control for mobile sensing networks,” in *Proceedings of the {IEEE} International Conference on Robotics and Automation*, no. May, (Washington DC), pp. 1327–1332, may 2002.
- [58] M. Schwager, J.-J. Slotine, and J. J. Daniela Russell, “Consensus Learning for Distributed Coverage Control,” in *Proceedings of the International Conference on Robotics and Automation*, (Pasadena, CA), pp. 1042–1048, may 2008.
- [59] J. H. Evers, “Biological Inspiration for Agile Autonomous Air Vehicles,” in *Symposium on Platform Innovations and System Integration for Unmanned Air, Land, and Sea Vehicles*, (Florence, Italy), NATO Research and Technology Organization AVT-146, may 2007.
- [60] D. B. Barber, J. D. Redding, T. W. McLain, R. W. Beard, and C. N. Taylor, “Vision-based Target Geo-location using a Fixed-wing Miniature Air Vehicle,” *Journal of Intelligent and Robotic Systems*, vol. 47, pp. 361–382, dec 2006.
- [61] G. F. Franklin, J. D. Powell, and M. Workman, *Digital Control of Dynamic Systems*. Addison Wesley, third edit ed., 1998.
- [62] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry, *An Invitation to {3-D} Vision: From Images to Geometric Models*. Springer-Verlag, 2003.
- [63] P. Zarchan, *Tactical and Strategic Missile Guidance*, vol. 124 of *Progress in Astronautics and Aeronautics*. Washington DC: American Institute of Aeronautics and Astronautics, 1990.
- [64] M. Guelman, M. Idan, and O. M. Golan, “Three-Dimensional Minimum Energy Guidance,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 31, pp. 835–841, apr 1995.
- [65] J. G. Lee, H. S. Han, and Y. J. Kim, “Guidance Performance Analysis of Bank-To-Turn {(BTT)} Missiles,” in *Proceedings of the IEEE International Conference on Control Applications*, (Kohala, Hawaii), pp. 991–996, aug 1999.
- [66] J. Lopez, M. Markel, N. Siddiqi, G. Gebert, and J. Evers, “Performance of Passive Ranging from Image Flow,” in *Proceedings of the IEEE International Conference on Image Processing*, vol. 1, pp. I-929—I-932, sep 2003.
- [67] M. Pachter, N. Ceccarelli, and P. R. Chandler, “Vision-Based Target Geo-Location Using Camera Equipped MAVs,” in *Proceedings of the IEEE Conference on Decision and Control*, (New Orleans, LA), dec 2007.
- [68] R. J. Prazenica, A. J. Kurdila, R. C. Sharpley, P. Binev, M. H. Hielsberg, J. Lane, and J. Evers, “Vision-Based Receding Horizon Control for Micro Air Vehicles in Urban Environments,” *AIAA Journal of Guidance, Dynamics, and Control*.
- [69] I. H. Wang, V. N. Dobrokhotov, I. I. Kaminer, and K. D. Jones, “On Vision-Based Target Tracking and Range Estimation for Small {UAV}s,” in *2005 AIAA Guidance, Navigation, and Control Conference and Exhibit*, pp. 1–11, 2005.

- [70] Y. Watanabe, A. J. Calise, E. N. Johnson, and J. H. Evers, "Minimum-Effort Guidance for Vision-Based Collision Avoidance," in *Proceedings of the AIAA Atmospheric Flight Mechanics Conference and Exhibit*, (Keystone, Colorado), American Institute of Aeronautics and Astronautics, aug 2006.
- [71] Y. Watanabe, E. N. Johnson, and A. J. Calise, "Optimal {3-D} Guidance from a {2-D} Vision Sensor," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, (Providence, Rhode Island), American Institute of Aeronautics and Astronautics, aug 2004.
- [72] I. H. Whang, V. N. Dobrokhotov, I. I. Kaminer, and K. D. Jones, "On Vision-Based Tracking and Range Estimation for Small UAVs," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, (San Francisco, CA), aug 2005.
- [73] M. E. Campbell and M. Wheeler, "A Vision Based Geolocation Tracking System for UAVs," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Keystone, Colorado), aug 2006.
- [74] D. Murray and A. Basu, "Motion Tracking with an Active Camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, pp. 449–459, may 1994.
- [75] S. Hutchinson, G. D. Hager, and P. I. Corke, "A Tutorial on Visual Servo Control," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 651–670, oct 1996.
- [76] J. Oliensis, "A critique of structure-from-motion algorithms," *Computer Vision and Image Understanding (CVIU)*, vol. 80, no. 2, pp. 172–214, 2000.
- [77] J. Santos-Victor and G. Sandini, "Uncalibrated Obstacle Detection Using Normal Flow," *Machine Vision and Applications*, vol. 9, no. 3, pp. 130–137, 1996.
- [78] R. A. B. L.M. Lorigo and W. E. L. Grimson, "Visually-Guided Obstacle Avoidance in Unstructured Environments," in *Proceedings of IROS '97*, (Grenoble, France), sep 1997.
- [79] R. Nelson and Y. Aloimonos, "Obstacle Avoidance Using Flow Field Divergence," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 1102–1106, oct 1989.
- [80] K. H. Gabbiani F and L. G, "Computation of object approach by a wide field visual neuron," *J Neuroscience*, vol. 19, pp. 1122–1141, 1999.
- [81] R. W. Beard, J. W. Curtis, M. Eilders, J. Evers, and J. R. Cloutier, "Vision Aided Proportional Navigation for Micro Air Vehicles," in *Proceedings of the {AIAA} Guidance, Navigation and Control Conference*, (Hilton Head, North Carolina), American Institute of Aeronautics and Astronautics, aug 2007.
- [82] M. Guelman, "Proportional Navigation with a Maneuvering Target," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 8, pp. 364–371, may 1972.
- [83] C. F. Lin, *Modern Navigation, Guidance, and Control Processing*. Englewood Cliffs, New Jersey: Prentice Hall, 1991.

Index

- Absolute Pressure Sensor, 126
- Absolution Pressure Sensor, 105
- Accelerometers, 99, 127, 147
- Aerodynamic Coefficients, 53
- Aileron, 44
- AIR SPEED, 18
- Airspeed, 21, 23, 60, 63
- Altitude, 70
- Angle of Attack, 17, 21, 43, 63
- Autopilot
 - Airspeed Hold using Throttle, 90
 - Altitude Hold Using Pitch, 88
 - Course Hold, 81
 - Pitch Attitude Hold, 87
 - Side Slip Hold, 83
- Bandwidth Separation, 83
- Coordinate Frames, 9, 239
 - Body Frame, 15, 240
 - Camera Frame, 240
 - Gimbal Frame, 240
 - Gimbal-1 Frame, 239
 - Inertial Frame, 13
 - Stability Frame, 17
 - Vehicle Frame, 13
 - Vehicle-1 Frame, 13
 - Vehicle-2 Frame, 15
 - Wind Frame, 17
- Coordinated Turn, 67, 165
- Course Angle, 21, 22, 119
- Crab Angle, 21, 23
- Design Models, 4, 67, 163
- Differential Pressure Sensor, 108, 126
- Dubins airplane, 173
- Dubins Path
 - Computation, 205
 - Definition, 204
 - RRT Coverage, 234
- RRT Paths through Obstacle Field, 230
- Tracking, 210
- Dynamics
 - Guidance Model, 170
 - Rotational Motion, 34
 - Translational Motion, 32
- Ego Motion, 247
- Elevator, 44
- Estiamtion
 - Heading, 149
- Estimation
 - Airspeed, 126
 - Altitude, 126
 - Angular Rates, 125
 - Course, 128, 149
 - Groundspeed, 128
 - Position, 128, 149
 - Roll and Pitch Angles, 127, 147
 - Wind, 149
- Euler Angles, 13, 30
 - Heading (Yaw)), 13
 - Pitch, 15
 - Roll, 15
- Fillet, 199
- Flat Earth Model, 245, 252
- Flight Path Angle
 - Air Mass Referenced, 24
 - Inertial Referenced, 21, 164, 166
- Flight-path Angle
 - Air Mass Referenced, 23
 - Focal Length, 241
- Forces
 - Drag, 43, 46, 47, 49, 170
 - Gravitation, 42
 - Lift, 43, 46, 49, 170
- Gimbal
 - Azimuth Angle, 240, 244

- Dynamics, 242
- Elevation Angle, 240, 244
- GPS, 113, 149
- GROUND SPEED, 18
- Ground Speed, 23, 24, 60, 119
- Half Plane, 198, 201, 211
- Heading Angle, 13, 21, 24
- helical path following, 192
- Inertial Matrix, 35
- Kalman Filter
 - Basic Explanation, 138
 - Continuous-Discrete to Estimate Roll and Pitch, 147
 - Continuous-Discrete to Position, Course, Wind, Heading, 149
 - Derivation, 139
 - Extended Kalman Filter, 143
 - Geolocation, 246
- Kinematics
 - Guidance Model, 168
 - Position, 31, 69, 164, 165
 - Rotation, 32, 69
- Lateral Motion, 72, 74, 79
- Linearization, 74
- Load Factor, 164
- Longitudinal Motion, 45, 52, 72, 74, 86
- Low Pass Filter, 248
- Low-pass Filter, 124
- Minimum Turning Radius, 69
- Normalized Line of Sight Vector, 242
- $\mathcal{P}_{\text{line}}$, 178
- $\mathcal{P}_{\text{orbit}}$, 183
- Path Following, 177
- path following, 188
- PID: Digital Implementation, 91
- Pitch Angle, 15
- Pitching Moment, 46, 47
- Pitot Tube, 109
- Precision Landing, 253
- Proportional Navigation, 253, 255
- Rapidly Exploring Random Trees (RRT), 224
- 3-D Terrain, 229
- Point to Point Algorithm, 226
- Smoothing Algorithm, 226
- Using Dubins Paths, 230
- Rate Gyros, 103, 125, 147
- Right Handed Rotation, 11
- Roll Angle, 15
- Rotation Matrices, 10
- Body to Gimbal, 240
- Body to Gimbal-1, 240
- Body to Stability, 17
- Body to Wind, 18
- Gimbal to Camera, 241
- Gimbal-1 to Gimbal, 240
- Stability to Wind, 17
- Vehicle to Body, 16
- Vehicle to Vehicle-1, 14
- Vehicle-1 to Vehicle-2, 15
- Vehicle-2 to Body, 16
- Rudder, 44
- Saturation Constraints, 79
- Side Slip Angle, 17, 21, 43, 63
- Simulation Model, 4, 69, 287
- Stall, 48
- State Variables, 29
- State-space Models
 - Lateral, 74
 - Longitudinal, 74
- straight-line path following, 190
- Successive Loop Closure, 77
- Time to Collision, 251
- Transfer Functions
 - Elevator to Pitch, 87
 - Pitch to Altitude, 88
 - Roll to Course, 81
 - Rudder to Side Slip, 83
 - Throttle to Airspeed, 74, 90
- Trim, 69
- Vector Field
 - Orbit, 185
 - Straight Line, 182
- vector field guidance model, 189
- Voronoi Path Planning, 218
- Waypoint Configuration, 211
- Waypoint Path, 197

- Wind Gusts, 60
 - Dryden Model, 61
- WIND SPEED, 18
- Wind Speed, 23, 24, 60
- Wind Triangle, 21, 23, 150