

## 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	19
2.4 The Wind Triangle	21
2.5 Differentiation of a Vector	24
2.6 Chapter Summary	26
2.7 Design Project	26
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	39
4.1 Gravitational Forces	39
4.2 Aerodynamic Forces and Moments	40
4.3 Propulsion Forces and Moments	52
4.4 Atmospheric Disturbances	57
4.5 Chapter Summary	59
4.6 Design Project	62
5. Linear Design Models	63
5.1 Summary of Nonlinear Equations of Motion	63
5.2 Coordinated Turn	66
5.3 Trim Conditions	68
5.4 Transfer Function Models	70
5.5 Linear State-space Models	79
5.6 Chapter Summary	91
5.7 Design Project	92

6. Autopilot Design	95
6.1 Successive Loop Closure	95
6.2 Total Energy Control	109
6.3 LQR Control	110
6.4 Chapter Summary	113
6.5 Design Project	115
7. Sensors for MAVs	117
7.1 Accelerometers	117
7.2 Rate Gyros	120
7.3 Pressure Sensors	123
7.4 Magnetometer and Digital Compass	128
7.5 Global Positioning System	131
7.6 Chapter Summary	139
7.7 Design Project	139
8. State Estimation	141
8.1 Benchmark Maneuver	141
8.2 Low-pass Filters	142
8.3 State Estimation by Inverting the Sensor Model	143
8.4 Dynamic-observer Theory	147
8.5 Derivation of the <i>Continuous-discrete</i> Kalman Filter	149
8.6 Constraints and Pseudo-Measurements	155
8.7 Measurement Gating	156
8.8 Attitude Estimation	157
8.9 GPS Smoothing	159
8.10 Full State EKF	163
8.11 Chapter Summary	171
8.12 Design Project	173
9. Design Models for Guidance	175
9.1 Autopilot Model	175
9.2 Kinematic Model of Controlled Flight	176
9.3 Kinematic Guidance Models	179
9.4 The Dubins Airplane Model	181
9.5 Dynamic Guidance Model	182
9.6 Chapter Summary	186
9.7 Design Project	186
10. Straight-line, Orbit, and Helix Path Following	189
10.1 Straight-line Path Following	190
10.2 Orbit Following	195
10.3 3D Vector-field Path Following	199
10.4 Chapter Summary	204
10.5 Design Project	207
11. Path Manager	209
11.1 Transitions Between Waypoints	209

11.2 Dubins Paths	216
11.3 Chapter Summary	226
11.4 Design Project	227
<b>12. Path Planning</b>	<b>229</b>
12.1 Point-to-Point Algorithms	230
12.2 Chapter Summary	245
12.3 Design Project	246
<b>13. Vision-guided Navigation</b>	<b>247</b>
13.1 Gimbal and Camera Frames and Projective Geometry	247
13.2 Gimbal Pointing	250
13.3 Geolocation	252
13.4 Estimating Target Motion in the Image Plane	255
13.5 Time to Collision	258
13.6 Precision Landing	261
13.7 Chapter Summary	265
13.8 Design Project	266
<b>14. Multicopters</b>	<b>267</b>
14.1 Introduction	267
14.2 Coordinate Frames	267
14.3 Kinematics and Dynamics	268
14.4 Forces and Moments	272
14.5 Linear Design Models	279
14.6 Autopilot Design	282
14.7 Sensors	286
14.8 State Estimation	287
14.10 Trajectory Following	293
14.11 Path Manager	294
14.12 Path Planning	297
14.13 Vision Guided Navigation	298
14.14 Chapter Summary	302
<b>A. Nomenclature and Notation</b>	<b>303</b>
<b>B. Quaternions</b>	<b>311</b>
B.1 Another look at complex numbers	311
B.2 Introduction to Quaternions	312
B.3 Quaternion Rotations	312
B.4 Conversion Between Euler Angles and Quaternions	314
B.5 Conversion Between Quaternions and Rotation Matrices	315
B.6 Quaternion Kinematics	315
B.7 Aircraft Kinematic and Dynamic Equations	316
<b>C. Simulation Using Object Oriented Programming</b>	<b>319</b>
C.1 Introduction	319

C.2 Numerical Solutions to Differential Equations	319
C.3 Numerical Implementation of Transfer Functions	325
C.4 Numerical Implementation of PID loops	331
D. Animation	333
D.1 3D Animation Using Python	333
D.2 3D Animation Using Matlab	334
D.3 3D Animation Using Simulink	334
D.4 Handle Graphics in Matlab	334
D.5 Animation Example: Inverted Pendulum	335
D.6 Animation Example: Spacecraft using Lines	338
D.7 Animation Example: Spacecraft Using Vertices and Faces	341
E. Airframe Parameters	345
F. Trim and Linearization	347
F.1 Numerical Computation of the Jacobian	347
F.2 Transformations Between Euler and Quaternion Representations of Attitude	349
F.3 Trim and Linearization in Simulink	352
F.4 Trim and Linearization in Matlab	354
F.5 Trim and Linearization in Python	354
G. Essentials from Probability Theory	357
H. Sensor Parameters	359
H.1 Rate Gyros	359
H.2 Accelerometers	360
H.3 Pressure Sensors	360
H.4 Digital Compass/Magnetometer	360
H.5 GPS	361
I. Useful Formulas and Other Information	363
I.1 Conversion from knots to mph	363
I.2 Density of Air	363
Bibliography	365
Index	379

# Chapter One

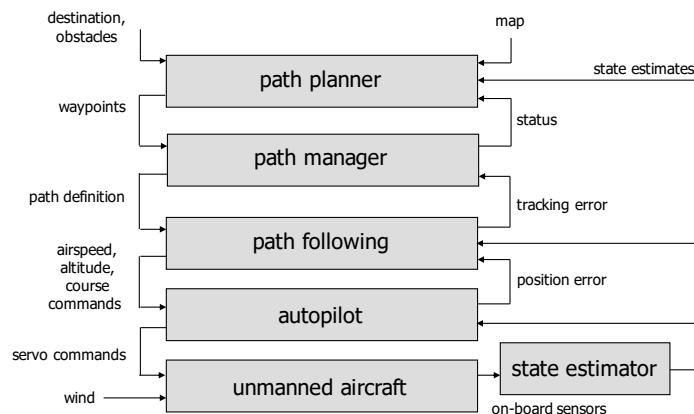
---

## 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. 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.

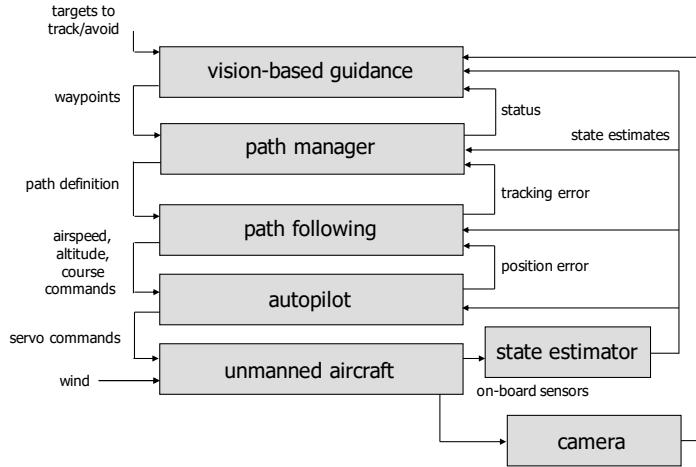
One of the primary challenges with MAVs is flight in windy conditions. Since airspeeds in the range of 20 to 40 mph are typical for MAVs, and since wind speeds at several hundred feet above ground level (AGL) almost

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.

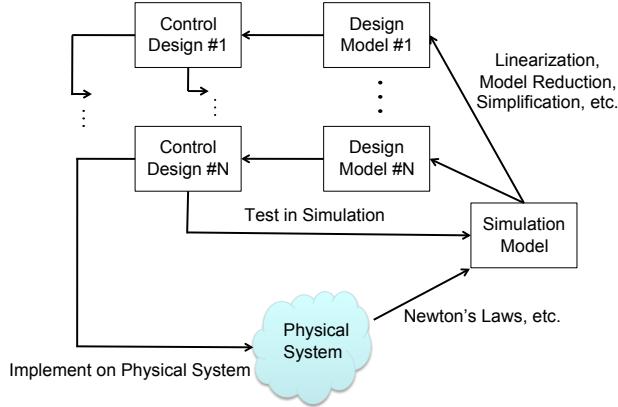


**Figure 1.2:** System architecture for vision-based navigation, guidance, and control. A video camera is added as an additional sensor and the path planner has been replaced with a vision-based guidance block.

**NEW MATERIAL:** Since the publication of the first edition of this book, multirotor vehicles have become inexpensive and ubiquitous. Therefore, in Chapter 14 we show how the principles outline in the first thirteen chapters of the book can be adapted to multirotor systems. There are, of course, many ways in which navigation, guidance, planning, and control algorithms can be developed for multirotor systems, and the literature in this area has become extensive. Our approach in Chapter 14 is to make as few adjustments as possible to the algorithms used in the rest of the book, to show simple, yet effective, techniques for multirotors.

## 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



**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.

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 for the simulation model, we should not assume that it will function properly for 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. 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.

**MODIFIED MATERIAL:** The design project involves building a MAV flight simulator from the ground up. The flight simulator can be developed in a variety of different languages and software packages. We have implemented the flight simulator in C/C++, in Matlab/Simulink, and in Python. Since new programming languages continue to appear, and since we hope to make the book independent of the choice of implementation, algorithms throughout the book are written in pseudocode. However, as a help to students, we have also developed a number of resources to show specifically how implementation can be accomplished in Matlab/Simulink and in Python. We have decided in the second edition of the book, to move all implementation specific resources to the website associated with this book, which can be accessed at <http://uavbook.byu.edu>. The website also contains template files for students, and lecture resources for instructors.

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. Tutorials on animating graphics are provided on the associated website. The assignment in chapter 3 is to drive the animation using a mathematical model of the rigid-body equations of motion. The attitude

of the aircraft is modeled throughout the textbook using Euler angles, but best-practices for implementation suggest using unit quaternions to model the attitude. Therefore, Appendix B provides a brief introduction to unit quaternions and how they are used to model aircraft attitude. Implementation of the flight simulator requires the solution of the nonlinear differential equations describing the design model. Programming languages like C/C++ and Python either require a specialized package or direct implementation of a differential equation solver. Therefore, in appendix C we show how to solve nonlinear differential equations using a first, second, and fourth order Runge-Kutta method.

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 find the trim conditions of the aircraft and to derive linear transfer-function and state-space models of the system. While there are specialized package in Simulink, for example, we show in appendix F how trim calculation can be accomplished from scratch. The associated website contains additional resources that are specific to Matlab/Simulink and Python. The assignment in chapter 6 adds autopilot functions that use feedback of the simulated true 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 airspeed, altitude, and course angle using only available sensor information. The assignment in chapter 9 is to approximate the closed-loop behavior of the full system using simple design models and to tune the parameters of the design models so that they essentially match 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

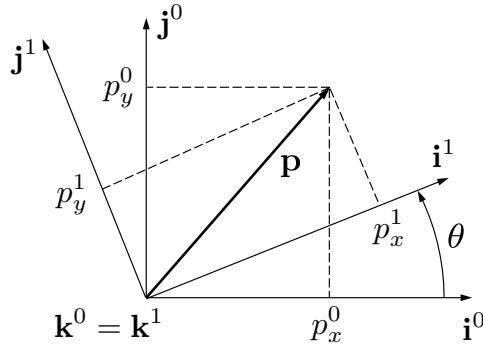
In studying unmanned aircraft systems, it is important to understand how different bodies are oriented relative to each other. Most obviously, we need to understand how the aircraft is oriented with respect to the earth. We may also want to know how a sensor (e.g., a camera) is oriented relative to the aircraft or how an antenna is oriented relative to a signal source on the ground. This chapter describes the various coordinate systems used to model 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



**Figure 2.1:** Rotation in 2D

vector  $\mathbf{p}$  can be expressed in both the  $\mathcal{F}^0$  frame (specified by  $(\mathbf{i}^0, \mathbf{j}^0, \mathbf{k}^0)$ ) 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 [that follow the right-handed rule  \$\mathbf{i} \times \mathbf{j} = \mathbf{k}\$](#) .

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 *special orthogonal* 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.

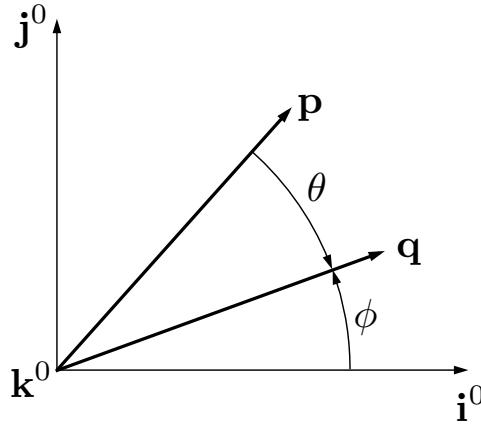
In the derivation of equation (2.1), note that the vector  $\mathbf{p}$  remains constant and the new coordinate frame  $\mathcal{F}^1$  was obtained by rotating  $\mathcal{F}^0$  through a *right-handed* rotation of angle  $\theta$ . Alternatively, rotation matrices can be used to rotate a vector through 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  $\theta$ , 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

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

and

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



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

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

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

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  $\theta$  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.

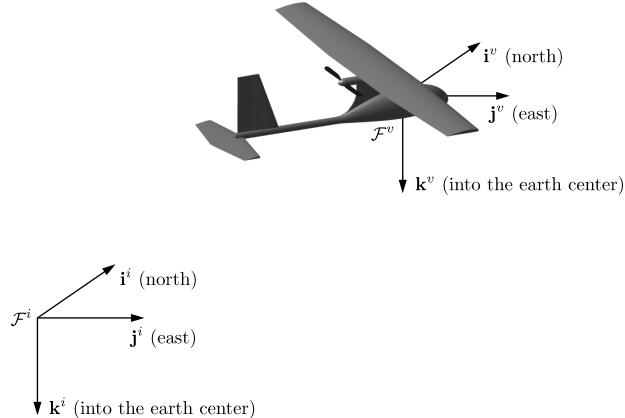
## 2.2 MAV COORDINATE FRAMES

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 translation, 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.



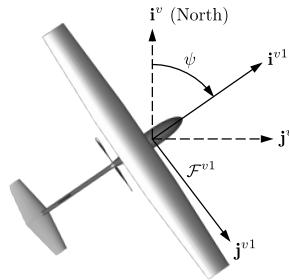
**Figure 2.3:** The inertial coordinate frame and the vehicle coordinate frame. The  $\mathbf{i}^i$  and  $\mathbf{i}^v$ -axes points [north](#), the  $\mathbf{j}^i$  and  $\mathbf{j}^v$ -axes points [east](#), and the  $\mathbf{k}^i$  and  $\mathbf{k}^v$ -axis points into the earth. The inertial frame is located at a fixed location usually on the ground. The origin of the vehicle frame is at the center of mass of the MAV.

### 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 2.3](#).

### 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  $\psi$ . In the absence of additional rotations,  $\mathbf{i}^{v1}$  points out the nose of the airframe,  $\mathbf{j}^{v1}$  points out 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 2.4.



**Figure 2.4:** 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

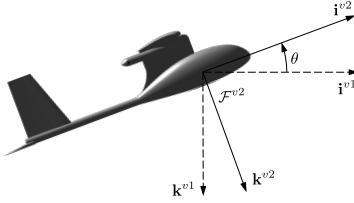
$$\mathcal{R}_v^{v1}(\psi) = \begin{pmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

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

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  $\mathbf{j}^{v1}$  axis by the pitch angle  $\theta$ . The unit vector  $\mathbf{i}^{v2}$  points out the nose of the aircraft,  $\mathbf{j}^{v2}$  points out the right wing, and  $\mathbf{k}^{v2}$  points out the belly, as shown in figure 2.5.

The transformation from  $\mathcal{F}^{v1}$  to  $\mathcal{F}^{v2}$  is given by

$$\mathbf{p}^{v2} = \mathcal{R}_{v1}^{v2}(\theta)\mathbf{p}^{v1},$$



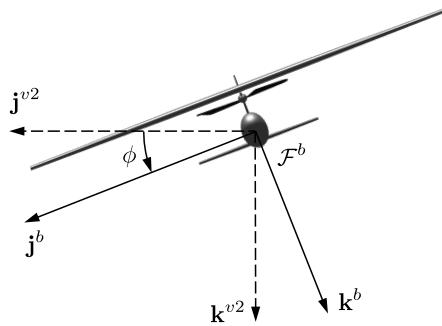
**Figure 2.5:** The vehicle-2 frame. The  $\mathbf{i}^{v2}$ -axis points out the nose of the aircraft, the  $\mathbf{j}^{v2}$ -axis points out the right wing, and the  $\mathbf{k}^{v2}$ -axis points out the belly.

where

$$\mathcal{R}_{v1}^{v2}(\theta) = \begin{pmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{pmatrix}.$$

### 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  $\mathbf{i}^{v2}$  by the roll angle  $\phi$ . Therefore, the origin is the center of mass,  $\mathbf{i}^b$  points out the nose of the airframe,  $\mathbf{j}^b$  points out the right wing, and  $\mathbf{k}^b$  points out the belly. The body frame is shown in figure 2.6. The directions indicated by the  $\mathbf{i}^b$ ,  $\mathbf{j}^b$ , and  $\mathbf{k}^b$  unit vectors are sometimes referred to as the body  $x$ , the body  $y$ , and the body  $z$  directions, respectively.



**Figure 2.6:** The body frame. The  $\mathbf{i}^b$ -axis points out the nose of the airframe, the  $\mathbf{j}^b$ -axis points out the right wing, and the  $\mathbf{k}^b$ -axis points out the belly.

The transformation from  $\mathcal{F}^{v2}$  to  $\mathcal{F}^b$  is given by

$$\mathbf{p}^b = \mathcal{R}_{v2}^b(\phi)\mathbf{p}^{v2},$$

where

$$\mathcal{R}_{v2}^b(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{pmatrix}.$$

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

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

where  $c_\phi \triangleq \cos \phi$  and  $s_\phi \triangleq \sin \phi$ . The angles  $\phi$ ,  $\theta$ , and  $\psi$  are typically referred to as the Euler angles, and are more correctly called the 3-2-1 Euler angles [2]. Euler angles are commonly used because they provide an intuitive means for representing the orientation of a body in three dimensions. The rotation sequence  $\psi$ - $\theta$ - $\phi$  is commonly used for aircraft and is just one of several Euler angle systems in use [3].

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  $\psi$ - $\theta$ - $\phi$  Euler angle sequence, there is a singularity when the pitch angle  $\theta$  is  $\pm 90$  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 unit quaternion. While the unit 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, and should be used when implementing the rotational kinematics of an aircraft in simulation.

### 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  $\alpha$ . As shown in figure 2.7, 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

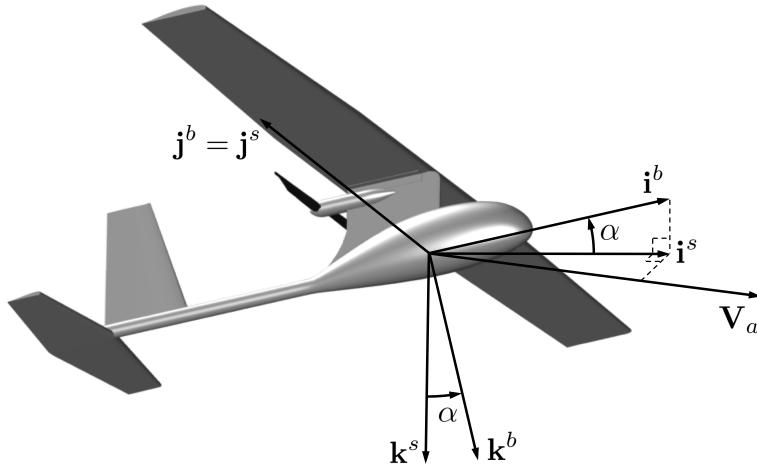
$$\mathbf{p}^b = \mathcal{R}_s^b(\alpha)\mathbf{p}^s,$$

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

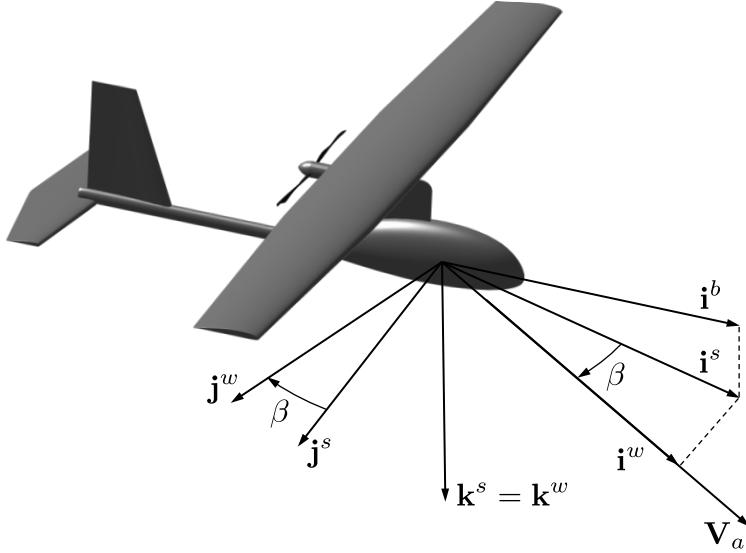
$$\mathcal{R}_b^s(\alpha) = (\mathcal{R}_s^b)^\top(\alpha) = \begin{pmatrix} \cos \alpha & 0 & \sin \alpha \\ 0 & 1 & 0 \\ -\sin \alpha & 0 & \cos \alpha \end{pmatrix}.$$



**Figure 2.7:** The stability frame. The  $\mathbf{i}^s$ -axis points along the projection of the airspeed vector onto the  $\mathbf{i}^b$ - $\mathbf{k}^b$  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.

### 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 sideslip angle and is denoted by  $\beta$ . As shown in [figure 2.8](#), the wind frame is obtained by rotating the stability frame by a right-handed rotation of  $\beta$  about  $\mathbf{k}^s$ . The unit vector  $\mathbf{i}^w$  is aligned with the airspeed vector  $\mathbf{V}_a$ .



**Figure 2.8:** The wind frame. The  $\mathbf{i}^w$ -axis points along the airspeed vector.

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

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

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 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

$$\mathbf{V}_g = \mathbf{V}_a + \mathbf{V}_w, \quad (2.5)$$

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  $\mathbf{i}^b$ ,  $\mathbf{j}^b$ , and  $\mathbf{k}^b$  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,<sup>1</sup> 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. **MODIFIED MATERIAL:** Combining expressions, we can express the airspeed vector body-frame components in terms of the airspeed magnitude, angle-of-attack, and sideslip angle as

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

which implies that

$$\begin{pmatrix} u_r \\ v_r \\ w_r \end{pmatrix} = V_a \begin{pmatrix} \cos \alpha \cos \beta \\ \sin \beta \\ \sin \alpha \cos \beta \end{pmatrix}. \quad (2.6)$$

Inverting this relationship gives

$$\begin{aligned} V_a &= \sqrt{u_r^2 + v_r^2 + w_r^2} \\ \alpha &= \tan^{-1} \left( \frac{w_r}{u_r} \right) \\ \beta &= \sin^{-1} \left( \frac{v_r}{\sqrt{u_r^2 + v_r^2 + w_r^2}} \right) = \tan^{-1} \left( \frac{v_r}{\sqrt{u_r^2 + w_r^2}} \right). \end{aligned} \quad (2.7)$$

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

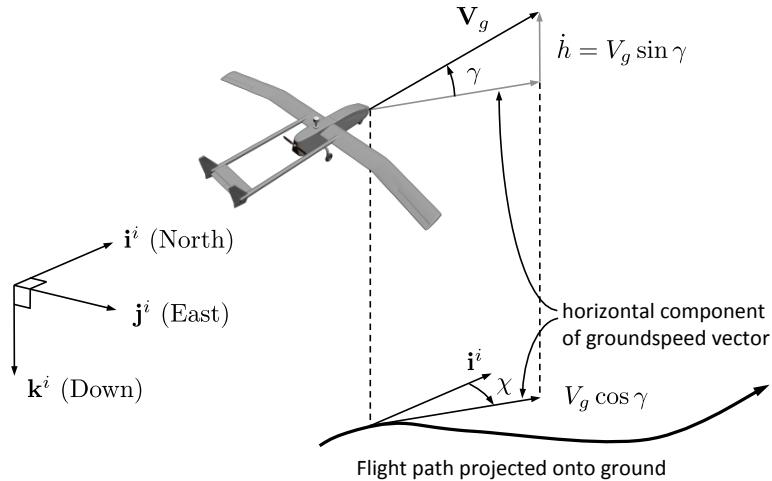
---

<sup>1</sup>Some 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.

## 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.

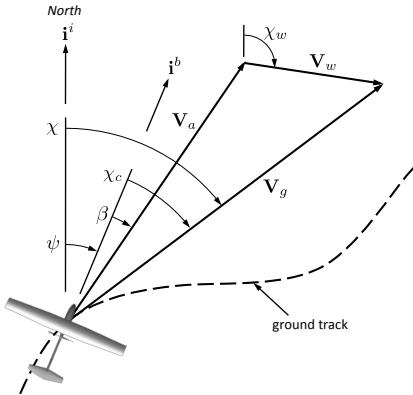
The direction of the ground speed vector relative to an inertial frame is specified using two angles. These angles are the course angle  $\chi$  and the (inertial referenced) flight path angle  $\gamma$ . Figure 2.9 shows how these two angles are defined. The flight path angle  $\gamma$  is defined as the angle between the horizontal plane and the ground velocity vector  $\mathbf{V}_g$ , while the course  $\chi$  is the angle between the projection of the ground velocity vector onto the horizontal plane and true **north**.



**Figure 2.9:** The flight path angle  $\gamma$  and the course angle  $\chi$ .

The relationship between the ground speed vector, the airspeed vector, and the wind vector, which is given by [equation \(2.5\)](#) is called the wind triangle. A more detailed depiction of the wind triangle is given in the horizontal plane in [figure 2.10](#) and in the vertical plane in [figure 2.11](#). Figure 2.10 shows an air vehicle following a ground track represented by the dashed line. The **north** direction is indicated by the  $i^i$  vector, and the direction that the vehicle is pointed is shown by the  $i^b$  vector, which is fixed in the direction of the body  $x$ -axis. For level flight, the heading (yaw) angle  $\psi$ , is the angle between  $i^i$  and  $i^b$  and defines the direction that the vehicle is pointed. The direction the vehicle is traveling with respect to the surrounding air mass is

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  $\beta$  is zero.

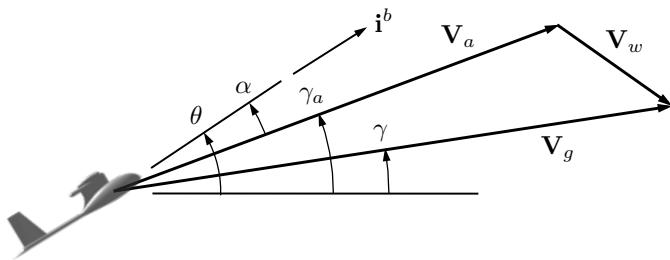


**Figure 2.10:** Heading is the direction that the MAV is pointed. Course is the direction of travel relative to the earth’s surface. The crab angle is the difference between course and heading. In the absence of wind, the crab angle is zero.

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  $\chi$ . 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*

$$\chi_c \triangleq \chi - \psi$$

is defined as the difference between the course and the heading angles.



**Figure 2.11:** The wind triangle projected onto the vertical plane.

Figure 2.11 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  $\gamma_a$ . 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

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

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

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

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

$$V_g \begin{pmatrix} \cos \chi \cos \gamma \\ \sin \chi \cos \gamma \\ -\sin \gamma \end{pmatrix} = V_a \begin{pmatrix} \cos \psi \cos \gamma_a \\ \sin \psi \cos \gamma_a \\ -\sin \gamma_a \end{pmatrix} + \begin{pmatrix} w_n \\ w_e \\ w_d \end{pmatrix}. \quad (2.8)$$

Equation (2.8) allows us to derive relationships between  $V_g$ ,  $V_a$ ,  $\chi$ ,  $\psi$ ,  $\gamma$  and  $\gamma_a$ . To find an expression for  $\gamma_a$ , multiply both sides of [equation \(2.8\)](#) 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  $\gamma_a$  to obtain

$$\gamma_a \approx \gamma + \sin^{-1} \left( \frac{1}{V_a} \begin{pmatrix} w_n \\ w_e \\ w_d \end{pmatrix}^\top \begin{pmatrix} \cos \chi \sin \gamma \\ \sin \chi \sin \gamma \\ \cos \gamma \end{pmatrix} \right). \quad (2.9)$$

To derive an expression for  $\psi$ , multiply both sides of [equation \(2.8\)](#) 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  $\psi$  gives

$$\psi = \chi - \sin^{-1} \left( \frac{1}{V_a \cos \gamma_a} \begin{pmatrix} w_n \\ w_e \end{pmatrix}^\top \begin{pmatrix} -\sin \chi \\ \cos \chi \end{pmatrix} \right). \quad (2.10)$$

An expression for groundspeed can be obtained by taking the squared norm of each side of [equation \(2.8\)](#) to get

$$V_g = \sqrt{V_a^2 + V_w^2 + 2V_a \begin{pmatrix} \cos \psi \cos \gamma_a \\ \sin \psi \cos \gamma_a \\ -\sin \gamma_a \end{pmatrix}^\top \begin{pmatrix} w_n \\ w_e \\ w_d \end{pmatrix}}, \quad (2.11)$$

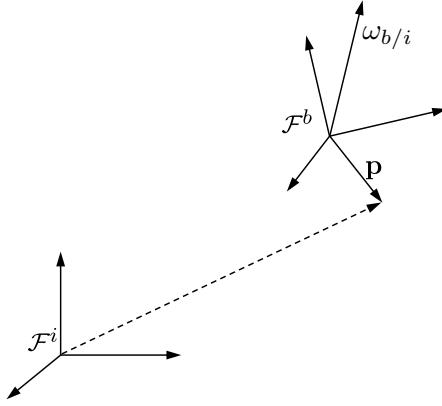
where  $V_w = \|\mathbf{V}_w\| = \sqrt{w_n^2 + w_e^2 + w_d^2}$  is the wind speed.

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  $\gamma = \gamma_a$ .

## 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 2.12](#). 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  $\boldsymbol{\omega}_{b/i}$  and express the vector  $\mathbf{p}$  in terms of its vector components as

$$\mathbf{p} = p_x \mathbf{i}^b + p_y \mathbf{j}^b + p_z \mathbf{k}^b. \quad (2.12)$$



**Figure 2.12:** A vector in a rotating reference frame.

The time derivative of  $\mathbf{p}$  with respect to frame  $\mathcal{F}^i$  can be found by differentiating [equation \(2.12\)](#) as

$$\frac{d}{dt_i} \mathbf{p} = \dot{p}_x \mathbf{i}^b + \dot{p}_y \mathbf{j}^b + \dot{p}_z \mathbf{k}^b + p_x \frac{d}{dt_i} \mathbf{i}^b + p_y \frac{d}{dt_i} \mathbf{j}^b + p_z \frac{d}{dt_i} \mathbf{k}^b, \quad (2.13)$$

where  $d/dt_i$  represents time differentiation with respect to the inertial frame. The first three terms on the right-hand side of [equation \(2.13\)](#) 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

$$\frac{d}{dt_b} \mathbf{p} = \dot{p}_x \mathbf{i}^b + \dot{p}_y \mathbf{j}^b + \dot{p}_z \mathbf{k}^b. \quad (2.14)$$

The next three terms on the right-hand side of [equation \(2.13\)](#) 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 [4] 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 \(2.13\)](#) as

$$\begin{aligned}p_x \dot{\mathbf{i}}^b + p_y \dot{\mathbf{j}}^b + p_z \dot{\mathbf{k}}^b &= p_x (\boldsymbol{\omega}_{b/i} \times \mathbf{i}^b) + p_y (\boldsymbol{\omega}_{b/i} \times \mathbf{j}^b) + p_z (\boldsymbol{\omega}_{b/i} \times \mathbf{k}^b) \\ &= \boldsymbol{\omega}_{b/i} \times \mathbf{p}.\end{aligned} \quad (2.15)$$

Combining results from [equations](#) (2.13), (2.14), and (2.15), we obtain the desired relation

$$\frac{d}{dt_i} \mathbf{p} = \frac{d}{dt_b} \mathbf{p} + \boldsymbol{\omega}_{b/i} \times \mathbf{p}, \quad (2.16)$$

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 ( $\psi$ ,  $\theta$ , and  $\phi$ ) as a means to rotate from the inertial coordinate frame to the body frame. We have also introduced the [angle-of-attack](#)  $\alpha$  and the [sideslip](#) angle  $\beta$  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-referenced 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 [\[5\]](#). Overviews of attitude representations are included in [\[2, 3\]](#). The definition of the different aircraft frames can be found in [\[1, 6, 7, 8\]](#). Vector differentiation is discussed in most textbooks on mechanics, including [\[4, 9, 10, 11\]](#).

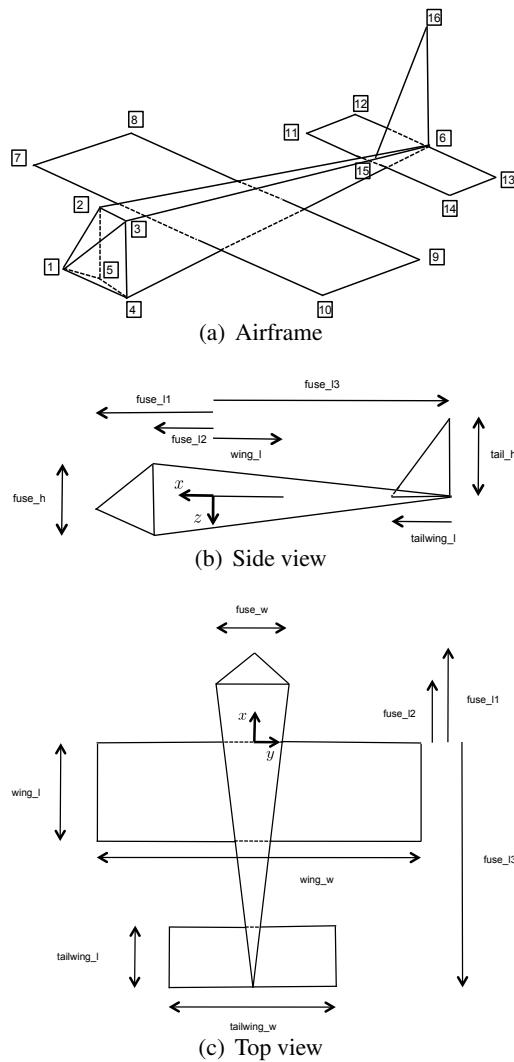
## 2.7 DESIGN PROJECT

### MODIFIED MATERIAL:

The objective of this assignment is to create a 3D rendering of a MAV that correctly displays the rotated and translated state of the MAV. Creating

animations is described in Appendix D and example files are contained on the textbook website.

- 2.1 Read Appendix D and study carefully the spacecraft animation on the textbook website.
- 2.2 Create an animation drawing of the aircraft shown in Figure 2.13.
- 2.3 Exercise your animation model by moving your aircraft through various sequences of rotations and translations. Verify that your results make sense.



**Figure 2.13:** Specifications for animation of aircraft for the design 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 [ground 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  $\phi$ , pitch  $\theta$ , and heading (yaw)  $\psi$ —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

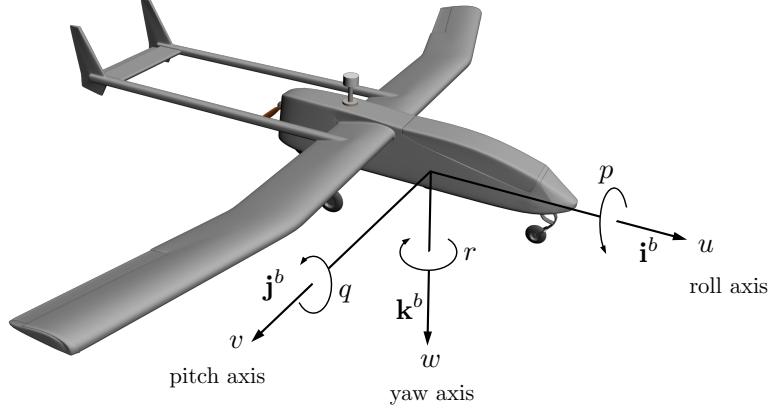
Name	Description
$p_n$	Inertial <b>north</b> position of the MAV along $\mathbf{i}^i$ in $\mathcal{F}^i$
$p_e$	Inertial <b>east</b> position of the MAV along $\mathbf{j}^i$ in $\mathcal{F}^i$
$p_d$	Inertial <b>down</b> position (negative of altitude) of the aircraft measured along $\mathbf{k}^i$ in $\mathcal{F}^i$
$u$	<b>Ground velocity</b> measured along $\mathbf{i}^b$ in $\mathcal{F}^b$
$v$	<b>Ground velocity</b> measured along $\mathbf{j}^b$ in $\mathcal{F}^b$
$w$	<b>Ground velocity</b> measured along $\mathbf{k}^b$ in $\mathcal{F}^b$
$\phi$	Roll angle defined with respect to $\mathcal{F}^{v2}$
$\theta$	Pitch angle defined with respect to $\mathcal{F}^{v1}$
$\psi$	Heading (yaw) angle defined with respect to $\mathcal{F}^v$
$p$	<b>Angular</b> rate measured along $\mathbf{i}^b$ in $\mathcal{F}^b$
$q$	<b>Angular</b> rate measured along $\mathbf{j}^b$ in $\mathcal{F}^b$
$r$	<b>Angular</b> 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 ( $\phi, \theta, \psi$ ). As we will show in the following section,  $p = \dot{\phi}$ ,  $q = \dot{\theta}$ , and  $r = \dot{\psi}$  only when  $\phi = \theta = 0$ . 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  $\phi$  and  $\theta$ . The remainder of this chapter is devoted to formulating the equations of motion corresponding to each of the states listed in table 3.1.

**MODIFIED MATERIAL:** When the equations of motion are implemented in simulation, unit quaternions are used to represent the attitude of the aircraft. Appendix B gives a brief introduction to unit quaternions and the equations of motion related to attitude kinematics. Simulating the aircraft requires numerical integration of the equations of motion described in this chapter. Appendix C provides a brief introduction to Runge-Kutta methods for numerically integrating nonlinear ordinary differential equations.

### 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



**Figure 3.1:** Definition of axes of motion.

and position requires differentiation and a rotational transformation

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

which using equation (2.4) 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.

The relationship between the Euler angles  $\phi$ ,  $\theta$ , and  $\psi$  and the angular rates  $p$ ,  $q$ , and  $r$  is 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 Euler angles are defined in three different coordinate frames.

**MODIFIED MATERIAL:** The roll angle  $\phi$  is a rotation from  $\mathcal{F}^{v2}$  to  $\mathcal{F}^b$  about the  $\mathbf{i}^{v2} = \mathbf{i}^b$  axis. Therefore, the small deviation  $\dot{\phi}$  from  $\phi$  is about the body  $\mathbf{i}^b$  axis. The pitch angle  $\theta$  is a rotation from  $\mathcal{F}^{v1}$  to  $\mathcal{F}^{v2}$  about the  $\mathbf{j}^{v1} = \mathbf{j}^{v2}$  axis. Therefore, the small deviation  $\dot{\theta}$  from  $\theta$  is about the  $v2 \mathbf{j}^{v2}$  axis. The yaw angle  $\psi$  is a rotation from  $\mathcal{F}^v$  to  $\mathcal{F}^{v1}$  about the  $\mathbf{k}^v = \mathbf{k}^{v1}$  axis. Therefore, the small deviation  $\dot{\psi}$  from  $\psi$  is about the  $v1 \mathbf{k}^{v1}$  axis.

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

formations are carried out as follows:

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

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  $\phi$  and  $\theta$  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

$$m \frac{d\mathbf{V}_g}{dt_i} = \mathbf{f}, \quad (3.4)$$

where  $m$  is the mass of the MAV,<sup>1</sup>  $\frac{d}{dt_i}$  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 \(2.16\)](#) 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 [in](#) 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

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

---

<sup>1</sup>Mass 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.

Expanding the cross product in [equation \(3.6\)](#) and rearranging terms, we get

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

### 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 \(2.16\)](#) 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 [12, 13].

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

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

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  $\mathbf{m}^b \triangleq (M_x, M_y, M_z)^\top$ , we can write equation (3.11) in component form as

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

where  $\Gamma_1 = \frac{J_{xz}(J_x - J_y + J_z)}{\Gamma}$ ,  $\Gamma_2 = \frac{J_z(J_z - J_y) + J_{xz}^2}{\Gamma}$ ,  $\Gamma_3 = \frac{J_z}{\Gamma}$ ,  $\Gamma_4 = \frac{J_{xz}}{\Gamma}$ ,  $\Gamma_5 = \frac{J_z - J_x}{J_y}$ ,  $\Gamma_6 = \frac{J_{xz}}{J_y}$ ,  $\Gamma_7 = \frac{(J_x - J_y)J_x + J_{xz}^2}{\Gamma}$ ,  $\Gamma_8 = \frac{J_x}{\Gamma}$ .

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 summarized as follows:

$$\begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{p}_d \end{pmatrix} = \begin{pmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix} \quad (3.13)$$

$$\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} = \begin{pmatrix} rv - qw \\ pw - ru \\ qu - pv \end{pmatrix} + \frac{1}{m} \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} \quad (3.14)$$

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & \sin \phi \tan \theta & \cos \phi \tan \theta \\ 0 & \cos \phi & -\sin \phi \\ 0 & \frac{\sin \phi}{\cos \theta} & \frac{\cos \phi}{\cos \theta} \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} \quad (3.15)$$

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} \Gamma_1 pq - \Gamma_2 qr \\ \Gamma_5 pr - \Gamma_6(p^2 - r^2) \\ \Gamma_7 pq - \Gamma_1 qr \end{pmatrix} + \begin{pmatrix} \Gamma_3 & 0 & \Gamma_4 \\ 0 & \frac{1}{J_y} & 0 \\ \Gamma_4 & 0 & \Gamma_8 \end{pmatrix} \begin{pmatrix} M_x \\ M_y \\ M_z \end{pmatrix}. \quad (3.16)$$

Equations (3.13)-(3.16) 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 unit quaternions to represent the MAV attitude is given. The unit quaternion formulation should be used to simulate the dynamics.

### 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, 14], space dynamics [15, 16], flight dynamics [1, 7, 8, 17, 18, 19] and robotics [5, 20].

Equations (3.13) and (3.14) 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

$$\begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{p}_d \end{pmatrix} = \mathcal{R}_b^v(\phi, \theta, \psi) \begin{pmatrix} u_r \\ v_r \\ w_r \end{pmatrix} + \begin{pmatrix} w_n \\ w_e \\ w_d \end{pmatrix} \quad (3.17)$$

$$\begin{pmatrix} \dot{u}_r \\ \dot{v}_r \\ \dot{w}_r \end{pmatrix} = \begin{pmatrix} rv_r - qw_r \\ pw_r - ru_r \\ qu_r - pv_r \end{pmatrix} + \frac{1}{m} \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} - \mathcal{R}_v^b(\phi, \theta, \psi) \begin{pmatrix} \dot{w}_n \\ \dot{w}_e \\ \dot{w}_d \end{pmatrix}, \quad (3.18)$$

where

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

The choice of which equations to use to express the aircraft kinematics is a matter of personal preference. In equations (3.13) and (3.14), the velocity states  $u$ ,  $v$ , and  $w$  represent the aircraft motion with respect to the ground (inertial frame). In equations (3.17) and (3.18), 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 (3.13) through (3.16). Assume that the inputs 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. The equations of motion should be implemented using unit quaternions to represent attitude as described in Appendix B.
- 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 simulated 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  $M_x$  and  $M_y$  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 [18], 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

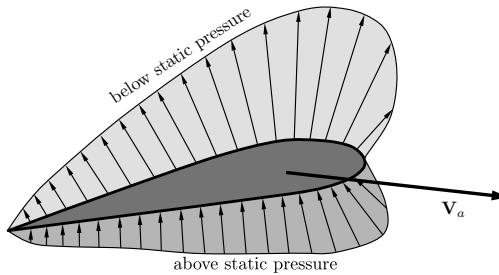
force into its body-frame components to give

$$\mathbf{f}_g^b = \mathcal{R}_v^b \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} = mg \begin{pmatrix} -\sin \theta \\ \cos \theta \sin \phi \\ \cos \theta \cos \phi \end{pmatrix} = mg \begin{pmatrix} 2(e_x e_z - e_y e_0) \\ 2(e_y e_z + e_x e_0) \\ e_z^2 + e_0^2 - e_x^2 - e_y^2 \end{pmatrix},$$

where the first expression is in terms of Euler angles and second expression is in terms of the unit quaternion. Since the gravity force acts through the center of mass of the MAV, there are no moments produced by gravity.

## 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  $\rho$  is the air density and  $V_a$  is the speed of the MAV through the surrounding air mass.

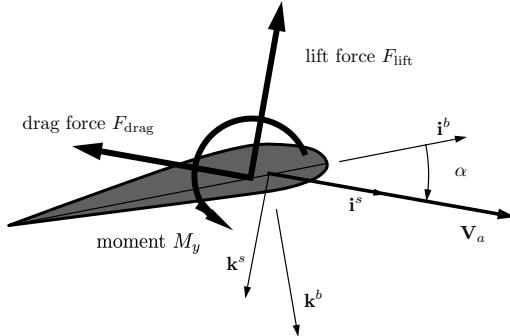


**Figure 4.1:** Pressure distribution around an airfoil.

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 ( $i^b$ - $k^b$ ) 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.

The lift, drag, and moment are commonly expressed as

$$\begin{aligned} F_{\text{lift}} &= \frac{1}{2}\rho V_a^2 S C_L \\ F_{\text{drag}} &= \frac{1}{2}\rho V_a^2 S C_D \end{aligned} \tag{4.1}$$



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

$$M_y = \frac{1}{2} \rho V_a^2 S c C_{M_y},$$

where  $C_L$ ,  $C_D$ , and  $C_{M_y}$  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  $\alpha$  and  $\beta$ ; 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.

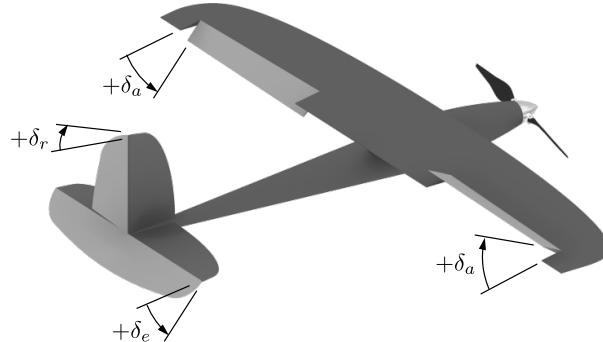
#### 4.2.1 Control Surfaces

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  $\delta_a$ , the elevator deflection is denoted by  $\delta_e$ , and the rudder deflection is denoted by  $\delta_r$ . 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  $\delta_a$  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  $\delta_a$  is produced when the left aileron is trailing edge down and the right aileron is trailing edge up.

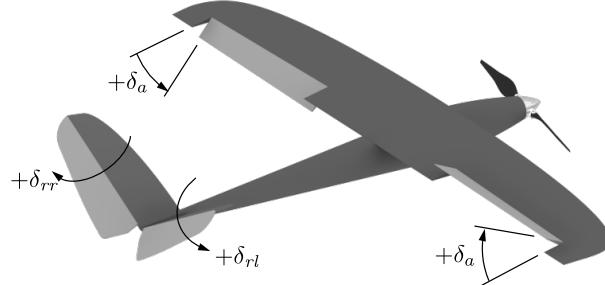


**Figure 4.3:** Control surfaces for a standard aircraft configuration. The ailerons are used to control the roll angle  $\phi$ . The elevators are used to control the pitch angle  $\theta$ . The rudder directly effects the yaw angle  $\psi$ .

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

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

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.

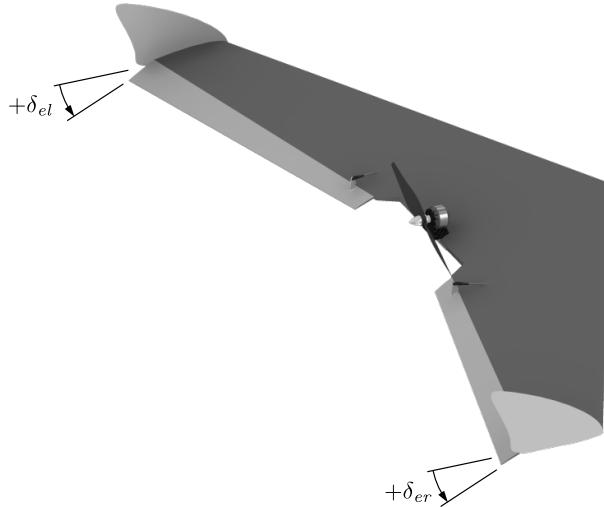
The other standard configuration for small aircraft is the flying wing depicted in figure 4.5. The control surfaces for a flying wing are called elevons. The angular deflection of the right elevon is denoted as  $\delta_{er}$ , and the angular deflection of the left elevon is denoted as  $\delta_{el}$ . 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  $\mathbf{j}^b$ . **MODIFIED MATERIAL:** Mathematically, we can convert between elevons and aileron-elevator signals with

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

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  $\mathbf{i}^b\text{-}\mathbf{k}^b$  plane, also known as the pitch plane. They are the aerodynamic forces 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  $\mathbf{j}^s$  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  $\delta_e$  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



**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.

dependence on  $\alpha$ ,  $q$  and  $\delta_e$  as

$$F_{\text{lift}} = \frac{1}{2} \rho V_a^2 S C_L(\alpha, q, \delta_e)$$

$$F_{\text{drag}} = \frac{1}{2} \rho V_a^2 S C_D(\alpha, q, \delta_e)$$

$$M_y = \frac{1}{2} \rho V_a^2 S c C_{M_y}(\alpha, q, \delta_e).$$

In general, these force and moment equations are nonlinear. For small [angles-of-attack](#), however, the flow over the wing will remain attached. Under these conditions, the lift, drag, and pitching moment coefficients can be modeled with acceptable accuracy using linear approximations. Working with the lift equation as an example, a first-order Taylor series approximation 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  $\alpha$  and  $\delta_e$  (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  $C_{L_0}$ ,  $C_{L_\alpha} \triangleq \frac{\partial C_L}{\partial \alpha}$ ,  $C_{L_q} \triangleq \frac{\partial C_L}{\partial \frac{qc}{2V_a}}$ , and  $C_{L_{\delta_e}} \triangleq \frac{\partial C_L}{\partial \delta_e}$  are dimensionless quantities.  $C_{L_\alpha}$  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

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

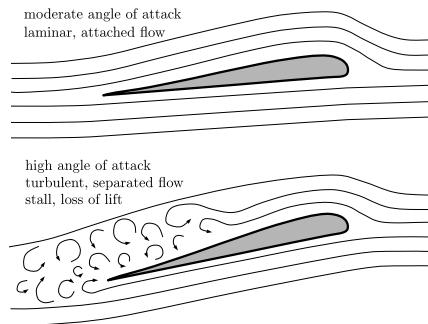
$$M_y = \frac{1}{2}\rho V_a^2 Sc \left[ C_{M_{y0}} + C_{M_{y\alpha}} \alpha + C_{M_{yq}} \frac{c}{2V_a} q + C_{M_{y\delta_e}} \delta_e \right]. \quad (4.5)$$

[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. They 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  $\alpha$ . 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 nonlinear functions of  $\alpha$ . 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 [19]

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

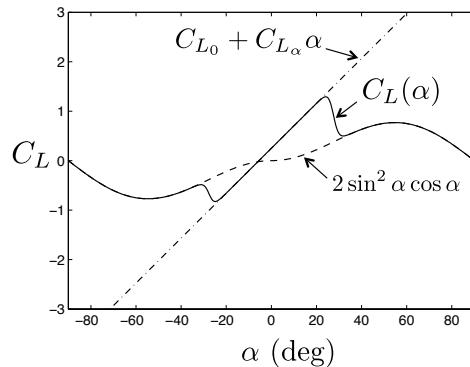
To obtain an accurate model of lift versus [angle-of-attack](#) for a specific wing design over a large range of [angles-of-attack](#) requires either wind tunnel testing or a detailed computational study. While for many simulation purposes it may not be necessary to have a high-fidelity lift model specific 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  $\alpha_0$  are positive constants. The sigmoid function in (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,



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

the linear lift coefficient can be reasonably approximated as

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

where  $A_R \triangleq b^2/S$  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 [19]. The parasitic drag, generated by the shear stress of air moving over the wing and other effects, is roughly constant and is denoted by  $C_{D_p}$ .<sup>1</sup> 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 [MODIFIED MATERIAL](#):

$$\begin{aligned} C_D(\alpha) &= C_{D_p} + \frac{(C_{L_0} + C_{L_\alpha}\alpha)^2}{\pi e_{os} A_R} \\ &= \left( C_{D_p} + \frac{C_{L_0}^2}{\pi e_{os} A_R} \right) + \left( \frac{2C_{L_0}C_{L_\alpha}}{\pi e_{os} A_R} \right) \alpha + \left( \frac{C_{L_\alpha}^2}{\pi e_{os} A_R} \right) \alpha^2, \end{aligned} \quad (4.11)$$

which is approximately valid for  $-\frac{\pi}{2} \leq \alpha \leq \frac{\pi}{2}$ . The parameter  $e_{os}$  is the Oswald efficiency factor, which ranges between 0.8 and 1.0 [8].

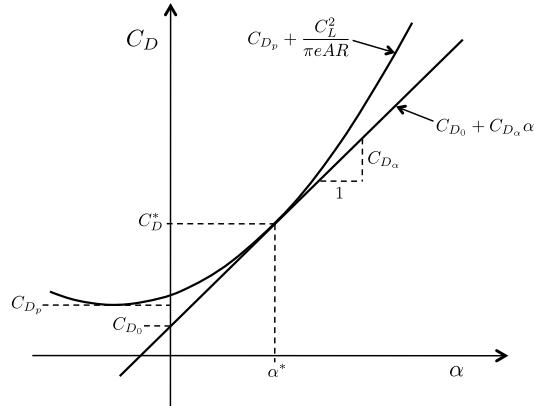
Figure 4.8 shows typical plots of drag coefficient versus [angle-of-attack](#) for quadratic and linear models. The quadratic model correctly models the drag force as an odd function with respect to  $\alpha$ . The drag force is always opposite the forward velocity of the aircraft, independent of the 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  $\alpha^*$  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  $\alpha$ , the linear model is sometimes used because of its simplicity and its fidelity under typical flight conditions.

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](#):

$$\begin{pmatrix} f_x \\ f_z \end{pmatrix} = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix} \begin{pmatrix} -F_{\text{drag}} \\ -F_{\text{lift}} \end{pmatrix}$$

---

<sup>1</sup>The 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 [8, pp. 179–180] for a detailed explanation.



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

$$= \frac{1}{2} \rho V_a^2 S \begin{pmatrix} [-C_D(\alpha) \cos \alpha + C_L(\alpha) \sin \alpha] \\ + [-C_{D_q} \cos \alpha + C_{L_q} \sin \alpha] \frac{c}{2V_a} q \\ + [-C_{D_{\delta_e}} \cos \alpha + C_{L_{\delta_e}} \sin \alpha] \delta_e \\ \cdots \\ [-C_D(\alpha) \sin \alpha - C_L(\alpha) \cos \alpha] \\ + [-C_{D_q} \sin \alpha - C_{L_q} \cos \alpha] \frac{c}{2V_a} q \\ + [-C_{D_{\delta_e}} \sin \alpha - C_{L_{\delta_e}} \cos \alpha] \delta_e \end{pmatrix}.$$

The functions  $C_L(\alpha)$  and  $C_D(\alpha)$  used in the force model above are the 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 the derivative of angle-of-attack  $\dot{\alpha}$  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

$$C_{M_y}(\alpha) = C_{M_{y0}} + C_{M_{y\alpha}} \alpha,$$

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

### 4.2.3 Lateral-Directional Aerodynamics

The lateral-directional 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-directional aerodynamics are most significantly influenced by the [sideslip angle](#)  $\beta$ . They are also influenced by the roll rate  $p$ , the yaw rate  $r$ , the deflection of the aileron  $\delta_a$ , and the deflection of the rudder  $\delta_r$ . Denoting the lateral force as  $f_y$  and the roll and yaw moments as  $M_x$  and  $M_z$ , respectively, we have

$$f_y = \frac{1}{2}\rho V_a^2 S C_Y(\beta, p, r, \delta_a, \delta_r)$$

$$M_x = \frac{1}{2}\rho V_a^2 S b C_{M_x}(\beta, p, r, \delta_a, \delta_r)$$

$$M_z = \frac{1}{2}\rho V_a^2 S b C_{M_z}(\beta, p, r, \delta_a, \delta_r),$$

where  $C_Y$ ,  $C_{M_x}$ , and  $C_{M_z}$  are nondimensional aerodynamic coefficients, and  $b$  is the wingspan of the aircraft. As with the longitudinal aerodynamic forces and moments, the coefficients  $C_Y$ ,  $C_{M_x}$ , and  $C_{M_z}$  are nonlinear in their constitutive parameters, in this case  $\beta$ ,  $p$ ,  $r$ ,  $\delta_a$ , and  $\delta_r$ . 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

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

$$\begin{aligned} M_x &= \frac{1}{2}\rho V_a^2 S b \left[ C_{M_{x0}} + C_{M_{x\beta}} \beta + C_{M_{xp}} \frac{b}{2V_a} p + C_{M_{xr}} \frac{b}{2V_a} r \right. \\ &\quad \left. + C_{M_{x\delta_a}} \delta_a + C_{M_{x\delta_r}} \delta_r \right] \end{aligned} \quad (4.15)$$

$$\begin{aligned} M_z &= \frac{1}{2}\rho V_a^2 S b \left[ C_{M_{z0}} + C_{M_{z\beta}} \beta + C_{M_{zp}} \frac{b}{2V_a} p + C_{M_{zr}} \frac{b}{2V_a} r \right. \\ &\quad \left. + C_{M_{z\delta_a}} \delta_a + C_{M_{z\delta_r}} \delta_r \right]. \end{aligned} \quad (4.16)$$

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. **MODIFIED MATERIAL:** The coefficient  $C_{Y_0}$  is the value of the lateral force coefficient  $C_Y$  when  $\beta = p = r = \delta_a = \delta_r = 0$ . For aircraft that are symmetrical about the  $i^b$ - $k^b$  plane,  $C_{Y_0}$  is typically zero. The coefficients  $C_{M_{x0}}$  and  $C_{M_{z0}}$  are defined similarly and are also typically zero for symmetric aircraft.

#### 4.2.4 Aerodynamic Coefficients

The aerodynamic coefficients  $C_{M_{y\alpha}}$ ,  $C_{M_{x\beta}}$ ,  $C_{M_{z\beta}}$ ,  $C_{M_{yq}}$ ,  $C_{M_{xp}}$ , and  $C_{M_{zr}}$  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  $C_{M_{y\alpha}}$ ,  $C_{M_{x\beta}}$ , and  $C_{M_{z\beta}}$  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  $\alpha$  and  $\beta$ .

$C_{M_{y\alpha}}$  is referred to as the longitudinal static stability derivative. For the MAV to be statically stable,  $C_{M_{y\alpha}}$  must be less than zero. In this case, an increase in  $\alpha$  due to an updraft would cause the MAV to nose down in order to maintain the nominal [angle-of-attack](#).

$C_{M_{x\beta}}$  is called the roll static stability derivative and is typically associated with dihedral in the wings. For static stability in roll,  $C_{M_{x\beta}}$  must be negative. A negative value for  $C_{M_{x\beta}}$  will result in rolling moments that roll the MAV away from the direction of [sideslip](#), thereby driving the [sideslip](#) angle  $\beta$  to zero.

$C_{M_{z\beta}}$  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 weathercock). The value of  $C_{M_{z\beta}}$  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  $C_{M_{z\beta}}$  will be. For the MAV to be stable in yaw,  $C_{M_{z\beta}}$  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  $C_{M_{y\alpha}}$ ,  $C_{M_{x\beta}}$ , and  $C_{M_{z\beta}}$  behave like torsional springs, while the derivatives  $C_{M_{yq}}$ ,  $C_{M_{xp}}$ , and  $C_{M_{zr}}$  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.

$C_{M_{yq}}$  is referred to as the pitch damping derivative,  $C_{M_{xp}}$  is called the roll damping derivative, and  $C_{M_{zr}}$  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  $C_{M_{y\delta_e}}$ ,  $C_{M_{x\delta_a}}$ , and  $C_{M_{z\delta_r}}$  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  $\delta_e$  is a pitching moment  $M_y$ .  $C_{M_{x\delta_r}}$  and  $C_{M_{z\delta_a}}$  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,  $C_{M_{y\delta_e}}$  will be negative,  $C_{M_{x\delta_a}}$  will be positive, and  $C_{M_{z\delta_r}}$  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 [21], the standard model for the thrust and torque produced by a propeller is given by

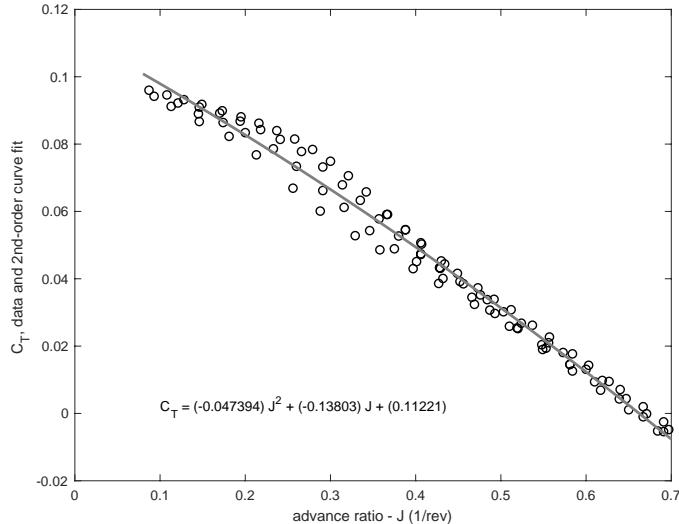
$$T_p(\Omega_p, C_T) = \frac{\rho D^4}{4\pi^2} \Omega_p^2 C_T$$

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

where  $\rho$  is the density of air,  $D$  is the propeller diameter,  $\Omega_p$  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 approx-



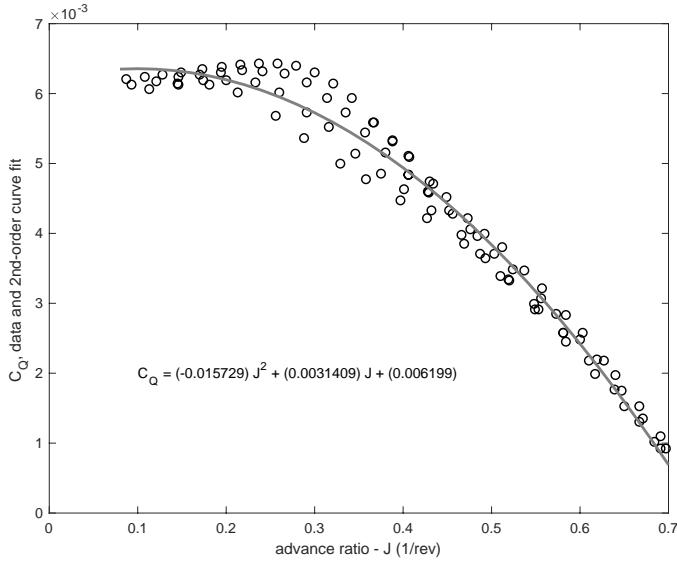
**Figure 4.9:** Thrust coefficient versus advance ratio for APC Thin Electric 10x5 propeller [22].

imated as quadratic functions of  $J$ . Accordingly we have

$$C_T(J) \approx C_{T0} + C_{T1}J + C_{T2}J^2$$

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

where the coefficients  $C_{T*}$  and  $C_{Q*}$  are unitless coefficients that are determined experimentally from data. The quadratic approximations are shown



**Figure 4.10:** Torque coefficient versus advance ratio for APC Thin Electric 10x5 propeller [22].

as solid lines in figures 4.9 and 4.10. Combining these equations, the thrust 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  $\Omega_p$  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,  $\Omega_m$  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 energy domains. If consistent units (such as SI units) are utilized,  $K_Q$  and  $K_V$  are identical in value.<sup>2</sup> 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

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

Denoting the coefficients of this quadratic equation as

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

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  $V_{in}$ .

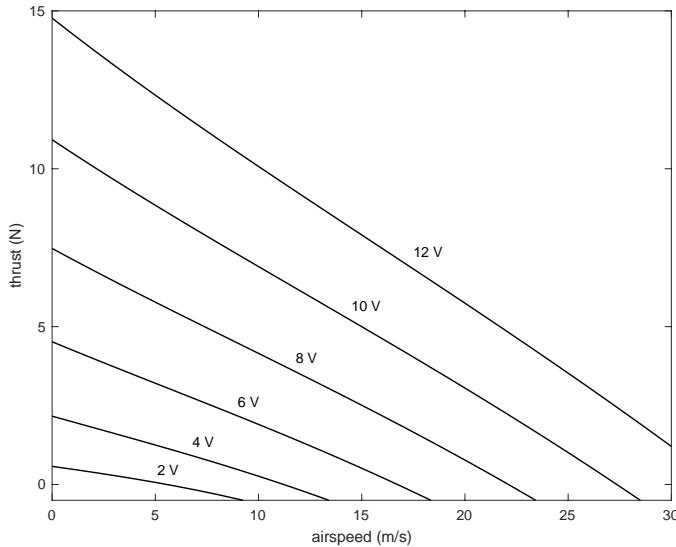
Using the approach above to find the operating speed, the thrust and torque characteristics of the motor/propeller combination over the full range of voltage and airspeed inputs can be quantified. A family of curves, one for each level of voltage input, is plotted versus airspeed and shows the thrust 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

---

<sup>2</sup>We assume the torque constant  $K_Q$  has units of N-m/A and back-emf constant  $K_V$  has units of V-sec/rad. In RC aircraft motor datasheets, the voltage constant  $K_V^*$  (the inverse of  $K_V$ ) is commonly specified with units of rpm/V, in which case  $K_Q = 60/(2\pi K_V^*)$ .

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.



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

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.

---

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

---

- Compute  $V_{in}$  using equation (4.22)
  - Compute  $\Omega_p$  using equation (4.21)
  - Compute  $T_p$  using equation (4.17)
  - Compute  $Q_p$  using equation (4.18)
-

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, 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 (2.5), 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

**Table 4.1:** Dryden gust model parameters [23].

gust description	altitude (m)	$L_u = L_v$ (m)	$L_w$ (m)	$\sigma_u = \sigma_v$ (m/s)	$\sigma_w$ (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

directions. The gust portion of the wind can be written in terms of its body-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 [19]. 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  $\sigma_u$ ,  $\sigma_v$ , and  $\sigma_w$  are the intensities of the turbulence along the body 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 [23] 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 (2.4). From the components of the wind velocity  $\mathbf{V}_w^b$  and the 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 (2.7) as

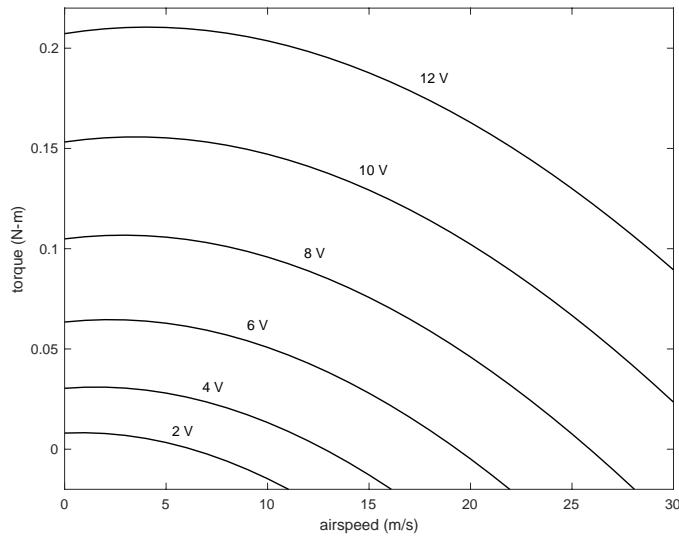
$$\begin{aligned} V_a &= \sqrt{u_r^2 + v_r^2 + w_r^2} \\ \alpha &= \tan^{-1} \left( \frac{w_r}{u_r} \right) \\ \beta &= \sin^{-1} \left( \frac{v_r}{\sqrt{u_r^2 + v_r^2 + w_r^2}} \right). \end{aligned}$$

These expressions for  $V_a$ ,  $\alpha$ , and  $\beta$  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.

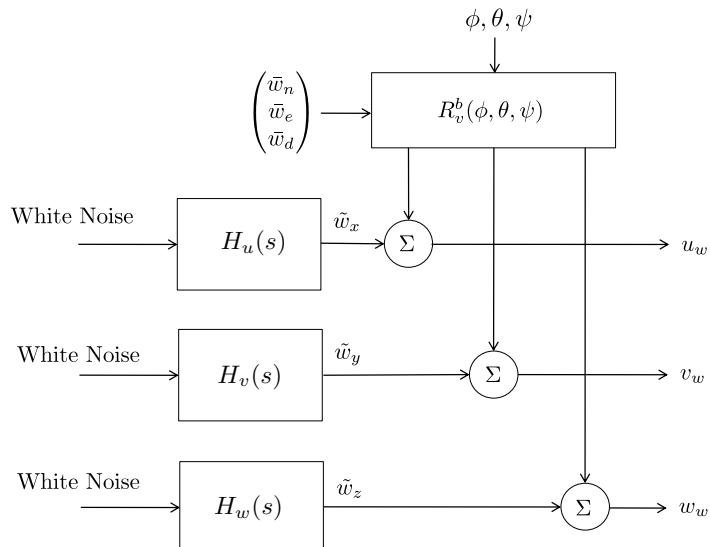
Numerical implementation of transfer function is discussed in [Appendix C.3](#).

## 4.5 CHAPTER SUMMARY

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



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



**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.

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

where

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

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:

$$\begin{pmatrix} M_x \\ M_y \\ M_z \end{pmatrix} = \frac{1}{2} \rho V_a^2 S \begin{pmatrix} b \left[ C_{M_{x0}} + C_{M_{x\beta}} \beta + C_{M_{xp}} \frac{b}{2V_a} p + C_{M_{xr}} \frac{b}{2V_a} r \right] \\ c \left[ C_{M_{y0}} + C_{M_{y\alpha}} \alpha + C_{M_{yq}} \frac{c}{2V_a} q \right] \\ b \left[ C_{M_{z0}} + C_{M_{z\beta}} \beta + C_{M_{zp}} \frac{b}{2V_a} p + C_{M_{zr}} \frac{b}{2V_a} r \right] \end{pmatrix} + \frac{1}{2} \rho V_a^2 S \begin{pmatrix} b \left[ C_{M_{x\delta_a}} \delta_a + C_{M_{x\delta_r}} \delta_r \right] \\ c \left[ C_{M_{y\delta_e}} \delta_e \right] \\ b \left[ C_{M_{z\delta_a}} \delta_a + C_{M_{z\delta_r}} \delta_r \right] \end{pmatrix} + \begin{pmatrix} -Q_p(\delta_t, V_a) \\ 0 \\ 0 \end{pmatrix}. \quad (4.26)$$

## NOTES AND REFERENCES

The material in this chapter can be found in most textbooks on flight dynamics, including [8, 19, 7, 17, 18, 1, 24]. Our discussion on lift, drag, and moment coefficients is drawn primarily from [19]. Decomposing the wind

vector into a constant and a random term follows [19]. 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 [24].

#### 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 the 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](#) 5.1. Because of their complexity, designing controllers based on these equations 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, [and the a yaw rate damper](#). 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](#) 5.3. Transfer functions for both the lateral and longitudinal dynamics are derived in [section](#) 5.4. State-space models are derived in [section](#) 5.5.

#### 5.1 SUMMARY OF NONLINEAR EQUATIONS OF MOTION

**MODIFIED MATERIAL:** A variety of models for the aerodynamic forces and moments appear in the literature ranging from linear, uncoupled models to highly nonlinear models with significant cross coupling. In this section, we summarize the six-degree-of-freedom, 12-state equations of motion with the quasi-linear aerodynamic and propulsion models developed in [chapter](#) 4.

We characterize them as quasi-linear because the lift and drag terms are nonlinear in the angle-of-attack, and the propeller thrust is nonlinear in the throttle command. For completeness, we will also present the linear models for lift and drag that are commonly used. Incorporating the aerodynamic and propulsion models described in chapter 4 into equations (3.13)–(3.16), we get the following equations of motion:

$$\begin{aligned}\dot{p}_n &= (\cos \theta \cos \psi)u + (\sin \phi \sin \theta \cos \psi - \cos \phi \sin \psi)v \\ &\quad + (\cos \phi \sin \theta \cos \psi + \sin \phi \sin \psi)w\end{aligned}\tag{5.1}$$

$$\begin{aligned}\dot{p}_e &= (\cos \theta \sin \psi)u + (\sin \phi \sin \theta \sin \psi + \cos \phi \cos \psi)v \\ &\quad + (\cos \phi \sin \theta \sin \psi - \sin \phi \cos \psi)w\end{aligned}\tag{5.2}$$

$$\dot{h} = u \sin \theta - v \sin \phi \cos \theta - w \cos \phi \cos \theta\tag{5.3}$$

$$\begin{aligned}\dot{u} &= rv - qw - g \sin \theta + \frac{\rho V_a^2 S}{2m} \left[ C_X(\alpha) + C_{X_q}(\alpha) \frac{cq}{2V_a} \right. \\ &\quad \left. + C_{X_{\delta_e}}(\alpha) \delta_e \right] + \frac{T_p(\delta_t, V_a)}{m}\end{aligned}\tag{5.4}$$

$$\begin{aligned}\dot{v} &= pw - ru + g \cos \theta \sin \phi + \frac{\rho V_a^2 S}{2m} \left[ C_{Y_0} + C_{Y_\beta} \beta \right. \\ &\quad \left. + C_{Y_p} \frac{bp}{2V_a} + C_{Y_r} \frac{br}{2V_a} + C_{Y_{\delta_a}} \delta_a + C_{Y_{\delta_r}} \delta_r \right]\end{aligned}\tag{5.5}$$

$$\begin{aligned}\dot{w} &= qu - pv + g \cos \theta \cos \phi + \frac{\rho V_a^2 S}{2m} \left[ C_Z(\alpha) \right. \\ &\quad \left. + C_{Z_q}(\alpha) \frac{cq}{2V_a} + C_{Z_{\delta_e}}(\alpha) \delta_e \right]\end{aligned}\tag{5.6}$$

$$\dot{\phi} = p + q \sin \phi \tan \theta + r \cos \phi \tan \theta\tag{5.7}$$

$$\dot{\theta} = q \cos \phi - r \sin \phi\tag{5.8}$$

$$\dot{\psi} = q \sin \phi \sec \theta + r \cos \phi \sec \theta\tag{5.9}$$

$$\begin{aligned}\dot{p} &= \Gamma_1 pq - \Gamma_2 qr + \frac{1}{2} \rho V_a^2 S b \left[ C_{p_0} + C_{p_\beta} \beta + C_{p_p} \frac{bp}{2V_a} \right. \\ &\quad \left. + C_{p_r} \frac{br}{2V_a} + C_{p_{\delta_a}} \delta_a + C_{p_{\delta_r}} \delta_r \right] - \Gamma_3 Q_p(\delta_t, V_a)\end{aligned}\tag{5.10}$$

$$\begin{aligned}\dot{q} &= \Gamma_5 pr - \Gamma_6 (p^2 - r^2) + \frac{\rho V_a^2 Sc}{2J_y} \left[ C_{M_{y0}} + C_{M_{y\alpha}} \alpha \right. \\ &\quad \left. + C_{M_{yq}} \frac{cq}{2V_a} + C_{M_{y\delta_e}} \delta_e \right]\end{aligned}\tag{5.11}$$

$$\dot{r} = \Gamma_7 pq - \Gamma_1 qr + \frac{1}{2} \rho V_a^2 S b \left[ C_{r_0} + C_{r_\beta} \beta + C_{r_p} \frac{bp}{2V_a} \right]$$

$$+ C_{r_r} \frac{br}{2V_a} + C_{r_{\delta_a}} \delta_a + C_{r_{\delta_r}} \delta_r \Big] - \Gamma_4 Q_p(\delta_t, V_a), \quad (5.12)$$

where  $h = -p_d$  is the altitude and

$$\begin{aligned} C_{p_0} &= \Gamma_3 C_{M_{x0}} + \Gamma_4 C_{M_{z0}} \\ C_{p_\beta} &= \Gamma_3 C_{M_{x\beta}} + \Gamma_4 C_{M_{z\beta}} \\ C_{p_p} &= \Gamma_3 C_{M_{xp}} + \Gamma_4 C_{M_{zp}} \\ C_{p_r} &= \Gamma_3 C_{M_{xr}} + \Gamma_4 C_{M_{zr}} \\ C_{p_{\delta_a}} &= \Gamma_3 C_{M_{x\delta_a}} + \Gamma_4 C_{M_{z\delta_a}} \\ C_{p_{\delta_r}} &= \Gamma_3 C_{M_{x\delta_r}} + \Gamma_4 C_{M_{z\delta_r}} \\ C_{r_0} &= \Gamma_4 C_{M_{x0}} + \Gamma_8 C_{M_{z0}} \\ C_{r_\beta} &= \Gamma_4 C_{M_{x\beta}} + \Gamma_8 C_{M_{z\beta}} \\ C_{r_p} &= \Gamma_4 C_{M_{xp}} + \Gamma_8 C_{M_{zp}} \\ C_{r_r} &= \Gamma_4 C_{M_{xr}} + \Gamma_8 C_{M_{zr}} \\ C_{r_{\delta_a}} &= \Gamma_4 C_{M_{x\delta_a}} + \Gamma_8 C_{M_{z\delta_a}} \\ C_{r_{\delta_r}} &= \Gamma_4 C_{M_{x\delta_r}} + \Gamma_8 C_{M_{z\delta_r}}. \end{aligned}$$

The inertia parameters specified by  $\Gamma_1, \Gamma_2, \dots, \Gamma_8$  are defined in [equation \(??\)](#). 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

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

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  $\alpha_0$  are positive constants.

**MODIFIED MATERIAL:** Further, it is common to model drag as a non-linear 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  $A_R$  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 used 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.

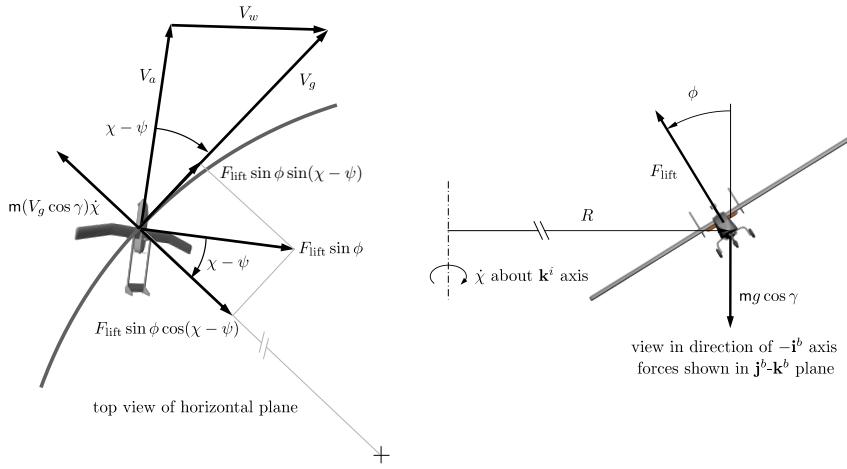
## 5.2 COORDINATED TURN

Referring to [equation \(5.9\)](#), 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 relationships in coming sections of this chapter. The coordinated-turn condition provides this relationship. The coordinated turn is a sought-after flight con-

dition 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 [24]. During a coordinated turn, the bank angle  $\phi$  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 to the horizontal component of the lift force. Summing forces in the horizontal direction gives

$$\begin{aligned} F_{\text{lift}} \sin \phi \cos(\chi - \phi) &= m \frac{v^2}{R} \\ &= mv\omega \\ &= m(V_g \cos \gamma) \dot{\chi}, \end{aligned} \quad (5.13)$$

where  $F_{\text{lift}}$  is the lift force,  $\gamma$  is the flight path angle,  $V_g$  is the ground speed, and  $\chi$  is the course angle. **MODIFIED MATERIAL:** The centrifugal force



**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  $\gamma$  with the nose of the MAV directed out of the page. The forces shown are in the  $j^b$ - $k^b$  plane.

is calculated using the angular rate  $\dot{\chi}$  about the inertial frame  $k^i$  axis and the horizontal component of the ground speed,  $V_g \cos \gamma$ . Similarly, the vertical component of the lift force has to cancel the projection of the gravitational force onto the  $j^b$ - $k^b$  plane as shown in figure 5.1 to remain at a climb rate, and therefore

$$F_{\text{lift}} \cos \phi = mg \cos \gamma. \quad (5.14)$$

Dividing [equation \(5.13\)](#) by [equation \(5.14\)](#) and solving for  $\dot{\chi}$  gives

$$\dot{\chi} = \frac{g}{V_g} \tan \phi \cos(\chi - \psi), \quad (5.15)$$

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

$$R = \frac{V_g^2 \cos \gamma}{g \tan \phi \cos(\chi - \phi)}. \quad (5.16)$$

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

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

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 [24, 25, 26], and we will have more to say about coordinated turns in [section 9.2](#) where we will show that

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

also holds true in the presence of wind.

### 5.3 TRIM CONDITIONS

Given a nonlinear system described by the differential equations

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

where  $f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ ,  $x$  is the state of the system, and  $\mathbf{u}$  is the input, the system is said to be in equilibrium at the state  $\mathbf{x}^*$  and input  $\mathbf{u}^*$  if

$$f(\mathbf{x}^*, \mathbf{u}^*) = 0.$$

When a MAV is in constant-altitude, wings-level steady flight, a subset of its states are in equilibrium. In particular, the altitude  $h = -p_d$ ; the body frame velocities  $u, v, w$ ; the Euler angles  $\phi, \theta, \psi$ ; and the angular rates  $p, q$ , and  $r$  are all constant. In the aerodynamics literature, an aircraft in equilibrium is said to be in trim. In general, trim conditions may include states that are not constant. For example, in steady-climb, wings-level flight,  $h$  is constant and  $h$  grows linearly. Also, in a constant turn  $\dot{\psi}$  is constant and

$\psi$  has linear growth. Therefore, in general, the conditions for trim are given by

$$\dot{\mathbf{x}}^* = f(\mathbf{x}^*, \mathbf{u}^*).$$

**MODIFIED MATERIAL:** In the process of performing trim calculations for the aircraft, we will assume that there is no wind, and therefore that  $V_a = V_g$ ,  $\psi = \chi$ , and  $\gamma = \gamma_a$ .

The most common scenario where trim values are needed are in wings-level flight. Therefore, the objective is to compute trim states and inputs when the aircraft simultaneously satisfies the following conditions:

- It is traveling at a constant speed  $V_a^*$ ,
- It is climbing at a constant flight path angle of  $\gamma^*$ .

The parameters  $V_a^*$  and  $\gamma^*$  are inputs to the trim calculations. For constant altitude flight, we have that  $\gamma^* = 0$ .

For fixed-wing aircraft, the states are given by

$$\mathbf{x} \triangleq (p_n, p_e, p_d, u, v, w, \phi, \theta, \psi, p, q, r)^\top \quad (5.17)$$

and the inputs are given by

$$\mathbf{u} \triangleq (\delta_e, \delta_a, \delta_r, \delta_t)^\top, \quad (5.18)$$

and  $f(\mathbf{x}, \mathbf{u})$  is specified by the right-hand side of equations (5.1)–(5.12). Note however, that the right-hand side of equations (5.1)–(5.12) are independent of  $p_n$ ,  $p_e$ ,  $p_d$ . Therefore, trimmed flight is independent of position. In addition, since only  $\dot{p}_n$  and  $\dot{p}_e$  are dependent on  $\psi$ , trimmed flight is also independent of the heading  $\psi$ .

In wings-level flight the speed of the aircraft is not changing, which implies that  $\dot{u}^* = \dot{v}^* = \dot{w}^* = 0$ . Similarly, since the roll and pitch angles will be constant, we have that  $\dot{\phi}^* = \dot{\theta}^* = \dot{p}^* = \dot{q}^* = \dot{r}^* = 0$ . Finally, the climb rate is constant, and is given by

$$\dot{h}^* = V_a^* \sin \gamma^*. \quad (5.19)$$

Therefore, given the parameters  $V_a^*$  and  $\gamma^*$ , it is possible to specify  $\dot{x}^*$ , as

$$\dot{\mathbf{x}}^* = \begin{pmatrix} \dot{p}_n^* \\ \dot{p}_e^* \\ \dot{h}^* \\ \dot{u}^* \\ \dot{v}^* \\ \dot{w}^* \\ \dot{\phi}^* \\ \dot{\theta}^* \\ \dot{\psi}^* \\ \dot{p}^* \\ \dot{q}^* \\ \dot{r}^* \end{pmatrix} = \begin{pmatrix} [\text{don't care}] \\ [\text{don't care}] \\ V_a^* \sin \gamma^* \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \quad (5.20)$$

The problem of finding  $\mathbf{x}^*$  (with the exception of  $p_n^*$ ,  $p_e^*$ ,  $h^*$  and  $\psi^*$ ) and  $\mathbf{u}^*$  such that  $\dot{\mathbf{x}}^* = f(\mathbf{x}^*, \mathbf{u}^*)$ , reduces to solving a nonlinear algebraic system of equations. There are numerous numerical techniques for solving this system of equations. Appendix F describes two methods for solving this system of equations.

## 5.4 TRANSFER FUNCTION MODELS

Transfer function models for the lateral dynamics are derived in section 5.4.1. These describe the motion of the aircraft in the horizontal plane. Transfer function models for the longitudinal dynamics that describe the motion of the aircraft in the vertical plane are derived in section 5.4.2.

### 5.4.1 Lateral Transfer Functions

For the lateral dynamics, the variables of interest are the roll angle  $\phi$ , the roll rate  $p$ , the heading angle  $\psi$ , and the yaw rate  $r$ . The control surfaces used to influence the lateral dynamics are the ailerons  $\delta_a$ , and the rudder  $\delta_r$ . The ailerons are primarily used to influence the roll rate  $p$ , while the rudder is primarily used to control the yaw  $\psi$  of the aircraft.

Our first task is to derive a transfer function from the ailerons  $\delta_a$  to the roll angle  $\phi$ . From equation (5.7), we have

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

Since in most flight conditions,  $\theta$  will be small, the primary influence on  $\dot{\phi}$  is the roll rate  $p$ . Defining

$$d_{\phi_1} \triangleq q \sin \phi \tan \theta + r \cos \phi \tan \theta$$

and considering  $d_{\phi_1}$  as a disturbance gives

$$\dot{\phi} = p + d_{\phi_1}. \quad (5.21)$$

Differentiating equation (5.21) and using equation (5.10) we get

$$\begin{aligned} \ddot{\phi} &= \ddot{p} + \dot{d}_{\phi_1} \\ &= \Gamma_1 pq - \Gamma_2 qr + \frac{1}{2} \rho V_a^2 Sb \left[ C_{p_0} + C_{p_\beta} \beta + C_{p_p} \frac{bp}{2V_a} \right. \\ &\quad \left. + C_{p_r} \frac{br}{2V_a} + C_{p_{\delta_a}} \delta_a + C_{p_{\delta_r}} \delta_r \right] - \Gamma_3 Q_p + \dot{d}_{\phi_1} \\ &= \Gamma_1 pq - \Gamma_2 qr + \frac{1}{2} \rho V_a^2 Sb \left[ C_{p_0} + C_{p_\beta} \beta + C_{p_p} \frac{b}{2V_a} (\dot{\phi} - d_{\phi_1}) \right. \\ &\quad \left. + C_{p_r} \frac{br}{2V_a} + C_{p_{\delta_a}} \delta_a + C_{p_{\delta_r}} \delta_r \right] - \Gamma_3 Q_p + \dot{d}_{\phi_1} \\ &= \left( \frac{1}{2} \rho V_a^2 Sb C_{p_p} \frac{b}{2V_a} \right) \dot{\phi} + \left( \frac{1}{2} \rho V_a^2 Sb C_{p_{\delta_a}} \right) \delta_a \\ &\quad + \left\{ \Gamma_1 pq - \Gamma_2 qr + \frac{1}{2} \rho V_a^2 Sb \left[ C_{p_0} + C_{p_\beta} \beta - C_{p_p} \frac{b}{2V_a} (d_{\phi_1}) \right. \right. \\ &\quad \left. \left. + C_{p_r} \frac{br}{2V_a} + C_{p_{\delta_r}} \delta_r \right] - \Gamma_3 Q_p + \dot{d}_{\phi_1} \right\} \\ &= -a_{\phi_1} \dot{\phi} + a_{\phi_2} \delta_a + d_{\phi_2}, \end{aligned}$$

where

$$a_{\phi_1} \triangleq -\frac{1}{2} \rho V_a^2 Sb C_{p_p} \frac{b}{2V_a} \quad (5.22)$$

$$a_{\phi_2} \triangleq \frac{1}{2} \rho V_a^2 Sb C_{p_{\delta_a}} \quad (5.23)$$

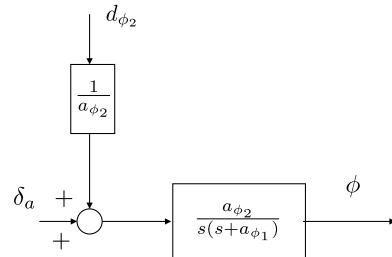
$$\begin{aligned} d_{\phi_2} &\triangleq \Gamma_1 pq - \Gamma_2 qr + \frac{1}{2} \rho V_a^2 Sb \left[ C_{p_0} + C_{p_\beta} \beta - C_{p_p} \frac{b}{2V_a} (d_{\phi_1}) \right. \\ &\quad \left. + C_{p_r} \frac{br}{2V_a} + C_{p_{\delta_r}} \delta_r \right] - \Gamma_3 Q_p + \dot{d}_{\phi_1}, \end{aligned}$$

where  $d_{\phi_2}$  is considered a disturbance on the system.

In the Laplace domain, we have

$$\phi(s) = \left( \frac{a_{\phi_2}}{s(s + a_{\phi_1})} \right) \left( \delta_a(s) + \frac{1}{a_{\phi_2}} d_{\phi_2}(s) \right). \quad (5.24)$$

A block diagram is shown in figure 5.2, where the inputs to the block diagram are the ailerons  $\delta_a$  and the disturbance  $d_{\phi_2}$ .



**Figure 5.2:** Block diagram for roll dynamics. The inputs are the ailerons  $\delta_a$  and the disturbance  $d_{\phi_2}$ .

We can also derive a transfer function from the roll angle  $\phi$  to the course angle  $\chi$ . From [equation \(5.15\)](#), we have that in a coordinated turn

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

This equation can be rewritten as

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

where

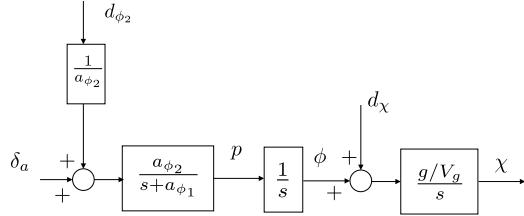
$$d_\chi = \tan \phi - \phi$$

is a disturbance. In the Laplace domain, we have

$$\chi(s) = \frac{g/V_g}{s} (\phi(s) + d_\chi(s)). \quad (5.25)$$

This leads to the block diagram for the lateral dynamics controlled by the aileron shown in [figure 5.3](#). To implement this transfer function, we need a value for the ground speed  $V_g$ . Since we have assumed zero wind, and we can further assume that the aircraft will track its commanded airspeed, we can use the commanded airspeed as the value for  $V_g$ . In [chapter 6](#), we will design control laws to control the flight path of the aircraft relative to the ground. This, combined with the fact that course measurements are readily available from GPS, has lead us to express the transfer function in [equation \(5.25\)](#) in terms of the course angle  $\chi$ . Again drawing on the assumption of  $V_w = 0$ , this transfer function could alternatively be expressed as

$$\psi(s) = \frac{g/V_a}{s} (\phi(s) + d_\chi(s)).$$



**Figure 5.3:** Block diagram for lateral dynamics. The roll rate  $p$  is shown explicitly because it can be obtained directly from the rate gyros and will be used as a feed-back signal in [chapter 6](#).

A second component of the lateral dynamics is the yaw behavior in response to rudder inputs. In the absence of wind,  $v = V_a \sin \beta$ . For a constant airspeed, this results in  $\dot{v} = (V_a \cos \beta) \dot{\beta}$ . Therefore, from [equation \(5.5\)](#), we have

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

With the reasonable assumption that  $\beta$  is small, this leads to  $\cos \beta \approx 1$  and

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

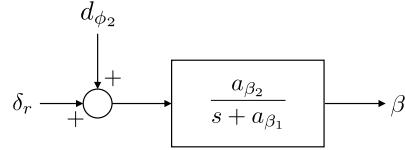
where

$$\begin{aligned} a_{\beta_1} &= -\frac{\rho V_a S}{2m} C_{Y_\beta} \\ a_{\beta_2} &= \frac{\rho V_a S}{2m} C_{Y_{\delta_r}} \\ d_\beta &= \frac{1}{V_a} (pw - ru + g \cos \theta \sin \phi) \\ &\quad + \frac{\rho V_a S}{2m} \left[ C_{Y_0} + C_{Y_p} \frac{bp}{2V_a} + C_{Y_r} \frac{br}{2V_a} + C_{Y_{\delta_a}} \delta_a \right]. \end{aligned}$$

In the Laplace domain, we have

$$\beta(s) = \frac{a_{\beta_2}}{s + a_{\beta_1}} (\delta_r(s) + d_\beta(s)). \quad (5.26)$$

This transfer function is depicted in block diagram form in [figure 5.4](#).



**Figure 5.4:** Block diagram for the rudder-sideslip dynamics.

#### 5.4.2 Longitudinal Transfer Functions

In this section we will derive transfer function models for the longitudinal dynamics. The variables of interest are the pitch angle  $\theta$ , the pitch rate  $q$ , the altitude  $h = -p_d$ , and the airspeed  $V_a$ . The control signals used to influence the longitudinal dynamics are the elevator  $\delta_e$  and the throttle  $\delta_t$ . The elevator will be used to directly influence the pitch angle  $\theta$ . As we will show below, the pitch angle can be used to manipulate both the altitude  $h$  and the airspeed  $V_a$ . The airspeed can be used to manipulate the altitude, and the throttle is used to influence the airspeed. The transfer functions derived in this section will be used in [chapter 6](#) to design an altitude control strategy.

We begin by deriving a simplified relationship between the elevator  $\delta_e$  and the pitch angle  $\theta$ . From [equation \(5.8\)](#), we have

$$\begin{aligned}\dot{\theta} &= q \cos \phi - r \sin \phi \\ &= q + q(\cos \phi - 1) - r \sin \phi \\ &\triangleq q + d_{\theta_1},\end{aligned}$$

where  $d_{\theta_1} \triangleq q(\cos \phi - 1) - r \sin \phi$  and where  $d_{\theta_1}$  is small for small roll angles  $\phi$ . Differentiating, we get

$$\ddot{\theta} = \dot{q} + \dot{d}_{\theta_1}.$$

Using [equation \(5.11\)](#) and the relationship  $\theta = \alpha + \gamma_a$ , where  $\gamma_a$  is the air-referenced flight path angle, we get

$$\begin{aligned}\ddot{\theta} &= \Gamma_6(r^2 - p^2) + \Gamma_5 pr + \frac{\rho V_a^2 c S}{2 J_y} \left[ C_{m_0} + C_{m_\alpha} \alpha + C_{m_q} \frac{cq}{2 V_a} + C_{m_{\delta_e}} \delta_e \right] + \dot{d}_{\theta_1} \\ &= \Gamma_6(r^2 - p^2) + \Gamma_5 pr + \frac{\rho V_a^2 c S}{2 J_y} \left[ C_{m_0} + C_{m_\alpha} (\theta - \gamma_a) + C_{m_q} \frac{c}{2 V_a} (\dot{\theta} - d_{\theta_1}) + C_{m_{\delta_e}} \delta_e \right] + \dot{d}_{\theta_1} \\ &= \left( \frac{\rho V_a^2 c S}{2 J_y} C_{m_q} \frac{c}{2 V_a} \right) \dot{\theta} + \left( \frac{\rho V_a^2 c S}{2 J_y} C_{m_\alpha} \right) \theta + \left( \frac{\rho V_a^2 c S}{2 J_y} C_{m_{\delta_e}} \right) \delta_e\end{aligned}$$

$$\begin{aligned}
& + \left\{ \Gamma_6(r^2 - p^2) + \Gamma_5 pr + \frac{\rho V_a^2 c S}{J_y} \left[ C_{m_0} - C_{m_\alpha} \gamma_a \right. \right. \\
& \quad \left. \left. - C_{m_q} \frac{c}{2V_a} d_{\theta_1} \right] + \dot{d}_{\theta_1} \right\} \\
& = -a_{\theta_1} \dot{\theta} - a_{\theta_2} \theta + a_{\theta_3} \delta_e + d_{\theta_2},
\end{aligned}$$

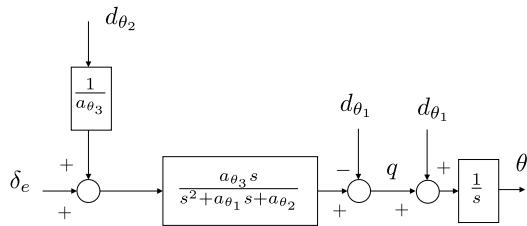
where

$$\begin{aligned}
a_{\theta_1} &\triangleq -\frac{\rho V_a^2 c S}{2J_y} C_{m_q} \frac{c}{2V_a} \\
a_{\theta_2} &\triangleq -\frac{\rho V_a^2 c S}{2J_y} C_{m_\alpha} \\
a_{\theta_3} &\triangleq \frac{\rho V_a^2 c S}{2J_y} C_{m_{\delta_e}} \\
d_{\theta_2} &\triangleq \Gamma_6(r^2 - p^2) + \Gamma_5 pr + \frac{\rho V_a^2 c S}{2J_y} \left[ C_{m_0} - C_{m_\alpha} \gamma_a \right. \\
& \quad \left. - C_{m_q} \frac{c}{2V_a} d_{\theta_1} \right] + \dot{d}_{\theta_1}.
\end{aligned}$$

We have derived a linear model for the evolution of the pitch angle. Taking the Laplace transform, we have

$$\theta(s) = \left( \frac{a_{\theta_3}}{s^2 + a_{\theta_1}s + a_{\theta_2}} \right) \left( \delta_e(s) + \frac{1}{a_{\theta_3}} d_{\theta_2}(s) \right). \quad (5.27)$$

Note that in straight and level flight,  $r = p = \phi = \gamma_a = 0$ . In addition, airframes are usually designed so that  $C_{m_0} = 0$ , which implies that  $d_{\theta_2} = 0$ . Using the fact that  $\dot{\theta} = q + d_{\theta_1}$ , we get the block diagram shown in figure 5.5. The model shown in figure 5.5 is useful because the pitch rate  $q$  is directly available from the rate gyros for feedback and therefore needs to be accessible in the model.



**Figure 5.5:** Block diagram for the transfer function from the elevator to the pitch angle. The pitch rate  $q$  is shown explicitly because it is available from the rate gyros and will be used as a feedback signal in chapter 6.

For a constant airspeed, the pitch angle directly influences the climb rate of the aircraft. Therefore, we can develop a transfer function from pitch angle to altitude. From [equation \(5.3\)](#), we have

$$\begin{aligned}\dot{h} &= u \sin \theta - v \sin \phi \cos \theta - w \cos \phi \cos \theta \\ &= V_a \theta + (u \sin \theta - V_a \theta) - v \sin \phi \cos \theta - w \cos \phi \cos \theta \\ &= V_a \theta + d_h,\end{aligned}\tag{5.28}$$

where

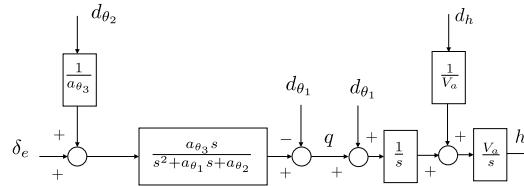
$$d_h \triangleq (u \sin \theta - V_a \theta) - v \sin \phi \cos \theta - w \cos \phi \cos \theta.$$

Note that in straight and level flight, where  $v = 0$ ,  $w \approx 0$ ,  $u \approx V_a$ ,  $\phi = 0$  and  $\sin \theta \approx \theta$ , we have  $d_h \approx 0$ .

If we assume that  $V_a$  is constant and that  $\theta$  is the input, then in the Laplace domain [equation \(5.28\)](#) becomes

$$h(s) = \frac{V_a}{s} \left( \theta + \frac{1}{V_a} d_h \right),\tag{5.29}$$

and the resulting block diagram for the longitudinal dynamics from the elevator to the altitude is shown in [figure 5.6](#). Alternatively, if the pitch angle is



**Figure 5.6:** Block diagram for longitudinal dynamics.

held constant, then increasing the airspeed will result in increased lift over the wings, resulting in a change in altitude. To derive the transfer function from airspeed to altitude, hold  $\theta$  constant in [equation \(5.28\)](#) and consider  $V_a$  as the input to obtain

$$h(s) = \frac{\theta}{s} \left( V_a + \frac{1}{\theta} d_h \right).\tag{5.30}$$

The altitude controllers discussed in [chapter 6](#) will regulate altitude using both pitch angle and airspeed. Similarly, the airspeed will be regulated using both the throttle setting and the pitch angle. For example, when the pitch angle is constant, increasing the throttle will increase the thrust, which increases the airspeed of the vehicle. On the other hand, if the throttle is

held constant, then pitching the nose down will decrease the lift causing the aircraft to accelerate downward under the influence of gravity, thereby increasing its airspeed.

**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 (2.6), we get

$$\begin{aligned}\dot{V}_a &= \dot{u} \cos \alpha \cos \beta + \dot{v} \sin \beta + \dot{w} \sin \alpha \cos \beta \\ &= \dot{u} \cos \alpha + \dot{w} \sin \alpha + d_{V_1},\end{aligned}\tag{5.31}$$

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 (5.4) and (5.6) in equation (5.31), 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 (2.6) and the linear approximation  $C_D(\alpha) \approx C_{D_0} + C_{D_\alpha} \alpha$ , and simplifying, we get

$$\dot{V}_a = rV_a \cos \alpha \sin \beta - pV_a \sin \alpha \sin \beta$$

$$\begin{aligned}
& -g \cos \alpha \sin \theta + g \sin \alpha \cos \theta \cos \phi \\
& + \frac{\rho V_a^2 S}{2m} \left[ -C_{D_0} - C_{D_\alpha} \alpha - C_{D_q} \frac{cq}{2V_a} - C_{D_{\delta_e}} \delta_e \right] \\
& + \frac{1}{m} T_p(\delta_t, V_a) \cos \alpha + d_{V_1} \\
= & (rV_a \cos \alpha - pV_a \sin \alpha) \sin \beta \\
& - g \sin(\theta - \alpha) - g \sin \alpha \cos \theta (1 - \cos \phi) \\
& + \frac{\rho V_a^2 S}{2m} \left[ -C_{D_0} - C_{D_\alpha} \alpha - C_{D_q} \frac{cq}{2V_a} - C_{D_{\delta_e}} \delta_e \right] \\
& + \frac{1}{m} T_p(\delta_t, V_a) \cos \alpha + d_{V_1} \\
= & -g \sin \gamma + \frac{\rho V_a^2 S}{2m} \left[ -C_{D_0} - C_{D_\alpha} \alpha - C_{D_q} \frac{cq}{2V_a} - C_{D_{\delta_e}} \delta_e \right] \\
& + \frac{1}{m} T_p(\delta_t, V_a) + d_{V_2}, \tag{5.32}
\end{aligned}$$

where

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

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  $\delta_t$  and the pitch angle  $\theta$ . Since equation (5.32) is nonlinear in  $V_a$  and  $\delta_t$ , we must first linearize before we can find the desired transfer functions. Following the approach outlined in section 5.5.1, we can linearize equation (5.32) by letting  $\bar{V}_a \triangleq V_a - V_a^*$  be the deviation of  $V_a$  from trim,  $\bar{\theta} \triangleq \theta - \theta^*$  be the deviation of  $\theta$  from trim, and  $\bar{\delta}_t \triangleq \delta_t - \delta_t^*$  be the deviation of the throttle from trim. Equation (5.32) can then be linearized around the wings-level, constant-altitude ( $\gamma^* = 0$ ) trim condition to give

$$\dot{\bar{V}}_a = -g \cos(\theta^* - \alpha^*) \bar{\theta} \tag{5.33}$$

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

$$= -a_{V_1} \bar{V}_a + a_{V_2} \bar{\delta}_t - a_{V_3} \bar{\theta} + d_V, \tag{5.34}$$

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

$$\bar{V}_a(s) = \frac{1}{s + a_{V_1}} (a_{V_2} \bar{\delta}_t(s) - a_{V_3} \bar{\theta}(s) + d_V(s)). \quad (5.35)$$

## 5.5 LINEAR STATE-SPACE MODELS

In this section we will derive linear state-space models for both longitudinal and lateral motion by linearizing equations (5.1)–(5.12) about trim conditions. Section 5.5.1 discusses general linearization techniques. Section 5.5.2 derives the state-space equations for the lateral dynamics, and section 5.5.3 derives the state-space equations for the longitudinal dynamics. Finally, section 5.5.4 describes the reduced-order modes including the short-period mode, the phugoid mode, the dutch-roll mode, and the spiral-divergence mode.

### 5.5.1 Linearization

Given the general nonlinear system of equations

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

where  $x \in \mathbb{R}^n$  is the state and  $u \in \mathbb{R}^m$  is the control vector, and suppose that, using the techniques discussed in section 5.3, a trim input  $u^*$  and state  $x^*$  can be found such that

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

Letting  $\bar{x} \triangleq x - x^*$ , we get

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

Taking the Taylor series expansion of the first term about the trim state, we get

$$\begin{aligned}\dot{x} &= f(x^*, u^*) + \frac{\partial f(x^*, u^*)}{\partial x} \bar{x} + \frac{\partial f(x^*, u^*)}{\partial u} \bar{u} + H.O.T - f(x^*, u^*) \\ &\approx \frac{\partial f(x^*, u^*)}{\partial x} \bar{x} + \frac{\partial f(x^*, u^*)}{\partial u} \bar{u}.\end{aligned}\quad (5.36)$$

Therefore, the linearized dynamics are determined by finding  $\frac{\partial f}{\partial x}$  and  $\frac{\partial f}{\partial u}$ , evaluated at the trim conditions.

### 5.5.2 Lateral-Directional State-Space Equations

For the lateral state-space equations, the state is given by

$$\dot{x}_{\text{lat}} \triangleq (v, p, r, \phi, \psi)^\top,$$

and the input vector is defined as

$$u_{\text{lat}} \triangleq (\delta_a, \delta_r)^\top.$$

Expressing [equations](#) (5.5), (5.10), (5.12), (5.7), and (5.9) in terms of  $x_{\text{lat}}$  and  $u_{\text{lat}}$ , we get

$$\begin{aligned}\dot{v} &= pw - ru + g \cos \theta \sin \phi + \frac{\rho \sqrt{u^2 + v^2 + w^2} S}{2m} \frac{b}{2} [C_{Y_p} p + C_{Y_r} r] \\ &\quad + \frac{\rho(u^2 + v^2 + w^2) S}{2m} [C_{Y_0} + C_{Y_\beta} \tan^{-1} \left( \frac{v}{\sqrt{u^2 + w^2}} \right) \\ &\quad + C_{Y_{\delta_a}} \delta_a + C_{Y_{\delta_r}} \delta_r]\end{aligned}\quad (5.37)$$

$$\begin{aligned}\dot{p} &= \Gamma_1 pq - \Gamma_2 qr + \frac{\rho \sqrt{u^2 + v^2 + w^2} S}{2} \frac{b^2}{2} [C_{p_p} p + C_{p_r} r] \\ &\quad + \frac{1}{2} \rho(u^2 + v^2 + w^2) S b [C_{p_0} + C_{p_\beta} \tan^{-1} \left( \frac{v}{\sqrt{u^2 + w^2}} \right) \\ &\quad + C_{p_{\delta_a}} \delta_a + C_{p_{\delta_r}} \delta_r] - \Gamma_3 Q_p\end{aligned}\quad (5.38)$$

$$\begin{aligned}\dot{r} &= \Gamma_7 pq - \Gamma_1 qr + \frac{\rho \sqrt{u^2 + v^2 + w^2} S}{2} \frac{b^2}{2} [C_{r_p} p + C_{r_r} r] \\ &\quad + \frac{1}{2} \rho(u^2 + v^2 + w^2) S b [C_{r_0} + C_{r_\beta} \tan^{-1} \left( \frac{v}{\sqrt{u^2 + w^2}} \right) \\ &\quad + C_{r_{\delta_a}} \delta_a + C_{r_{\delta_r}} \delta_r] - \Gamma_3 Q_p\end{aligned}\quad (5.39)$$

$$\dot{\phi} = p + q \sin \phi \tan \theta + r \cos \phi \tan \theta \quad (5.40)$$

$$\dot{\psi} = q \sin \phi \sec \theta + r \cos \phi \sec \theta, \quad (5.41)$$

where we have utilized the zero-wind expressions

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

The Jacobians of equations (5.37)–(5.41) are given by

$$\begin{aligned}\frac{\partial f_{\text{lat}}}{\partial x_{\text{lat}}} &= \begin{pmatrix} \frac{\partial \dot{v}}{\partial v} & \frac{\partial \dot{v}}{\partial p} & \frac{\partial \dot{v}}{\partial r} & \frac{\partial \dot{v}}{\partial \phi} & \frac{\partial \dot{v}}{\partial \psi} \\ \frac{\partial \dot{p}}{\partial v} & \frac{\partial \dot{p}}{\partial p} & \frac{\partial \dot{p}}{\partial r} & \frac{\partial \dot{p}}{\partial \phi} & \frac{\partial \dot{p}}{\partial \psi} \\ \frac{\partial \dot{r}}{\partial v} & \frac{\partial \dot{r}}{\partial p} & \frac{\partial \dot{r}}{\partial r} & \frac{\partial \dot{r}}{\partial \phi} & \frac{\partial \dot{r}}{\partial \psi} \\ \frac{\partial \dot{\phi}}{\partial v} & \frac{\partial \dot{\phi}}{\partial p} & \frac{\partial \dot{\phi}}{\partial r} & \frac{\partial \dot{\phi}}{\partial \phi} & \frac{\partial \dot{\phi}}{\partial \psi} \\ \frac{\partial \dot{\psi}}{\partial v} & \frac{\partial \dot{\psi}}{\partial p} & \frac{\partial \dot{\psi}}{\partial r} & \frac{\partial \dot{\psi}}{\partial \phi} & \frac{\partial \dot{\psi}}{\partial \psi} \end{pmatrix} \\ \frac{\partial f_{\text{lat}}}{\partial u_{\text{lat}}} &= \begin{pmatrix} \frac{\partial \dot{v}}{\partial \delta_a} & \frac{\partial \dot{v}}{\partial \delta_r} \\ \frac{\partial \dot{p}}{\partial \delta_a} & \frac{\partial \dot{p}}{\partial \delta_r} \\ \frac{\partial \dot{r}}{\partial \delta_a} & \frac{\partial \dot{r}}{\partial \delta_r} \\ \frac{\partial \dot{\phi}}{\partial \delta_a} & \frac{\partial \dot{\phi}}{\partial \delta_r} \\ \frac{\partial \dot{\psi}}{\partial \delta_a} & \frac{\partial \dot{\psi}}{\partial \delta_r} \end{pmatrix}.\end{aligned}$$

Toward that end, note that

$$\frac{\partial}{\partial v} \tan^{-1} \left( \frac{v}{\sqrt{u^2 + w^2}} \right) = \frac{\sqrt{u^2 + w^2}}{u^2 + v^2 + w^2} = \frac{\sqrt{u^2 + w^2}}{V_a^2}.$$

Working out the derivatives, we get that the linearized state-space equations are

$$\begin{pmatrix} \dot{\bar{v}} \\ \dot{\bar{p}} \\ \dot{\bar{r}} \\ \dot{\bar{\phi}} \\ \dot{\bar{\psi}} \end{pmatrix} = \begin{pmatrix} Y_v & Y_p & Y_r & A_{14} & 0 \\ L_v & L_p & L_r & 0 & 0 \\ N_v & N_p & N_r & 0 & 0 \\ 0 & 1 & A_{43} & A_{44} & 0 \\ 0 & 0 & A_{53} & A_{54} & 0 \end{pmatrix} \begin{pmatrix} \bar{v} \\ \bar{p} \\ \bar{r} \\ \bar{\phi} \\ \bar{\psi} \end{pmatrix} + \begin{pmatrix} Y_{\delta_a} & Y_{\delta_r} \\ L_{\delta_a} & L_{\delta_r} \\ N_{\delta_a} & N_{\delta_r} \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \bar{\delta}_a \\ \bar{\delta}_r \end{pmatrix}, \quad (5.42)$$

where the coefficients are given in table 5.1.

The lateral equations are often given in terms of  $\bar{\beta}$  instead of  $\bar{v}$ . From equation (2.6), we have

$$v = V_a \sin \beta.$$

Linearizing around  $\beta = \beta^*$ , we get

$$\bar{v} = V_a^* \cos \beta^* \bar{\beta},$$

**Table 5.1:** Lateral state-space model coefficients

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

which implies that

$$\dot{\bar{\beta}} = \frac{1}{V_a^* \cos \beta^*} \dot{\bar{v}}.$$

Therefore, we can write the state-space equations in terms of  $\bar{\beta}$  instead of  $\bar{v}$  as

$$\begin{aligned} \begin{pmatrix} \dot{\bar{\beta}} \\ \dot{\bar{p}} \\ \dot{\bar{r}} \\ \dot{\bar{\phi}} \\ \dot{\bar{\psi}} \end{pmatrix} &= \begin{pmatrix} Y_v & \frac{Y_p}{V_a^* \cos \beta^*} & \frac{Y_r}{V_a^* \cos \beta^*} & \frac{A_{14}}{V_a^* \cos \beta^*} & 0 \\ L_v V_a^* \cos \beta^* & L_p & L_r & 0 & 0 \\ N_v V_a^* \cos \beta^* & N_p & N_r & 0 & 0 \\ 0 & 1 & A_{43} & A_{44} & 0 \\ 0 & 0 & A_{53} & A_{54} & 0 \end{pmatrix} \begin{pmatrix} \bar{\beta} \\ \bar{p} \\ \bar{r} \\ \bar{\phi} \\ \bar{\psi} \end{pmatrix} \\ &\quad + \begin{pmatrix} \frac{Y_{\delta_a}}{V_a^* \cos \beta^*} & \frac{Y_{\delta_r}}{V_a^* \cos \beta^*} \\ L_{\delta_a} & L_{\delta_r} \\ N_{\delta_a} & N_{\delta_r} \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \bar{\delta}_a \\ \bar{\delta}_r \end{pmatrix}. \end{aligned} \tag{5.43}$$

### 5.5.3 Longitudinal State-Space Equations

For the longitudinal state-space equations, the state is given by

$$\dot{x}_{\text{lon}} \triangleq (u, w, q, \theta, h)^\top,$$

and the input vector is defined as

$$u_{\text{lon}} \triangleq (\delta_e, \delta_t)^\top.$$

Expressing [equations](#) (5.4), (5.6), (5.11), (5.8), and (5.3) in terms of  $x_{\text{lon}}$  and  $u_{\text{lon}}$ , we get

$$\begin{aligned} \dot{u} &= rv - qw - g \sin \theta \\ &\quad + \frac{\rho V_a^2 S}{2m} \left[ C_{X_0} + C_{X_\alpha} \alpha + C_{X_q} \frac{cq}{2V_a} + C_{X_{\delta_e}} \delta_e \right] + T_p(\delta_t, V_a) \\ \dot{w} &= qu - pv + g \cos \theta \cos \phi \\ &\quad + \frac{\rho V_a^2 S}{2m} \left[ C_{Z_0} + C_{Z_\alpha} \alpha + C_{Z_q} \frac{cq}{2V_a} + C_{Z_{\delta_e}} \delta_e \right] \\ \dot{q} &= \frac{J_{xz}}{J_y} (r^2 - p^2) + \frac{J_z - J_x}{J_y} pr \\ &\quad + \frac{1}{2J_y} \rho V_a^2 c S \left[ C_{m_0} + C_{m_\alpha} \alpha + C_{m_q} \frac{cq}{2V_a} + C_{m_{\delta_e}} \delta_e \right] \end{aligned}$$

$$\begin{aligned}\dot{\theta} &= q \cos \phi - r \sin \phi \\ \dot{h} &= u \sin \theta - v \sin \phi \cos \theta - w \cos \phi \cos \theta.\end{aligned}$$

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

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

from equation (2.6) gives

$$\begin{aligned}\dot{u} &= -qw - g \sin \theta + \frac{\rho(u^2 + w^2)S}{2m} \left[ C_{X_0} + C_{X_\alpha} \tan^{-1} \left( \frac{w}{u} \right) + C_{X_{\delta_e}} \delta_e \right] \\ &\quad + \frac{\rho \sqrt{u^2 + w^2} S}{4m} C_{X_q} cq + T_p(\delta_t, \sqrt{u^2 + w^2})\end{aligned}\quad (5.44)$$

$$\begin{aligned}\dot{w} &= qu + g \cos \theta + \frac{\rho(u^2 + w^2)S}{2m} \left[ C_{Z_0} + C_{Z_\alpha} \tan^{-1} \left( \frac{w}{u} \right) + C_{Z_{\delta_e}} \delta_e \right] \\ &\quad + \frac{\rho \sqrt{u^2 + w^2} S}{4m} C_{Z_q} cq\end{aligned}\quad (5.45)$$

$$\begin{aligned}\dot{q} &= \frac{1}{2J_y} \rho(u^2 + w^2) c S \left[ C_{m_0} + C_{m_\alpha} \tan^{-1} \left( \frac{w}{u} \right) + C_{m_{\delta_e}} \delta_e \right] \\ &\quad + \frac{1}{4J_y} \rho \sqrt{u^2 + w^2} S C_{m_q} c^2 q\end{aligned}\quad (5.46)$$

$$\dot{\theta} = q \quad (5.47)$$

$$\dot{h} = u \sin \theta - w \cos \theta. \quad (5.48)$$

The Jacobians of equations (5.44)–(5.48) are given by

$$\begin{aligned}\frac{\partial f_{\text{lon}}}{\partial x_{\text{lon}}} &= \begin{pmatrix} \frac{\partial \dot{u}}{\partial u} & \frac{\partial \dot{u}}{\partial w} & \frac{\partial \dot{u}}{\partial q} & \frac{\partial \dot{u}}{\partial \theta} & \frac{\partial \dot{u}}{\partial h} \\ \frac{\partial \dot{w}}{\partial u} & \frac{\partial \dot{w}}{\partial w} & \frac{\partial \dot{w}}{\partial q} & \frac{\partial \dot{w}}{\partial \theta} & \frac{\partial \dot{w}}{\partial h} \\ \frac{\partial \dot{q}}{\partial u} & \frac{\partial \dot{q}}{\partial w} & \frac{\partial \dot{q}}{\partial q} & \frac{\partial \dot{q}}{\partial \theta} & \frac{\partial \dot{q}}{\partial h} \\ \frac{\partial \dot{\theta}}{\partial u} & \frac{\partial \dot{\theta}}{\partial w} & \frac{\partial \dot{\theta}}{\partial q} & \frac{\partial \dot{\theta}}{\partial \theta} & \frac{\partial \dot{\theta}}{\partial h} \\ \frac{\partial \dot{h}}{\partial u} & \frac{\partial \dot{h}}{\partial w} & \frac{\partial \dot{h}}{\partial q} & \frac{\partial \dot{h}}{\partial \theta} & \frac{\partial \dot{h}}{\partial h} \end{pmatrix} \\ \frac{\partial f_{\text{lon}}}{\partial u_{\text{lon}}} &= \begin{pmatrix} \frac{\partial \dot{u}}{\partial \delta_e} & \frac{\partial \dot{u}}{\partial \delta_t} \\ \frac{\partial \dot{w}}{\partial \delta_e} & \frac{\partial \dot{w}}{\partial \delta_t} \\ \frac{\partial \dot{q}}{\partial \delta_e} & \frac{\partial \dot{q}}{\partial \delta_t} \\ \frac{\partial \dot{\theta}}{\partial \delta_e} & \frac{\partial \dot{\theta}}{\partial \delta_t} \\ \frac{\partial \dot{h}}{\partial \delta_e} & \frac{\partial \dot{h}}{\partial \delta_t} \end{pmatrix}.\end{aligned}$$

Note that

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

where we have used [equation](#) (2.7) and the fact that  $v = 0$ . Calculating the derivatives, we get the linearized state-space equations

$$\begin{aligned}\begin{pmatrix} \dot{\bar{u}} \\ \dot{\bar{w}} \\ \dot{\bar{q}} \\ \dot{\bar{\theta}} \\ \dot{\bar{h}} \end{pmatrix} &= \begin{pmatrix} X_u & X_w & X_q & -g \cos \theta^* & 0 \\ Z_u & Z_w & Z_q & -g \sin \theta^* & 0 \\ M_u & M_w & M_q & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \sin \theta^* & -\cos \theta^* & 0 & u^* \cos \theta^* + w^* \sin \theta^* & 0 \end{pmatrix} \begin{pmatrix} \bar{u} \\ \bar{w} \\ \bar{q} \\ \bar{\theta} \\ \bar{h} \end{pmatrix} \\ &\quad + \begin{pmatrix} X_{\delta_e} & X_{\delta_t} \\ Z_{\delta_e} & 0 \\ M_{\delta_e} & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \bar{\delta}_e \\ \bar{\delta}_t \end{pmatrix},\end{aligned}\tag{5.49}$$

where the coefficients are given in [table](#) 5.2.

The longitudinal equations are often given in terms of  $\bar{\alpha}$  instead of  $\bar{w}$ . From [equation](#) (2.6), we have

$$w = V_a \sin \alpha \cos \beta = V_a \sin \alpha,$$

where we have set  $\beta = 0$ . Linearizing around  $\alpha = \alpha^*$ , we get

$$\bar{w} = V_a^* \cos \alpha^* \bar{\alpha},$$

which implies that

$$\dot{\bar{\alpha}} = \frac{1}{V_a^* \cos \alpha^*} \dot{\bar{w}}.$$

We can, therefore, write the the state-space equations in terms of  $\bar{\alpha}$  instead

**Table 5.2:** Longitudinal state-space model coefficients

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

of  $\bar{w}$  as

$$\begin{pmatrix} \dot{\bar{u}} \\ \dot{\bar{\alpha}} \\ \dot{\bar{q}} \\ \dot{\bar{\theta}} \\ \dot{\bar{h}} \end{pmatrix} = \begin{pmatrix} X_u & X_w V_a^* c_{\alpha^*} & X_q & -g c_{\theta^*} & 0 \\ \frac{Z_u}{V_a^* c_{\alpha^*}} & Z_w & \frac{Z_q}{V_a^* c_{\alpha^*}} & \frac{-g s_{\theta^*}}{V_a^* c_{\alpha^*}} & 0 \\ M_u & M_w V_a^* c_{\alpha^*} & M_q & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ s_{\theta^*} & -V_a^* c_{\theta^*} c_{\alpha^*} & 0 & u^* c_{\theta^*} + w^* s_{\theta^*} & 0 \end{pmatrix} \begin{pmatrix} \bar{u} \\ \bar{\alpha} \\ \bar{q} \\ \bar{\theta} \\ \bar{h} \end{pmatrix} + \begin{pmatrix} X_{\delta_e} & X_{\delta_t} \\ \frac{Z_{\delta_e}}{V_a^* c_{\alpha^*}} & 0 \\ M_{\delta_e} & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \bar{\delta}_e \\ \bar{\delta}_t \end{pmatrix}. \quad (5.50)$$

#### 5.5.4 Reduced-order Modes

The traditional literature on aircraft dynamics and control define several open-loop aircraft dynamic modes. These include the short-period mode, the phugoid mode, the rolling mode, the spiral-divergence mode, and the dutch-roll mode. In this section we will briefly describe each of these modes and show how to approximate the eigenvalues associated with these modes.

##### *Short-period Mode*

If we assume a constant altitude and a constant thrust input, then we can simplify the longitudinal state-space model in [equation \(5.50\)](#) to

$$\begin{pmatrix} \dot{\bar{u}} \\ \dot{\bar{\alpha}} \\ \dot{\bar{q}} \\ \dot{\bar{\theta}} \end{pmatrix} = \begin{pmatrix} X_u & X_w V_a^* \cos \alpha^* & X_q & -g \cos \theta^* \\ \frac{Z_u}{V_a^* \cos \alpha^*} & Z_w & \frac{Z_q}{V_a^* \cos \alpha^*} & \frac{-g \sin \theta^*}{V_a^* \cos \alpha^*} \\ M_u & M_w V_a^* \cos \alpha & M_q & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \bar{u} \\ \bar{\alpha} \\ \bar{q} \\ \bar{\theta} \end{pmatrix} + \begin{pmatrix} X_{\delta_e} \\ \frac{Z_{\delta_e}}{V_a^* \cos \alpha^*} \\ M_{\delta_e} \\ 0 \\ 0 \end{pmatrix} \bar{\delta}_e. \quad (5.51)$$

If we compute the eigenvalues of the  $A$  matrix, we will find that there is one fast, damped mode and one slow, lightly damped mode. The fast mode is called the *short period* mode. The slow, lightly damped mode is called the *phugoid* mode.

For the short-period mode, we will assume that  $u$  is constant (i.e.,  $\bar{u} = \dot{\bar{u}} = 0$ ). It follows that the state-space equations in [equation \(5.51\)](#) can be

written as

$$\begin{aligned}\dot{\bar{\alpha}} &= Z_w \bar{\alpha} + \frac{Z_q}{V_a^* \cos \alpha^*} \dot{\bar{\theta}} - \frac{g \sin \theta^*}{V_a^* \cos \alpha^*} \bar{\theta} + \frac{Z_{\delta_e}}{V_a^* \cos \alpha^*} \bar{\delta}_e \\ \ddot{\bar{\theta}} &= M_w V_a^* \cos \alpha^* \bar{\alpha} + M_q \dot{\bar{\theta}},\end{aligned}$$

where we have substituted  $\bar{q} = \dot{\bar{\theta}}$ . Taking the Laplace transform of these equations gives

$$\begin{pmatrix} s - Z_w & -\frac{Z_q s}{V_a^* \cos \alpha^*} + \frac{g \sin \theta^*}{V_a^* \cos \alpha^*} \\ -M_w V_a^* \cos \alpha^* & s^2 - M_q s \end{pmatrix} \begin{pmatrix} \bar{\alpha}(s) \\ \bar{\theta}(s) \end{pmatrix} = \begin{pmatrix} \frac{Z_{\delta_e}}{V_a^* \cos \alpha^*} \\ 0 \end{pmatrix} \bar{\delta}_e(s),$$

which implies that

$$\begin{pmatrix} \bar{\alpha}(s) \\ \bar{\theta}(s) \end{pmatrix} = \frac{\begin{pmatrix} s^2 - M_q s & \frac{Z_q s}{V_a^* c_{\alpha^*}} - \frac{g s_{\theta^*}}{V_a^* c_{\alpha^*}} \\ M_w V_a^* c_{\alpha^*} & s - Z_w \end{pmatrix}}{(s^2 - M_q s)(s - Z_w) + M_w V_a^* c_{\alpha^*} \left( -\frac{Z_q s}{V_a^* c_{\alpha^*}} + \frac{g s_{\theta^*}}{V_a^* c_{\alpha^*}} \right)} \begin{pmatrix} \frac{Z_{\delta_e}}{V_a^* c_{\alpha^*}} \\ 0 \end{pmatrix} \bar{\delta}_e(s).$$

Assuming that we have linearized around level flight (i.e.,  $\theta^* = 0$ ), the characteristic equation becomes

$$s(s^2 + (-Z_w - M_q)s + M_q Z_w - M_w Z_q) = 0.$$

Therefore, the short-period poles are approximately equal to

$$\lambda_{\text{short}} = \frac{Z_w + M_q}{2} \pm \sqrt{\left( \frac{Z_w + M_q}{2} \right)^2 - M_q Z_w + M_w Z_q}.$$

#### Phugoid Mode

Assuming that  $\alpha$  is constant (i.e.,  $\bar{\alpha} = \dot{\bar{\alpha}} = 0$ ), then  $\alpha = \alpha^*$  and [equation \(5.51\)](#) becomes

$$\begin{aligned}\begin{pmatrix} \dot{\bar{u}} \\ 0 \\ \dot{\bar{q}} \\ \dot{\bar{\theta}} \end{pmatrix} &= \begin{pmatrix} X_u & X_w V_a^* s_{\alpha^*} & X_q & -g c_{\theta^*} \\ \frac{Z_u}{V_a^* c_{\alpha^*}} & Z_w & \frac{Z_q}{V_a^* c_{\alpha^*}} & \frac{-g s_{\theta^*}}{V_a^* c_{\alpha^*}} \\ M_u & M_w V_a^* c_{\alpha^*} & M_q & 0 \\ 0 & 0 & 1 & 0 \\ -s_{\theta^*} & -V_a^* c_{\theta^*} c_{\alpha^*} & 0 & u^* c_{\theta^*} + w^* s_{\theta^*} \end{pmatrix} \begin{pmatrix} \bar{u} \\ 0 \\ \bar{q} \\ \bar{\theta} \end{pmatrix} \\ &+ \begin{pmatrix} X_{\delta_e} \\ \frac{Z_{\delta_e}}{V_a^* c_{\alpha^*}} \\ M_{\delta_e} \\ 0 \\ 0 \end{pmatrix} \bar{\delta}_e.\end{aligned}$$

Taking the Laplace transform of the first two equations gives

$$\begin{pmatrix} s - X_u & -X_q s + g \cos \theta^* \\ -Z_u & -Z_q s + g \sin \theta^* \end{pmatrix} \begin{pmatrix} \bar{u}(s) \\ \bar{\theta}(s) \end{pmatrix} = \begin{pmatrix} X_{\delta_e} \\ Z_{\delta_e} \end{pmatrix} \bar{\delta}_e.$$

Again assuming that  $\theta^* = 0$ , we get that the characteristic equation is given by

$$s^2 + \left( \frac{Z_u X_q - X_u Z_q}{Z_q} \right) s - \frac{g Z_u}{Z_q} = 0.$$

The poles of the phugoid mode are approximately given by

$$\lambda_{\text{phugoid}} = -\frac{Z_u X_q - X_u Z_q}{2 Z_q} \pm \sqrt{\left( \frac{Z_u X_q - X_u Z_q}{2 Z_q} \right)^2 + \frac{g Z_u}{Z_q}}.$$

### Roll Mode

If we ignore the heading dynamics and assume a constant pitch angle (i.e.,  $\bar{\theta} = 0$ ), then [equation \(5.43\)](#) becomes

$$\begin{aligned} \begin{pmatrix} \dot{\bar{\beta}} \\ \dot{\bar{p}} \\ \dot{\bar{r}} \\ \dot{\bar{\phi}} \end{pmatrix} &= \begin{pmatrix} Y_v & \frac{Y_p}{V_a^* \cos \beta^*} & \frac{Y_r}{V_a^* \cos \beta^*} & \frac{g \cos \theta^* \cos \phi^*}{V_a^* \cos \beta^*} \\ L_v V_a^* \cos \beta^* & L_p & L_r & 0 \\ N_v V_a^* \cos \beta^* & N_p & N_r & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \bar{\beta} \\ \bar{p} \\ \bar{r} \\ \bar{\phi} \end{pmatrix} \\ &+ \begin{pmatrix} \frac{Y_{\delta_a}}{V_a^* \cos \beta^*} & \frac{Y_{\delta_r}}{V_a^* \cos \beta^*} \\ L_{\delta_a} & L_{\delta_r} \\ N_{\delta_a} & N_{\delta_r} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \bar{\delta}_a \\ \bar{\delta}_r \end{pmatrix}. \end{aligned} \quad (5.52)$$

The dynamics for  $\bar{p}$  are obtained from [equation \(5.52\)](#) as

$$\dot{\bar{p}} = L_v V_a^* \cos \beta^* \bar{\beta} + L_p \bar{p} + L_r \bar{r} + L_{\delta_a} \bar{\delta}_a + L_{\delta_r} \bar{\delta}_r.$$

The rolling mode is obtained by assuming that  $\bar{\beta} = \bar{r} = \bar{\delta}_r = 0$ :

$$\dot{\bar{p}} = +L_p \bar{p} + L_{\delta_a} \bar{\delta}_a.$$

The transfer function is therefore

$$\bar{p}(s) = \frac{L_{\delta_a}}{s - L_p} \bar{\delta}_a(s).$$

An approximation of the eigenvalue for the rolling mode is therefore given by

$$\lambda_{\text{rolling}} = L_p.$$

### Spiral-divergence Mode

For the spiral-divergence mode we assume that  $\dot{p} = \bar{p} = 0$ , and that the rudder command is negligible. Therefore, from the second and third equations in [equation \(5.52\)](#), we get

$$0 = L_v V_a^* \cos \beta^* \bar{\beta} + L_r \bar{r} + L_{\delta_a} \bar{\delta}_a \quad (5.53)$$

$$\dot{\bar{r}} = N_v V_a^* \cos \beta^* \bar{\beta} + N_r \bar{r} + N_{\delta_a} \bar{\delta}_a. \quad (5.54)$$

Solving [equation \(5.53\)](#) for  $\bar{\beta}$  and substituting into [equation \(5.54\)](#), we obtain

$$\dot{\bar{r}} = \left( \frac{N_r L_v - N_v L_r}{L_v} \right) \bar{r} + \left( \frac{N_{\delta_a} L_v - N_v L_{\delta_a}}{L_v} \right) \bar{\delta}_a.$$

In the frequency domain, we have

$$\bar{r}(s) = \frac{\left( \frac{N_{\delta_a} L_v - N_v L_{\delta_a}}{L_v} \right)}{s - \left( \frac{N_r L_v - N_v L_r}{L_v} \right)} \bar{\delta}_a(s).$$

From this, the pole of the spiral mode is approximately

$$\lambda_{\text{spiral}} = \frac{N_r L_v - N_v L_r}{L_v},$$

which is typically in the right half of the complex plane and is, therefore, an unstable mode.

### Dutch-roll Mode

For the dutch-roll mode, we neglect the rolling motions and focus on the equations for sideslip and yaw. From [equation \(5.52\)](#), we have

$$\begin{pmatrix} \dot{\bar{\beta}} \\ \dot{\bar{r}} \end{pmatrix} = \begin{pmatrix} Y_v & \frac{Y_r}{V_a^* \cos \beta^*} \\ N_v V_a^* \cos \beta^* & N_r \end{pmatrix} \begin{pmatrix} \bar{\beta} \\ \bar{r} \end{pmatrix} + \begin{pmatrix} \frac{Y_{\delta_r}}{V_a^* \cos \beta^*} \\ N_{\delta_r} \end{pmatrix} \bar{\delta}_r. \quad (5.55)$$

The characteristic equation is given by

$$\begin{aligned} \det \left( sI - \begin{pmatrix} Y_v & \frac{Y_r}{V_a^* \cos \beta^*} \\ N_v V_a^* \cos \beta^* & N_r \end{pmatrix} \right) \\ = s^2 + (-Y_v - N_r)s + (Y_v N_r - N_v Y_r) = 0. \end{aligned}$$

Therefore, the poles of the dutch-roll mode are approximated by

$$\lambda_{\text{dutch roll}} = \frac{Y_v + N_r}{2} \pm \sqrt{\left( \frac{Y_v + N_r}{2} \right)^2 - (Y_v N_r - N_v Y_r)}.$$

### NEW MATERIAL:

#### *Yaw Damping Mode*

For small aircraft where sideslip angle is difficult to measure, the rudder is typically used to reduce oscillations in the yaw rate, which can be measured by a rate gyro. Using the fact that for the state-space equations

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

the associated transfer function is

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

we can derive the transfer function from the rudder command to the yaw rate by letting  $C = (0, 1)$ , and using the state-space equations (5.55) to obtain

$$\begin{aligned}\bar{r}(s) &= (0 \ 1) \left( sI - \begin{pmatrix} Y_v & \frac{Y_r}{V_a^* \cos \beta^*} \\ N_v V_a^* \cos \beta^* & N_r \end{pmatrix} \right)^{-1} \begin{pmatrix} \frac{Y_{\delta_r}}{V_a^* \cos \beta^*} \\ \frac{N_{\delta_r}}{N_{\delta_r}} \end{pmatrix} \bar{\delta}_r(s) \\ &= \left( \frac{N_{\delta_r}s + (Y_{\delta_r}N_v - N_{\delta_r}Y_v)}{s^2 - (Y_v + N_r)s + (Y_vN_r - Y_rN_v)} \right) \bar{\delta}_r(s).\end{aligned}\quad (5.56)$$

### 5.6 CHAPTER SUMMARY

The objective of this chapter is to develop design models that can be used to develop low-level autopilot models for a fixed-wing miniature air vehicle. In particular we focus on linear models about trim conditions. In section 5.1 we summarized the nonlinear equations of motion developed in chapters 3 and 4. In section 5.2 we introduced the notion of a coordinated turn, which was used later in the chapter to model the relationship between roll angle and course rate. In section 5.3 we introduced the notion of trim states and inputs. In section 5.4 we linearized the nonlinear model and developed transfer functions that model the dominant relationships. The motion of the aircraft was decomposed into lateral and longitudinal dynamics. The transfer functions for the lateral dynamics are given by equations (5.24) and (5.25), which express the relationship between the aileron deflection and the roll angle, and the relationship between the roll angle and the course angle, respectively. For aircraft that have a rudder and the ability to measure the sideslip angle, equation (5.26) expresses the relationship between the

rudder deflection and the sideslip angle. The transfer functions for the longitudinal dynamics are given by [equations](#) (5.27), (5.29), (5.30), and (5.35), which model the relationship between the elevator deflection and the pitch angle, the pitch angle and the altitude, the airspeed and the altitude, and the throttle and pitch angle to the airspeed, respectively. In [section](#) 5.5 we developed state-space models linearized about the trim condition. The state-space model for the lateral dynamics is given in [equation](#) (5.42). The state-space model for the longitudinal dynamics is given in [equation](#) (5.49). In [section](#) 5.5.4 we discussed the modes associated with the linear models developed in this chapter, as they are defined in the traditional aeronautics literature. For the lateral dynamics, the modes are the roll mode, the dutch-roll mode, and the spiral-divergence mode. For the longitudinal dynamics, the modes are the short-period mode and the phugoid mode.

### NOTES AND REFERENCES

The models that we have developed in this chapter are standard. An excellent discussion of trim including algorithms for computing trim is contained in [1]. The transfer function models are discussed in detail in [6]. The state-space models are derived in [7, 17, 18, 27, 1, 8], which also discuss the reduced-order modes derived in [section](#) 5.5.4.

### 5.7 DESIGN PROJECT

#### MODIFIED MATERIAL:

- 5.1 Create a function that computes the trim state and the trim inputs for a desired airspeed of  $V_a^*$  and a desired flight path angle of  $\gamma^*$ . 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_1on` 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  $\omega_n$  and  $\zeta$  from the two complex conjugate pairs of poles. The pair with the larger  $\omega_n$  correspond to the short-period mode, and the pair with the smaller  $\omega_n$  correspond to the phugoid mode. The phugoid and short-period modes can be excited by starting the simulation in a wings-level, constant-altitude trim condition, and placing an impulse on the elevator. By placing an impulse on the elevator, convince yourself that the eigenvalues of `A_lon` adequately predict the short period and phugoid modes.

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



# *Chapter Six*

---

## Autopilot Design

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.

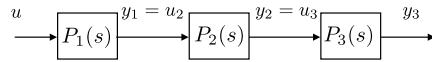
This chapter presents several autopilot design methods suitable for the sensors and computational resources available to MAVs. In section 6.1, we describe the successive loop closure approach and utilize it to design lateral-directional and longitudinal autopilots. In section 6.2, the concept of total energy control is explained and applied to the design of longitudinal autopilots. Section 6.3 describes the implementation of linear quadratic regulators for the lateral-directional and longitudinal autopilots.

### 6.1 SUCCESSIVE LOOP CLOSURE

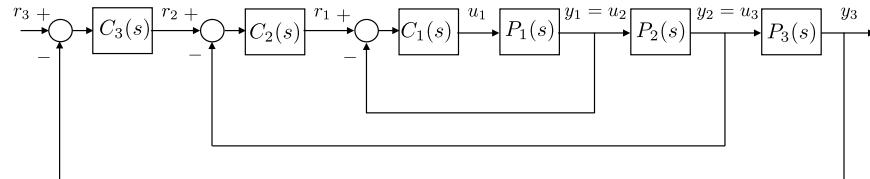
**MODIFIED MATERIAL:** The primary goal in autopilot design is to control the inertial position ( $p_n$ ,  $p_e$ ,  $h$ ) and attitude ( $\phi$ ,  $\theta$ ,  $\chi$ ) of the MAV. For most flight maneuvers of interest, autopilots designed on the assumption of decoupled dynamics yield good performance. 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.



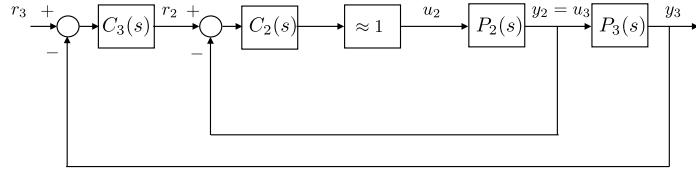
**Figure 6.1:** Open-loop transfer function modeled as a cascade of three transfer functions.



**Figure 6.2:** Three-stage successive loop closure design.

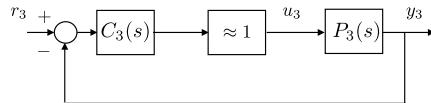
Examining the inner loop shown in figure 6.2, the goal is to design a closed-loop system from  $r_1$  to  $y_1$  having a bandwidth  $\omega_{BW1}$ . The key assumption we make is that for frequencies well below  $\omega_{BW1}$ , the closed-loop transfer function from  $r_1(s)$  to  $y_1(s)$  can be 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 outer loop so that it is a factor of  $W$  smaller than the inner loop, where  $W$  is typically in the range of 5 to 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, the transfer function from  $r_2(s)$  to  $y_2(s)$  is approximately one, and the transfer function from  $r_2(s)$  to  $y_2(s)$  can be replaced with a gain of 1 for the design of the outermost loop, as shown in figure 6.4. Again, there is a bandwidth constraint on the design of the outer loop:  $\omega_{BW3} < \frac{1}{W_2}\omega_{BW2}$ . Because each of the plant



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

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.

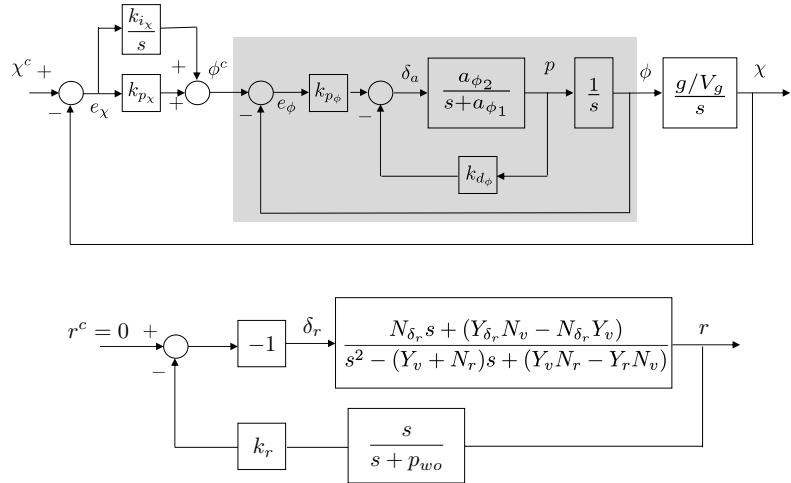
The following sections discuss the design of a lateral autopilot and a longitudinal autopilot. Transfer functions modeling the lateral and longitudinal dynamics were developed in section 5.4 and will be used to design the autopilots in this chapter.

### 6.1.1 Lateral-directional Autopilot

#### MODIFIED MATERIAL:

Figure 6.5 shows the block diagram for a lateral autopilot using successive loop closure. There are six design parameters 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  $k_r$  and high-pass filter pole  $p_{wo}$ . The idea with successive loop closure is that the gains are successively chosen beginning with the inner loop and working outward. In particular,  $k_{d\phi}$  and  $k_{p\phi}$  are usually selected first, and  $k_{p_x}$  and  $k_{i_x}$  are usually chosen second. The pole  $p_{wo}$  and gain  $k_r$  is selected independently of the other gains.

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



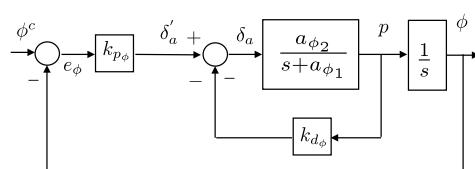
**Figure 6.5:** Autopilot for lateral control using successive loop closure.

#### Roll Hold using the Aileron

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

#### MODIFIED MATERIAL:

If the transfer function coefficients  $a_{\phi_1}$  and  $a_{\phi_2}$  are known, then there is a systematic method for selecting the control gains  $k_{d_\phi}$  and  $k_{p_\phi}$  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  $\phi^c$  to  $\phi$  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, i.e.,  $k_{DC} \triangleq \lim_{s \rightarrow 0} H_{\phi/\phi^c}(s)$ , 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  $\omega_{n_\phi}$  and the damping ratio  $\zeta_\phi$  are design parameters.

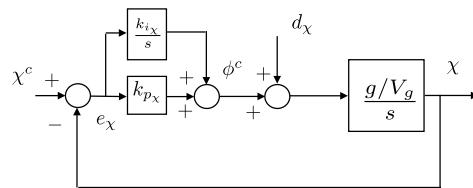
The control signal for the roll attitude hold loop is therefore

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

#### *Course Hold using Commanded Roll*

#### MODIFIED MATERIAL:

The next step in the successive-loop-closure design of the lateral autopilot is to design the course-hold outer loop. If the inner loop from  $\phi^c$  to  $\phi$  has been adequately tuned, then  $H_{\phi/\phi^c}(j\omega) \approx 1$  over the range of frequencies from 0 to  $\omega_{n_\phi}$ . Under this assumption, 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 control loop.



**Figure 6.7:** Course hold outer feedback loop.

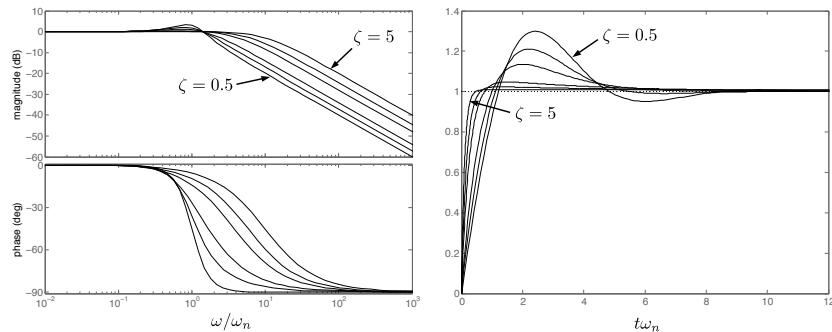
The objective of the course hold design is to select  $k_{p_\chi}$  and  $k_{i_\chi}$  in figure 6.5 so that the course  $\chi$  asymptotically tracks steps in the commanded course  $\chi^c$ . From the simplified block diagram, the transfer functions from the inputs  $\chi^c$  and  $d_\chi$  to the output  $\chi$  are given by

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

Note that if  $d_\chi(t)$  and  $\chi^c(t)$  are constants, then the final value theorem implies that  $\chi(t) \rightarrow \chi^c(t)$ . The transfer function from  $\chi^c(s)$  to  $\chi(s)$  has the form

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

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  $k_{p_\chi}$  and  $k_{i_\chi}$ . Figure 6.8 shows the frequency response and the step response associated with a canonical transfer of the form given in Equation (6.4). Note that because of the finite zero in the numerator, the standard intuition for the selection of  $\zeta$  does not hold for this transfer function. Larger  $\zeta$  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

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

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

Solving these expressions for  $k_{p_\chi}$  and  $k_{i_\chi}$ , 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  $W_\chi$  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 course ( $\chi$ ) loop (lower  $\omega_{n_\chi}$ ), or faster response in the roll ( $\phi$ ) loop (higher  $\omega_{n_\phi}$ ). 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 control signal associated with the course hold loop is therefore

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

#### *Yaw Damper using the Rudder*

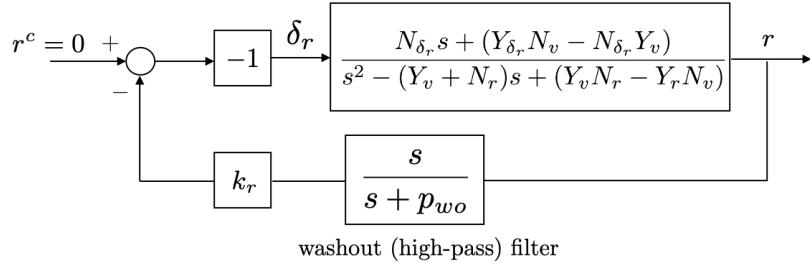
**NEW MATERIAL:** As discussed in the previous section, the aileron is used to control the roll angle, and subsequently the course angle. In deriving the transfer function from the roll angle to the course angle we assumed that the sideslip angle is zero. For large aircraft where the sideslip angle can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

Recall from Equation (5.56) that the transfer function from the rudder to the yaw rate is given by

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

The yaw-damper control loop is shown in Figure 6.9, where  $p_{wo}$  is the pole of the washout-filter, and  $k_r > 0$  is the control gain from yaw-rate to rudder. The effect of the washout filter is to remove the feedback signal over the frequency band below  $p_{wo}$  and to pass with unity gain the feedback signal for frequencies above  $p_{wo}$ . The negative sign in the top path is due to the fact that negative  $\delta_r$  results in a positive yaw rate (i.e.,  $N_{\delta_r} < 0$ ).

The selection of  $p_{wo}$  and  $k_r$  can be found from physical considerations. As discussed in Chapter 5, the poles of the transfer function from  $\delta_r$  to  $r$  shown in Equation 6.7 correspond to the lightly-damped dutch roll mode.



**Figure 6.9:** Yaw damper with washout filter.

Essentially, the objective of the yaw damper is damp out the dutch-roll mode. Therefore, the pole of the washout filter should be selected to be below the natural frequency of the dutch-roll mode. The canonical expression  $s^2 + 2\zeta_{dr}\omega_{n_{dr}}s + \omega_{n_{dr}}^2 = s^2 - (Y_v + N_r)s + (Y_vN_r - Y_rN_v)$  approximates the natural frequency of the dutch-roll mode as

$$\omega_{n_{dr}} \approx \sqrt{Y_vN_r - Y_rN_v}.$$

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

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

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

where the closed-loop poles are given by

$$p_{\text{yaw-damper}} = \frac{Y_v + N_r + k_rN_{\delta_r}}{2} \pm j \sqrt{(Y_vN_r - Y_rN_v - k_r(Y_{\delta_r} - N_vN_{\delta_r}Y_v)) - \left( \frac{Y_v + N_r + k_rN_{\delta_r}}{2} \right)^2}.$$

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

$$\begin{aligned} & \left( \frac{Y_v + N_r + k_rN_{\delta_r}}{2} \right)^2 \\ &= (Y_vN_r - Y_rN_v - k_r(Y_{\delta_r} - N_vN_{\delta_r}Y_v)) - \left( \frac{Y_v + N_r + k_rN_{\delta_r}}{2} \right)^2. \end{aligned}$$

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

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

resulting in

$$k_r = -\frac{N_r N_{\delta_r} + Y_{\delta_r} N_v}{N_{\delta_r}^2}$$

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

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

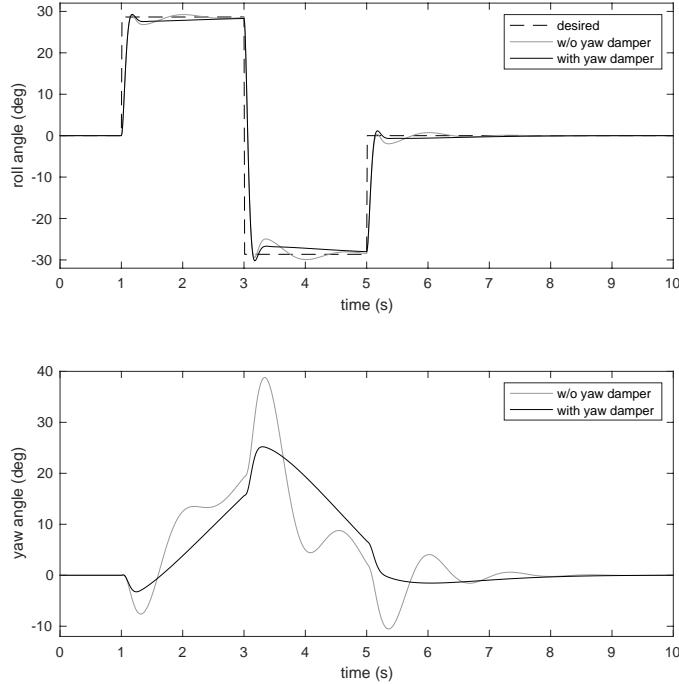
Figure 6.10 shows the lateral-directional performance of the Aerosonde aircraft in response to a doublet roll command, both without the yaw damper and with the yaw damper. The upper plot shows the roll response of the aircraft to the doublet roll command. Without the yaw damper, the lightly damped dutch-roll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

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.

In summary, the yaw damper is represented in the Laplace domain by

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

See appendix C for a description of how to implement a transfer function in computer code.



**Figure 6.10:** Turning performance with and without yaw damper.

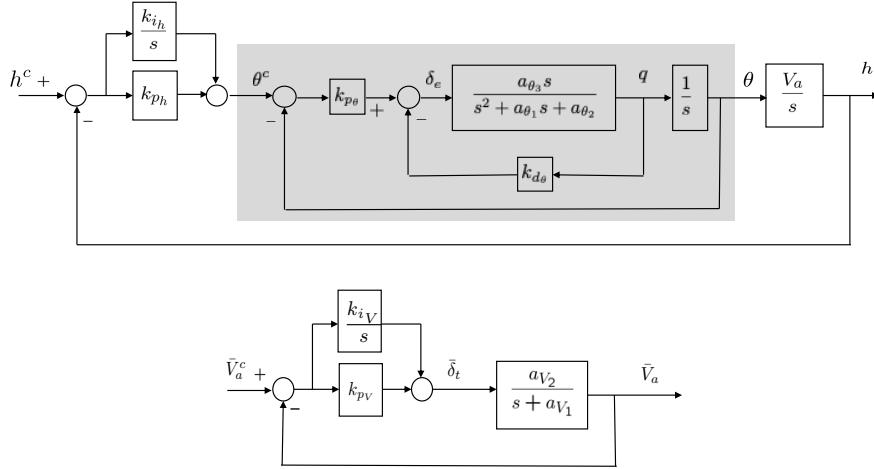
### 6.1.2 Longitudinal Autopilot

**MODIFIED MATERIAL:** Figure 6.11 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 inner most 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  $k_{pv_a}$  and  $k_{iv_a}$ . 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  $k_{ph}$  and  $k_{ih}$  are usually chosen second. The gains  $k_{pv_a}$  and  $k_{iv_a}$  are selected independently of the other gains.

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

#### *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

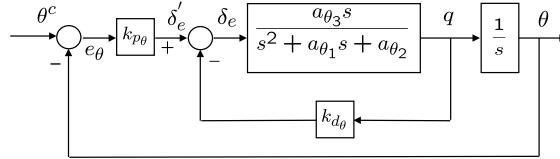


**Figure 6.11:** Autopilot for longitudinal control using successive loop closure.

development. From figure 6.12, the transfer function from  $\theta^c$  to  $\theta$  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 6.12:** 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  $k_{p_\theta}$ ,  $k_{d_\theta}$ , 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  $\omega_{n_\theta}$  and  $\zeta_\theta$  are design parameters.

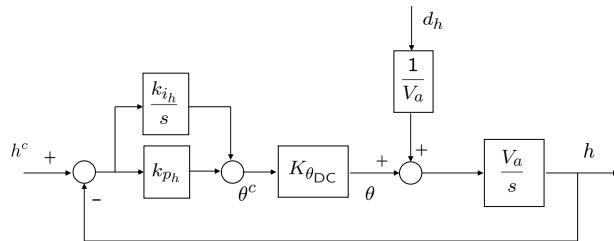
An integral feedback term could be employed to ensure unity DC gain on the inner loop. The addition of an integral term, however, can severely limit the bandwidth of the inner loop, and therefore is not used. Note however, that in the design project, the actual pitch angle will not converge to the commanded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

The output of the pitch-attitude controller is therefore

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

#### 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 6.11. 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 6.13.



**Figure 6.13:** The altitude-hold loop using the commanded pitch angle.

In the Laplace domain, we have

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

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

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

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

where the bandwidth separation  $W_h$  is a design parameter that is usually close to 10. 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

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

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

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

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

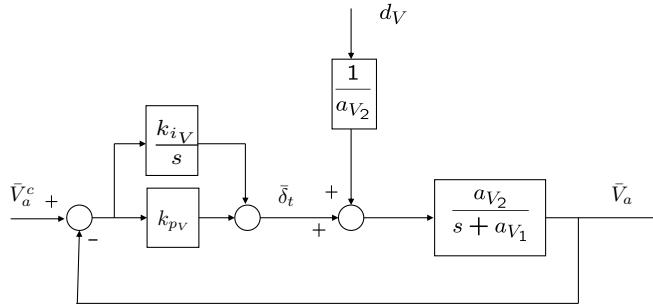
Therefore, selecting the desired damping ratio  $\zeta_h$  and the bandwidth separation  $W_h$  fixes the value for  $k_{p_h}$  and  $k_{i_h}$ .

The commanded pitch angle is therefore

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

#### Airspeed Hold using Throttle

**MODIFIED MATERIAL:** Using PI control for the airspeed loop results in the closed-loop system is shown in figure 6.14. If we use proportional



**Figure 6.14:** Airspeed hold using throttle.

control, then

$$\bar{V}_a(s) = \left( \frac{a_{V_2} k_{pV}}{s + (a_{V_1} + a_{V_2} k_{pV})} \right) \bar{V}_a^c(s) + \left( \frac{1}{s + (a_{V_1} + a_{V_2} k_{pV})} \right) d_V(s),$$

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

$$\begin{aligned} \bar{V}_a = & \left( \frac{a_{V_2}(k_{pV}s + k_{iV})}{s^2 + (a_{V_1} + a_{V_2}k_{pV})s + a_{V_2}k_{iV}} \right) \bar{V}_a^c \\ & + \left( \frac{1}{s^2 + (a_{V_1} + a_{V_2}k_{pV})s + a_{V_2}k_{iV}} \right) d_V. \end{aligned}$$

resulting in a DC gain of one with step disturbance rejection. If  $a_{V_1}$  and  $a_{V_2}$  are known, then the gains  $k_{pV}$  and  $k_{iV}$  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

$$\begin{aligned} \omega_{nV}^2 &= a_{V_2} k_{iV} \\ 2\zeta_V \omega_{nV} &= a_{V_1} + a_{V_2} k_{pV}. \end{aligned}$$

Inverting these expressions gives the control gains

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

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

The design parameters for this loop are the damping coefficient  $\zeta_V$  and the natural frequency  $\omega_{nV}$ .

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 6.14 is

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

Therefore, the control loop shown in figure 6.14 can be implemented without knowledge of the trim velocity  $V_a^*$ . If the throttle trim value  $\delta_t^*$  is known, then the throttle command is

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

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

The throttle command is therefore

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

## 6.2 TOTAL ENERGY CONTROL

**NEW MATERIAL:** An alternative design for the longitudinal autopilot that is used in the Ardupilot autopilot<sup>1</sup>, is the total energy control scheme (TECS) proposed in [28, 29]. The advantage of TECS for the longitudinal autopilot is that there is an implicit coupling between the altitude and airspeed loops that results in better performance.

We will assume that the pitch attitude hold loop described in the previous section is active. Therefore, for TECS, the control signals are the throttle and the commanded pitch angle. Both of these quantities have a significant effect on 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

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

---

<sup>1</sup><https://ardupilot.org/plane/docs/tecs-total-energy-control-system-for-speed-height-tuning-guide.html>

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  $U_{\text{error}}$ , and  $\text{sat}$  is the saturation function.

### 6.3 LQR CONTROL

**NEW MATERIAL:** 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

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

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  $z^r$  and further suppose that  $z^r$  is a step, i.e.,  $\dot{z}^r = 0$ .

The first step is to augment the state with the integrator

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

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, using the augmented cost function

$$J(x_0) = \min_{u(t)} \int_0^\infty \xi^\top(\tau) \bar{Q} \xi(\tau) + u^\top(\tau) R u(\tau) d\tau.$$

### 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  $u_{lat} = (\delta_a, \delta_r)^\top$ . For small aircraft we typically actuate the rudder using a yaw damper, and let  $u_{lat} = \delta_a$  and ignore the second column of  $B_{lat}$  in Equation (5.42). The objective of the lateral autopilot is to drive the course  $\chi$  to the commanded course  $\chi^c$ . 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

$$Q = \text{diag}([q_v, q_p, q_r, q_\phi, q_\chi, q_I])$$

$$R = r_{\delta_a}.$$

### 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

**NEW MATERIAL:** In this chapter we have described several different designs for the low-level autopilot of a small aircraft. We have decomposed the autopilot into lateral control for side-to-side motion actuated by the ailerons and rudder, and longitudinal control for up-down motion using the elevator and throttle. The traditional method of autopilot design is successive loop

closure and has the advantage that a model of the aircraft is not required since the PID gains can be tuned without knowledge of the model. One of the disadvantages of successive loop closure is that time constants for inner and outer loops must be adequately separated. In the longitudinal autopilot, this leads to undesirable coupling between airspeed and altitude control. The total energy control system addresses this problem by directly coupling airspeed and altitude control. If a linear state space model of the aircraft is known, then LQR controllers can exploit that knowledge resulting in significantly better performance as compared to successive loop closure and total energy control.

#### NOTES AND REFERENCES

**NEW MATERIAL:** Successive-loop-closure is a traditional design technique described in [7, 30, 17].

Total energy control system (TECS) was proposed in [31, 32, 33] in the early 1980's by Lambregts who realized that the airspeed and altitude of an aircraft could be controlled by manipulating the kinetic and potential energy of the system to couple the the altitude and airspeed of the longitudinal autopilot. TECS control schemes have been successfully used and tested on a variety of airframes [34, 35]. While most of these controllers are of PI type, other variants have been developed such as adding pitch damping [36]. TECS concepts have also been applied to the lateral control of an aircraft [37, 38] and for the longitudinal control of a helicopter [39], and are implemented in the Ardupilot autopilot firmware. A nonlinear version of TECS is described in [40].

There are many excellent references on the linear quadratic regulator (LQR) including [41, 42, 43, 44]. Application of LQR to flight control is described in many references including [1, 45].

## 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  $\pm 180$  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, to quantify what they measure, and to describe how they can be simulated. 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.

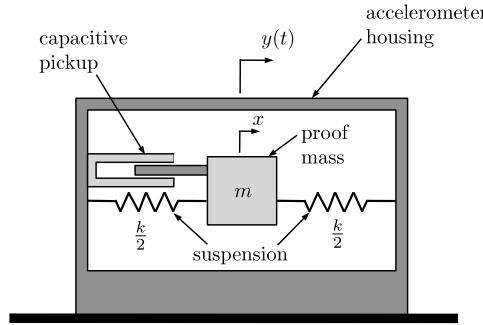
The following sensors are often found on MAVs:

- 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 [casing](#) of the accelerometer experiences an acceleration, the proof mass



**Figure 7.1:** Conceptual depiction of MEMS accelerometer.

moves relative to the [casing](#) 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

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

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

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

At frequencies corresponding to  $\omega < \sqrt{k/m}$ , the transfer function  $H_{A_X/A_Y} \approx 1$  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

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

where  $y_{\text{accel}}$  is in volts,  $k_{\text{accel}}$  is a gain,  $A$  is the acceleration in meters per second squared,  $\beta_{\text{accel}}$  is a bias term, and  $\eta'_{\text{accel}}$  is zero-mean Gaussian noise. The gain  $k_{\text{accel}}$  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  $\beta_{\text{accel}}$  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.

**NEW MATERIAL:** In practice it is difficult to exactly align the sensitive axes of the accelerometer with the vehicle body axes. The misalignment can be represented by a constant rotation matrix  $R_b^a$  from the body to the accelerometer frame. The output of a 3-axis accelerometer is modeled as

$$\mathbf{y}'_{\text{accel}} = k_{\text{accel}} R_b^a \left( \frac{1}{m} \mathbf{f}^b - R_v^b \mathbf{g}^i + \boldsymbol{\beta}_{\text{accel}} + \boldsymbol{\eta}_{\text{accel}} \right),$$

where the biases  $\boldsymbol{\beta}_{\text{accel}}$  is assumed to be slowly varying and unknown, and  $\boldsymbol{\eta}_{\text{accel}}$  is a zero-mean white Gaussian random process with instantaneous covariance  $Q_{\text{accel}}$ .

In the remainder of this book, we will assume that  $k_{\text{accel}}$  is known, and that  $R_b^a$ , and  $\boldsymbol{\beta}_{\text{accel}}$  have been determined through a calibration process, and

that they can therefore be removed from the measurement to produce

$$\begin{aligned}\mathbf{y}_{\text{accel}} &= \frac{1}{k_{\text{accel}}} R_b^{a\top} \mathbf{y}'_{\text{accel}} - \boldsymbol{\beta}_{\text{accel}} \\ &= \frac{1}{m} \mathbf{f}^b - R_v^b \mathbf{g}^i + \boldsymbol{\eta}_{\text{accel}}.\end{aligned}$$

The resulting equations in component form are

$$\begin{aligned}y_{\text{accel},x} &= \frac{f_x}{m} + g \sin \theta + \eta_{\text{accel},x} \\ y_{\text{accel},y} &= \frac{f_y}{m} - g \cos \theta \sin \phi + \eta_{\text{accel},y} \\ y_{\text{accel},z} &= \frac{f_z}{m} - g \cos \theta \cos \phi + \eta_{\text{accel},z},\end{aligned}\tag{7.1}$$

where  $f_x$ ,  $f_y$ , and  $f_z$  are given in equation (4.24), and 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. After calibration, the units of  $y_{\text{accel},x}$ ,  $y_{\text{accel},y}$ , and  $y_{\text{accel},z}$  are in  $m/s^2$ . With the exception of the noise terms, the terms on the right hand sides of equation (7.1) 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 (7.1) 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.

Using the relationship  $\mathbf{f}^b = m \left( \frac{d\mathbf{v}}{dt_b} + \boldsymbol{\omega}_{b/i} \times \mathbf{v} \right)$  from Equation (3.6), we can also write the outputs as

$$\begin{aligned}y_{\text{accel},x} &= \dot{u} + qw - rv + g \sin \theta + \eta_{\text{accel},x} \\ y_{\text{accel},y} &= \dot{v} + ru - pw - g \cos \theta \sin \phi + \eta_{\text{accel},y} \\ y_{\text{accel},z} &= \dot{w} + pv - qu - g \cos \theta \cos \phi + \eta_{\text{accel},z},\end{aligned}\tag{7.2}$$

which is an expression that we will use in the next chapter in the derivation a two-state extended Kalman filter to estimate the roll and pitch angles.

## 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

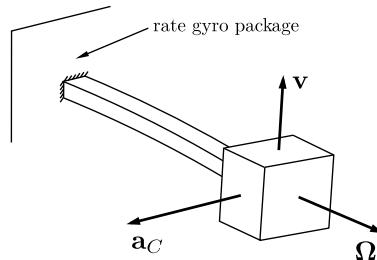
$$\mathbf{a}_C = 2\boldsymbol{\Omega} \times \mathbf{v}, \quad (7.3)$$

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

$$v = A\omega_n \sin(\omega_n t),$$

where  $A$  is the amplitude of the oscillation and  $\omega_n$  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 (7.3) 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.



**Figure 7.2:** Conceptual depiction of proof mass rate gyro.  $\boldsymbol{\Omega}$  is the angular velocity of the sensor package to be measured.  $\mathbf{v}$  is the actuated vibration velocity of the cantilever.  $\mathbf{a}_C$  is the Coriolis acceleration that results as the sensor package undergoes an angular velocity.

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

acceleration, 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

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

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

The output of a single axis rate gyro can be modeled as

$$y_{\text{gyro}} = k_{\text{gyro}} \Omega + \beta_{\text{gyro}} + \eta_{\text{gyro}},$$

where  $y_{\text{gyro}}$  corresponds to the measured rate of rotation in volts,  $k_{\text{gyro}}$  is a gain converting the rate in radians per second to volts,  $\Omega$  is the angular rate in radians per second,  $\beta_{\text{gyro}}$  is a bias term, and  $\eta_{\text{gyro}}$  is zero-mean Gaussian noise. An approximate value for the gain  $k_{\text{gyro}}$  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  $\beta_{\text{gyro}}$  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  $\Upsilon_{\text{gyro}}$  averages to zero over a period of 100 or so samples, or by explicitly estimating the bias using an estimation scheme as discussed in the next chapter.

**NEW MATERIAL:** As with accelerometers, in practice it is difficult to exactly align the sensitive axes of the rate gyros with the vehicle body axes. The misalignment can be represented by a constant rotation matrix  $R_b^g$  from the body to the rate gyro frame. The output of a 3-axis rate gyro is modeled as

$$\mathbf{y}'_{\text{gyro}} = k_{\text{gyro}} R_b^g (\boldsymbol{\omega}_{b/i}^b + \boldsymbol{\beta}_{\text{gyro}} + \boldsymbol{\eta}_{\text{gyro}}),$$

where the biases  $\beta_{\text{gyro}}$  are assumed to be slowly varying and unknown, and  $\eta_{\text{gyro}}$  is a zero-mean white Gaussian random process with instantaneous covariance  $Q_{\text{gyro}}$ .

In the remainder of this book, we will assume that  $k_{\text{gyro}}$  is known, that  $R_b^g$  has been determined through a calibration process, and that  $\beta_{\text{accel}}$  is estimated, and that they can therefore be removed from the measurement to produce

$$\begin{aligned}\mathbf{y}_{\text{gyro}} &= \frac{1}{k_{\text{gyro}}} R_b^{g\top} \mathbf{y}'_{\text{gyro}} - \beta_{\text{gyro}} \\ &= \boldsymbol{\omega}_{b/i}^b + \boldsymbol{\eta}_{\text{gyro}}.\end{aligned}$$

The resulting equations in component form are

$$\begin{aligned}y_{\text{gyro},x} &= p + \eta_{\text{gyro},x} \\ y_{\text{gyro},y} &= q + \eta_{\text{gyro},y} \\ y_{\text{gyro},z} &= r + \eta_{\text{gyro},z},\end{aligned}\tag{7.4}$$

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.

### 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

$$P_2 - P_1 = \rho g(z_2 - z_1),\tag{7.5}$$

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 \(7.5\)](#) 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 \(7.5\)](#), the change in pressure due to a change in altitude is given by

$$\begin{aligned} P - P_{\text{ground}} &= -\rho g(h - h_{\text{ground}}) \\ &= -\rho g h_{\text{AGL}}, \end{aligned} \quad (7.6)$$

where  $h$  is the absolute altitude of the aircraft,  $h_{\text{ground}}$  is the absolute altitude of the ground,  $h_{\text{AGL}} = h - h_{\text{ground}}$ , and  $h$  and  $h_{\text{ground}}$  are measured with respect to sea level and  $P$  is the corresponding absolute pressure measurement. The change in sign between [equations \(7.5\)](#) and (7.6) 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 [practice](#)  $P_{\text{ground}}$  is the atmospheric pressure measured at ground level prior to take off and  $\rho$  is the air density at the flight location.

[Equation \(7.6\)](#) 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.

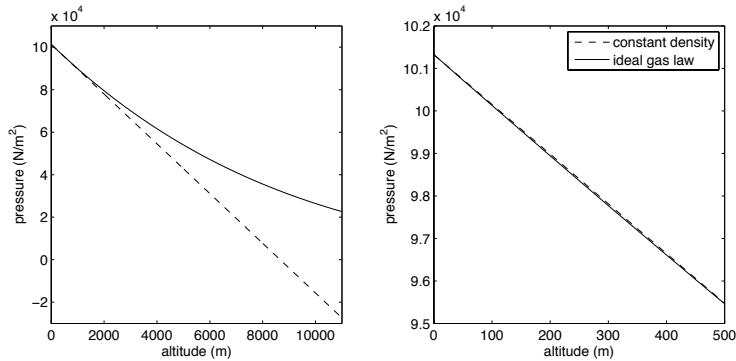
Below altitudes of 11,000 m above sea level, the pressure of the atmosphere can be calculated using the barometric formula [\[46\]](#). This formula takes into account the change in density and pressure due to decreasing temperature with altitude and is given by

$$P = P_0 \left[ \frac{T_0}{T_0 + L_0 h_{\text{ASL}}} \right]^{\frac{gM}{RL_0}}, \quad (7.7)$$

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  $h_{\text{ASL}}$  is referenced to sea level.

The relative significance of the constant-density assumption can be seen by comparing pressures calculated using [equations \(7.6\)](#) and (7.7) 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 \(7.6\)](#) 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 \(7.6\)](#) 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 \(7.6\)](#) 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

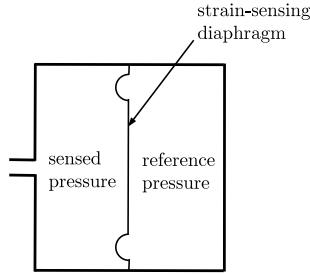
$$\rho = \frac{MP}{RT},$$

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<sup>2</sup>. Typical weather data reports pressure in inches of mercury (Hg). The conversion factor is 3385 N/m<sup>2</sup> = 1 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 sensor consists of two volumes separated by a diaphragm. The volume on



**Figure 7.4:** Schematic of an absolute pressure sensor.

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 (7.6), the output of the absolute pressure sensor of interest is given by

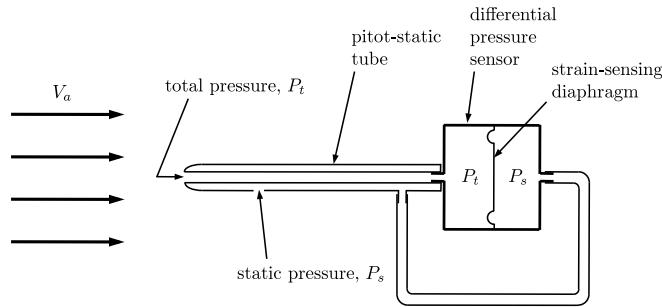
$$\begin{aligned} y_{\text{abs pres}} &= (P_{\text{ground}} - P) + \beta_{\text{abs pres}} + \eta_{\text{abs pres}} \\ &= \rho g h_{\text{AGL}} + \beta_{\text{abs pres}} + \eta_{\text{abs pres}} \end{aligned} \quad (7.8)$$

where  $h_{\text{AGL}}$  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  $\sigma_{\text{abs pres}}^2$ .  $P_{\text{ground}}$  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. In the remainder of this book we will assume that the bias  $\beta_{\text{abs pres}}$  is zero or has been removed through calibration.

### 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  $\rho$  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/m<sup>2</sup>, we can model the output of the differential pressure sensor as

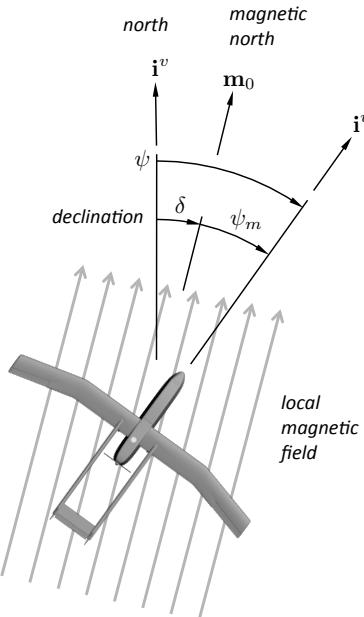
$$y_{\text{diff pres}} = \frac{\rho V_a^2}{2} + \beta_{\text{diff pres}} + \eta_{\text{diff pres}}, \quad (7.9)$$

where  $\beta_{\text{diff pres}}$  is a temperature-related bias drift,  $\eta_{\text{diff pres}}$  is zero-mean Gaussian noise with variance  $\sigma_{\text{diff pres}}^2$ . In the remainder of this book we will assume that the bias  $\beta_{\text{diff pres}}$  is zero or has been removed through calibration. 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 MAGNETOMETER AND DIGITAL COMPASS

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 AD. Compasses appeared in Europe around the 11th century AD 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.

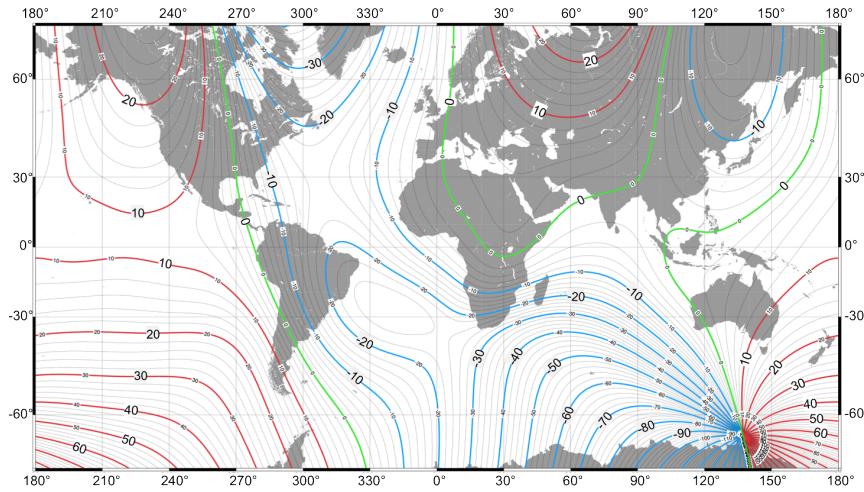
The magnetic field around the earth behaves similarly to that of a common magnetic dipole with the magnetic field lines running normal to the earth's surface at the poles and parallel to the earth's surface near the equator. Except near the poles, the earth's magnetic field points to magnetic north. A compass measures the direction of the magnetic field locally and provides an indication of heading relative to magnetic north denoted as  $\psi_m$ , as depicted schematically in figure 7.6. The declination angle  $\delta$  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 of the magnetic field is 21,053 nT. ■

the east component is 4520 nT, and the down component is 47,689 nT, and 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 angle that the magnetic field makes with the horizontal plane, is called the *inclination angle* and is denoted as  $\iota$ . A map of the inclination angle over the surface of the earth is shown in Figure 7.8. As an example, in Provo Utah the inclination is  $\iota = 65.7$  degrees.



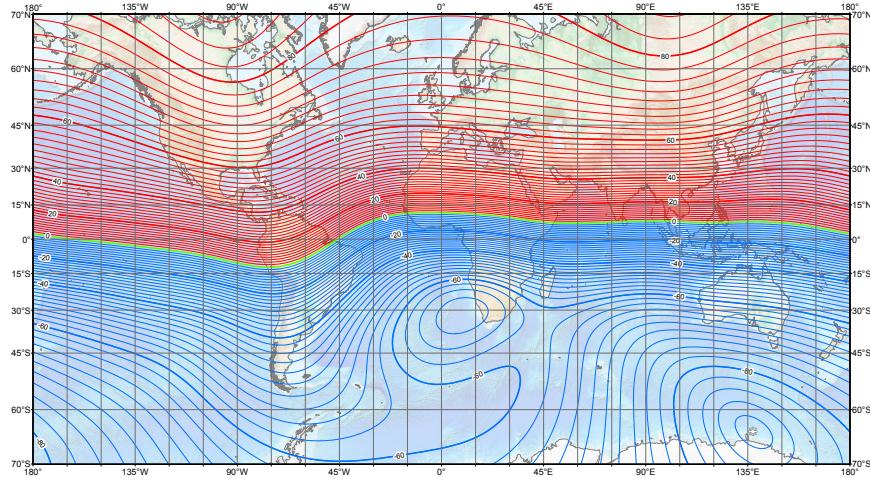
**Figure 7.7:** Lines of constant declination angles  $\delta$  of the earth's magnetic field, according to the US/UK World Magnetic Model. Adapted from [47].

Therefore, the magnetic field in the inertial frame can be written as

$$\begin{aligned} \mathbf{m}^i &= \begin{pmatrix} \cos \delta & -\sin \delta & 0 \\ \sin \delta & \cos \delta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(-\iota) & 0 & \sin(-\iota) \\ 0 & 1 & 0 \\ -\sin(-\iota) & 0 & \cos(-\iota) \end{pmatrix} \begin{pmatrix} M \\ 0 \\ 0 \end{pmatrix}, \\ &= M \begin{pmatrix} \cos \delta \cos \iota \\ \sin \delta \cos \iota \\ \sin \iota \end{pmatrix}, \end{aligned}$$

where  $M$  is the strength of the magnetic field,  $\delta$  is the declination angle,  $\iota$  is the inclination angle, and  $(-\iota)$  is used in the second matrix because the inclination angle is defined as a left-handed rotation, where positive  $\iota$  points into the earth. A 3-axis magnetometer measures the magnetic field in the body frame as

$$\mathbf{y}_{mag}^b = R_v^b(\phi, \theta, \psi) (\mathbf{m}^i + \boldsymbol{\beta}_{mag}^i) + \boldsymbol{\eta}_{mag}, \quad (7.10)$$



**Figure 7.8:** Lines of constant inclination angles  $\iota$  of the earth's magnetic field, according to the US/UK World Magnetic Model. Adapted from [47].

where  $\beta_{\text{mag}}^i$  is a constant bias due to uncertainty in the inclination and declination angles and other electronic biases, and  $\eta_{\text{mag}}$  is zero-mean Gaussian noise.

Modern digital compasses use three-axis magnetometers to measure the strength of the magnetic field along three orthogonal axes, and embedded accelerometers to project the magnetic field onto the local-level, or vehicle-1, plane, assuming that the compass is not accelerating. The magnetic field in the vehicle-1 frame is given by

$$\begin{aligned}
 \mathbf{m}^{v1} &= \begin{pmatrix} m_x^{v1} \\ m_y^{v1} \\ m_z^{v1} \end{pmatrix} = R_{v2}^{v1}(\theta) R_b^{v2}(\phi) \mathbf{m}^b \\
 &= R_{v2}^{v1}(\theta) R_b^{v2}(\phi) R_{v2}^b(\phi) R_{v1}^{v2}(\theta) R_v^{v1}(\psi) \mathbf{m}^i \\
 &= R_v^{v1}(\psi) \mathbf{m}^i \\
 &= M \begin{pmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \delta \cos \iota \\ \sin \delta \cos \iota \\ \sin \iota \end{pmatrix} \\
 &= M \begin{pmatrix} \cos(\psi - \delta) \cos \iota \\ -\sin(\psi - \delta) \cos \iota \\ \sin \iota \end{pmatrix}.
 \end{aligned}$$

The measurement of the magnetic heading  $\psi_m \triangleq \psi - \delta$  is therefore given

by

$$\begin{aligned} y_{\text{compass}} &= \psi_m + \beta_{\text{compass}} + \eta_{\text{compass}} \\ &= -\text{atan2}(m_y^{v1}, m_x^{v1}) + \beta_{\text{compass}} + \eta_{\text{compass}}, \end{aligned} \quad (7.11)$$

where the four-quadrant inverse tangent function  $\text{atan2}(y,x)$  returns the arc-tangent of  $y/x$  in the range  $[-\pi, \pi]$  using the signs of both arguments to determine the quadrant of the return value, and where  $\eta_{\text{compass}}$  is a zero-mean Gaussian process with variance  $\sigma_{\text{compass}}^2$ , and  $\beta_{\text{compass}}$  is a bias error due to uncertainty in the declination and inclination angles and other electronic biases. A digital compass typically communicates over a serial link with the autopilot at sample rate  $T_s$ .

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 digital compass, that in their sophistication with full-featured versions incorporating tilt compensation and automatic declination/inclination calculation from latitude and longitude data.

## 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 US 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., [48, 49, 50, 51]). 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 [48]. 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 *pseudo-range* to distinguish it from the true range.

#### MODIFIED MATERIAL:

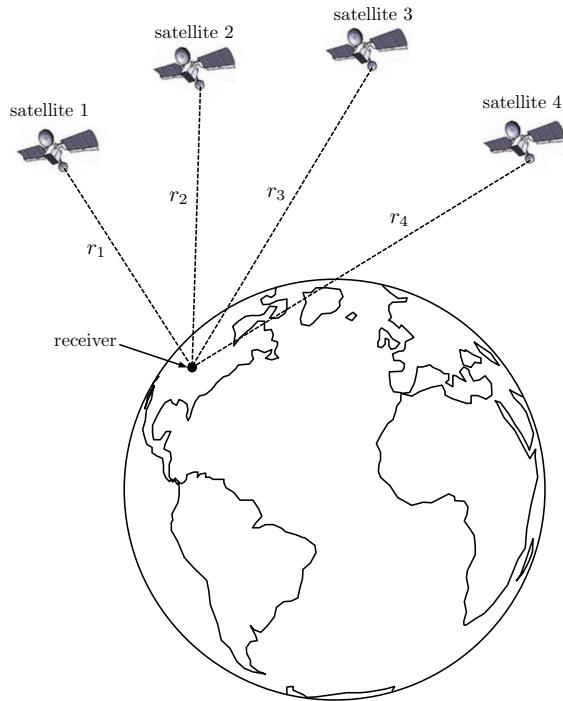
Because of clock synchronization errors between the satellites (whose atomic clocks are almost perfectly synchronized) and the receiver, four independent pseudo-range measurements are required to trilaterate the position of the receiver as depicted in figure 7.9. Why are four pseudo-range measurements required? One range measurement narrows the receiver position to a sphere around the satellite. A second measurement from another satellite narrows the possible positions to the intersection of the two spheres around the satellites, which is a circle. A third measurement narrows the possible positions to the intersection of the three spheres around the satellites, which is two points. The point closest to the surface of the earth is taken as the position. To resolve position in three dimensions with a receiver clock offset error, at least four measurements are needed. The geometry associated with the pseudo-range measurements from four different satellites form a system of four nonlinear algebraic equations in four unknowns: latitude, longitude, and altitude of the GPS receiver, and receiver clock time offset [48].

##### 7.5.1 GPS Measurement Error

The accuracy of a GPS position measurement is affected by the accuracy of the satellite **pseudo-range** measurements and by the geometry of the satellites from which **pseudo-range** measurements are taken. The effect of satellite geometry is taken into consideration through a factor called dilution of precision (DOP). **Pseudo-range** 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 [48, 49].

#### EPHEMERIS DATA

The satellite ephemeris is a mathematical description of its orbit. Calculation of the receiver location requires the satellite locations to be known.



**Figure 7.9:** Pseudo-range measurements from four satellites used to [trilaterate](#) the position of a receiver.

Ephemeris errors in the [pseudo-range](#) 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 [pseudo-range](#) 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.

### 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.

[Pseudo-range](#) 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 [pseudo-range](#) measurement is called the user-equivalent range error (UERE). Parkinson, et al. [49] 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 ( $1-\sigma$ ) [48].

The [pseudo-range](#) 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) [48]. 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

**Table 7.1:** Standard *pseudo-range* error model ( $1-\sigma$ , in meters) [49]

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

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 vertical 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 [48].

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

$$\begin{aligned} E_{n-e,\text{rms}} &= \text{HDOP} \times \text{UERE}_{\text{rms}} \\ &= (1.3)(5.1 \text{ m}) \\ &= 6.6 \text{ m}. \end{aligned} \tag{7.12}$$

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

$$\begin{aligned} E_{h,\text{rms}} &= \text{VDOP} \times \text{UERE}_{\text{rms}} \\ &= (1.8)(5.1 \text{ m}) \\ &= 9.2 \text{ m}. \end{aligned} \tag{7.13}$$

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 \(7.12\)](#) 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.

**MODIFIED MATERIAL:** To model the transient behavior of the error, we follow the approach of [52] and model the error as a Gauss-Markov process. In continuous time, a Gauss-Markov process follows the dynamics given by

$$\dot{\nu} = -k_{\text{GPS}}\nu + \eta_{\text{GPS}},$$

where where  $\nu(t)$  is the error being modeled,  $1/k_{\text{GPS}}$  is the time constant of the process, and  $\eta_{\text{GPS}}(t)$  is modeled as a zero-mean, wide sense stationary, white random process with autocorrelation function  $R(\tau) = \sigma^2\delta(\tau)$ , where  $\delta(t)$  is the impulse distribution. The Gauss-Markov process can be approximated in discrete time using the difference equation

$$\nu[n + 1] = e^{-k_{\text{GPS}}T_s}\nu[n] + w[n] \quad (7.14)$$

where  $\nu[n]$  is the positioning error at sample  $n$ , and where  $w[n]$  is a zero-mean Gaussian random variable with variance  $T_s\sigma^2$ . <sup>1</sup> Figure 7.10 shows results from the Gauss-Markov GPS altitude error model given by [equation \(7.14\)](#). 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

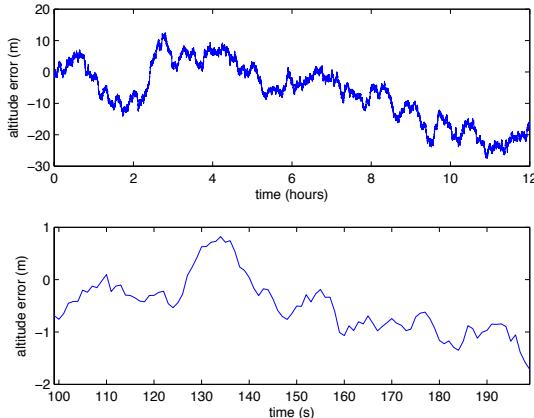
---

<sup>1</sup>Digital implementation will typically call a standard random variable library that implements a normal random variable with zero mean and standard deviation of one. In that case,  $w[n]$  is implemented by multiplying the normal random variable by  $\sqrt{T_s}\sigma$ .

**Table 7.2:** Gauss-Markov error model parameters

Direction	Nominal $1-\sigma$ 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

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

Drawing on the error model in equation (7.14) and the parameters in table 7.2, we can create position error models for the north, east, and altitude measurements from GPS:  $\nu_n$ ,  $\nu_e$ , and  $\nu_h$ . Accordingly, a model for GPS measurements that is suitable for simulation purposes is given by

$$y_{\text{GPS},n}[n] = p_n[n] + \nu_n[n] \quad (7.15)$$

$$y_{\text{GPS},e}[n] = p_e[n] + \nu_e[n] \quad (7.16)$$

$$y_{\text{GPS},h}[n] = -p_d[n] + \nu_h[n], \quad (7.17)$$

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. **MODIFIED MATERIAL:** Horizontal ground speed and course are calculated from the north and east velocity components from GPS as

$$V_g = \sqrt{V_n^2 + V_e^2} \quad (7.18)$$

$$\chi = \text{atan2}(V_e, V_n), \quad (7.19)$$

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

Using basic principles of uncertainty analysis [53], 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

$$\sigma_{V_g} = \sigma_V \quad (7.20)$$

$$\sigma_\chi = \frac{\sigma_V}{V_g}. \quad (7.21)$$

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 (7.18)–(7.19) and (7.20)–(7.21), we can model the ground speed and course measurements available from GPS as

$$y_{\text{GPS}, V_g} = \sqrt{(V_a \cos \psi + w_n)^2 + (V_a \sin \psi + w_e)^2} + \eta_V \quad (7.22)$$

$$y_{\text{GPS}, \chi} = \text{atan2}(V_a \sin \psi + w_e, V_a \cos \psi + w_n) + \eta_\chi, \quad (7.23)$$

where  $\eta_V$  and  $\eta_\chi$  are zero-mean Gaussian processes with variances  $\sigma_{V_g}^2$  and  $\sigma_\chi^2$ .

## 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 [54, 55, 56]. 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 [48, 49, 50, 51]. Specific information for sensor models can be found in manufacturer data sheets for the devices of interest.

## 7.7 DESIGN PROJECT

**MODIFIED MATERIAL:** 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, and add a sensors function to the implementation of the MAV dynamics.
- 7.2 Using the sensor parameters listed in appendix H, modify the sensors implementation to simulate the output of the rate gyros (Eq. (7.4)), the accelerometers (Eq. (7.1)), and the pressure sensors (Eq (7.8) and (7.9)).
- 7.3 Using the sensor parameters listed in appendix H, modify the sensors implementation to simulate the position measurement output of the GPS sensor (Eq. (7.15)–(7.17)) and the ground speed and course output of the GPS sensor (Eq. (7.22)–(7.23)). The GPS signal should be output at a different sample rate that is specific to GPS (e.g. 4 Hz).

- 7.4 Using the sensor viewer, 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. Demonstrate and record the output of your working sensors.

## *Chapter Eight*

---

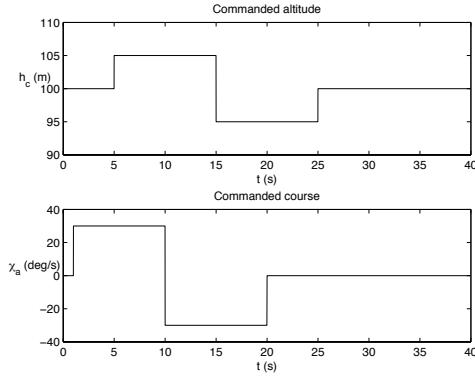
### State Estimation

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 and [subtracting the bias](#). 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 8.4](#) 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 8.5](#). For those with limited exposure to stochastic processes, [appendix G](#) provides an overview of basic concepts from probability theory with a focus on Gaussian stochastic processes. The last [three](#) sections of the chapter describe applications of the Kalman filter. In [section 8.8](#), an extended Kalman filter is designed to estimate the roll and pitch attitude of the MAV, and in [section 8.9](#), 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. [Since the combination of these two filters cannot be used to estimate sensor biases and misalignment errors, in Section 8.10 we derive a full-state extended Kalman filter for estimating all states, biases, and misalignment errors at once.](#)

#### 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 perfor-



**Figure 8.1:** Altitude and heading commands that define the benchmark maneuver used to evaluate and tune the state estimation scheme.

mance of the different estimators developed in this chapter, we can evaluate 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 [first-order](#) low-pass filter. The Laplace transform representation of a [first-order](#) 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 equation (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, and using the notation  $z_k \triangleq z(kT_s)$ , results in the expression

$$\begin{aligned} y_{k+1} &= e^{-aT_s} y_k + a \int_0^{T_s} e^{-a(T_s-\tau)} d\tau u_k \\ &= e^{-aT_s} y_k + (1 - e^{-aT_s}) u_k. \end{aligned} \quad (8.2)$$

If we let  $\alpha_{LPF} = e^{-aT_s} \in [0, 1]$  then we get the simple form

$$y_{k+1} = \alpha_{LPF} y_k + (1 - \alpha_{LPF}) u_k. \quad (8.3)$$

Equation (8.3) is called an alpha-filter, and has a nice physical interpretation: the new filtered value of  $y$  is a weighted average of the old value filtered value of  $y$  and the unfiltered value of  $u$ . If  $u$  is noisy, then  $\alpha_{LPF} \in [0, 1]$  should be close to unity. However, if  $u$  is relatively noise free, then  $\alpha_{LPF} \in [0, 1]$  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

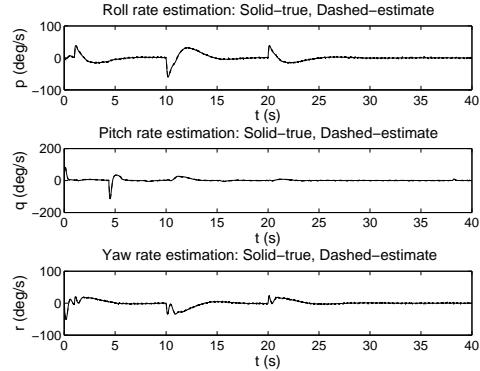
The angular rates  $p$ ,  $q$ , and  $r$  can be estimated by low-pass filtering the rate gyro signals given by equation (7.4) to obtain

$$\hat{p} = LPF(y_{gyro,x}) - \beta_{gyro,x} \quad (8.4)$$

$$\hat{q} = LPF(y_{gyro,y}) - \beta_{gyro,y} \quad (8.5)$$

$$\hat{r} = LPF(y_{gyro,z}) - \beta_{gyro,z} \quad (8.6)$$

where we assume that  $\beta_{gyro,x}$ ,  $\beta_{gyro,y}$ , and  $\beta_{gyro,z}$  have been determined by a calibration process. 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 \(7.8\)](#) and dividing by  $\rho g$  we get

$$\hat{h} = \frac{LPF(y_{\text{static pres}})}{\rho g}. \quad (8.7)$$

### 8.3.3 Airspeed

The airspeed can be estimated by applying a low-pass filter to the differential pressure sensor represented by [equation \(7.9\)](#), and inverting to obtain

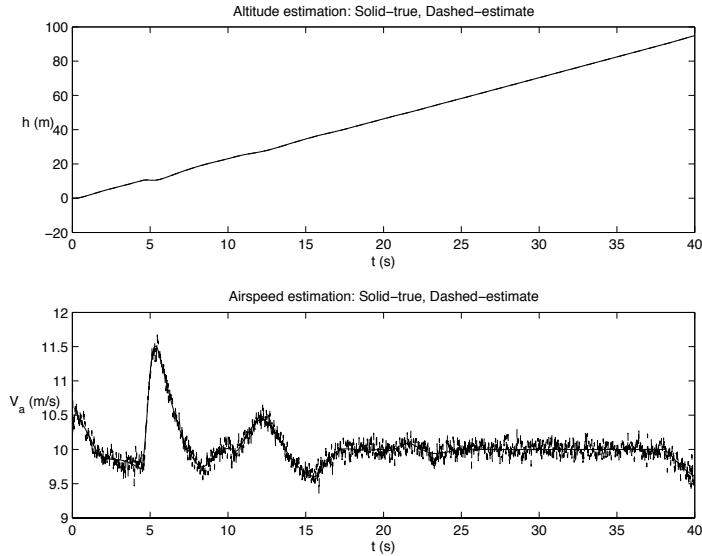
$$\hat{V}_a = \sqrt{\frac{2}{\rho} LPF(y_{\text{diff pres}})}. \quad (8.8)$$

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 flight can be derived as follows. Recall from [equation \(7.2\)](#) that

$$y_{\text{accel,x}} = \dot{u} + qw - rv + g \sin \theta + \eta_{\text{accel,x}}$$



**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.

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

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

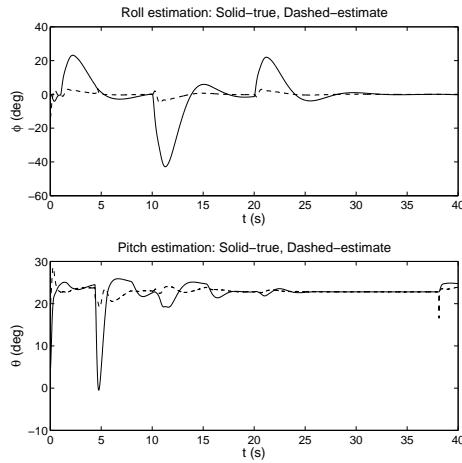
$$\begin{aligned} LPF(y_{\text{accel},x}) &= g \sin \theta \\ LPF(y_{\text{accel},y}) &= -g \cos \theta \sin \phi \\ LPF(y_{\text{accel},z}) &= -g \cos \theta \cos \phi. \end{aligned}$$

Solving for  $\phi$  and  $\theta$  we get

$$\hat{\phi}_{\text{accel}} = \tan^{-1} \left( \frac{LPF(y_{\text{accel},y})}{LPF(y_{\text{accel},z})} \right) \quad (8.9)$$

$$\hat{\theta}_{\text{accel}} = \sin^{-1} \left( \frac{LPF(y_{\text{accel},x})}{g} \right). \quad (8.10)$$

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 sections 8.8 and 8.10 we will use the 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](#) (7.15) and (7.16). The biases due to multipath, clock, and satellite geometry will not be removed. The estimate of the position variables is therefore given by

$$\hat{p}_n = LPF(y_{GPS,n}) \quad (8.11)$$

$$\hat{p}_e = LPF(y_{GPS,e}). \quad (8.12)$$

Similarly, an estimate of the course angle and ground speed of the MAV can be obtained by low-pass filtering [equations](#) (7.23) and (7.22) to obtain

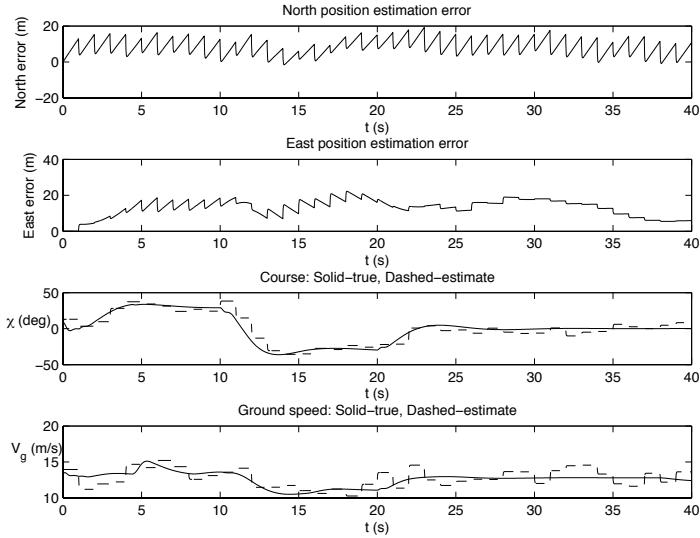
$$\hat{\chi} = LPF(y_{GPS,\chi}) \quad (8.13)$$

$$\hat{V}_g = LPF(y_{GPS,V_g}). \quad (8.14)$$

The primary downside of low-pass filtering GPS signals is that since the sample rate is slow (usually on the order of [1-5 Hz](#)), there is significant delay in the estimate. [The extended Kalman filters described in sections 8.9 and 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 8.5](#). The application of the Kalman filter to attitude estimation is in [section 8.8](#), the application of the Kalman filter to position, course, and ground speed estimation is in [section 8.9](#), and a full state Kalman filter is derived in [section 8.10](#).

#### 8.4 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

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

$$y = Cx.$$

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

$$\dot{\hat{x}} = \underbrace{A\hat{x} + Bu}_{\text{copy of the model}} + \underbrace{L(y - C\hat{x})}_{\text{correction due to sensor reading}}, \quad (8.15)$$

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 \(8.15\)](#) to account for sampled sensor readings? One approach is to propagate the system model between samples using the prediction equation

$$\dot{\hat{x}} = A\hat{x} + Bu, \quad (8.16)$$

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

$$\begin{aligned} \hat{y}^- &= C\hat{x}^- \\ \hat{x}^+ &= \hat{x}^- + L(y(t_k) - \hat{y}^-), \end{aligned} \quad (8.17)$$

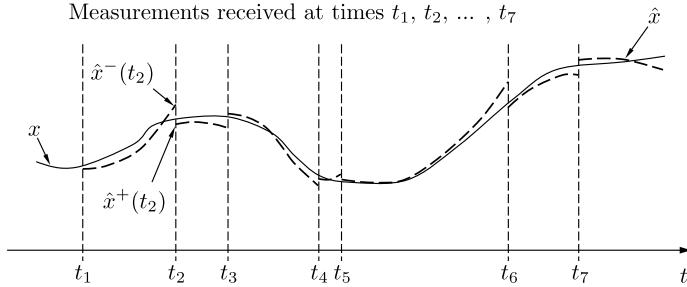
where  $t_k$  is the instant in time that the measurement is received and  $\hat{x}^-$  is the state estimate produced by [equation \(8.16\)](#) at time  $t_k$ . Equation (8.16) is then re-instantiated with initial conditions given by  $\hat{x}^+$ . If the system is nonlinear, then the propagation and measurement update equations become

$$\dot{\hat{x}} = f(\hat{x}, u) \quad (8.18)$$

$$\begin{aligned} \hat{y}^- &= h(\hat{x}^-) \\ \hat{x}^+ &= \hat{x}^- + L(y(t_k) - \hat{y}^-). \end{aligned} \quad (8.19)$$

The [estimation](#) process is shown graphically in [figure 8.6](#). Note that it is not necessary to have a fixed sample rate. The continuous-discrete observer can be implemented using [algorithm 2](#).

A pseudo-code implementation of the resulting continuous-discrete observer is shown in [algorithm 2](#). In [line 1](#) the state estimate is initialized to



**Figure 8.6:** 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 (8.18). When a measurement is received, the state is updated using equation (8.19).

zero. If additional information is known, then the state can be initialized accordingly. The ODE in equation (8.18) is propagated between samples with the for-loop in lines 4–6 using an Euler integration method. When a measurement is received, the state is updated using equation (8.19) in lines 8–9.

#### MODIFIED MATERIAL:

---

#### 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**
  - 5:      $\hat{x} = \hat{x} + \left(\frac{T_{out}}{N}\right) f(\hat{x}, u)$                        $\triangleright$  Propagate the state equation.
  - 6: **end for**
  - 7: **if** A measurement has been received from sensor  $i$  **then**
  - 8:      $\hat{y} = h_i(\hat{x})$
  - 9:      $\hat{x} = \hat{x} + L_i (y_i - \hat{y}^-)$                                $\triangleright$  Measurement Update
  - 10: **end if**
- 

## 8.5 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

$$\begin{aligned}\dot{x} &= Ax + B(u + \xi_u) + \xi \\ y_k &= Cx_k + \eta_k,\end{aligned}\tag{8.20}$$

where  $y_k = y(t_k)$  is the  $k^{\text{th}}$  sample of  $y$ ,  $x_k = x(t_k)$  is the  $k^{\text{th}}$  sample of  $x$ ,  $\eta_k$  is the measurement noise at time  $t_k$ ,  $\xi$  is a zero-mean Gaussian random process with covariance  $E\{\xi(t)\xi(\tau)\} = Q\delta(t, \tau)$  where  $\delta(t, \tau)$  is the Dirac delta function,  $\xi_u$  is a zero-mean Gaussian random process with covariance  $E\{\xi_u(t)\xi_u(\tau)\} = Q_u\delta(t, \tau)$ , and  $\eta_k$  is a zero-mean Gaussian random variable with covariance  $R$ . The random process  $\xi$  is called the process noise and represents modeling error and disturbances on the system. The random variable  $\eta$  is called the measurement noise and represents noise on the sensors. The random process  $\xi_u$  represents noise on the input signal  $u$ . For UAV and other robotic applications that derive the observer using IMU mechanization, the input  $u$  is itself derived from noisy measurements. The covariances  $R$  and  $Q_u$  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 (8.16) and (8.17) the continuous-discrete Kalman filter has the form

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

Define the estimation error as  $\tilde{x} = x - \hat{x}$ . Assuming that  $E\{\tilde{x}(t)\} = 0$ , the covariance of the estimation error at time  $t$  is given by

$$P(t) \triangleq E\{\tilde{x}(t)\tilde{x}(t)^\top\}.\tag{8.21}$$

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  $\lambda_i$  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  $\text{tr}(P)$ .

*Between Measurements:*

MODIFIED MATERIAL: Differentiating  $\tilde{x}$  we get

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

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

$$\tilde{x}(t) = e^{At}\tilde{x}_0 + \int_0^t e^{A(t-\tau)}(B\xi_u(\tau) + \xi(\tau)) d\tau. \quad (8.22)$$

We can compute the evolution of the error covariance  $P$  as

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

We can compute  $E\{\tilde{x}(\xi_u^\top B^\top + \xi^\top)\}$  as

$$\begin{aligned}E\{\tilde{x}(\xi_u^\top B^\top + \xi^\top)\} &= E\left\{e^{At}\tilde{x}_0(\xi_u^\top(t)B^\top + \xi^\top(t))\right. \\ &\quad \left.+ \int_0^t e^{A(t-\tau)}(B\xi_u(\tau) + \xi(\tau))(\xi_u^\top(\tau)B^\top + \xi^\top(\tau)) d\tau\right\} \\ &= \int_0^t e^{A(t-\tau)}(BQ_uB^\top + Q)\delta(t-\tau) d\tau \\ &= \frac{1}{2}(BQ_uB^\top + Q),\end{aligned}$$

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

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

To solve this differential equation numerically, Euler's method could be used to give

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

However, given the numerical inaccuracy due to the Euler approximation,  $P$  may not remain positive definite due to the  $AP_k + P_k A^\top$  term, which is not guaranteed to be positive definite. We can solve this problem by discretizing in a different way. In Equation (8.22), let  $t = kT_s$ , 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  $\xi_{u,k} = \xi_u(kT_s)$ , and  $\xi_k = \xi(kT_s)$ , we get

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

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

$$\tilde{x}_{k+1} = A_d \tilde{x}_k + T_s (B \xi_{u,k} + \xi_k).$$

Defining the error covariance as

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

the covariance update can be approximated as

$$\begin{aligned} P_{k+1} &= E\{\tilde{x}_{k+1} \tilde{x}_{k+1}^\top\} \\ &= E\{(A_d \tilde{x}_k + T_s (B \xi_{u,k} + \xi_k))(A_d \tilde{x}_k + T_s (B \xi_{u,k} + \xi_k))^\top\} \\ &= E\{A_d \tilde{x}_k \tilde{x}_k^\top A_d^\top + T_s (B \xi_{u,k} + \xi_k) \tilde{x}_k^\top A_d^\top + T_s A_d \tilde{x}_k (B \xi_{u,k} + \xi_k)^\top \\ &\quad + T_s^2 (B \xi_{u,k} + \xi_k) (B \xi_{u,k} + \xi_k)^\top\} \\ &= A_d E\{\tilde{x}_k \tilde{x}_k^\top\} A_d^\top + T_s (B E\{\xi_{u,k} \tilde{x}_k^\top\} + E\{\xi_k \tilde{x}_k^\top\}) A_d^\top \\ &\quad + T_s A_d (E\{\tilde{x}_k \xi_{u,k}^\top\} B^\top + E\{\tilde{x}_k \xi_k^\top\}) \\ &\quad + T_s^2 (B E\{\xi_{u,k} \xi_{u,k}^\top\} B^\top + E\{\xi_k \xi_k^\top\}) \\ &= A_d P_k A_d^\top + T_s^2 (B Q_u B^\top + Q), \end{aligned}$$

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

$$P_{k+1} = A_d P_k A_d^\top + T_s^2 (B Q_u B^\top + Q)$$

ensures that the error covariance remains symmetric and positive definite.

*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

$$\begin{aligned}P^+ &= E\{\tilde{x}^+\tilde{x}^{+T}\} \\ &= E\left\{(\tilde{x}^- - LC\tilde{x}^- - L\eta)(\tilde{x}^- - LC\tilde{x}^- - L\eta)^T\right\} \\ &= E\left\{\tilde{x}^-\tilde{x}^{-T} - \tilde{x}^-\tilde{x}^{-T}C^T L^T - \tilde{x}^-\eta^T L^T\right. \\ &\quad \left.- LC\tilde{x}^-\tilde{x}^{-T} + LC\tilde{x}^-\tilde{x}^{-T}C^T L^T + LC\tilde{x}^-\eta^T L^T\right. \\ &\quad \left.- L\eta\tilde{x}^{-T} + L\eta\tilde{x}^{-T}C^T L^T + L\eta\eta^T L^T\right\} \\ &= P^- - P^-C^T L^T - LCP^- + LCP^-C^T L^T + LRL^T, \quad (8.23)\end{aligned}$$

$$= (\mathbf{I} - LC)P^-(\mathbf{I} - LC)^T + LRL^T, \quad (8.24)$$

where we have used the fact that since  $\eta$  and  $\tilde{x}^-$  are independent, i.e.,

$$E\left\{\tilde{x}^-\eta^T L^T\right\} = E\left\{L\eta\tilde{x}^{-T}\right\} = 0.$$

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

$$\frac{\partial}{\partial A} \text{tr}(BAD) = B^T D^T \quad (8.25)$$

$$\frac{\partial}{\partial A} \text{tr}(ABA^T) = 2AB, \text{ if } B = B^T. \quad (8.26)$$

Our objective is to pick  $L$  to minimize  $\text{tr}(P^+)$ . From equation (8.23), 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 (8.23) and simplifying gives the standard form

$$P^+ = (I - LC)P^-,$$

however from a practical point of view, it is better to use equation (8.24), which is called the Joseph's stabilized form, to ensure that  $P^+$  is symmetric positive definite.

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

$$\begin{aligned}\dot{\hat{x}} &= A\hat{x} + Bu \\ A_d &= e^{AT_s} \approx I + AT_s + A^2 \frac{T_s^2}{2} \\ P_{k+1} &= A_d P_k A_d^\top + T_s^2 (BQ_u B^\top + Q)\end{aligned}$$

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

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

where  $L_i$  is called the Kalman gain for sensor  $i$ .

To this point, we have assumed that the system propagation model 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 (8.20) becomes

$$\dot{x} = f(x, u + \xi_u) + \xi \quad (8.27)$$

$$y_k = h(x_k) + \eta_k. \quad (8.28)$$

For this case, the state propagation and update laws use the nonlinear model, but the propagation and update of the error covariance use the Jacobians of  $f$  for  $A$  and  $B$ , 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 4–10. The update equations for the  $i^{th}$  sensor are given in lines 11–17. The application of algorithm 3 to roll and pitch angle estimation is

described in section 8.8. The application of algorithm 3 to position, heading, ground speed, course, and wind estimation is described in section 8.9. The application of algorithm 3 to estimate all states, biases, and misalignment errors is described in section 8.10.

---

**Algorithm 3** Continuous-discrete Extended Kalman Filter

---

```

1: Initialize:  $\hat{x} = 0$ .
2: Pick an output sample rate  $T_{out}$  that is much less than the sample rates
   of the sensors, and let  $T_p = T_{out}/N$ 
3: At each sample time  $T_{out}$ :
4: for  $i = 1$  to  $N$  do                                 $\triangleright$  Prediction Step
5:    $\hat{x} \leftarrow \hat{x} + T_p f(\hat{x}, u)$ 
6:    $A \leftarrow \frac{\partial f}{\partial x}(\hat{x}, u)$ 
7:    $B \leftarrow \frac{\partial f}{\partial u}(\hat{x}, u)$ 
8:    $A_d \leftarrow I + AT_p + A^2T_p^2$ 
9:    $P \leftarrow A_d P A_d^\top + T_p^2 (B Q_u B^\top + Q)$ 
10: end for                                          $\triangleright$  Measurement Update
11: if Measurement  $y_i$  is received from sensor  $i$  then
12:    $\hat{y}_i \leftarrow h_i(\hat{x}^-)$ 
13:    $C_i \leftarrow \frac{\partial h_i}{\partial x}(\hat{x}^-)$ 
14:    $S_i \leftarrow R_i + C_i P C_i^\top$ 
15:    $L_i \leftarrow P C_i^\top S_i^{-1}$ 
16:    $\hat{x} \leftarrow \hat{x} + L_i (y_i - \hat{y}_i)$ 
17:    $P \leftarrow (I - L_i C_i) P (I - L_i C_i)^\top + L_i R_i L_i^\top$ 
18: end if

```

---

## 8.6 CONSTRAINTS AND PSEUDO-MEASUREMENTS

**NEW MATERIAL:** The extended Kalman filter assumes that the state variables evolve according to the dynamic model given in Equation (8.27), and that the dynamic model represents the only constraints on the system. It is often the case that there are additional constraints on the state variables. For example, if the state  $x$  contained the four elements of a unit quaternion, then those four elements would also satisfy a unit length constraint. As another example, in Section 8.9 we will estimate the groundspeed, course, heading, and the north-east components of the wind. These variables are constrained by the wind triangle, which is not represented in the dynamic equations of motion. Constraints can be included in an EKF through the use of so-called *pseudo-measurements*.

Suppose that the states  $x$  that satisfy Equation (8.27) also satisfy the constraint

$$\lambda(x) = 0.$$

The basic idea of a pseudo-measurement is to pretend that there is an additional sensor that measures the constraint. If such a sensor were to exist, then it would always measure

$$y_{pseudo} = \lambda(x) = 0.$$

Therefore, the nonlinear measurement given in Equation (8.28) is augmented as

$$y'_k = \begin{pmatrix} y_{real} \\ y_{pseudo} \end{pmatrix} = \begin{pmatrix} h(x_k) \\ \lambda(x_k) \end{pmatrix} + \begin{pmatrix} \eta_k \\ \eta_k^\lambda \end{pmatrix}, \quad (8.29)$$

where  $y_{real}$  are the real measurements, and where  $\eta_k^\lambda$  represents an artificial noise term on the pseudo-measurements where the covariance of  $\eta_k^\lambda$  represents confidence that the constraint is satisfied, and can be used as an additional tuning parameter.

We will demonstrate the use of pseudo-measurements in the Examples given in Sections 8.9 and 8.10.

## 8.7 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

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

and note that since  $\tilde{x}_k^-$  and  $\eta_k$  are zero mean, it is implied that  $v_k$  is zero mean. Therefore, the covariance of the innovation sequence  $v_k$  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  $\eta_k$  and  $\tilde{x}_k^-$  are uncorrelated.

The random variable

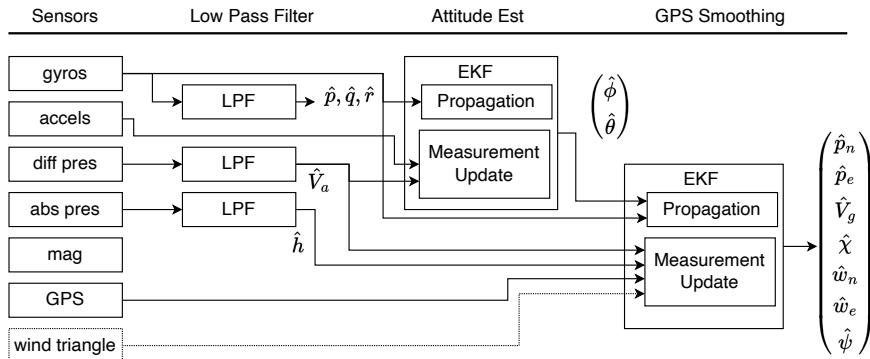
$$z_k = S_k^{-\frac{1}{2}} v_k$$

is therefore an  $m$ -dimensional Gaussian random variable with zero mean and covariance of  $I$ . The random variable  $w_k = z_k^\top z_k = \nu_k^\top S^{-1} \nu_k$  is a  $\chi^2$  (chi-squared) random variable with  $m$  degrees-of-freedom. The idea of measurement gating is that when measurement  $y_k$  is received, the probability that  $w_k = (y_k - \hat{y}_k)^\top S^{-1} (y - \hat{y}_k)$  comes from a chi-squared distribution with  $m$ -degrees-of-freedom is computed, and the measurement update is performed only if the probability is above a user-specified threshold.

## 8.8 ATTITUDE ESTIMATION

### MODIFIED MATERIAL:

In this and the next section we will develop two cascaded extended Kalman filters that can be used, together with low-pass filtering, to estimate the state of the MAV. The two stage approach is shown in Figure 8.7. The EKF that estimates attitude will be derived in this section, and the EKF that estimates the other states will be derived in Section 8.9. In Section 8.10 we will derive a full state EKF that estimates all of the states, plus the biases in one filter.



**Figure 8.7:** Block diagram for the two-stage extended Kalman filter. In the first stage, an EKF is used to estimate the roll and pitch angles from rate gyros and accelerometers. In the second stage, GPS is used to estimate position, groundspeed, course, wind, and heading.

To apply the continuous-discrete extended Kalman filter derived in section 8.5 to roll and pitch estimation, we use the nonlinear propagation model

in Equations (5.7) and (5.8):

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u} + \xi_{\mathbf{u}}) + \xi$$

where  $\mathbf{x} \triangleq (\phi, \theta)^T$ ,  $\mathbf{u} \triangleq (p, q, r)^T$ ,  $\xi_{\mathbf{u}} = (\xi_p, \xi_q, \xi_r)^T$ , and  $\xi = (\xi_\phi, \xi_\theta)^T$ , and where

$$f(\mathbf{x}, \mathbf{u}) \triangleq \begin{pmatrix} p + q \sin \phi \tan \theta + r \cos \phi \tan \theta \\ q \cos \phi - r \sin \phi \end{pmatrix},$$

and where we have added the noise terms  $\xi_p$ ,  $\xi_q$  and  $\xi_r$  to model the noise on the gyro measurements  $p$ ,  $q$ , and  $r$  with covariance  $Q_{\mathbf{u}}$ , and the noise terms  $\xi_\phi$  and  $\xi_\theta$  represent model uncertainty with covariance  $Q$ .

In this case, the Jacobians of  $f$  can be calculated as

$$\begin{aligned} \frac{\partial f}{\partial \mathbf{x}} &= \begin{pmatrix} q \cos \phi \tan \theta - r \sin \phi \tan \theta, & \frac{q \sin \phi + r \cos \phi}{\cos^2 \theta} \\ -q \sin \phi - r \cos \phi, & 0 \end{pmatrix} \\ \frac{\partial f}{\partial \mathbf{u}} &= \begin{pmatrix} 1 & \sin \phi \tan \theta & \cos \phi \tan \theta \\ 0 & \cos \phi & -\sin \phi \end{pmatrix}. \end{aligned}$$

For the measurement update, we will use the accelerometer, which from equation (7.2) we have

$$y_{\text{accel}} = \begin{pmatrix} \dot{u} + qw - rv + g \sin \theta \\ \dot{v} + ru - pw - g \cos \theta \sin \phi \\ \dot{w} + pv - qu - g \cos \theta \cos \phi \end{pmatrix} + \eta_{\text{accel}}. \quad (8.30)$$

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 (2.6) we have

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

where the last expression comes by assuming that  $\alpha \approx \theta$  and  $\beta \approx 0$ . Substituting into equation (8.30), we get

$$\begin{aligned} y_{\text{accel}} &= h(\mathbf{x}, \mathbf{u}) + \eta_{\text{accel}} \\ &\triangleq \begin{pmatrix} qV_a \sin \theta + g \sin \theta \\ rV_a \cos \theta - pV_a \sin \theta - g \cos \theta \sin \phi \\ -qV_a \cos \theta - g \cos \theta \cos \phi \end{pmatrix} + \eta_{\text{accel}}, \end{aligned}$$

where  $\eta_{\text{accel}} = (\eta_x, \eta_y, \eta_z)$  is the measurement noise on the accelerometer with covariance  $R_{\text{accel}}$ . The Jacobian of the output function is given by

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

In this case, where the measurement equation also contains noisy measurement inputs  $\bar{\mathbf{u}} = (p, q, r, V_a)$  with noise  $\xi_{\bar{\mathbf{u}}}$ , a more accurate model of the output noise is modeled as

$$\eta_{out} = \eta_{accel} + \frac{\partial h}{\partial \bar{\mathbf{u}}} \xi_{\bar{\mathbf{u}}}$$

with covariance

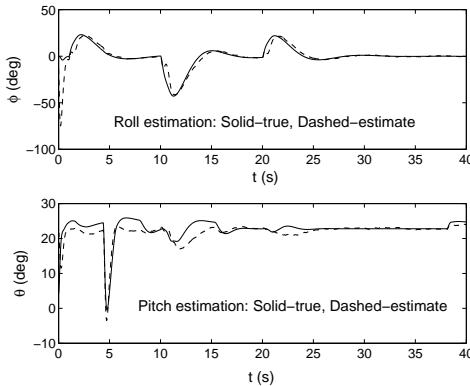
$$R_{out} = R_{accel} + \frac{\partial h}{\partial \bar{\mathbf{u}}} Q_{\bar{\mathbf{u}}} \frac{\partial h^T}{\partial \bar{\mathbf{u}}},$$

where

$$\frac{\partial h}{\partial \bar{\mathbf{u}}} = \begin{pmatrix} 1 & V_a \sin \theta & 0 & q \sin \theta \\ -V_a \sin \theta & 0 & V_a \cos \theta & (r \cos \theta - q \sin \theta) \\ 0 & -V_a \cos \theta & 0 & -q \cos \theta \end{pmatrix}.$$

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 8.8. Comparing figure 8.8 with figure 8.4 shows that the continuous-discrete extended Kalman filter produces much better results during accelerated flight.



**Figure 8.8:** Estimation error on the roll and pitch angles using continuous-discrete extended Kalman filter.

## 8.9 GPS SMOOTHING

**MODIFIED MATERIAL:** This section describes the second extended Kalman filter labeled *GPS Smoothing* in Figure 8.7. This filter uses 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 (7.18) we get that the evolution of the ground speed is given by

$$\begin{aligned}\dot{V}_g &= \frac{d}{dt} \sqrt{(V_a \cos \psi + w_n)^2 + (V_a \sin \psi + w_e)^2} \\ &= \frac{1}{V_g} \left[ (V_a \cos \psi + w_n)(\dot{V}_a \cos \psi - V_a \dot{\psi} \sin \psi + \dot{w}_n) \right. \\ &\quad \left. + (V_a \sin \psi + w_e)(\dot{V}_a \sin \psi + V_a \dot{\psi} \cos \psi + \dot{w}_e) \right].\end{aligned}$$

Assuming that wind and airspeed are constant we get

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

From equation (5.15) the evolution of  $\chi$  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 (5.9) the evolution of  $\psi$  is given by

$$\dot{\psi} = q \frac{\sin \phi}{\cos \theta} + r \frac{\cos \phi}{\cos \theta}. \quad (8.31)$$

Defining the state as  $\mathbf{x} = (p_n, p_e, V_g, \chi, w_n, w_e, \psi)^\top$ , and the input as  $\mathbf{u} = (V_a, q, r, \phi, \theta)^\top$ , the nonlinear propagation model is given by

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u} + \xi_{\mathbf{u}}) + \xi,$$

where

$$f(\mathbf{x}, \mathbf{u}) \triangleq \begin{pmatrix} V_g \cos \chi \\ V_g \sin \chi \\ \frac{V_a \dot{\psi}(w_e \cos \psi - w_n \sin \psi)}{V_g} \\ \frac{g}{V_g} \tan \phi \\ 0 \\ 0 \\ q \frac{\sin \phi}{\cos \theta} + r \frac{\cos \phi}{\cos \theta} \end{pmatrix},$$

and where  $\xi_{\mathbf{u}} = (\xi_{V_a}, \xi_q, \xi_r, \xi_\phi, \xi_\theta)$  model the noise on the input variables from the first stages of Figure 8.7 with covariance  $Q_{\mathbf{u}}$ , and where  $\xi$  represents model uncertain with covariance  $Q$ . The Jacobians of  $f$  can be calculated as

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

$$\frac{\partial f}{\partial \mathbf{u}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{\partial \dot{V}_g}{\partial V_a} & \frac{\partial \dot{V}_g}{\partial q} & \frac{\partial \dot{V}_g}{\partial r} & \frac{\partial \dot{V}_g}{\partial \phi} & \frac{\partial \dot{V}_g}{\partial \theta} \\ 0 & 0 & 0 & \frac{g}{V_g} \frac{1}{1+\phi^2} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\sin \phi}{\cos \theta} & \frac{\cos \phi}{\cos \theta} & \frac{\partial \dot{\psi}}{\partial \phi} & \frac{\partial \dot{\psi}}{\partial \theta} \end{pmatrix}$$

where  $\dot{\psi}$  is given in equation (8.31) and

$$\begin{aligned} \frac{\partial \dot{V}_g}{\partial \psi} &= \frac{-\dot{\psi} V_a (w_n \cos \psi + w_e \sin \psi)}{V_g} \\ \frac{\partial \dot{V}_g}{\partial V_a} &= \frac{\dot{\psi} (w_e \cos \psi - w_n \sin \psi) \sin \phi}{V_g \cos \theta} \\ \frac{\partial \dot{V}_g}{\partial q} &= \frac{V_a (w_e \cos \psi - w_n \sin \psi) \sin \phi}{V_g \cos \theta} \\ \frac{\partial \dot{V}_g}{\partial r} &= \frac{V_a (w_e \cos \psi - w_n \sin \psi) \cos \phi}{V_g \cos \theta} \end{aligned}$$

$$\begin{aligned}\frac{\partial \dot{V}_g}{\partial \phi} &= \frac{V_a(w_e \cos \psi - w_n \sin \psi)(q \cos \phi - r \sin \phi)}{V_g \cos \theta} \\ \frac{\partial \dot{V}_g}{\partial \theta} &= \frac{V_a(w_e \cos \psi - w_n \sin \psi)(q \sin \phi + r \cos \phi)(-\sin \theta)}{V_g \cos^2 \theta} \\ \frac{\partial \dot{\psi}}{\partial \phi} &= \frac{q \cos \phi - r \sin \phi}{\cos \theta} \\ \frac{\partial \dot{\psi}}{\partial \theta} &= \frac{-(q \sin \phi + r \cos \phi) \sin \theta}{\cos^2 \theta}.\end{aligned}$$

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 (2.8) as a pseudo-measurement. 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

$$y_{\text{GPS}} = h(\mathbf{x}, \mathbf{u}) + \eta_{\text{GPS}},$$

where  $y_{\text{GPS}} = (y_{\text{GPS},n}, y_{\text{GPS},e}, y_{\text{GPS},V_g}, y_{\text{GPS},\chi}, y_{\text{wind},n}, y_{\text{wind},e})^\top$ , and

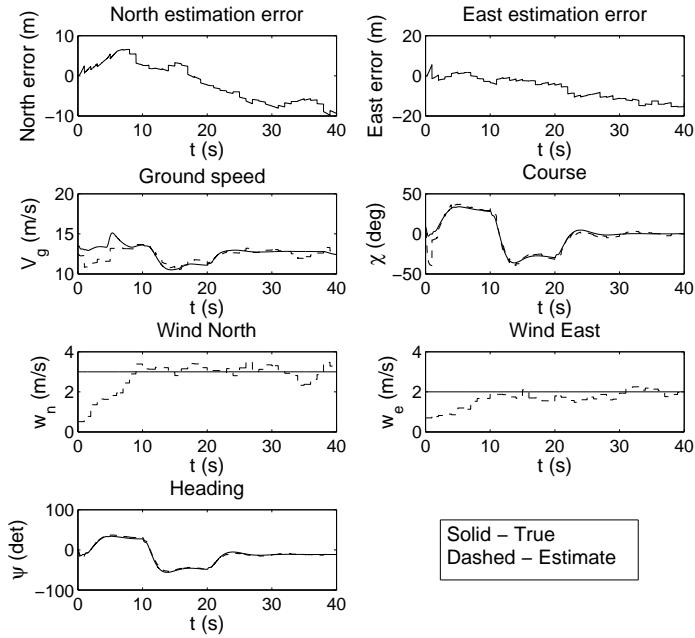
$$h(\mathbf{x}, \mathbf{u}) = \begin{pmatrix} p_n \\ p_e \\ V_g \\ \chi \\ V_a \cos \psi + w_n - V_g \cos \chi \\ V_a \sin \psi + w_e - V_g \sin \chi \end{pmatrix},$$

and where the Jacobian is given by

$$\frac{\partial h}{\partial \mathbf{x}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -\cos \chi & V_g \sin \chi & 1 & 0 & -V_a \sin \psi \\ 0 & 0 & -\sin \chi & -V_g \cos \chi & 0 & 1 & V_a \cos \psi \end{pmatrix}.$$

The extended Kalman filter to estimate  $p_n$ ,  $p_e$ ,  $V_g$ ,  $\chi$ ,  $w_n$ ,  $w_e$ , and  $\psi$  is implemented using algorithm 3.

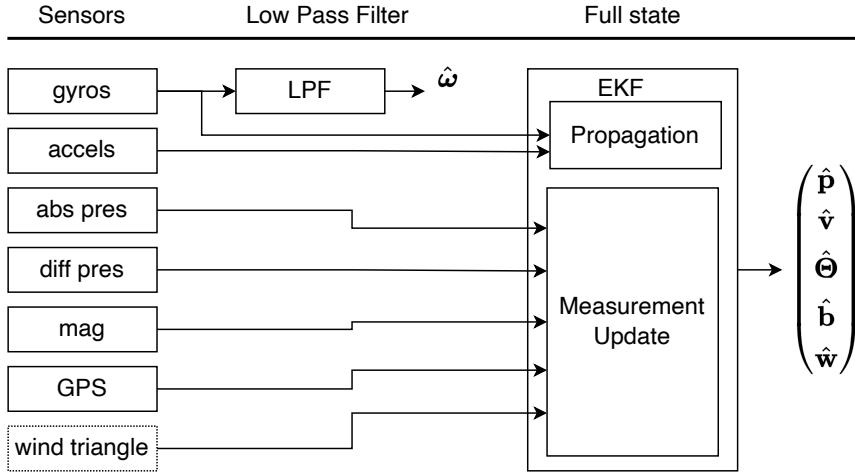
For the benchmark maneuver discussed in section 8.1, the estimation error for the position and heading using algorithm 3 is shown in figure 8.9. Comparing figure 8.9 with figure 8.5, shows that the continuous-discrete extended Kalman filter produces much better results than a simple low-pass filter.



**Figure 8.9:** Estimation error for position, ground speed, course, wind, and heading using continuous-discrete extended Kalman filter.

## 8.10 FULL STATE EKF

**NEW MATERIAL:** In this section we will expand upon the previous discussion to include the full aircraft state. The block diagram for the full state EKF is shown in Figure 8.11, which can be contrasted with Figure 8.7.



**Figure 8.10:** Block diagram for the full state extended Kalman filter.

### 8.10.1 Aircraft and sensor models

In this section we will use a more compact vector notation in an attempt to make the presentation of the full state EKF more clear. Define the position vector as  $\mathbf{p} = (p_n, p_e, p_d)^\top$ , the velocity vector as  $\mathbf{v} = (u, v, w)^\top$ , the attitude vector as  $\Theta = (\phi, \theta, \psi)^\top$ , the angular velocity vector as  $\boldsymbol{\omega} = (p, q, r)^\top$ , the rate gyro bias vector as  $\mathbf{b}_g = (b_{gx}, b_{gy}, b_{gz})^\top$ , the accelerometer bias vector as  $\mathbf{b}_a = (b_{ax}, b_{ay}, b_{az})^\top$ , and the wind vector as  $\mathbf{w} = (w_n, w_e)^\top$ . Then we can write the dynamics as

$$\begin{aligned}\dot{\mathbf{p}} &= R(\Theta)\mathbf{v} \\ \dot{\mathbf{v}} &= \mathbf{v}^\times \boldsymbol{\omega} + \mathbf{a} + R^\top(\Theta)\mathbf{g} \\ \dot{\Theta} &= S(\Theta)\boldsymbol{\omega} \\ \dot{\mathbf{b}}_g &= 0_{3 \times 1} \\ \dot{\mathbf{b}}_a &= 0_{3 \times 1} \\ \dot{\mathbf{w}} &= 0_{2 \times 1},\end{aligned}\tag{8.32}$$

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

$$R(\Theta) = \begin{pmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{pmatrix}$$

$$S(\Theta) = \begin{pmatrix} 1 & \sin \phi \tan \theta & \cos \phi \tan \theta \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi \sec \theta & \cos \phi \sec \theta \end{pmatrix},$$

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 magnetometer, the GPS north and east measurements, and the GPS ground-speed and course measurement. We will also use the wind triangle as a pseudo-measurement to constrain the relationship between airspeed, groundspeed, course, heading, and wind.

### 8.10.2 Propagation model for the EKF

The measurement models for the gyroscope and accelerometer are given by

$$\mathbf{y}_{\text{accel}} = \mathbf{a} + \mathbf{b}_g + \boldsymbol{\eta}_{\text{accel}}$$

$$\mathbf{y}_{\text{gyro}} = \boldsymbol{\omega} + \mathbf{b}_g + \boldsymbol{\eta}_{\text{gyro}}$$

Using these models, the equations of motion in Equation (8.32) can be written as

$$\dot{\mathbf{p}} = R(\Theta) \mathbf{v}$$

$$\dot{\mathbf{v}} = \mathbf{v}^\times (\mathbf{y}_{\text{gyro}} - \mathbf{b}_g - \boldsymbol{\eta}_{\text{gyro}}) + (\mathbf{y}_{\text{accel}} - \mathbf{b}_a - \boldsymbol{\eta}_{\text{accel}}) + R^\top(\Theta) \mathbf{g}$$

$$\dot{\Theta} = S(\Theta) (\mathbf{y}_{\text{gyro}} - \mathbf{b}_g - \boldsymbol{\eta}_{\text{gyro}})$$

$$\dot{\mathbf{b}}_g = 0_{3 \times 1}$$

$$\dot{\mathbf{b}}_a = 0_{3 \times 1}$$

$$\dot{\mathbf{w}} = 0_{2 \times 1}.$$

Defining

$$\mathbf{x} = (\mathbf{p}^\top, \mathbf{v}^\top, \Theta^\top, \mathbf{b}_g^\top, \mathbf{b}_a^\top, \mathbf{w}^\top)^\top$$

$$\mathbf{u} = (\mathbf{y}_{\text{accel}}^\top, \mathbf{y}_{\text{gyro}}^\top)^\top,$$

we get

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u} + \boldsymbol{\xi}_u) + \boldsymbol{\xi},$$

where  $\boldsymbol{\xi}_u = (\boldsymbol{\eta}_{\text{accel}}^\top, \boldsymbol{\eta}_{\text{gyro}}^\top)^\top$  is the measurement noise on the accelerometers and rate gyros with covariance

$$Q_u = \text{diag}(\sigma_{\text{accel},x}^2, \sigma_{\text{accel},y}^2, \sigma_{\text{accel},z}^2, \sigma_{\text{gyro},x}^2, \sigma_{\text{gyro},y}^2, \sigma_{\text{gyro},z}^2),$$

$\boldsymbol{\xi}$  is the process noise with covariance

$$Q = \text{diag}([\sigma_{p_n}^2, \sigma_{p_e}^2, \sigma_{p_d}^2, \sigma_u^2, \sigma_v^2, \sigma_w^2, \sigma_\phi^2, \sigma_\theta^2, \sigma_\psi^2, \sigma_{b_x}^2, \sigma_{b_y}^2, \sigma_{b_z}^2, \sigma_{w_n}^2, \sigma_{w_e}^2])$$

and represents modeling errors, and where

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

The Jacobians of  $f$  are given by

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

$$\frac{\partial f}{\partial \mathbf{u}} = \begin{pmatrix} 0_{3 \times 3} & 0_{3 \times 3} \\ -I_{3 \times 3} & -\mathbf{v}^\times \\ 0_{3 \times 3} & -S(\Theta) \\ 0_{3 \times 3} & 0_{3 \times 3} \\ 0_{3 \times 3} & 0_{3 \times 3} \\ 0_{2 \times 3} & 0_{2 \times 3} \end{pmatrix}.$$

It is straightforward to show that when  $\mathbf{z} \in \mathbb{R}^3$  and  $A(\mathbf{z})$  is a  $3 \times 3$  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

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

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

### 8.10.3 Sensor Update Equations

#### *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

$$\frac{\partial h_{\text{static}}}{\partial \mathbf{x}} = ((0, 0, -\rho g), \quad 0_{1 \times 3}, \quad 0_{1 \times 3}, \quad 0_{1 \times 3}, \quad 0_{1 \times 3}, \quad 0_{1 \times 2}).$$

#### *Differential Pressure Sensor*

The model for the differential pressure sensor is

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

where

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

Therefore

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

The associated Jacobian is given by

$$\frac{\partial h_{\text{diff}}}{\partial \mathbf{x}} = \rho \begin{pmatrix} 0_{3 \times 1} \\ \mathbf{v} - R^T(\Theta)\mathbf{w} \\ -\frac{\partial [R^T(\Theta)\mathbf{w}]}{\partial \Theta}^\top (\mathbf{v} - R^T(\Theta)\mathbf{w}) \\ 0_{3 \times 1} \\ 0_{3 \times 1} \\ - (I_{2 \times 2}, \quad 0_{2 \times 1}) R(\Theta) (\mathbf{v} - R^T(\Theta)\mathbf{w}) \end{pmatrix}^\top.$$

#### *Magnetometer*

From Equation (7.10), the model for the magnetometer is

$$h_{\text{mag}}(\mathbf{x}) = R(\Theta)^\top \mathbf{m}^i,$$

where we have assumed that the magnetic bias vector  $\beta_{\text{mag}}^i$  is known through a calibration process, and that the constant magnetic vector  $\mathbf{m}^i$  is also known. ■  
The associated Jacobian is given by

$$\frac{\partial h_{\text{mag}}}{\partial \mathbf{x}} = \left( 0_{3 \times 3}, \quad 0_{3 \times 3}, \quad \frac{\partial [R^T(\Theta)\mathbf{m}^i]}{\partial \Theta}^\top, \quad 0_{3 \times 3}, \quad 0_{3 \times 3}, \quad 0_{3 \times 2} \right),$$

where

$$\frac{\partial [R^T(\Theta)\mathbf{m}^i]}{\partial \Theta} = \left( \frac{\partial R^T(\Theta)}{\partial \phi} \mathbf{m}^i, \quad \frac{\partial R^T(\Theta)}{\partial \theta} \mathbf{m}^i, \quad \frac{\partial R^T(\Theta)}{\partial \psi} \mathbf{m}^i \right).$$

#### *GPS Position Sensor*

The model for the GPS position sensor is

$$h_{\text{GPS, pos}}(\mathbf{x}) = (p_n, p_e, p_d)^\top$$

with associated Jacobian

$$\frac{\partial h_{\text{GPS, pos}}}{\partial \mathbf{x}} = (I_{3 \times 3}, \quad 0_{3 \times 2}).$$

#### *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  $P = [I_{2 \times 2}, \ 0_{2 \times 1}]$ , and the associated Jacobian is given by

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

where  $\frac{\partial V_g(\mathbf{v}, \Theta)^\top}{\partial \Theta}$  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  $3 \times 1$  vector with one in the  $i^{\text{th}}$  location and zeros elsewhere, and  $\Delta$  is a small number.

#### *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

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

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

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

#### *Pseudo Measurement for Sideslip Angle*

With the given sensors, the sideslip 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 sideslip angle by assuming that it is zero.

The sideslip 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

$$\frac{\partial h_\beta}{\partial \mathbf{x}} = \begin{pmatrix} 0_{3 \times 1} \\ \frac{\partial v_r(\mathbf{v}, \Theta, \mathbf{w})}{\partial \mathbf{v}} \\ \frac{\partial v_r(\mathbf{v}, \Theta, \mathbf{w})}{\partial \Theta} \\ 0_{3 \times 1} \\ 0_{3 \times 1} \\ \frac{\partial v_r(\mathbf{v}, \Theta, \mathbf{w})}{\partial \mathbf{w}} \end{pmatrix}^\top,$$

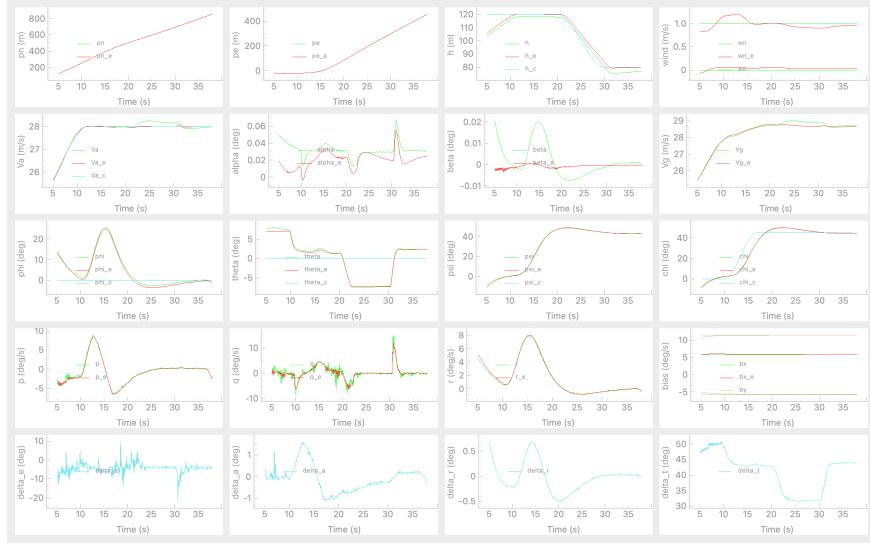
where the partial derivatives are computed numerically as

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

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

The extended Kalman filter to estimate  $\mathbf{p}$ ,  $\mathbf{v}$ ,  $\Theta$ ,  $\mathbf{b}_a$ ,  $\mathbf{b}_g$ , and  $\mathbf{w}$  is implemented using algorithm 3. For the benchmark maneuver discussed in section 8.1, the estimation errors using algorithm 3 is shown in figure 8.11. Comparing figure 8.11 with figures 8.9 and 8.5, shows that the full continuous-discrete extended Kalman filter produces much better results than the other filters presented in this chapter.

NEW MATERIAL:



**Figure 8.11:** Estimation error for the full-state continuous-discrete extended Kalman filter.

## 8.11 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 8.8](#) we showed how a two-state EKF can be used to estimate the roll and pitch angles. In [section 8.9](#) we showed how the position, ground speed, course, wind, and heading can be estimated using a seven-state EKF based on GPS measurements. The two-stage EKF assumes that there is not a bias in the gyroscopes or accelerometers. [When there are unknown biases, a full-state EKF, as derived in section 8.10, must be used.](#)

## NOTES AND REFERENCES

The Kalman filter was first introduced in [57]. There are many excellent texts on the Kalman filter including [58, 59, 60, 61]. Some of the results in this chapter have been discussed previously in [62, 63]. State estimation

using computer vision instead of GPS has been discussed in [64, 65, 66]. As explained in [https://docs.px4.io/main/en/advanced\\_config/tuning\\_the\\_ecl\\_ekf.html](https://docs.px4.io/main/en/advanced_config/tuning_the_ecl_ekf.html), the EKF used in the PX4 and ArduPilot firmware, is similar to the full-state EKF explained in section 8.10, with the exception that the attitude is represented a unit quaternion, and with additional states to estimate the magnetic field in the inertial and body frames. The PX4 and ArduPilot firmware also allows sensor measurements to be delayed.

## 8.12 DESIGN PROJECT

### MODIFIED MATERIAL:

- 8.1 Download the simulation files for this chapter from the web..
- 8.2 Implement the simple schemes described in section 8.3 using low-pass filters and model inversion to estimate the states  $p_n$ ,  $p_e$ ,  $h$ ,  $V_a$ ,  $\phi$ ,  $\theta$ ,  $\chi$ ,  $p$ ,  $q$ , and  $r$ . Tune the bandwidth of the low-pass filter to observe the effect. Different states may require a different filter bandwidth.
- 8.3 Modify the observer file to implement the extended Kalman filter for roll and pitch angles described in section 8.8. Tune the filter until you are satisfied with the performance.
- 8.4 Modify the observer file to implement the extended Kalman filter for position, heading, and wind described in section 8.9. Tune the filter until you are satisfied with the performance.
- 8.5 Change the simulation to use the estimated state in the autopilot as opposed to the true states. Tune autopilot and estimator gains if necessary. By changing the bandwidth of the low-pass filter, note that the stability of the closed loop system is heavily influenced by this value.



## Chapter Nine

---

### Design Models for Guidance

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  $b_\phi$  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  $b_\chi$  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  $\gamma$  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  $b_\gamma$  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 2.9](#) 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  $\psi$  and  $V_a$ . With reference to the wind triangle expression in [equation \(2.8\)](#), 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 5.2](#) we showed that the coordinated-turn condition is described by

$$\dot{\chi} = \frac{g}{V_g} \tan \phi \cos(\chi - \psi). \quad (9.10)$$

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 \(2.8\)](#) to get

$$\begin{aligned} & \begin{pmatrix} \cos \chi \cos \gamma & -V_g \sin \chi \cos \gamma & -V_g \cos \chi \sin \gamma \\ \sin \chi \cos \gamma & V_g \cos \chi \cos \gamma & -V_g \sin \chi \sin \gamma \\ -\sin \gamma & 0 & -\cos \gamma \end{pmatrix} \begin{pmatrix} \dot{V}_g \\ \dot{\chi} \\ \dot{\gamma} \end{pmatrix} \\ &= \begin{pmatrix} \cos \psi \cos \gamma_a & -V_g \sin \psi \cos \gamma_a & -V_g \cos \psi \sin \gamma_a \\ \sin \psi \cos \gamma_a & V_g \cos \psi \cos \gamma_a & -V_g \sin \psi \sin \gamma_a \\ -\sin \gamma_a & 0 & -\cos \gamma_a \end{pmatrix} \begin{pmatrix} \dot{V}_a \\ \dot{\psi} \\ \dot{\gamma}_a \end{pmatrix}. \quad (9.11) \end{aligned}$$

Under the condition of constant-altitude flight and no down component of wind, where  $\gamma$ ,  $\gamma_a$ ,  $\dot{\gamma}$ ,  $\dot{\gamma}_a$ , 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

$$\dot{V}_g = \frac{\dot{V}_a}{\cos(\chi - \psi)} + V_g \dot{\chi} \tan(\chi - \psi) \quad (9.12)$$

$$\dot{\psi} = \frac{\dot{V}_a}{V_a} \tan(\chi - \psi) + \frac{V_g \dot{\chi}}{V_a \cos(\chi - \psi)}. \quad (9.13)$$

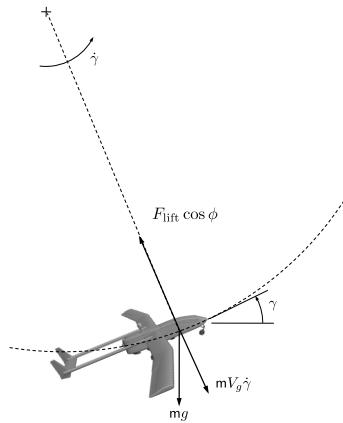
If we assume that the airspeed is constant, then from [equations \(9.13\)](#) and [\(9.10\)](#), we have

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

This is the familiar coordinated-turn expression. Most notable is that this equation is true in the presence of wind.

### 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 9.4. Since the airframe is rolled



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

at an angle of  $\phi$ , the projection of the lift vector onto  $\mathcal{P}$  is  $F_{\text{lift}} \cos \phi$ . The centripetal force due to the pull-up maneuver is  $mV_g \dot{\gamma}$ . 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 9.5](#).

### 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 [sideslip](#) 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 groundspeed, 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 as in [equation \(9.7\)](#), then we will use [equations \(2.9\) through \(2.11\)](#) 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

$$\begin{aligned} \dot{p}_n &= V_a \cos \psi + w_n \\ \dot{p}_e &= V_a \sin \psi + w_e \\ \ddot{\chi} &= b_{\dot{\chi}}(\dot{\chi}^c - \dot{\chi}) + b_\chi(\chi^c - \chi) \\ \ddot{h} &= b_{\dot{h}}(\dot{h}^c - \dot{h}) + b_h(h^c - h) \\ \dot{V}_a &= b_{V_a}(V_a^c - V_a), \end{aligned} \quad (9.18)$$

where the inputs are the commanded altitude  $h^c$ , the commanded airspeed  $V_a^c$ , and the commanded course  $\chi^c$ , and  $\psi$  is given by [equation \(2.10\)](#), 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  $\phi^c$  is the commanded roll angle,  $\psi$  is given by [equation \(2.10\)](#), and  $V_g$  is given by [equation \(2.11\)](#), 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  $\gamma^c$  is the commanded (inertial referenced) flight-path angle, and where  $\gamma_a$ ,  $\psi$ , and  $V_g$  are given by [equations \(2.9\)](#), [\(2.10\)](#), and [\(2.11\)](#), 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

$$\begin{aligned}\dot{p}_n &= V_a \cos \psi \cos \gamma_a + w_n \\ \dot{p}_e &= V_a \sin \psi \cos \gamma_a + w_e \\ \dot{h} &= V_a \sin \gamma_a - w_d \\ \dot{\chi} &= \frac{g}{V_g} \tan \phi\end{aligned}\tag{9.21}$$

$$\begin{aligned}\dot{\gamma} &= \frac{g}{V_g} (n_{lf} \cos \phi - \cos \gamma) \\ \dot{V}_a &= b_{V_a} (V_a^c - V_a) \\ \dot{\phi} &= b_\phi (\phi^c - \phi) \\ \dot{n}_{lf} &= b_n (n_{lf}^c - n_{lf}),\end{aligned}$$

where  $n_{lf}^c$  is the commanded load factor, and where  $\gamma_a$ ,  $\psi$ , and  $V_g$  are given by equations (2.9), (2.10), and (2.11) respectively. It is common in these equations to use the coordinated turn condition  $\dot{\psi} = g/V_a \tan \phi$  rather than  $\dot{\chi} = g/V_g \tan \phi$  and to use  $\gamma$  instead of  $\gamma_a$  in the kinematic evolution of position and altitude.

#### 9.4 THE DUBINS AIRPLANE MODEL

**NEW MATERIAL:** A particularly simple model that is often used for planning purposes is the so-called *Dubin's Airplane Model*. For this model, the wind is ignored, and the roll angle and flight path angle are assumed to be the control variables.

Accordingly, 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, the following kinematic model is a good description of the UAV motion

$$\begin{aligned}\dot{r}_n &= V_a \cos \psi \cos \gamma^c \\ \dot{r}_e &= V_a \sin \psi \cos \gamma^c \\ \dot{r}_d &= -V_a \sin \gamma^c \\ \dot{\psi} &= \frac{g}{V_a} \tan \phi^c,\end{aligned}\tag{9.22}$$

where we assume that the commanded roll and flight path angles satisfy

$$|\phi^c| \leq \bar{\phi}\tag{9.23}$$

$$|\gamma^c| \leq \bar{\gamma}.\tag{9.24}$$

It is also common to ignore the influence of the flight path angle on the lateral kinematics of the aircraft resulting in

$$\begin{aligned}\dot{r}_n &= V_a \cos \psi \\ \dot{r}_e &= V_a \sin \psi \\ \dot{r}_d &= -V_a \sin \gamma^c \\ \dot{\psi} &= \frac{g}{V_a} \tan \phi^c,\end{aligned}\tag{9.25}$$

with input constraints (9.23) and (9.24), which is equivalent to the Dubin's airplane model originally proposed in [67]. The Dubin's airplane model is well suited high level path planning and path following algorithms, as will be discussed in Chapters 10–12.

## 9.5 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. Thrust, angle-of-attack, side-slip, and roll angles are used as control variables in these models.

### MODIFIED MATERIAL:

If we let  $V_g \triangleq \|\mathbf{v}^i\|$  be the magnitude of the ground velocity vector, then the ground velocity vector expressed in NED inertial coordinates is

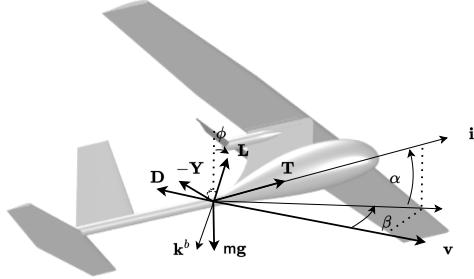
$$\mathbf{v}^i = V_g \begin{pmatrix} \cos \chi \cos \gamma \\ \sin \chi \cos \gamma \\ -\sin \gamma \end{pmatrix}.$$

Therefore, the kinematic expression for the evolution of the position vector is

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

A free-body diagram is shown in Figure 9.2, from which we see that the thrust in the body frame is  $\mathbf{T}^b = (T, 0, 0)^\top$ . The thrust in the wind frame is given by

$$\begin{aligned} \mathbf{T}^w &= R_b^w \mathbf{T}^b = \begin{pmatrix} \cos \alpha \cos \beta & \sin \beta & \sin \alpha \cos \beta \\ -\cos \alpha \sin \beta & \cos \beta & -\sin \alpha \sin \beta \\ -\sin \alpha & 0 & \cos \alpha \end{pmatrix} \begin{pmatrix} T \\ 0 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} T \cos \alpha \cos \beta \\ -T \cos \alpha \sin \beta \\ -T \sin \alpha \end{pmatrix}. \end{aligned}$$



**Figure 9.2:** Free-body diagram indicating external forces impacting the evolution of the magnitude of velocity.  $\mathbf{L}$  is lift,  $\mathbf{D}$  is drag,  $\mathbf{T}$  is thrust, and  $\mathbf{g}$  is the gravity vector.

As shown in Figure 9.2 the lift, drag, and sideslip forces all act in the wind frame. The drag force  $\mathbf{D}$  acts in the opposite direction of the velocity vector or along the negative  $i^w$  axis, the lift force  $\mathbf{L}$  acts perpendicular to the velocity vector and along the symmetry plane of the aircraft, or the negative  $k^w$  axis, and the sideslip force  $\mathbf{Y}$  acts perpendicular to both  $\mathbf{D}$  and  $\mathbf{L}$  along the negative  $j^w$  axis. Therefore, in the wind frame, the aerodynamic forces are

$$\mathbf{F}_{aero}^w = \begin{pmatrix} -D \\ -Y \\ -L \end{pmatrix},$$

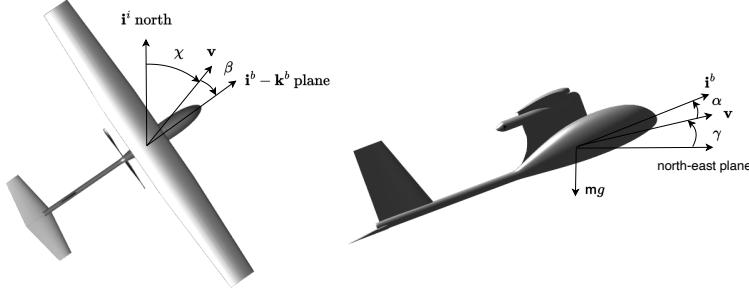
where  $L$ ,  $D$ , and  $Y$  are the magnitude of the lift, drag, and sideslip forces. Therefore the total aerodynamic and thrust forces in the wind frame are

$$\mathbf{F}^w = \begin{pmatrix} -D + T \cos \alpha \cos \beta \\ -Y - T \cos \alpha \sin \beta \\ -L - T \sin \alpha \end{pmatrix}.$$

The flight path angle  $\gamma$  and the course angle  $\chi$  are inertial quantities that are independent of the roll of the aircraft. Therefore, to express the inertial forces we need to express the forces in the unrolled or vehicle-2 frame, i.e.,

$$\begin{aligned} \mathbf{F}^{v^2} &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & -\sin \phi & \cos \phi \end{pmatrix} \begin{pmatrix} -D + T \cos \alpha \cos \beta \\ -Y - T \cos \alpha \sin \beta \\ -L - T \sin \alpha \end{pmatrix} \\ &= \begin{pmatrix} -D + T \cos \alpha \cos \beta \\ -(Y + T \cos \alpha \sin \beta) \cos \phi - (L + T \sin \alpha) \sin \phi \\ (Y + T \cos \alpha \sin \beta) \sin \phi - (L + T \sin \alpha) \cos \phi \end{pmatrix}. \end{aligned}$$

The magnitude of the velocity vector evolves along the  $i^{v^2}$  axis. Since the



**Figure 9.3:** The course angle  $\chi$  is the angle between the inertial north axis and the projection of the velocity vector  $\mathbf{v}_g$  onto the north-east plane. The side-slip angle  $\beta$  is the angle between the velocity vector  $\mathbf{v}_g$  and the body  $i^b - k^b$  plane. The flight path angle  $\gamma$  is the angle between the inertial north-east plane and the velocity vector  $\mathbf{v}_g$ . The angle-of-attack  $\alpha$  is the angle between the body  $i^b$  axis and the projection of the velocity vector  $\mathbf{v}_g$  onto the body  $i^b - k^b$  plane.

gravity vector is perpendicular to the north-east plane, from Figure 9.3 we see that the projection of the gravity vector along the velocity vector, or the  $i^{v^2}$ -axis, is  $-mg \sin \gamma$ . Therefore, from Newton's law, the magnitude of the velocity vector evolves according to

$$\begin{aligned} m\dot{V}_g &= T \cos \alpha \cos \beta - D - mg \sin \gamma \\ \implies \dot{V}_g &= \frac{T}{m} \cos \alpha \cos \beta - \frac{D}{m} - g \sin \gamma. \end{aligned} \quad (9.26)$$

From the free body diagrams shown in Figure 9.4 we get that

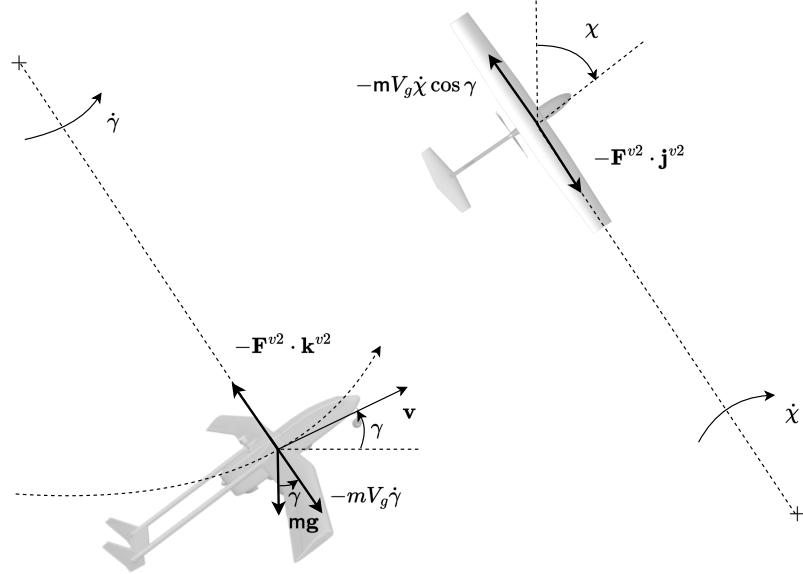
$$\begin{aligned} mV_g\dot{\gamma} &= -(Y + T \cos \alpha \sin \beta) \sin \phi + (L + T \sin \alpha) \cos \phi - mg \cos \gamma \\ \implies \dot{\gamma} &= \frac{1}{mV_g} [(L + T \sin \alpha) \cos \phi - (Y + T \cos \alpha \sin \beta) \sin \phi - mg \cos \gamma]. \end{aligned} \quad (9.27)$$

Along the lateral direction, projecting the velocity vector onto the north-east plane, gives from Figure 9.4

$$\begin{aligned} mV_g\dot{\chi} \cos \gamma &= (Y + T \cos \alpha \sin \beta) \cos \phi + (L + T \sin \alpha) \sin \phi \\ \implies \dot{\chi} &= \frac{1}{mV_g \cos \gamma} [(Y + T \cos \alpha \sin \beta) \cos \phi + (L + T \sin \alpha) \sin \phi]. \end{aligned}$$

In summary the point-mass model is given by

$$\dot{p}_n = V_g \cos \chi \cos \gamma$$



**Figure 9.4:** Free-body diagram for a pull-up maneuver. The MAV is at a roll angle of  $\phi$ .

$$\begin{aligned}
 \dot{p}_e &= V_g \sin \chi \cos \gamma \\
 \dot{h} &= V_g \sin \gamma \\
 \dot{V}_g &= \frac{T}{m} \cos \alpha \cos \beta - \frac{D(\alpha)}{m} - g \sin \gamma \\
 \dot{\chi} &= \frac{1}{m V_g \cos \gamma} [(Y(\beta) + T \cos \alpha \sin \beta) \cos \phi + (L(\alpha) + T \sin \alpha) \sin \phi] \\
 \dot{\gamma} &= \frac{1}{m V_g} [(L(\alpha) + T \sin \alpha) \cos \phi - (Y(\beta) + T \cos \alpha \sin \beta) \sin \phi - m g \cos \gamma],
 \end{aligned} \tag{9.29}$$

where the lift, drag, and sideslip forces are given by

$$\begin{aligned}
 L(\alpha) &= \frac{1}{2} \rho V_a^2 S (C_{L_0} + C_{L_\alpha} \alpha) \\
 D(\alpha) &= \frac{1}{2} \rho V_a^2 S (C_{D_0} + C_{D_\alpha} \alpha + C_{D_{\alpha^2}} \alpha^2) \\
 Y(\beta) &= \frac{1}{2} \rho V_a^2 S C_{Y_\beta} \beta,
 \end{aligned}$$

and where  $\rho$  is the density of air,  $V_a$  is the airspeed, and  $S$  is a vehicle reference area, and where  $C_{L_0}$ ,  $C_{L_\alpha}$ ,  $C_{D_0}$ ,  $C_{D_\alpha}$ ,  $C_{D_{\alpha^2}}$ , and  $C_{Y_\beta}$  are aero-

dynamic coefficients defined in Chapter 4. The control variables are thrust  $T$ , roll angle  $\phi$ , angle-of-attack  $\alpha$ , and sideslip angle  $\beta$ .

When there is no wind, and when the sideslip angle is being regulated to zero, we get the simplified model

$$\begin{aligned}\dot{p}_n &= V_a \cos \psi \cos \gamma \\ \dot{p}_e &= V_a \sin \chi \cos \gamma \\ \dot{h} &= V_a \sin \gamma \\ \dot{V}_a &= \frac{T}{m} \cos \alpha - \frac{D(\alpha)}{m} - g \sin \gamma \\ \dot{\psi} &= \frac{1}{mV_a \cos \gamma} (L(\alpha) + T \sin \alpha) \sin \phi \\ \dot{\gamma} &= \frac{1}{mV_a} [(L(\alpha) + T \sin \alpha) \cos \phi - mg \cos \gamma].\end{aligned}\tag{9.30}$$

## 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), with the particularly useful Dubin's airplane model given in [equation](#) (9.22). The dynamic point-mass model often found in the literature is given in [equations](#) (9.29) and (9.30).

## NOTES AND REFERENCES

Material supporting the development of the guidance models discussed in this chapter can be found in [17, 24, 19, 68]. The derivation of accelerating climb in [section](#) 9.2.2 draws on the discussion in [17, p. 227–228]. The Dubin's airplane model was first proposed in [67]. The model given in this chapter is from [69].

## 9.7 DESIGN PROJECT

### MODIFIED MATERIAL:

The objective of the assignment in this chapter is to estimate the autopilot constants  $b_*$  and to develop a reduced-order design models that can be used to test and debug the guidance algorithm discussed in later chapters, prior

to implementation on the full simulation model. We will focus primarily on the models given in equations (9.18) and (9.19).

- 9.1 Create simulation files that implements the model given in equation (9.18) and add the model to the MAV simulator. For different inputs  $\chi^c$ ,  $h^c$ , and  $V_a^c$ , compare the output of the two models, and tune the autopilot coefficients  $b_{V_a}$ ,  $b_{\dot{h}}$ ,  $b_h$ ,  $b_{\dot{\chi}}$ , and  $b_\chi$  to obtain similar behavior.



## Chapter Ten

---

### Straight-line, Orbit, and Helix Path 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 [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 airframe is the most aerodynamic.

cally 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

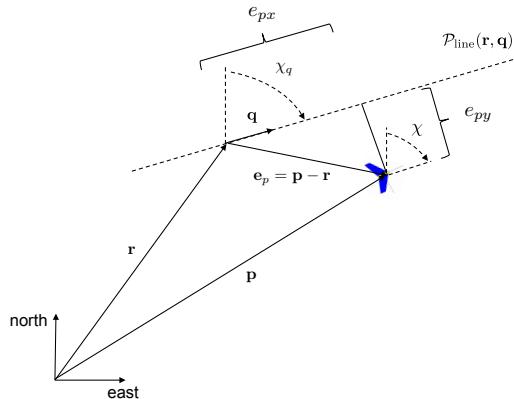
A straight-line path is described by two vectors in  $\mathbb{R}^3$ , namely

$$\mathcal{P}_{\text{line}}(\mathbf{r}, \mathbf{q}) = \{\mathbf{x} \in \mathbb{R}^3 : \mathbf{x} = \mathbf{r} + \lambda \mathbf{q}, \lambda \in \mathbb{R}\}, \quad (10.1)$$

where  $\mathbf{r} \in \mathbb{R}^3$  is the origin of the path resolved in the inertial frame, and  $\mathbf{q} \in \mathbb{R}^3$  is a unit vector whose direction indicates the desired direction of travel, also resolved in the inertial frame. 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

$$\chi_q \triangleq \text{atan2}(q_e, q_n), \quad (10.2)$$

where  $\mathbf{q} = (q_n \ q_e \ q_d)^\top$  expresses the north, east, and down components of the unit direction vector, and 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.



**Figure 10.1:** The configuration of the MAV indicated by  $(\mathbf{p}, \chi)$ , 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 the transformation from the inertial frame to the path frame is given by

$$\mathcal{R}_i^P \triangleq \begin{pmatrix} \cos \chi_q & \sin \chi_q & 0 \\ -\sin \chi_q & \cos \chi_q & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

and the relative error expressed in the path frame is

$$\mathbf{e}_p = \begin{pmatrix} e_{px} \\ e_{py} \\ e_{pz} \end{pmatrix} \triangleq \mathcal{R}_i^P (\mathbf{p}^i - \mathbf{r}^i).$$

The relative error dynamics in the north-east inertial plane, expressed in the path frame, are given by

$$\begin{aligned} \begin{pmatrix} \dot{e}_{px} \\ \dot{e}_{py} \end{pmatrix} &= \begin{pmatrix} \cos \chi_q & \sin \chi_q \\ -\sin \chi_q & \cos \chi_q \end{pmatrix} \begin{pmatrix} V_g \cos \chi \cos \gamma \\ V_g \sin \chi \cos \gamma \end{pmatrix} \\ &= V_g \begin{pmatrix} \cos(\chi - \chi_q) \cos \gamma \\ \sin(\chi - \chi_q) \cos \gamma \end{pmatrix}. \end{aligned} \quad (10.3)$$

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

$$\dot{e}_{py} = V_g \sin(\chi - \chi_q) \cos \gamma \quad (10.4)$$

$$\ddot{\chi} = b_{\dot{\chi}}(\dot{\chi}^c - \dot{\chi}) + b_\chi(\chi^c - \chi). \quad (10.5)$$

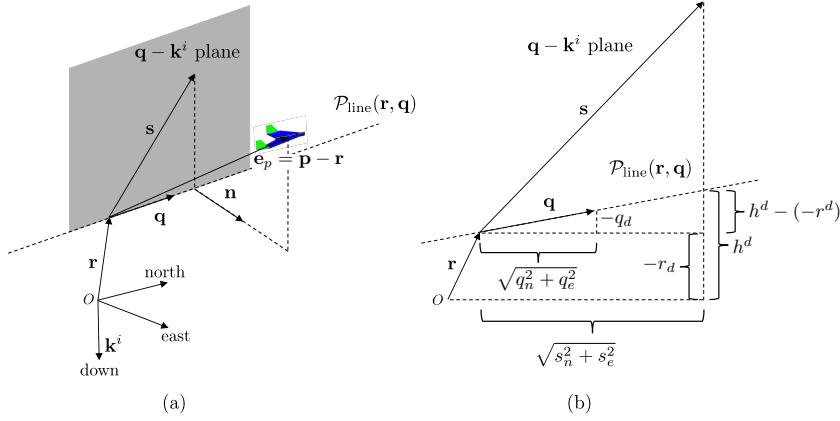
The lateral straight-line path following problem is to select  $\chi^c$  so that  $e_{py} \rightarrow 0$  when  $\chi_q$  is known.

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  $\mathbf{s}$  given by

$$\mathbf{s}^i = \begin{pmatrix} s_n \\ s_e \\ s_d \end{pmatrix} = (I - \mathbf{n}\mathbf{n}^\top) \mathbf{e}_p^i,$$

where

$$\mathbf{e}_p^i = \begin{pmatrix} e_{pn} \\ e_{pe} \\ e_{pd} \end{pmatrix} \triangleq \mathbf{p}^i - \mathbf{r}^i = \begin{pmatrix} p_n - r_n \\ p_e - r_e \\ p_d - r_d \end{pmatrix}$$



**Figure 10.2:** Desired altitude calculation for straight-line path following in longitudinal direction.

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:** Referring to the vertical plane containing the path shown in figure 10.2(b) and using similar triangles, we have the relationship

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

Rearranging gives that 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

$$h_d(\mathbf{r}, \mathbf{p}, \mathbf{q}) = -r_d - \sqrt{s_n^2 + s_e^2} \left( \frac{q_d}{\sqrt{q_n^2 + q_e^2}} \right). \quad (10.6)$$

Since the altitude dynamics are

$$\ddot{h} = b_{\dot{h}}(\dot{h}^c - \dot{h}) + b_h(h^c - h), \quad (10.7)$$

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](#) (10.6) and the dynamics modeled by [equation](#) (10.7), we will show that letting  $h^c = h_d(\mathbf{r}, \mathbf{p}, \mathbf{q})$  good path-following performance will result, with zero steady-state error in altitude for straight-line paths.

In the altitude hold zone, where there is no saturation, pitch attitude is used to control the altitude of the MAV. Assuming that successive loop closure has been properly implemented, [figure](#) 6.13 shows a simplified representation of the outer-loop dynamics, which in the Laplace domain is

$$h(s) = \left( \frac{b_h s + b_h}{s^2 + b_h s + b_h} \right) h^c(s).$$

**MODIFIED MATERIAL:** Defining the altitude error as

$$e_h \triangleq h_d(\mathbf{r}, \mathbf{p}, \mathbf{q}) - h = h^c - h,$$

we get that

$$\begin{aligned} e_h(s) &= (1 - h(s)) h^c(s) \\ &= \left( \frac{s^2 + b_h s + b_h}{s^2 + b_h s + b_h} - \frac{b_h s + b_h}{s^2 + b_h s + b_h} \right) h^c(s) \\ &= \left( \frac{s^2}{s^2 + b_h s + b_h} \right) h^c(s). \end{aligned}$$

By applying the final value theorem, we find that

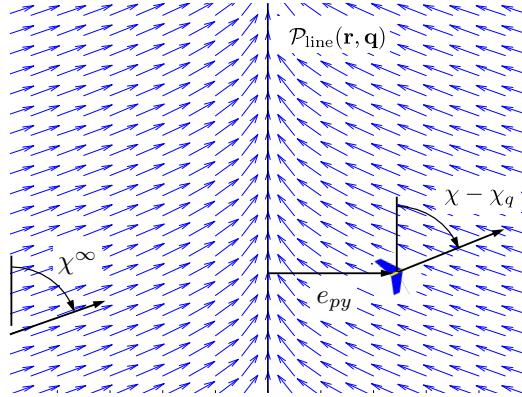
$$\begin{aligned} e_{h,ss} &= \lim_{s \rightarrow 0} s \frac{s^2}{s^2 + b_h s + b_h} h^c \\ &= 0, \quad \text{for } h^c = \frac{H_0}{s}, \frac{H_0}{s^2}. \end{aligned}$$

The analysis in [chapter](#) 6 also shows that constant disturbances are rejected implying that 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  $\chi^c$  in [equation](#) (10.5) so that  $e_{py}$  in [equation](#) (10.4) 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  $\chi^\infty$  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

$$\chi_d(e_{py}) = -\chi^\infty \frac{2}{\pi} \tan^{-1}(k_{\text{path}} e_{py}), \quad (10.8)$$

where  $k_{\text{path}}$  is a positive constant that influences the rate of the transition from  $\chi^\infty$  to zero. Figure 10.4 shows how the choice of  $k_{\text{path}}$  affects the rate of transition. Large values of  $k_{\text{path}}$  result in short, abrupt transitions, while small values of  $k_{\text{path}}$  cause long, smooth transitions in the desired course.

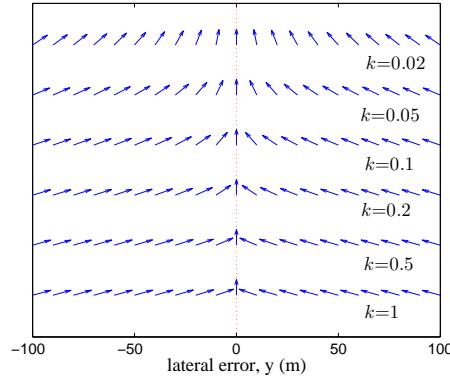
If  $\chi^\infty$  is restricted to be in the range  $\chi^\infty \in (0, \frac{\pi}{2}]$ , then clearly

$$-\frac{\pi}{2} < \chi^\infty \frac{2}{\pi} \tan^{-1}(k_{\text{path}} e_{py}) < \frac{\pi}{2}$$

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

$$\dot{W} = -V_a e_{py} \sin \left( \chi^\infty \frac{2}{\pi} \tan^{-1}(k_{\text{path}} e_{py}) \right) \cos \gamma$$

is less than zero for  $e_{py} \neq 0$ . The command for lateral path following is



**Figure 10.4:** Vector fields for various values of  $k_{\text{path}}$ . Large values of  $k_{\text{path}}$  yield abrupt transitions from  $\chi^\infty$  to zero, while small values of  $k_{\text{path}}$  give smooth transitions.

therefore given by

$$\chi^c(t) = \chi_q - \chi^\infty \frac{2}{\pi} \tan^{-1}(k_{\text{path}} e_{py}(t)). \quad (10.9)$$

Before moving to orbit following, we note that using equation (10.9) may result in undesirable behavior if  $\chi_q$  is computed directly from equation (10.2), where atan2 returns an angle between  $\pm\pi$ . As an example, consider the scenario shown in figure 10.5, where  $\chi_q$  is a positive number slightly smaller than  $+\pi$ . Since the current course is negative, equation (10.9) will cause the MAV to turn right to align with the waypoint path. As an alternative, if  $\chi_q$  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,  $\chi_q$  should be computed as

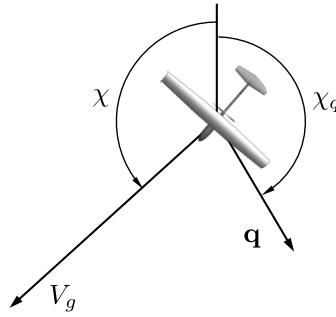
$$\chi_q = \text{atan2}(q_e, q_n) + 2\pi m,$$

where  $m \in \mathcal{N}$  is selected so that  $-\pi \leq \chi_q - \chi \leq \pi$ , and atan2 is the four-quadrant inverse tangent 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

$$\mathcal{P}_{\text{orbit}}(\mathbf{c}, \rho, \lambda) = \left\{ \mathbf{p} \in \mathbb{R}^3 : \mathbf{p} = \mathbf{c} + \lambda \rho \begin{pmatrix} \cos \varphi & \sin \varphi & 0 \end{pmatrix}^\top, \varphi \in [0, 2\pi) \right\},$$



**Figure 10.5:** The calculation of  $\chi_q$  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  $\chi_q$  is computed with  $\text{atan}2$ , 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 strategy for orbit following is best derived in polar coordinates. Let

$$d \triangleq \sqrt{(p_n - c_n)^2 + (p_e - c_e)^2}$$

be the lateral distance from the desired center of the orbit to the MAV, and let

$$\varphi \triangleq \tan^{-1} \left( \frac{p_e - c_e}{p_n - c_n} \right) \quad (10.10)$$

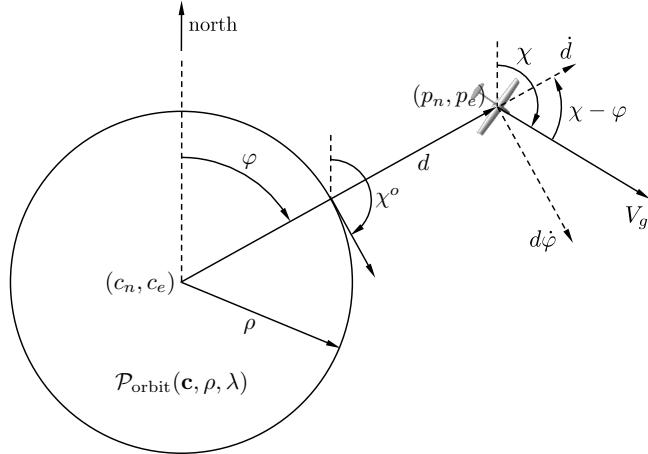
be the phase angle of the relative position, as shown in figure 10.6.

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  $\varphi$  so that the equations of motion represent the MAV motion in the normal and tangential directions to the orbit:

$$\begin{pmatrix} \dot{d} \\ d\dot{\varphi} \end{pmatrix} = \begin{pmatrix} \cos \varphi & \sin \varphi \\ -\sin \varphi & \cos \varphi \end{pmatrix} \begin{pmatrix} \dot{p}_n \\ \dot{p}_e \end{pmatrix}$$



**Figure 10.6:** Orbital path with center  $(c_n, c_e)$ , and radius  $\rho$ . The distance from the orbit center to the MAV is  $d$ , and the angular position of the MAV relative to the orbit is  $\varphi$ .

$$\begin{aligned} &= \begin{pmatrix} \cos \varphi & \sin \varphi \\ -\sin \varphi & \cos \varphi \end{pmatrix} \begin{pmatrix} V_g \cos \chi \\ V_g \sin \chi \end{pmatrix} \\ &= \begin{pmatrix} V_g \cos(\chi - \varphi) \\ V_g \sin(\chi - \varphi) \end{pmatrix}. \end{aligned}$$

These expressions can also be derived from the geometry illustrated in figure 10.6. The MAV dynamics in polar coordinates are therefore given by

$$\dot{d} = V_g \cos(\chi - \varphi) \quad (10.11)$$

$$\dot{\varphi} = \frac{V_g}{d} \sin(\chi - \varphi) \quad (10.12)$$

$$\ddot{\chi} = -b_\chi \dot{\chi} + b_\chi (\chi^o - \chi). \quad (10.13)$$

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  $\rho$  and to drive the course angle  $\chi(t)$  to  $\chi^o$  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

$$\chi_d(d - \rho, \lambda) = \chi^o + \lambda \tan^{-1} \left( k_{\text{orbit}} \left( \frac{d - \rho}{\rho} \right) \right), \quad (10.14)$$

where  $k_{\text{orbit}} > 0$  is a constant that specifies the rate of transition from  $\lambda\pi/2$  to zero. This expression for  $\chi_d$  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

$$\chi^c(t) = \varphi + \lambda \left[ \frac{\pi}{2} + \tan^{-1} \left( k_{\text{orbit}} \left( \frac{d - \rho}{\rho} \right) \right) \right]. \quad (10.15)$$

Similar to the computation of the path angle  $\chi_q$ , if the angular position in the orbit  $\varphi$  is computed to be between  $\pm\pi$ , then there will be a sudden jump of  $2\pi$  in the commanded course as the MAV transitions from  $\varphi = \pi$  to  $\varphi = -\pi$ . To alleviate this problem,  $\varphi$  should be computed as

$$\varphi = \text{atan2}(p_e - c_e, p_n - c_n) + 2\pi m,$$

where  $m \in \mathbb{N}$  is selected so that  $-\pi \leq \varphi - \chi \leq \pi$ .

### 10.2.1 Feedforward Roll Control for Orbit Tracking

**NEW MATERIAL:** Suppose that the aircraft is exactly tracking the orbit, then the commanded course angle is  $\chi^c = \chi_0$ , and if actual course angle is equal to  $\chi^c$ , then the commanded roll angle will be zero. Since the roll angle is zero, the course will not change, and the aircraft will leave the orbit. The result is an oscillatory behavior about the orbit path. This oscillatory

behavior can be avoided by adding feedforward roll command as shown in Figure 10.7.

For a UAV flying a circular orbit with radius  $\rho$  at a constant ground speed of  $V_g$  with no wind, the course rate is given by

$$\dot{\chi} = \lambda \frac{V_g}{\rho}. \quad (10.16)$$

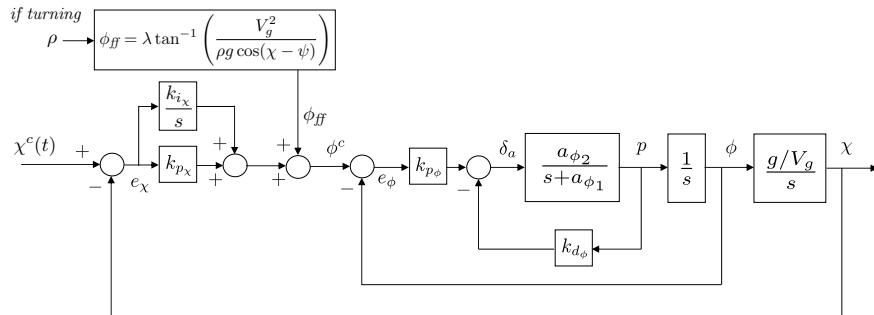
Assuming that the UAV is flying a coordinated turn, the kinematics of the UAV are given by equation (5.15) as

$$\dot{\chi} = \frac{g}{V_g} \tan \phi \cos(\chi - \psi). \quad (10.17)$$

By equating the right-hand sides of Equations (10.16) and (10.17) and solving for the roll angle  $\phi$ , we can express the feedforward roll command as

$$\phi_{ff} = \lambda \tan^{-1} \left( \frac{V_g^2}{\rho g \cos(\chi - \psi)} \right). \quad (10.18)$$

This feedforward roll command can be incorporated into the lateral-directional control block diagram as shown in figure 10.7.



**Figure 10.7:** Lateral-directional control block diagram with roll feedforward command for orbit tracking.

When wind is present, the feedforward roll command is similarly effective. The feedforward roll command will no longer remain constant for a constant radius as the roll angle will depend on the direction of the wind relative to the aircraft. Similarly, the ground speed and course rate will vary with time.

### 10.3 3D VECTOR-FIELD PATH FOLLOWING

### NEW MATERIAL:

This section shows how to develop guidance laws to ensure that the kinematic model (9.22) follows straight-line and helical paths. The guidance strategy will use the vector-field methodology proposed in [70], and this section provides a brief overview.

In [70], the path to be followed in  $\mathbb{R}^3$  is specified as the intersection of two two-dimensional manifolds given by  $\alpha_1(\mathbf{p}) = 0$  and  $\alpha_2(\mathbf{p}) = 0$  where  $\alpha_1$  and  $\alpha_2$  have bounded second partial derivatives, and where  $\mathbf{p} \in \mathbb{R}^3$ . An underlying assumption is that the path given by the intersection is connected and one-dimensional. Defining the function

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

gives

$$\frac{\partial V}{\partial \mathbf{p}} = \alpha_1(\mathbf{p}) \frac{\partial \alpha_1}{\partial \mathbf{p}}(\mathbf{p}) + \alpha_2(\mathbf{p}) \frac{\partial \alpha_2}{\partial \mathbf{p}}(\mathbf{p}),$$

where we note that the negative gradient  $-\frac{\partial V}{\partial \mathbf{p}}$  is a vector that points toward the path. Therefore following the negative gradient will transition the aircraft onto the path. However simply following the negative gradient is insufficient for path following since the gradient is zero on the path. When the MAV is on the path, its direction of motion should be perpendicular to both  $\frac{\partial \alpha_1}{\partial \mathbf{p}}$  and  $\frac{\partial \alpha_2}{\partial \mathbf{p}}$ . Following [70] the desired velocity vector  $\mathbf{u}' \in \mathbb{R}^3$  can be chosen as

$$\mathbf{u}' = -K_1 \frac{\partial V}{\partial \mathbf{p}} + K_2 \frac{\partial \alpha_1}{\partial \mathbf{p}} \times \frac{\partial \alpha_2}{\partial \mathbf{p}}, \quad (10.19)$$

where  $K_1$  and  $K_2$  are symmetric tuning matrices. It is shown in [70] that the dynamics  $\dot{\mathbf{r}} = \mathbf{u}'$  where  $\mathbf{u}'$  is given by equation (10.19), results in  $\mathbf{r}$  asymptotically converging to a trajectory that follows the intersection of  $\alpha_1$  and  $\alpha_2$  if  $K_1$  is positive definite, and where the definiteness of  $K_2$  determines the direction of travel along the path defined by the intersection.

The problem with equation (10.19) is that the magnitude of the desired velocity  $\mathbf{u}'$  may not equal  $V_a$ , the velocity of the Dubins airplane. Therefore  $\mathbf{u}'$  is normalized as

$$\mathbf{u} = V_a \frac{\mathbf{u}'}{\|\mathbf{u}'\|}. \quad (10.20)$$

Fortunately, the stability proof in [70] is still valid when  $\mathbf{u}'$  is normalized.

Setting the NED components of the velocity of the Dubins airplane model given in equation (9.22) to  $\mathbf{u} = (u_1, u_2, u_3)^\top$  gives

$$V_a \cos \chi^c \cos \gamma^c = u_1$$

$$\begin{aligned} V_a \sin \chi^c \cos \gamma^c &= u_2 \\ -V_a \sin \gamma^c &= u_3. \end{aligned}$$

Solving for the commanded flight-path angle  $\gamma^c$ , and the commanded course angle  $\chi^c$  results in the expressions

$$\gamma^c = -\text{sat}_{\bar{\gamma}} \left[ \sin^{-1} \left( \frac{u_3}{V_a} \right) \right] \quad (10.21)$$

$$\chi^c = \text{atan2}(u_2, u_1), \quad (10.22)$$

where  $\text{atan2}$  is the four quadrant inverse tangent, and where the saturation function is defined as

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

Sections 10.3.1 and 10.3.2 applies the framework described in this section to straight-line following and helix following, respectively.

### 10.3.1 Straight-line Paths

Similar to Section 10.1, let the straight-line path be given by Equation (10.1) where  $\mathbf{r}$  is a point on the line, and the direction of the line is given by the unit vector

$$\mathbf{q} = \begin{pmatrix} q_n \\ q_e \\ q_d \end{pmatrix} \triangleq \begin{pmatrix} \cos \chi_q \cos \gamma_q \\ \sin \chi_q \cos \gamma_q \\ -\sin \gamma_q \end{pmatrix}$$

where  $\chi_q$  is the desired course angle, and  $\gamma_q$  is the desired flight path angle. A unit vector that is perpendicular to the longitudinal plane defined by  $\mathbf{q}$  is given by

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

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

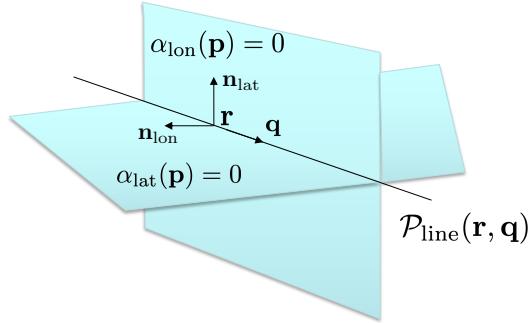
$$\mathbf{n}_{\text{lat}} \triangleq \mathbf{n}_{\text{lon}} \times \mathbf{q} = \begin{pmatrix} -\cos \chi_q \sin \gamma_q \\ -\sin \chi_q \sin \gamma_q \\ -\cos \gamma_q \end{pmatrix}.$$

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

$$\alpha_{\text{lon}}(\mathbf{p}) \triangleq \mathbf{n}_{\text{lon}}^\top (\mathbf{p} - \mathbf{r}) = 0 \quad (10.23)$$

$$\alpha_{\text{lat}}(\mathbf{p}) \triangleq \mathbf{n}_{\text{lat}}^\top (\mathbf{p} - \mathbf{r}) = 0. \quad (10.24)$$

Figure 10.8 shows  $\mathbf{q}$ ,  $\mathbf{r}$ , and the surfaces defined by  $\alpha_{\text{lon}}(\mathbf{p}) = 0$  and  $\alpha_{\text{lat}}(\mathbf{p}) = 0$ .



**Figure 10.8:** This figure shows how the straight-line path  $\mathcal{P}_{\text{line}}(\mathbf{r}, \mathbf{q})$  is defined by the intersection of the two surfaces given by  $\alpha_{\text{lon}}(\mathbf{p}) = 0$  and  $\alpha_{\text{lat}}(\mathbf{p}) = 0$ .

The gradients of  $\alpha_{\text{lon}}$  and  $\alpha_{\text{lat}}$  are given by

$$\frac{\partial \alpha_{\text{lon}}}{\partial \mathbf{p}} = \mathbf{n}_{\text{lon}}$$

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

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

$$\mathbf{u}'_{\text{line}} = K_1 \left( \mathbf{n}_{\text{lon}} \mathbf{n}_{\text{lon}}^\top + \mathbf{n}_{\text{lat}} \mathbf{n}_{\text{lat}}^\top \right) (\mathbf{p} - \mathbf{r}) + K_2 (\mathbf{n}_{\text{lon}} \times \mathbf{n}_{\text{lat}}). \quad (10.25)$$

### 10.3.2 Helical Paths

A time parameterized helical path is given by

$$\mathbf{r}(t) = \mathbf{c} + \begin{pmatrix} \rho \cos(\lambda t + \chi_h) \\ \rho \sin(\lambda t + \chi_h) \\ -t\rho \tan \gamma_h \end{pmatrix}, \quad (10.26)$$

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

helix at time  $t = 0$  is

$$\mathbf{r}(0) = \mathbf{c} + \begin{pmatrix} \rho \cos \chi_h \\ \rho_h \sin \chi_h \\ 0 \end{pmatrix},$$

and where  $\rho$  is the radius,  $\lambda = +1$  denotes a clockwise helix and  $\lambda_h = -1$  denotes a counter-clockwise helix,  $\gamma_h$  is the desired flight-path angle along the helix, and where  $\chi_h$  is the course angle at time  $t = 0$ .

To find two surfaces that define the helical path, the time parameterization in (10.26) needs to be eliminated. Equation (10.26) gives

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

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

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

Solving for  $t$  and plugging into the third component of (10.26) gives

$$r_d - c_d = -\lambda \rho \tan \gamma_h \left( \tan^{-1} \left( \frac{r_e - c_e}{r_n - c_n} \right) - \chi_h \right).$$

Therefore, normalizing these equations by  $\rho$  results in

$$\alpha_{\text{cyl}}(\mathbf{p}) = \left( \frac{p_n - c_n}{\rho} \right)^2 + \left( \frac{p_e - c_e}{\rho} \right)^2 - 1 \quad (10.27)$$

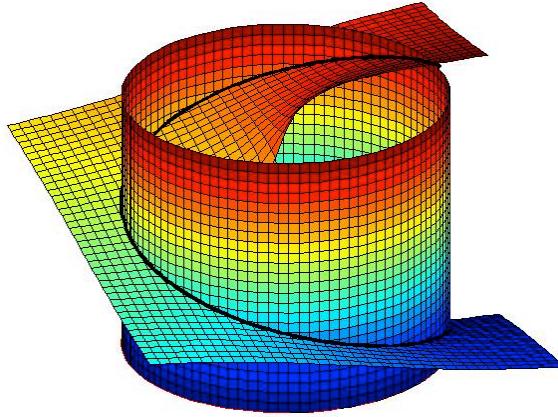
$$\alpha_{\text{pl}}(\mathbf{p}) = \left( \frac{p_d - c_d}{\rho} \right) + \lambda \tan \gamma_h \left( \tan^{-1} \left( \frac{p_e - c_e}{p_n - c_n} \right) - \chi_h \right). \quad (10.28)$$

Normalization by  $\rho$  makes the gains on the resulting control strategy invariant to the size of the orbit.

A helical path is then defined as

$$\mathcal{P}_{\text{helix}}(\mathbf{c}, \rho, \lambda, \chi_h, \gamma_h) = \{ \mathbf{p} \in \mathbb{R}^3 : \alpha_{\text{cyl}}(\mathbf{p}) = 0 \text{ and } \alpha_{\text{pl}}(\mathbf{p}) = 0 \}. \quad (10.29)$$

The two surfaces  $\alpha_{\text{cyl}}(\mathbf{p}) = 0$  and  $\alpha_{\text{pl}}(\mathbf{p}) = 0$  are shown in figure 10.9 for parameters  $\mathbf{c} = (0, 0, 0)^\top$ ,  $\rho = 30$  m,  $\lambda = +1$ ,  $\chi_h = 0$ , and  $\gamma_h = \frac{15\pi}{180}$  rad. The associated helical path is the intersection of the two surfaces.



**Figure 10.9:** A helical path for parameters  $\mathbf{c} = (0, 0, 0)^\top$ ,  $\rho = 30$  m,  $\lambda = +1$ ,  $\chi_h = 0$ , and  $\gamma_h = \frac{15\pi}{180}$  rad.

The gradients of  $\alpha_{\text{cyl}}$  and  $\alpha_{\text{pl}}$  are given by

$$\frac{\partial \alpha_{\text{cyl}}}{\partial \mathbf{p}} = \left( 2 \frac{p_n - c_n}{\rho}, \quad 2 \frac{p_e - c_e}{\rho}, \quad 0 \right)^\top \quad (10.30)$$

$$\frac{\partial \alpha_{\text{pl}}}{\partial \mathbf{p}} = \left( \lambda \tan \gamma_h \frac{-(p_e - c_e)}{(p_n - c_n)^2 + (p_e - c_e)^2}, \quad \lambda \tan \gamma_h \frac{(p_n - c_e)}{(p_n - c_n)^2 + (p_e - c_e)^2}, \quad \frac{1}{\rho} \right)^\top. \quad (10.31)$$

Before normalization, the desired velocity vector is given by

$$\mathbf{u}'_{\text{helix}} = K_1 \left( \alpha_{\text{cyl}} \frac{\partial \alpha_{\text{cyl}}}{\partial \mathbf{p}} + \alpha_{\text{pl}} \frac{\partial \alpha_{\text{pl}}}{\partial \mathbf{p}} \right) + \lambda K_2 \left( \frac{\partial \alpha_{\text{cyl}}}{\partial \mathbf{p}} \times \frac{\partial \alpha_{\text{pl}}}{\partial \mathbf{p}} \right), \quad (10.32)$$

where

$$\frac{\partial \alpha_{\text{cyl}}}{\partial \mathbf{p}} \times \frac{\partial \alpha_{\text{pl}}}{\partial \mathbf{p}} = \frac{2}{\rho} \left( \frac{p_e - c_e}{\rho}, \quad -\frac{p_n - c_n}{\rho}, \quad \lambda \tan \gamma_h \right)^\top.$$

## 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 5, 6 and 7. Each algorithm uses the wrap function, which is listed in algorithm 4.

**Algorithm 4** Wrap:

---

 $\hat{\chi}_{\text{new}} = \text{wrap}(\hat{\chi}, \chi)$ 


---

**Input:** Current course  $\chi \in \mathbb{R}$ , desired or commanded course  $\hat{\chi} \in (-\pi, \pi]$ .

- 1:  $\hat{\chi}_{\text{new}} \leftarrow \hat{\chi}$
  - 2: **while**  $\hat{\chi}_{\text{new}} - \chi < -\pi$  **do**
  - 3:      $\hat{\chi}_{\text{new}} \leftarrow \hat{\chi}_{\text{new}} + 2\pi$
  - 4: **end while**
  - 5: **while**  $\hat{\chi}_{\text{new}} - \chi > \pi$  **do**
  - 6:      $\hat{\chi}_{\text{new}} \leftarrow \hat{\chi}_{\text{new}} - 2\pi$
  - 7: **end while**
  - 8: **return**  $\hat{\chi}_{\text{new}}$
- 

**Algorithm 5** Straight-line following:

---

 $[h^c, \chi^c] = \text{followStraightLine}(\mathbf{r}, \mathbf{q}, \mathbf{p}, \chi)$ 


---

**Input:** Path parameters:  $\mathbf{r} = (r_n, r_e, r_d)^\top$  and  $\mathbf{q} = (q_n, q_e, q_d)^\top$ ,

**Input:** MAV states:  $\mathbf{p} = (p_n, p_e, p_d)^\top$  and  $\chi$ ,

**Input:** Control gains:  $\chi_\infty$  and  $k_{\text{path}}$ .

- 1: Compute commanded altitude  $h^c$  using [equation \(10.6\)](#).
  - 2:  $\chi_q \leftarrow \text{atan2}(q_e, q_n)$
  - 3:  $\chi_q \leftarrow \text{wrap}(\chi_q, \chi)$
  - 4:  $e_{py} \leftarrow -\sin \chi_q (p_n - r_n) + \cos \chi_q (p_e - r_e)$
  - 5: Compute commanded course angle  $\chi^c$  using [equation \(10.9\)](#).
  - 6: **return**  $h^c, \chi^c$
- 

**Algorithm 6** Circular orbit following:  $[h^c, \chi^c] = \text{followOrbit}(\mathbf{c}, \rho, \lambda, \mathbf{p}, \chi)$ 

**Input:** Orbit parameters:  $\mathbf{c} = (c_n, c_e, c_d)^\top$ ,  $\rho$ , and  $\lambda$ ,

**Input:** MAV states:  $\mathbf{p} = (p_n, p_e, p_d)^\top$  and  $\chi$ ,

**Input:** Control gain:  $k_{\text{orbit}}$ .

- 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:  $\varphi \leftarrow \text{wrap}(\varphi, \chi)$
  - 5: Compute commanded course angle  $\chi^c$  using [equation \(10.15\)](#).
  - 6: **return**  $h^c, \chi^c$
-

**Algorithm 7** Helix following:  $[\gamma^c, \chi^c] = \text{followHelix}(\mathbf{c}, \rho, \lambda, \chi_h, \gamma_h, \mathbf{p}, \chi)$ 

**Input:** Helix parameters:  $\mathbf{c} = (c_n, c_e, c_d)^\top$ ,  $\rho$ ,  $\lambda$ ,  $\chi_h$ , and  $\gamma_h$ ,

**Input:** MAV states:  $\mathbf{p} = (p_n, p_e, p_d)^\top$  and  $\chi$ ,

**Input:** Control gains:  $K_1$  and  $K_2$ .

- 1: Compute  $\alpha_{\text{cyl}}(\mathbf{p})$ ,  $\alpha_{\text{pl}}(\mathbf{p})$ ,  $\frac{\partial \alpha_{\text{cyl}}}{\partial \mathbf{p}}(\mathbf{p})$ , and  $\frac{\partial \alpha_{\text{pl}}}{\partial \mathbf{p}}(\mathbf{p})$  according to equations (10.27), (10.28), (10.30), and (10.31), respectively.
- 2: Compute  $\mathbf{u}'_{\text{helix}}$  from equation (10.32).
- 3: Compute  $\mathbf{u}_{\text{helix}}$  from equation (10.20).
- 4: Compute  $\gamma^c$  and  $\chi^c$  from equations (10.21) and (10.22) respectively.
- 5:  $\chi^c \leftarrow \text{wrap}(\chi^c, \chi)$
- 6: **return**  $\gamma^c$  and  $\chi^c$ .

**NOTES AND REFERENCES**

The methods described in sections 10.1 and 10.2 are variations on those described in [71, 72, 73] 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 [71] is given in [74], 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 [74], 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., [75]). It has also been suggested in [76] that potential fields can be used in UAV navigation for obstacle and collision avoidance applications. The method of [76] provides a way for groups of UAVs to use the gradient 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 [77]. 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 [78] 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 [25] presents a path-following method for UAVs that provides a constant line of sight between the UAV and an observation target.

The approach discussed in Section 10.3 is based on the results in [70], and was first presented in [69].

## 10.5 DESIGN PROJECT

### MODIFIED MATERIAL:

- 10.1 Download the simulation files for this chapter from the web.
- 10.2 Implement algorithm 5 for straight-line following and tune the parameters  $k_{\text{path}}$ ,  $\chi^\infty$ , first in zero wind, and then in significant constant wind (e.g.,  $w_n = 5$ ,  $w_e = 5$ ).
- 10.3 Implement algorithm 6 and tune the parameter  $k_{\text{orbit}}$ , first in zero wind, and then in significant constant wind.
- 10.4 Implement algorithm 7 and tune the parameter  $K_1$  and  $K_2$ , first in zero wind, and then in significant constant wind.



## 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

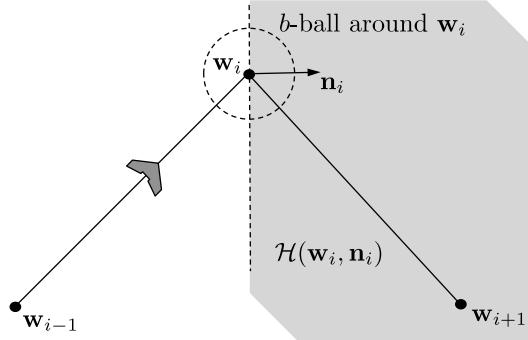
$$\mathcal{W} = \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_N\}, \quad (11.1)$$

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  $\mathbf{w}_i$ . **MODIFIED MATERIAL:** In other words, the guidance algorithm would switch at the first time instant when

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

where  $b$  is the size of the ball and  $\mathbf{p}(t)$  is the location of the MAV. 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 8. The first time that the algorithm is executed, the waypoint pointer is initialized to  $i = 2$  in 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 8** 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** **return**  $\mathbf{r}, \mathbf{q} = \mathbf{q}_{i-1}$  at each time step

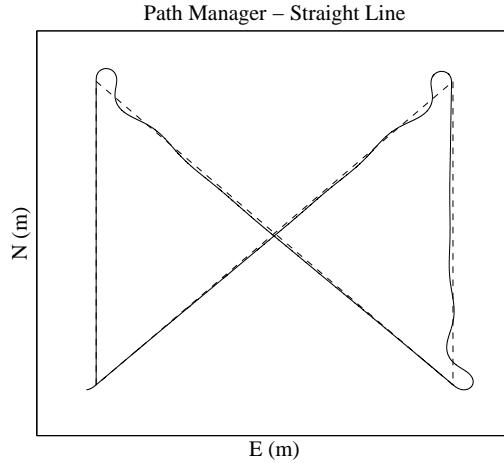
---

Algorithm 8 will produce paths like that shown in figure 11.2. The advantages of algorithm 8 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. **MODIFIED MATERIAL:** 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

$$\varrho \triangleq \cos^{-1} \left( -\mathbf{q}_{i-1}^\top \mathbf{q}_i \right). \quad (11.3)$$

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  $\varrho$  is given by  $R/\sin \frac{\varrho}{2} - R$ .

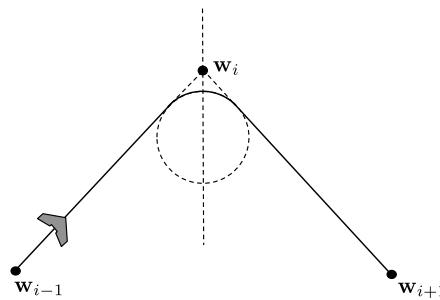


**Figure 11.2:** Path generated using the path-following approach given in algorithm 8. The MAV follows the straight-line path until reaching the waypoint, and then maneuvers onto the next straight-line section.

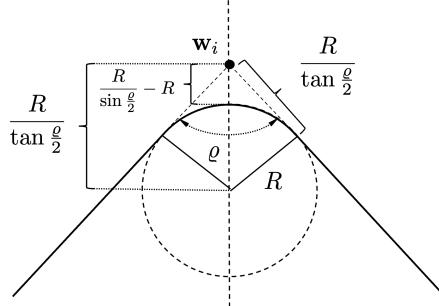
To implement the fillet maneuver using the path-following algorithms described in chapter 10, we will follow the straight-line segment  $\overline{w_{i-1}w_i}$  until entering the half plane  $\mathcal{H}_1$  shown in figure 11.5. The right-handed orbit of radius  $R$  is then followed until entering the half plane  $\mathcal{H}_2$  shown in figure 11.5, at which point the straight-line segment  $\overline{w_iw_{i+1}}$  is followed.

As shown in figure 11.5, the center of the fillet is given by

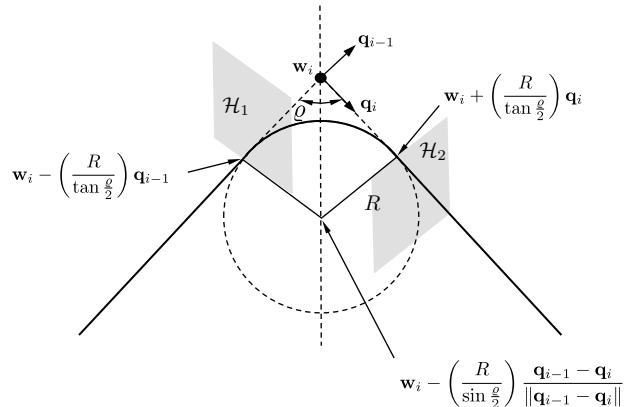
$$\mathbf{c} = \mathbf{w}_i - \left( \frac{R}{\sin \frac{\varrho}{2}} \right) \frac{\mathbf{q}_{i-1} - \mathbf{q}_i}{\|\mathbf{q}_{i-1} - \mathbf{q}_i\|}.$$



**Figure 11.3:** The transition from straight-line path  $\overline{w_{i-1}w_i}$  to  $\overline{w_iw_{i+1}}$  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

$$\mathbf{r}_2 = \mathbf{w}_i + \left( \frac{R}{\tan \frac{\varrho}{2}} \right) \mathbf{q}_i$$

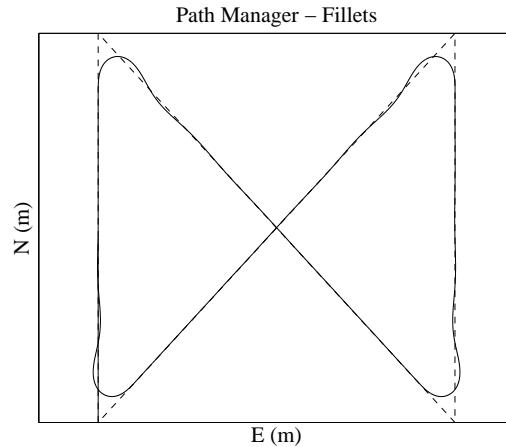
and the normal vector  $\mathbf{q}_i$ .

The algorithm for maneuvering along the waypoint path  $\mathcal{W}$  using fillets to smooth between the straight-line segments is given by algorithm 9. 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  $\varrho$  are computed in lines 4–6.

**MODIFIED MATERIAL:** 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  $\mathbf{q} = \mathbf{q}_{i-1}$  that 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  $\mathcal{H}_1$ , 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 9 produces paths like that shown in figure 11.6.



**Figure 11.6:** An example of the types of flight paths produced by algorithm 9.

One of the disadvantages of the fillet method as given in algorithm 9 is that the path length is changed when fillets are inserted. For certain applications, like the cooperative timing problems discussed in [79], 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 9** 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 return flag,  $\mathbf{r}, \mathbf{q}, \mathbf{c}, \rho, \lambda$ 

```

---

**MODIFIED MATERIAL:** To be precise, let

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

be defined as the length of the waypoint path  $\mathcal{W}$ . Define  $|\mathcal{W}|_F$  as the path length of the fillet-corrected waypoint path that will be obtained using algorithm 9. 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  $\varrho_i$  is given in equation (11.3).

## 11.2 DUBINS PATHS

### 11.2.1 Definition of Dubins Path

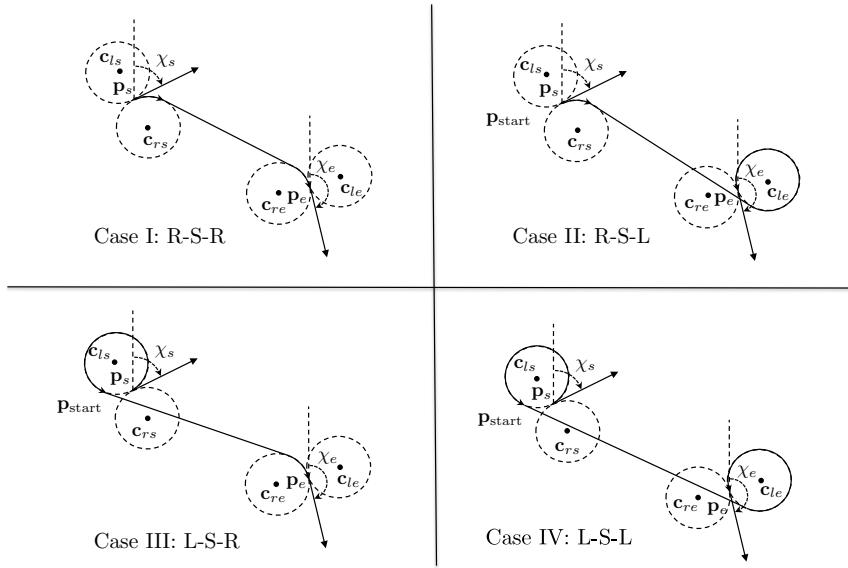
This section focuses on so-called Dubins paths, where, rather than following a waypoint path, the objective is to transition from one configuration (position and course) to another. It was shown in [80] that for a vehicle with kinematics given by

$$\begin{aligned} \dot{p}_n &= V \cos \vartheta \\ \dot{p}_e &= V \sin \vartheta \\ \dot{\vartheta} &= u, \end{aligned}$$

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-groundspeed 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  $(\mathbf{p}, \chi)$ , where  $\mathbf{p}$  is inertial position and  $\chi$  is course angle.

Given a start configuration denoted as  $(\mathbf{p}_s, \chi_s)$  and an end configuration denoted as  $(\mathbf{p}_e, \chi_e)$ , 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 the case with the shortest path length.



**Figure 11.7:** Given a start configuration  $(\mathbf{p}_s, \chi_s)$ , an end configuration  $(\mathbf{p}_e, \chi_e)$ , 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

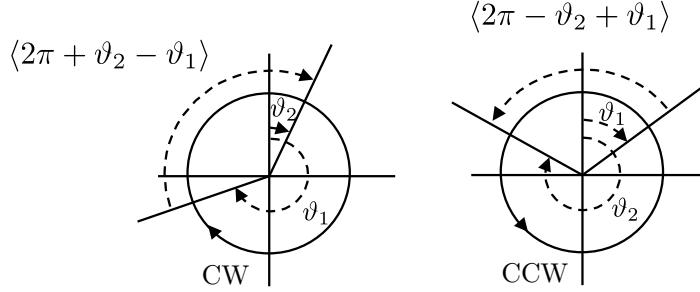
To determine the Dubins path, it is necessary to compute the path length for the four cases shown in figure 11.7. In this section, we will derive explicit formulas for the path length for each case. Given the position  $\mathbf{p}$ , the course

$\chi$ , 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  $\vartheta_1$  and  $\vartheta_2$  for clockwise ( $CW$ ) and counter clockwise ( $CCW$ ) circles.

assume that both  $\vartheta_1$  and  $\vartheta_2$  are between 0 and  $2\pi$ . For clockwise circles, the angular distance between  $\vartheta_1$  and  $\vartheta_2$  is given by

$$|\vartheta_2 - \vartheta_1|_{CW} \triangleq \langle 2\pi + \vartheta_2 - \vartheta_1 \rangle, \quad (11.7)$$

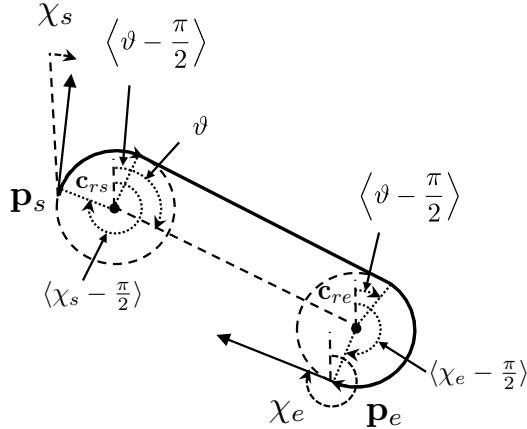
where

$$\langle \varphi \rangle \triangleq \varphi \bmod 2\pi.$$

Similarly, for counter clockwise circles, we get

$$|\vartheta_2 - \vartheta_1|_{CCW} \triangleq \langle 2\pi - \vartheta_2 + \vartheta_1 \rangle. \quad (11.8)$$

*Case I: R-S-R*



**Figure 11.9:** Dubins path, case I.

The geometry for case I is shown in figure 11.9, where  $\vartheta$  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

$$R\langle 2\pi + \langle \vartheta - \frac{\pi}{2} \rangle - \langle \chi_s - \frac{\pi}{2} \rangle \rangle.$$

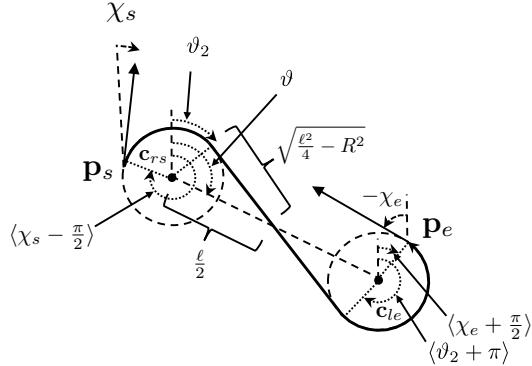
Similarly, using equation (11.7), the angular distance traveled along  $\mathbf{c}_{re}$  is given by

$$R\langle 2\pi + \langle \chi_e - \frac{\pi}{2} \rangle - \langle \vartheta - \frac{\pi}{2} \rangle \rangle.$$

The total path length for case I is therefore given by

$$L_1 = \|\mathbf{c}_{rs} - \mathbf{c}_{re}\| + R\langle 2\pi + \langle \vartheta - \frac{\pi}{2} \rangle - \langle \chi_s - \frac{\pi}{2} \rangle \rangle + R\langle 2\pi + \langle \chi_e - \frac{\pi}{2} \rangle - \langle \vartheta - \frac{\pi}{2} \rangle \rangle. \quad (11.9)$$

*Case II: R-S-L*



**Figure 11.10:** Dubins path, case II.

The geometry for case II is shown in figure 11.10, where  $\vartheta$  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

$$L_2 = \sqrt{\ell^2 - 4R^2} + R \langle 2\pi + \langle \vartheta_2 \rangle - \langle \chi_s - \frac{\pi}{2} \rangle \rangle + R \langle 2\pi + \langle \vartheta_2 + \pi \rangle - \langle \chi_e + \frac{\pi}{2} \rangle \rangle. \quad (11.10)$$

## Case III: L-S-R

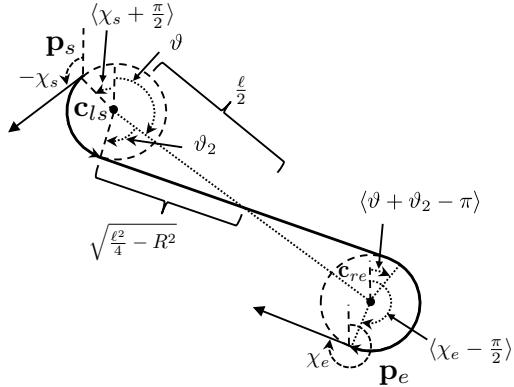


Figure 11.11: Dubins path, case III.

The geometry for case III is shown in figure 11.11, where  $\vartheta$  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

$$R\langle 2\pi + \langle \chi_s + \frac{\pi}{2} \rangle - \langle \vartheta + \vartheta_2 \rangle \rangle.$$

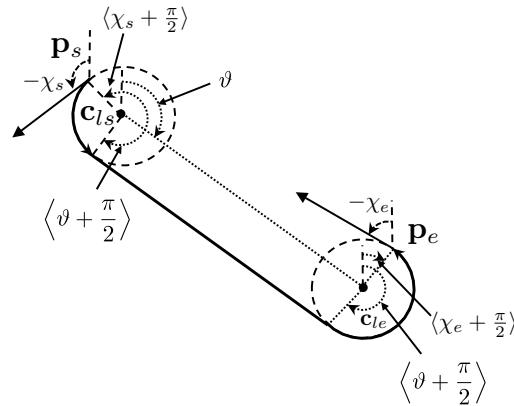
Similarly, using equation (11.7), the angular distance traveled along  $\mathbf{c}_{re}$  is given by

$$R\langle 2\pi + \langle \chi_e - \frac{\pi}{2} \rangle - \langle \vartheta + \vartheta_2 - \pi \rangle \rangle.$$

The total path length for case III is therefore given by

$$L_3 = \sqrt{\ell^2 - 4R^2} + R\langle 2\pi + \langle \chi_s + \frac{\pi}{2} \rangle - \langle \vartheta + \vartheta_2 \rangle \rangle + R\langle 2\pi + \langle \chi_e - \frac{\pi}{2} \rangle - \langle \vartheta + \vartheta_2 - \pi \rangle \rangle. \quad (11.11)$$

*Case IV: L-S-L*



**Figure 11.12:** Dubins path, case IV.

The geometry for case IV is shown in figure 11.12, where  $\vartheta$  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

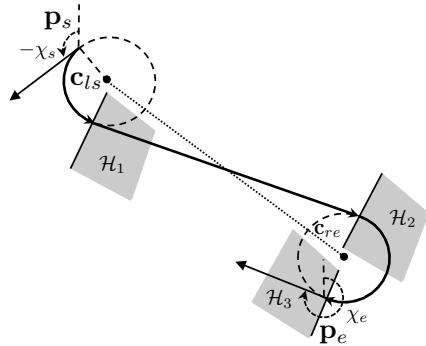
$$R\langle 2\pi + \langle \vartheta + \frac{\pi}{2} \rangle - \langle \chi_e + \frac{\pi}{2} \rangle \rangle.$$

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  $\lambda_s$ , the end circle  $\mathbf{c}_e$ , the direction of the end circle  $\lambda_e$ , 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  $(\mathbf{p}_s, \chi_s)$ , the end configuration  $(\mathbf{p}_e, \chi_e)$ , and the radius  $R$  are computed in algorithm 10. 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  $\vartheta$  about the  $z$ -axis and  $\vec{\epsilon}_1 = (1, 0, 0)^\top$ .

#### MODIFIED MATERIAL:

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 11. In line 5 the Dubins parameters are found for the current waypoint segment using algorithm 10. 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 8–10. The start circle is then followed in `state=2` until the MAV has crossed half plane  $\mathcal{H}_1$ , as shown in lines 11–14. After crossing into  $\mathcal{H}_1$ , the straight-line segment of the Dubins path is followed in `state=3` as shown in lines 15–19. Line 17 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

**Algorithm 10** 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  $(\mathbf{p}_s, \chi_s)$ , end configuration  $(\mathbf{p}_e, \chi_e)$ , radius  $R$

**Require:**  $\|\mathbf{p}_s - \mathbf{p}_e\| \geq 3R$

**Require:**  $R$  is larger than minimum turn radius of MAV

- 1:  $\mathbf{c}_{rs} \leftarrow \mathbf{p}_s + R\mathcal{R}_z\left(\frac{\pi}{2}\right)(\cos \chi_s, \sin \chi_s, 0)^\top$
  - 2:  $\mathbf{c}_{ls} \leftarrow \mathbf{p}_s + R\mathcal{R}_z\left(\frac{-\pi}{2}\right)(\cos \chi_s, \sin \chi_s, 0)^\top$
  - 3:  $\mathbf{c}_{re} \leftarrow \mathbf{p}_e + R\mathcal{R}_z\left(\frac{\pi}{2}\right)(\cos \chi_e, \sin \chi_e, 0)^\top$
  - 4:  $\mathbf{c}_{le} \leftarrow \mathbf{p}_e + R\mathcal{R}_z\left(-\frac{\pi}{2}\right)(\cos \chi_e, \sin \chi_e, 0)^\top$
  - 5: Compute  $L_1, L_2, L_3$ , and  $L_4$  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:    $\mathbf{z}_2 \leftarrow \mathbf{c}_e + R\mathcal{R}_z\left(\frac{\pi}{2}\right)\mathbf{q}_1$
  - 33: **end if**
  - 34:  $\mathbf{z}_3 \leftarrow \mathbf{p}_e$
  - 35:  $\mathbf{q}_3 \leftarrow \mathcal{R}_z(\chi_e)\mathbf{e}_1$
-

**Algorithm 11** 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:**  $N \geq 3$

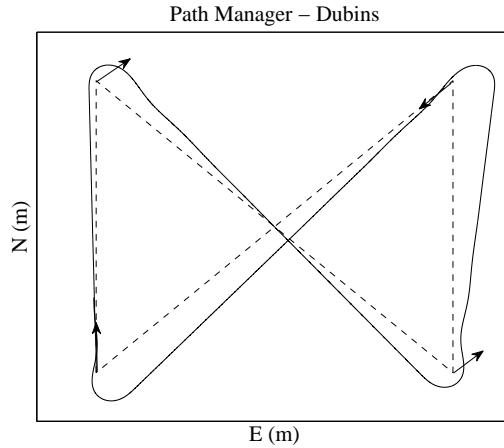
```

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:  $(L, \mathbf{c}_s, \lambda_s, \mathbf{c}_e, \lambda_e, \mathbf{z}_1, \mathbf{q}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{q}_3) \leftarrow$ 
5: findDubinsParameters( $\mathbf{w}_{i-1}, \chi_{i-1}, \mathbf{w}_i, \chi_i, R$ )
6: if state = 1 then
7:   flag  $\leftarrow 2$ ,  $\mathbf{c} \leftarrow \mathbf{c}_s$ ,  $\rho \leftarrow R$ ,  $\lambda \leftarrow \lambda_s$ 
8:   if  $\mathbf{p} \in \mathcal{H}(\mathbf{z}_1, -\mathbf{q}_1)$  then
9:     state  $\leftarrow 2$ 
10:  end if
11: else if state = 2 then
12:   if  $\mathbf{p} \in \mathcal{H}(\mathbf{z}_1, \mathbf{q}_1)$  then
13:     state  $\leftarrow 3$ 
14:   end if
15: else if state = 3 then
16:   flag  $\leftarrow 1$ ,  $\mathbf{r} \leftarrow \mathbf{z}_1$ ,  $\mathbf{q} \leftarrow \mathbf{q}_1$ 
17:   if  $\mathbf{p} \in \mathcal{H}(\mathbf{z}_2, \mathbf{q}_1)$  then
18:     state  $\leftarrow 4$ 
19:   end if
20: else if state = 4 then
21:   flag  $\leftarrow 2$ ,  $\mathbf{c} \leftarrow \mathbf{c}_e$ ,  $\rho \leftarrow R$ ,  $\lambda \leftarrow \lambda_e$ 
22:   if  $\mathbf{p} \in \mathcal{H}(\mathbf{z}_3, -\mathbf{q}_3)$  then
23:     state  $\leftarrow 5$ 
24:   end if
25: else if state = 5 then
26:   if  $\mathbf{p} \in \mathcal{H}(\mathbf{z}_3, \mathbf{q}_3)$  then
27:     state  $\leftarrow 1$ 
28:      $i \leftarrow (i + 1)$  until  $i = N$ 
29:      $(L, \mathbf{c}_s, \lambda_s, \mathbf{c}_e, \lambda_e, \mathbf{z}_1, \mathbf{q}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{q}_3) \leftarrow$ 
30: findDubinsParameters( $\mathbf{w}_{i-1}, \chi_{i-1}, \mathbf{w}_i, \chi_i, R$ )
31:   end if
32: end ifreturn flag,  $\mathbf{r}$ ,  $\mathbf{q}$ ,  $\mathbf{c}$ ,  $\rho$ ,  $\lambda$ .

```

---

entering  $\mathcal{H}_3$  as shown in lines 20–24. After the MAV has entered into  $\mathcal{H}_3$ , as detected in line 26, the waypoints are cycled and new Dubins parameters are found in lines 28–30. Figure 11.14 shows an example of a path generated using algorithm 11.



**Figure 11.14:** An example of the types of flight paths produced by algorithm 11.

### 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 [81]. Dubins paths were introduced in [80]. 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 [82] 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 [83], 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 [79], 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 [81].

#### 11.4 DESIGN PROJECT

The objective of this assignment is to implement algorithms 8 and 9 for following a set of waypoints denoted as  $\mathcal{W}$ , and algorithm 11 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 8 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 9 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 11 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 [84, 85, 86], and *reactive* motion planning, which uses behavioral methods to react to local sensor information [87, 88]. 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 ?? we will address coverage problems, where the objective is to plan a waypoint path so that

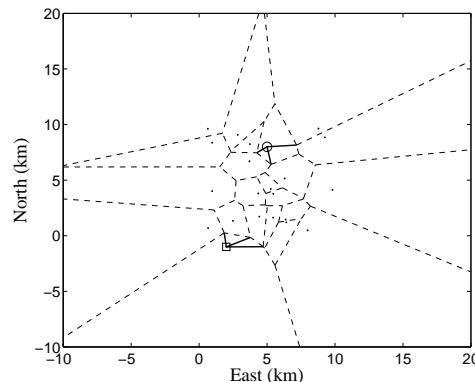
the MAV covers all of the area in a certain region. The output of the path planning algorithms developed in this chapter will be a sequence of either waypoints or configurations (waypoint plus orientation) and will therefore interface with the path management algorithms developed in [chapter 11](#).

## 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

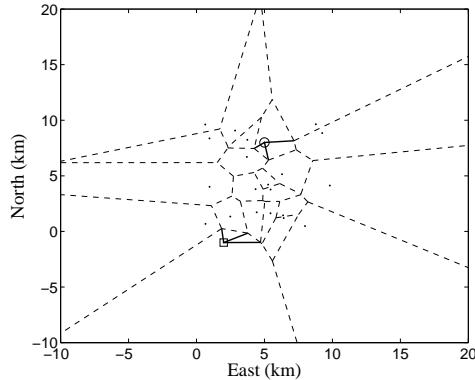
Voronoi graph may lead to unnecessarily long excursions. Finally, note that for two obstacle points positioned close to each other, such as those near the center of figure 12.1, the Voronoi graph produces an edge between the two points; however, since the edge is so close to the points, the corresponding 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 [89, 90, 91].

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

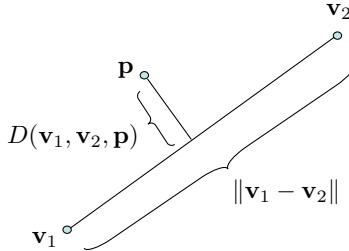
$$V^+ = V \cup \{\mathbf{p}_s, \mathbf{p}_e\},$$

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

$$\begin{aligned} D(\mathbf{v}_1, \mathbf{v}_2, \mathbf{p}) &\triangleq \min_{\sigma \in [0,1]} \|\mathbf{p} - \mathbf{w}(\sigma)\| \\ &= \min_{\sigma \in [0,1]} \sqrt{(\mathbf{p} - \mathbf{w}(\sigma))^\top (\mathbf{p} - \mathbf{w}(\sigma))} \\ &= \min_{\sigma \in [0,1]} \left[ \mathbf{p}^\top \mathbf{p} - 2(1 - \sigma)\sigma \mathbf{p}^\top \mathbf{v}_1 - \sigma \mathbf{p}^\top \mathbf{v}_2 \right. \\ &\quad \left. + (1 - \sigma)^2 \mathbf{v}_1^\top \mathbf{v}_1 + 2(1 - \sigma)\sigma \mathbf{v}_1^\top \mathbf{v}_2 + \sigma^2 \mathbf{v}_2^\top \mathbf{v}_2 \right]^{\frac{1}{2}} \\ &= \min_{\sigma \in [0,1]} \left[ \|\mathbf{p} - \mathbf{v}_1\|^2 + 2\sigma(\mathbf{p} - \mathbf{v}_1)^\top (\mathbf{v}_1 - \mathbf{v}_2) \right. \\ &\quad \left. + \sigma^2 \|\mathbf{v}_1 - \mathbf{v}_2\|^2 \right]^{\frac{1}{2}}. \end{aligned}$$

**MODIFIED MATERIAL:** If  $\sigma$  is unconstrained, then its optimizing value is

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

and

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

If we define

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

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

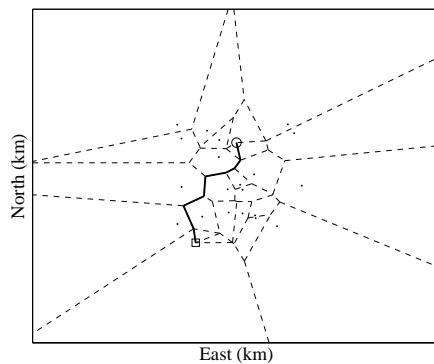
$$D(\mathbf{v}_1, \mathbf{v}_2, \mathcal{Q}) = \min_{\mathbf{p} \in \mathcal{Q}} D'(\mathbf{v}_1, \mathbf{v}_2, \mathbf{p}).$$

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 equation (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 [91]. A well-known algorithm with readily available code is Dijkstra's algorithm [92], 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 12. 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 12 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 12** Plan Voronoi Path:  $\mathcal{W} = \text{planVoronoi}(\mathcal{Q}, \mathbf{p}_s, \mathbf{p}_e)$ 

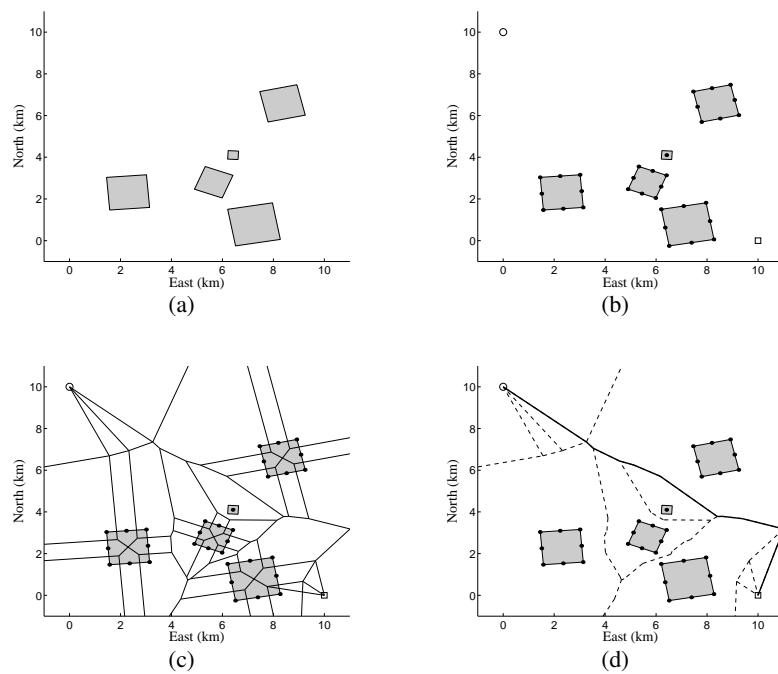

---

**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:  $\mathcal{W} = \text{DijkstraSearch}(V^+, E^+, \mathbf{J})$  **return**  $\mathcal{W}$
- 

One of the disadvantages of the Voronoi method described in [algorithm 12](#) 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.

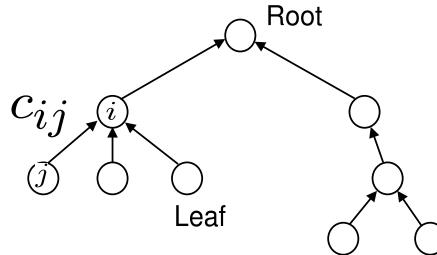


**Figure 12.5:** (a) An obstacle field with non-point obstacles. (b) The first step in using the Voronoi method to construct a path through the obstacle field is to insert points around the perimeter of the obstacles. (c) The resulting Voronoi graph includes many infeasible links contained inside the obstacles or that terminate on the obstacles. (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 13 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 2–16 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 8 and the cost matrix is updated in line 9. 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 12–13, and the algorithm ends in line 17 by returning the shortest waypoint path in  $G$ .

The result of implementing algorithm 13 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 13 sometimes wander needlessly and that eliminating some nodes may result in a more efficient path. Algorithm 14 gives a simple scheme for smoothing the paths generated by algorithm 13. The basic idea is to remove intermediate nodes if a feasible path still exists. The result of applying algorithm 14 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 [93], 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 13** 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  $V \leftarrow V \cup \{\mathbf{v}^+\}$ ,
- 8:          $E \leftarrow E \cup \{(\mathbf{v}^*, \mathbf{v}^+)\}$
- 9:         Update edge costs as  $C[(\mathbf{v}^*, \mathbf{v}^+)] \leftarrow \text{pathLength}(\mathbf{v}^*, \mathbf{v}^+)$
- 10:        **if**  $\text{existFeasiblePath}(\mathcal{T}, \mathbf{v}^+, \mathbf{p}_e)$  **then**
- 11:           Update graph  $G = (V, E)$  as  $V \leftarrow V \cup \{\mathbf{p}_e\}$ ,
- 12:            $E \leftarrow E \cup \{(\mathbf{v}^+, \mathbf{p}_e)\}$
- 13:           Update edge costs as  $C[(\mathbf{v}^+, \mathbf{p}_e)] \leftarrow \text{pathLength}(\mathbf{v}^+, \mathbf{p}_e)$
- 14:        **end if**
- 15:     **end if**
- 16: **end while**
- 17:  $\mathcal{W} = \text{findShortestPath}(G, C)$  **return**  $\mathcal{W}$

---



---

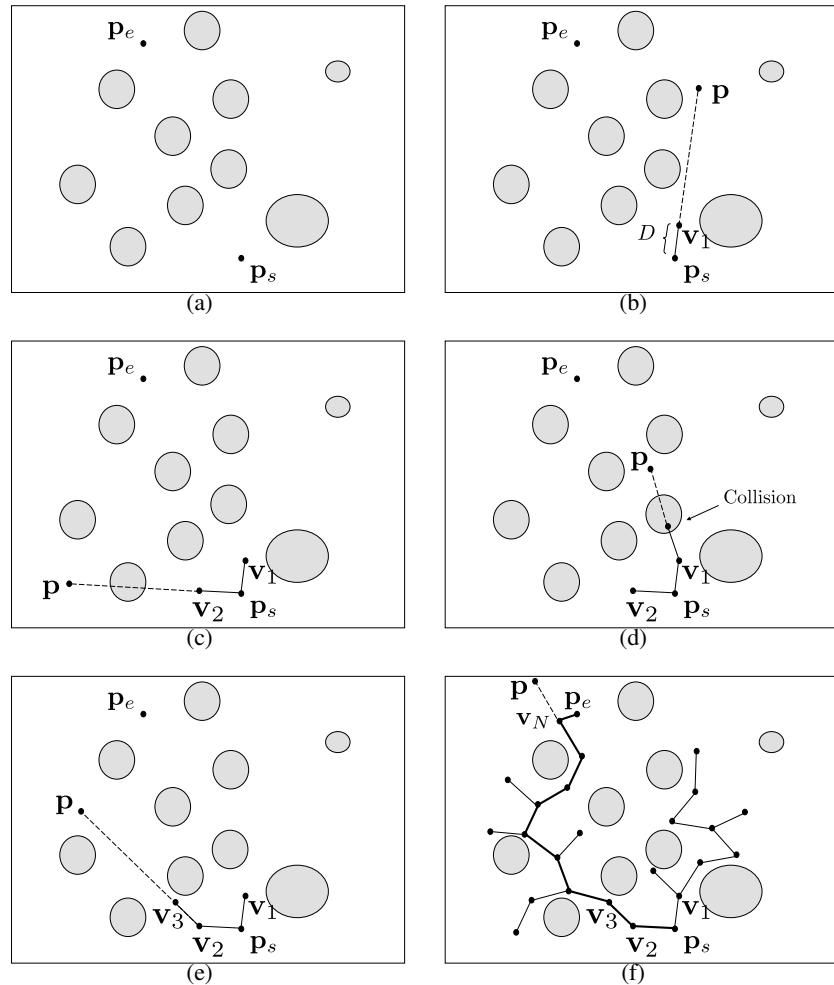
**Algorithm 14** Smooth RRT Path:  $(\mathcal{W}_s, C_s) = \text{smoothRRT}(\mathcal{T}, \mathcal{W}, C)$ 

---

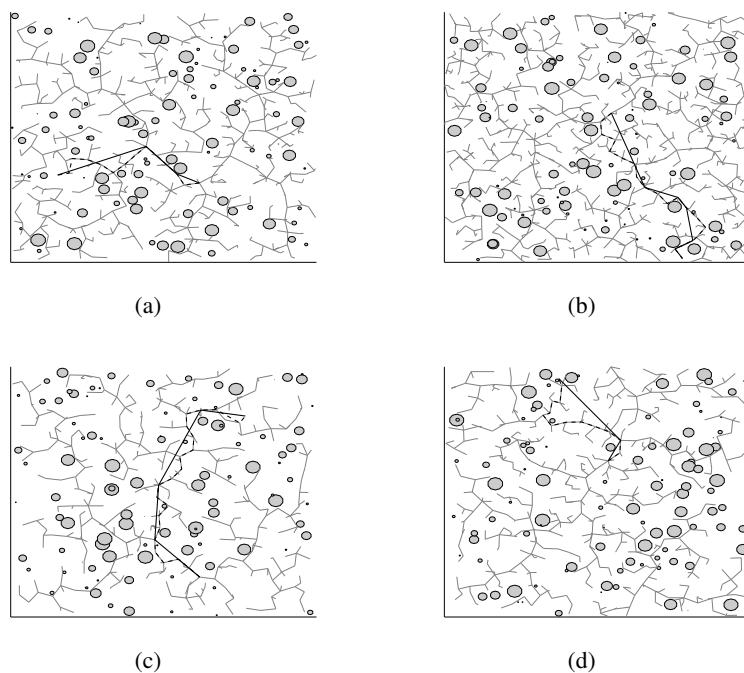
**Input:** Terrain map  $\mathcal{T}$ , waypoint path  $\mathcal{W} = \{\mathbf{w}_1, \dots, \mathbf{w}_N\}$ , cost matrix  $C$

- 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:         $i \leftarrow j$
- 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:  $C_s[(\mathbf{w}_i, \mathbf{w}_N)] \leftarrow \text{pathLength}(\mathbf{w}_i, \mathbf{w}_N)$  **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 RRT graph.



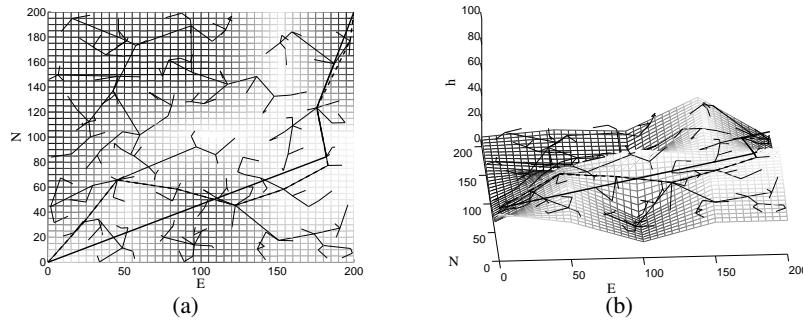
**Figure 12.8:** The results of algorithm 13 for four randomly generated obstacle fields and randomly generated start and end nodes are indicated by dashed lines. The smoothed paths generated by algorithm 14 are indicated by the solid lines.

### RRT Waypoint Planning over 3-D Terrain

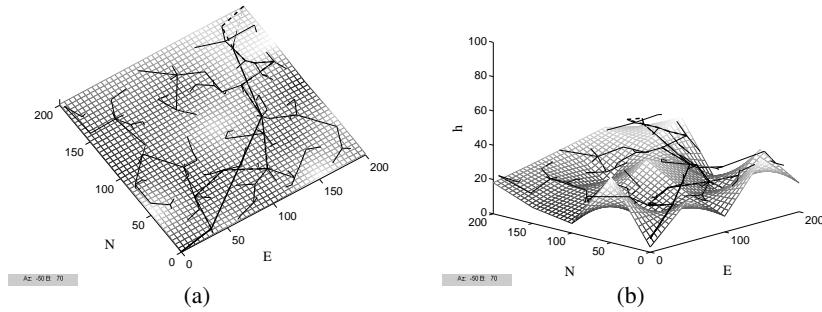
In this section we will consider the extension of the basic RRT algorithm to planning waypoint paths over 3-D terrain. We will assume that the terrain  $\mathcal{T}$  can be queried for the altitude of the terrain at any north-east position. The primary question that must be answered to extend the basic 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 13 will be a path that follows the terrain at a fixed altitude  $h_{AGL}$ . However, the smoothing step represented by algorithm 14 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 13 and 14 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 13, and the thick solid line is the smoothed path generated by algorithm 14.



**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 13, 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 13, and the thick solid line is the smoothed path. The RRT graph is generated at a fixed height above the terrain.

#### 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.

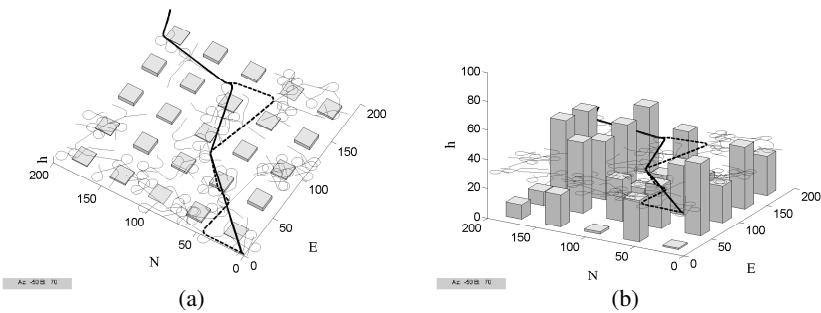
We will generate a random configuration as follows:

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.

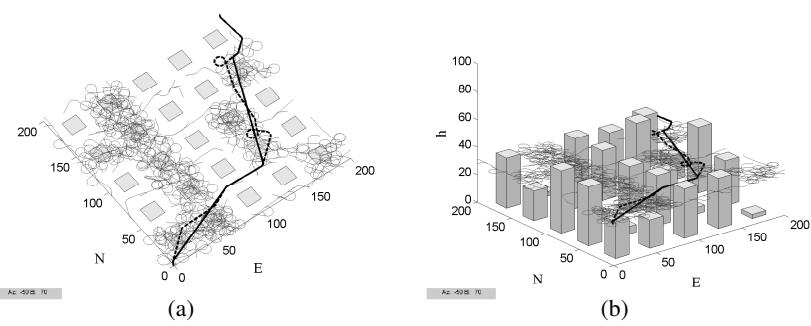
The RRT algorithm is then implemented as described in [algorithm 13](#), where the function `pathLength` returns the length of the Dubins path between configurations.

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 13, and the thick solid line is the smoothed path generated by algorithm 14.



**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 13, 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 13, and the thick solid line is the smoothed path. The RRT graph is generated at a fixed height above the terrain.

## 12.2 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 [94], Choset, et al. [95], and LaValle [96], which contain thorough reviews of related path planning research. An introduction to the Voronoi graph is contained in [90] and early applications of Voronoi techniques to UAV path planning are described in [83, 97, 98, 99]. Incremental construction of Voronoi graphs based on sensor measurements is described in [100, 101]. An effective search technique for Voronoi diagrams providing multiple path options is the Eppstein's  $k$ -shortest paths algorithm [102].

The RRT algorithm was first introduced in [103] and applied to nonholonomic robotic vehicles in [104, 93]. There are numerous applications of RRTs reported in the literature, as well as extensions to the basic algorithms [105]. A recent extension to the RRT algorithm that converges with probability one to the optimal path is described in [106]. The RRT algorithm is closely related to the probabilistic roadmap technique described in [107], which is applied to UAVs in [108].

There are several coverage algorithms discussed in the literature. Reference [109] 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 [110]. Multiple vehicle coverage algorithms are discussed in [111], and coverage algorithms in the context of mobile sensor networks are described in [112, 113].

### 12.3 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.

## *Chapter Thirteen*

---

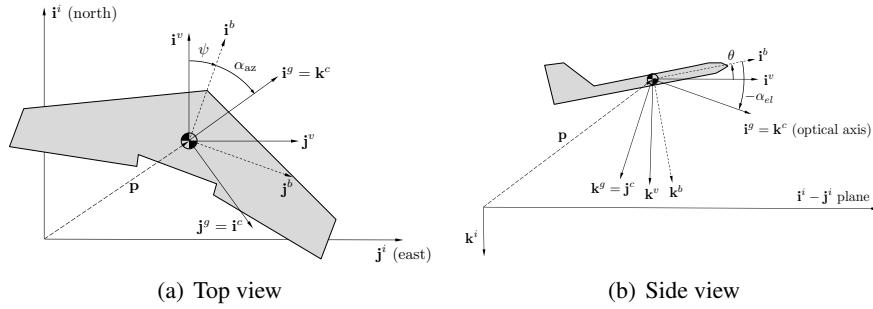
### Vision-guided Navigation

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 [114].

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 [115]. Figure 13.1 shows the relationship between the vehicle and body frames of the MAV and the gimbal and camera frames. **MODIFIED MATERIAL:** There are three frames of interest: the gimbal-



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

1 frame denoted by  $\mathcal{F}^{g1} = (\mathbf{i}^{g1}, \mathbf{j}^{g1}, \mathbf{k}^{g1})$ , the 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  $\alpha_{az}$ , which is called the gimbal azimuth angle. The rotation from the body to the gimbal-1 frame is given by

$$R_b^{g1}(\alpha_{az}) \triangleq \begin{pmatrix} \cos \alpha_{az} & \sin \alpha_{az} & 0 \\ -\sin \alpha_{az} & \cos \alpha_{az} & 0 \\ 0 & 0 & 1 \end{pmatrix}. \quad (13.1)$$

The gimbal frame is obtained by rotating the gimbal-1 frame about the  $\mathbf{j}^{g1}$  axis by an angle of  $\alpha_{el}$ , 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

$$R_{g1}^g(\alpha_{el}) \triangleq \begin{pmatrix} \cos \alpha_{el} & 0 & -\sin \alpha_{el} \\ 0 & 1 & 0 \\ \sin \alpha_{el} & 0 & \cos \alpha_{el} \end{pmatrix}. \quad (13.2)$$

The rotation from the body to the gimbal frame is, therefore, given by

$$R_b^g = R_{g1}^g R_b^{g1} = \begin{pmatrix} \cos \alpha_{el} \cos \alpha_{az} & \cos \alpha_{el} \sin \alpha_{az} & -\sin \alpha_{el} \\ -\sin \alpha_{az} & \cos \alpha_{az} & 0 \\ \sin \alpha_{el} \cos \alpha_{az} & \sin \alpha_{el} \sin \alpha_{az} & \cos \alpha_{el} \end{pmatrix}. \quad (13.3)$$

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

$$R_g^c = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}. \quad (13.4)$$

### 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  $\epsilon_x$  and  $\epsilon_y$  are the pixel location (in units of pixels) of the object. The distance from the origin of the camera frame to the pixel location  $(\epsilon_x, \epsilon_y)$ , as shown in [figure 13.2](#), is  $PF$  where

$$F = \sqrt{f^2 + \epsilon_x^2 + \epsilon_y^2}. \quad (13.6)$$

Using similar triangles in [figure 13.2](#), we get

$$\frac{\ell_x^c}{\mathbb{L}} = \frac{P\epsilon_x}{PF} = \frac{\epsilon_x}{F}. \quad (13.7)$$

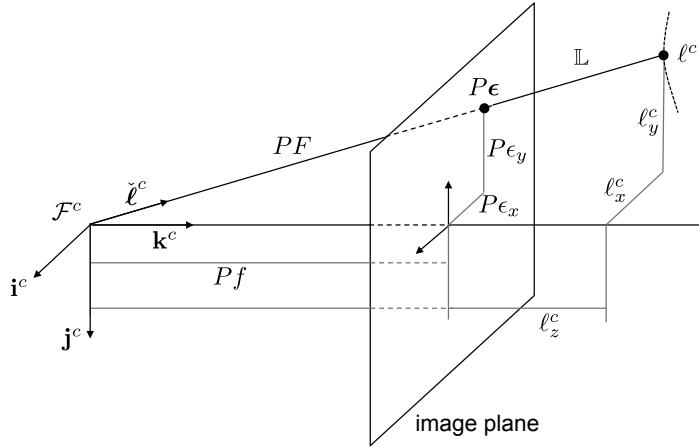
Similarly, we get that  $\ell_y^c/\mathbb{L} = \epsilon_y/F$  and  $\ell_z^c/\mathbb{L} = f/F$ . Combining, we have that

$$\ell^c = \frac{\mathbb{L}}{F} \begin{pmatrix} \epsilon_x \\ \epsilon_y \\ f \end{pmatrix}, \quad (13.8)$$

where  $\ell$  is the vector to the object of interest and  $\mathbb{L} = \|\ell\|$ .

Note that  $\ell^c$  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  $\ell^c$ . The projection of the target onto the image plane is represented by  $\epsilon$ . 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  $\mathbb{L}$ ,  $\epsilon$  and  $f$  are in units of pixels,  $\ell$  is in units of meters.

Since the unit vector  $\ell^c/\mathbb{L}$  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  $\ell$ .

### 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

$$\check{\boldsymbol{\ell}}_d^b = \frac{1}{\|\boldsymbol{\ell}_d^i\|} R_i^b \boldsymbol{\ell}_d^i.$$

For the second scenario, suppose that we desire to maneuver the gimbal so that the pixel location  $\epsilon$  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

$$\check{\boldsymbol{\ell}}_d^b = R_g^b R_c^g \check{\boldsymbol{\ell}}_d^c.$$

**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  $\alpha_{\text{az}}^c$  and  $\alpha_{\text{el}}^c$  so that

$$\check{\boldsymbol{\ell}}_d^b \triangleq \begin{pmatrix} \check{\ell}_{xd}^b \\ \check{\ell}_{yd}^b \\ \check{\ell}_{zd}^b \end{pmatrix} = R_g^b(\alpha_{\text{az}}^c, \alpha_{\text{el}}^c) R_c^g \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad (13.10)$$

$$= \begin{pmatrix} \cos \alpha_{\text{el}}^c \cos \alpha_{\text{az}}^c & -\sin \alpha_{\text{az}}^c & \sin \alpha_{\text{el}}^c \cos \alpha_{\text{az}}^c \\ \cos \alpha_{\text{el}}^c \sin \alpha_{\text{az}}^c & \cos \alpha_{\text{az}}^c & \sin \alpha_{\text{el}}^c \sin \alpha_{\text{az}}^c \\ -\sin \alpha_{\text{el}}^c & 0 & \cos \alpha_{\text{el}}^c \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad (13.11)$$

$$= \begin{pmatrix} \cos \alpha_{\text{el}}^c \cos \alpha_{\text{az}}^c \\ \cos \alpha_{\text{el}}^c \sin \alpha_{\text{az}}^c \\ -\sin \alpha_{\text{el}}^c \end{pmatrix}. \quad (13.12)$$

Solving for  $\alpha_{\text{el}}^c$  and  $\alpha_{\text{az}}^c$  gives the desired azimuth and elevation angles as

$$\alpha_{\text{az}}^c = \tan^{-1} \left( \frac{\tilde{\ell}_{yd}^b}{\tilde{\ell}_{xd}^b} \right) \quad (13.13)$$

$$\alpha_{\text{el}}^c = -\sin^{-1} \left( \frac{\tilde{\ell}_{zd}^b}{\tilde{\ell}_{xd}^b} \right). \quad (13.14)$$

The gimbal servo commands can be selected as

$$\begin{aligned} u_{\text{az}} &= k_{\text{az}}(\alpha_{\text{az}}^c - \alpha_{\text{az}}) \\ u_{\text{el}} &= k_{\text{el}}(\alpha_{\text{el}}^c - \alpha_{\text{el}}), \end{aligned} \quad (13.15)$$

where  $k_{\text{az}}$  and  $k_{\text{el}}$  are positive control gains.

### 13.3 GEOLOCATION

This section presents a method for determining the location of objects in world/inertial coordinates using a gimballed 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.

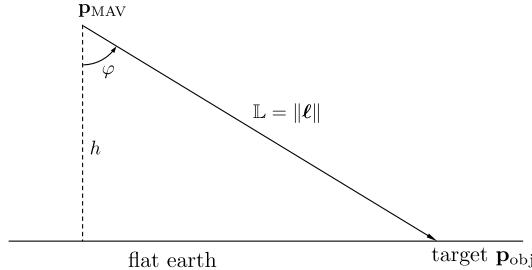
Following [section 13.1](#), let  $\ell = \mathbf{p}_{\text{obj}} - \mathbf{p}_{\text{MAV}}$  be the relative position vector between the target of interest and the MAV, and define  $\mathbb{L} = \|\ell\|$  and  $\check{\ell} = \ell/\mathbb{L}$ . From geometry, we have the relationship

$$\begin{aligned} \mathbf{p}_{\text{obj}}^i &= \mathbf{p}_{\text{MAV}}^i + R_b^i R_g^b R_c^g \ell^c \\ &= \mathbf{p}_{\text{MAV}}^i + \mathbb{L} \left( R_b^i R_g^b R_c^g \check{\ell}^c \right), \end{aligned} \quad (13.16)$$

where  $\mathbf{p}_{\text{MAV}}^i = (p_n, p_e, p_d)^\top$ ,  $R_b^i = R_b^i(\phi, \theta, \psi)$ , and  $R_g^b = R_g^b(\alpha_{\text{az}}, \alpha_{\text{el}})$ . 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 [115]. [Figure 13.3](#) shows the geometry of the situation, where  $h = -p_d$  is the height-above-ground, and  $\varphi$  is the angle between  $\ell$  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

$$\begin{aligned}\cos \varphi &= \mathbf{k}^i \cdot \check{\ell}^i \\ &= \mathbf{k}^i \cdot R_b^i R_g^b R_c^g \check{\ell}^c.\end{aligned}$$

Therefore, the range estimate using the flat-earth model is given by

$$\mathbb{L} = \frac{h}{\mathbf{k}^i \cdot R_b^i R_g^b R_c^g \check{\ell}^c}. \quad (13.17)$$

The geolocation estimate is given by combining equations (13.16) and (13.17) to obtain

$$\mathbf{p}_{\text{obj}}^i = \begin{pmatrix} p_n \\ p_e \\ p_d \end{pmatrix} + h \frac{R_b^i R_g^b R_c^g \check{\ell}^c}{\mathbf{k}^i \cdot R_b^i R_g^b R_c^g \check{\ell}^c}. \quad (13.18)$$

### 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

$$\mathbf{p}_{\text{MAV}}^i = \mathbf{p}_{\text{obj}}^i - \mathbb{L} \left( R_b^i R_g^b R_c^g \check{\ell}^c \right), \quad (13.19)$$

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  $\mathbb{L} = \|\mathbf{p}_{\text{obj}}^i - \mathbf{p}_{\text{MAV}}^i\|$ , we have

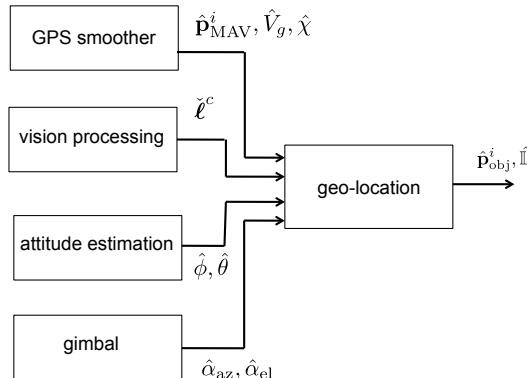
$$\begin{aligned}\dot{\mathbb{L}} &= \frac{d}{dt} \sqrt{(\mathbf{p}_{\text{obj}}^i - \mathbf{p}_{\text{MAV}}^i)^\top (\mathbf{p}_{\text{obj}}^i - \mathbf{p}_{\text{MAV}}^i)} \\ &= \frac{(\mathbf{p}_{\text{obj}}^i - \mathbf{p}_{\text{MAV}}^i)^\top (\dot{\mathbf{p}}_{\text{obj}}^i - \dot{\mathbf{p}}_{\text{MAV}}^i)}{\mathbb{L}} \\ &= -\frac{(\mathbf{p}_{\text{obj}}^i - \mathbf{p}_{\text{MAV}}^i)^\top \dot{\mathbf{p}}_{\text{MAV}}^i}{\mathbb{L}},\end{aligned}$$

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 8.9](#).

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 8.9](#), 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 8.8](#).



**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.

The geolocation algorithm is an extended Kalman filter with state  $\hat{x} = (\hat{\mathbf{p}}_{\text{obj}}^{i\top} \hat{\mathbb{L}})^{\top}$  and prediction equations given by

$$\begin{pmatrix} \dot{\hat{\mathbf{p}}}_{\text{obj}}^i \\ \dot{\hat{\mathbb{L}}} \end{pmatrix} = \begin{pmatrix} 0_{3 \times 1} \\ -\frac{(\hat{\mathbf{p}}_{\text{obj}}^i - \hat{\mathbf{p}}_{\text{MAV}}^i)^{\top} \dot{\hat{\mathbf{p}}}_{\text{MAV}}^i}{\hat{\mathbb{L}}} \end{pmatrix}.$$

The prediction Jacobian is therefore given by

$$\frac{\partial f}{\partial x} = \begin{pmatrix} 0_{3 \times 3} & 0_{3 \times 1} \\ -\frac{\dot{\hat{\mathbf{p}}}_{\text{MAV}}^i}{\hat{\mathbb{L}}} & \frac{(\hat{\mathbf{p}}_{\text{obj}}^i - \hat{\mathbf{p}}_{\text{MAV}}^i)^{\top} \dot{\hat{\mathbf{p}}}_{\text{MAV}}^i}{\hat{\mathbb{L}}^2} \end{pmatrix}.$$

The output equation is given by [equation \(13.19\)](#), where the Jacobian of the output equation is

$$\frac{\partial h}{\partial x} = (I_{3 \times 3} \quad -R_b^i R_g^b R_c^g \check{\ell}^c).$$

### 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 [116],

$$s \mapsto \frac{2}{T_s} \frac{z-1}{z+1},$$

to convert to the  $z$ -domain gives

$$\begin{aligned}\boldsymbol{\epsilon}[z] &= \frac{1}{\frac{2\tau}{T_s} \frac{z-1}{z+1} + 1} \bar{\boldsymbol{\epsilon}} = \frac{\frac{T_s}{2\tau+T_s}(z+1)}{z - \frac{2\tau-T_s}{2\tau+T_s}} \bar{\boldsymbol{\epsilon}} \\ \dot{\boldsymbol{\epsilon}}[z] &= \frac{\frac{2}{T_s} \frac{z-1}{z+1}}{\frac{2\tau}{T_s} \frac{z-1}{z+1} + 1} \bar{\boldsymbol{\epsilon}} = \frac{\frac{2}{2\tau+T_s}(z-1)}{z - \frac{2\tau-T_s}{2\tau+T_s}} \bar{\boldsymbol{\epsilon}}.\end{aligned}$$

Taking the inverse  $z$ -transform gives the difference equations

$$\begin{aligned}\boldsymbol{\epsilon}[n] &= \left( \frac{2\tau - T_s}{2\tau + T_s} \right) \boldsymbol{\epsilon}[n-1] + \left( \frac{T_s}{2\tau + T_s} \right) (\bar{\boldsymbol{\epsilon}}[n] + \bar{\boldsymbol{\epsilon}}[n-1]) \\ \dot{\boldsymbol{\epsilon}}[n] &= \left( \frac{2\tau - T_s}{2\tau + T_s} \right) \dot{\boldsymbol{\epsilon}}[n-1] + \left( \frac{2}{2\tau + T_s} \right) (\bar{\boldsymbol{\epsilon}}[n] - \bar{\boldsymbol{\epsilon}}[n-1]),\end{aligned}$$

where  $\boldsymbol{\epsilon}[0] = \bar{\boldsymbol{\epsilon}}[0]$  and  $\dot{\boldsymbol{\epsilon}}[0] = 0$ .

### 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  $\check{\boldsymbol{\ell}} \triangleq \boldsymbol{\ell}/\mathbb{L} = (\mathbf{p}_{\text{obj}} - \mathbf{p}_{\text{MAV}})/\|\mathbf{p}_{\text{obj}} - \mathbf{p}_{\text{MAV}}\|$  be the normalized relative position vector between the target and the MAV. Using the Coriolis formula in equation (2.16), we get

$$\frac{d\check{\boldsymbol{\ell}}}{dt_i} = \frac{d\check{\boldsymbol{\ell}}}{dt_c} + \boldsymbol{\omega}_{c/i} \times \check{\boldsymbol{\ell}}. \quad (13.22)$$

The expression on the left-hand side of equation (13.22) is the true relative translational 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

$$\frac{d\check{\ell}^c}{dt_i} = \frac{d\check{\ell}^c}{dt_c} + \omega_{c/i}^c \times \check{\ell}^c. \quad (13.23)$$

The first expression on the right-hand side of [equation \(13.23\)](#) can be computed as

$$\begin{aligned} \frac{d\check{\ell}^c}{dt_c} &= \frac{d}{dt_c} \frac{\begin{pmatrix} \epsilon_x \\ \epsilon_y \\ f \end{pmatrix}}{F} = \frac{F \begin{pmatrix} \dot{\epsilon}_x \\ \dot{\epsilon}_y \\ 0 \end{pmatrix} - \dot{F} \begin{pmatrix} \epsilon_x \\ \epsilon_y \\ f \end{pmatrix}}{F^2} = \frac{F \begin{pmatrix} \dot{\epsilon}_x \\ \dot{\epsilon}_y \\ 0 \end{pmatrix} - \frac{\epsilon_x \dot{\epsilon}_x + \epsilon_y \dot{\epsilon}_y}{F} \begin{pmatrix} \epsilon_x \\ \epsilon_y \\ f \end{pmatrix}}{F^2} \\ &= \frac{1}{F^3} \begin{pmatrix} F^2 - \epsilon_x^2 & -\epsilon_x \epsilon_y & 0 \\ -\epsilon_x \epsilon_y & F^2 - \epsilon_y^2 & 0 \\ -\epsilon_x f & -\epsilon_y f & 0 \end{pmatrix} \begin{pmatrix} \dot{\epsilon}_x \\ \dot{\epsilon}_y \\ 0 \end{pmatrix} = \frac{1}{F^3} \begin{pmatrix} \epsilon_y^2 + f^2 & -\epsilon_x \epsilon_y & 0 \\ -\epsilon_x \epsilon_y & \epsilon_x^2 + f^2 & 0 \\ -\epsilon_x f & -\epsilon_y f & 0 \end{pmatrix} \dot{\epsilon} \\ &= Z(\epsilon) \dot{\epsilon}, \end{aligned} \quad (13.24)$$

where

$$Z(\epsilon) \triangleq \frac{1}{F^3} \begin{pmatrix} \epsilon_y^2 + f^2 & -\epsilon_x \epsilon_y & 0 \\ -\epsilon_x \epsilon_y & \epsilon_x^2 + f^2 & 0 \\ -\epsilon_x f & -\epsilon_y f & 0 \end{pmatrix} \dot{\epsilon}.$$

To compute the second term on the right-hand side of [equation \(13.23\)](#), we need an expression for  $\omega_{c/i}^c$ , which can be decomposed as

$$\omega_{c/i}^c = \omega_{c/g}^c + \omega_{g/b}^c + \omega_{b/i}^c. \quad (13.25)$$

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

$$\omega_{b/i}^c = R_g^c R_b^g \omega_{b/i}^b = R_g^c R_b^g \begin{pmatrix} p \\ q \\ r \end{pmatrix}. \quad (13.26)$$

To derive an expression for  $\omega_{g/b}$  in terms of the measured gimbal angle rates  $\dot{\alpha}_{el}$  and  $\dot{\alpha}_{az}$ , recall that the azimuth angle  $\alpha_{az}$  is defined with respect to the body frame, and that the elevation angle  $\alpha_{el}$  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  $\alpha_{el}$ . Therefore,  $\dot{\alpha}_{az}$  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

$$\boldsymbol{\omega}_{g/b}^b = R_{g1}^b(\alpha_{\text{az}})R_g^{g1}(\alpha_{\text{el}}) \begin{pmatrix} 0 \\ \dot{\alpha}_{\text{el}} \\ 0 \end{pmatrix}^g + R_{g1}^b(\alpha_{\text{az}}) \begin{pmatrix} 0 \\ 0 \\ \dot{\alpha}_{\text{az}} \end{pmatrix}^{g1}.$$

Noting that  $R_g^{g1}$  is a  $y$ -axis rotation, we get

$$\begin{aligned} \boldsymbol{\omega}_{g/b}^b &= R_{g1}^b(\alpha_{\text{az}}) \begin{pmatrix} 0 \\ \dot{\alpha}_{\text{el}} \\ \dot{\alpha}_{\text{az}} \end{pmatrix} = \begin{pmatrix} \cos \alpha_{\text{az}} & -\sin \alpha_{\text{az}} & 0 \\ \sin \alpha_{\text{az}} & \cos \alpha_{\text{az}} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ \dot{\alpha}_{\text{el}} \\ \dot{\alpha}_{\text{az}} \end{pmatrix} \\ &= \begin{pmatrix} -\sin(\alpha_{\text{az}})\dot{\alpha}_{\text{el}} \\ \cos(\alpha_{\text{az}})\dot{\alpha}_{\text{el}} \\ \dot{\alpha}_{\text{az}} \end{pmatrix}, \end{aligned}$$

and it follows that

$$\boldsymbol{\omega}_{g/b}^c = R_g^c R_b^g \boldsymbol{\omega}_{g/b}^b = R_g^c R_b^g \begin{pmatrix} -\sin(\alpha_{\text{az}})\dot{\alpha}_{\text{el}} \\ \cos(\alpha_{\text{az}})\dot{\alpha}_{\text{el}} \\ \dot{\alpha}_{\text{az}} \end{pmatrix}. \quad (13.27)$$

Drawing on equations (13.24) and (13.27), equation (13.23) can be expressed as

$$\frac{d\dot{\ell}^c}{dt_i} = Z(\epsilon)\dot{\epsilon} + \dot{\ell}_{\text{app}}^c, \quad (13.28)$$

where

$$\dot{\ell}_{\text{app}}^c \triangleq \frac{1}{F} \left[ R_g^c R_b^g \begin{pmatrix} p - \sin(\alpha_{\text{az}})\dot{\alpha}_{\text{el}} \\ q + \cos(\alpha_{\text{az}})\dot{\alpha}_{\text{el}} \\ r + \dot{\alpha}_{\text{az}} \end{pmatrix} \right] \times \begin{pmatrix} \epsilon_x \\ \epsilon_y \\ f \end{pmatrix} \quad (13.29)$$

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

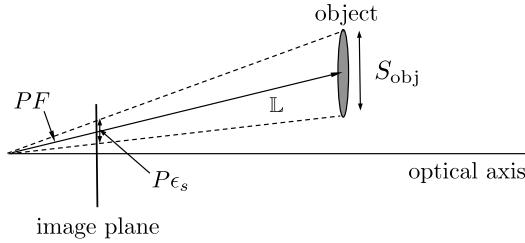
For collision avoidance algorithms and for the precision landing algorithm described in section 13.6, it is important to estimate the time to collision for objects in the camera field of view. If  $\mathbb{L}$  is the length of the line-of-sight vector between the MAV and an object, then the time to 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.

$$\frac{S_{\text{obj}}}{L} = \frac{P\epsilon_s}{PF} = \frac{\epsilon_s}{F}, \quad (13.30)$$

where the size of the target in meters is  $S_{\text{obj}}$ , and the size of the target in pixels is given by  $\epsilon_s$ . We assume that the size of the target  $S_{\text{obj}}$  is not changing in time. Differentiating equation (13.30) and solving for  $\dot{L}/L$ , we obtain

$$\begin{aligned} \frac{\dot{L}}{L} &= \frac{L}{S_{\text{obj}}} \left[ \frac{\epsilon_s \dot{F}}{F F} - \frac{\dot{\epsilon}_s}{F} \right] \\ &= \frac{F}{\epsilon_s} \left[ \frac{\epsilon_s \dot{F}}{F F} - \frac{\dot{\epsilon}_s}{F} \right] \\ &= \frac{\dot{F}}{F} - \frac{\dot{\epsilon}_s}{\epsilon_s} \\ &= \frac{\epsilon_x \dot{\epsilon}_x + \epsilon_y \dot{\epsilon}_y}{F} - \frac{\dot{\epsilon}_s}{\epsilon_s}, \end{aligned} \quad (13.31)$$

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 [117]. 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

$$\begin{aligned}\dot{\mathbb{L}} &= \frac{1}{\mathbb{L}} \left( \frac{\cos \varphi \dot{h} + h \dot{\varphi} \sin \varphi}{\cos^2 \varphi} \right) \\ &= \frac{\cos \varphi}{h} \left( \frac{\cos \varphi \dot{h} + h \dot{\varphi} \sin \varphi}{\cos^2 \varphi} \right) \\ &= \frac{\dot{h}}{h} + \dot{\varphi} \tan \varphi.\end{aligned}\tag{13.32}$$

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  $\dot{\varphi}$  gives

$$\dot{\varphi} = -\frac{1}{\sin \varphi} \frac{d}{dt_i} \check{\ell}_z^i.\tag{13.35}$$

Therefore,

$$\dot{\varphi} \tan \varphi = -\frac{1}{\cos \varphi} \frac{d}{dt_i} \check{\ell}_z^i = -\frac{1}{\check{\ell}_z^i} \frac{d}{dt_i} \check{\ell}_z^i,\tag{13.36}$$

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 [118]. 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),  $\chi$  is the course angle,  $\gamma$  is the flight path angle,  $\phi$  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  $\mathbf{p}_{\text{obj}}$ . Similarly, let  $\mathbf{p}_{\text{MAV}}$ ,  $\mathbf{v}_{\text{MAV}}$ ,  $\mathbf{a}_{\text{MAV}}$  denote the position, velocity, and acceleration of the MAV, respectively.

In the inertial frame, the position and velocity of the MAV can be ex-

pressed as

$$\mathbf{p}_{\text{MAV}}^i = (p_n, \ p_e, \ p_d)^\top$$

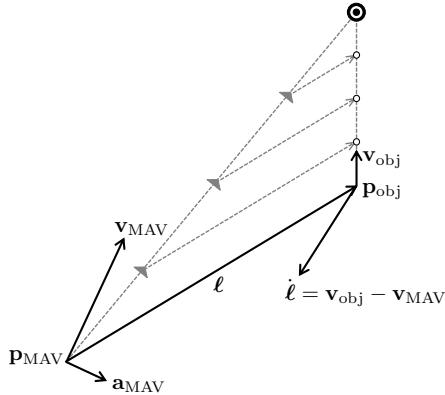
and

$$\mathbf{v}_{\text{MAV}}^i = (V_g \cos \chi \cos \gamma, \ V_g \sin \chi \cos \gamma, \ -V_g \sin \gamma)^\top.$$

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 key idea behind proportional navigation is that two approach-



**Figure 13.6:** The geometry associated with precision landing.

ing vehicles are on a collision course when their direct line-of-sight does not change direction as their range closes. To intercept or land on a moving object, the MAV needs to maneuver so that the rotation rate of the line-of-sight vector goes to zero. This happens when the line-of-sight rate  $\dot{\ell}$  is aligned with the negative of the line-of-sight vector  $-\ell$ . Proportional navigation dictates that the MAV velocity vector should rotate at a rate proportional to the rotation rate of the line of sight vector and in the same direction, bringing  $\dot{\ell}$  and  $\ell$  into alignment. This can be accomplished by making the rotation rate of  $\mathbf{v}_{\text{MAV}}$  proportional to the cross product of  $\ell$  and  $\dot{\ell}$ . However, since  $\ell$  and  $\dot{\ell}$  cannot be directly computed from vision data, we normalize both quantities and define

$$\Omega_\perp = \check{\ell} \times \frac{\dot{\ell}}{\mathbb{L}}. \quad (13.43)$$

With reference to [figure 13.6](#), note that  $\Omega_{\perp}$  is directed into the page. Because direct control of the ground speed can be adversely affected by wind and MAV propulsion limits, we will require that the commanded acceleration be perpendicular to the velocity vector of the MAV. Accordingly, let the commanded acceleration of the MAV be given by [119]

$$\mathbf{a}_{\text{MAV}} = N\Omega_{\perp} \times \mathbf{v}_{\text{MAV}}, \quad (13.44)$$

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  $\mathbf{a}_{\text{MAV}}$  must be resolved in the vehicle-2 frame as

$$\begin{aligned} \mathbf{a}_{\text{MAV}}^{v2} &= \mu N \Omega_{\perp}^{v2} \times \mathbf{v}_{\text{MAV}}^{v2} \\ &= \mu N \begin{pmatrix} \Omega_{\perp,x}^{v2} \\ \Omega_{\perp,y}^{v2} \\ \Omega_{\perp,z}^{v2} \end{pmatrix} \times \begin{pmatrix} V_g \\ 0 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ \mu N V_g \Omega_{\perp,z}^{v2} \\ -\mu N V_g \Omega_{\perp,y}^{v2} \end{pmatrix} \end{aligned} \quad (13.45)$$

where  $\mu$  is a scaling factor. 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  $\Omega_{\perp} = \dot{\ell} \times \frac{\dot{\ell}}{\|\dot{\ell}\|}$  must be estimated from video camera data. Our basic approach will be to estimate  $\Omega_{\perp}$  in the camera frame, and then transform to the vehicle-2 frame using the expression

$$\Omega_{\perp}^{v2} = R_b^{v2} R_g^b R_c^g \Omega_{\perp}^c. \quad (13.46)$$

The normalized line-of-sight vector  $\dot{\ell}^c$  can be estimated directly from camera data using [equation \(13.9\)](#). Differentiating  $\ell^c/\|\ell\|$  gives

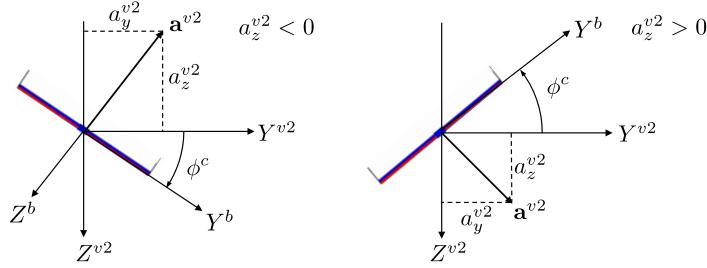
$$\frac{d}{dt_i} \left( \frac{\ell^c}{\|\ell\|} \right) = \frac{\|\dot{\ell}^c\| - \|\dot{\ell}\|^2}{\|\ell\|^2} = \frac{\dot{\ell}^c}{\|\ell\|} - \frac{\dot{\ell}}{\|\ell\|}, \quad (13.47)$$

which, when combined with [equation \(13.28\)](#), results in the expression

$$\frac{\dot{\ell}^c}{\|\ell\|} = \frac{\dot{\ell}}{\|\ell\|} + Z(\epsilon)\dot{\epsilon} + \dot{\ell}_{\text{app}}^c, \quad (13.48)$$

where the inverse of time to collision  $\dot{\ell}/\|\ell\|$  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. Here 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 [120], which is shown in figure 13.7. From figure 13.7 it



**Figure 13.7:** Polar converting logic that transforms an acceleration command  $\mathbf{a}^{v2}$  to a commanded roll angle  $\phi^c$  and a commanded normal acceleration  $V_g \dot{\gamma}^c$ .

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, 121, 122, 123, 124, 125, 115, 126, 127, 128, 129]). The gimbal-pointing algorithm described in this chapter was presented in [130]. Geolocation algorithms using small MAV are described in [115, 131, 132, 133, 125, 129]. The removal of apparent motion, or ego-motion, in the image plane is discussed in [134, 135, 124]. Time to collision can be estimated using structure from motion [136], ground-plane methods [137, 138], flow divergence [139], and insect-inspired methods [140]. Section 13.6 is taken primarily from [141]. Proportional navigation has been extensively analyzed in the literature. It has been shown to be optimal under certain conditions [142] and to produce zero-miss distances for a constant target acceleration[143]. If rich information regarding the time to go is available, augmented proportional navigation [144] improves the performance by adding terms that account for target and pursuer accelerations. A three-dimensional expression of PN can be found in [119, 145].

### 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](#).

# Chapter Fourteen

---

## Multicopters

### 14.1 INTRODUCTION

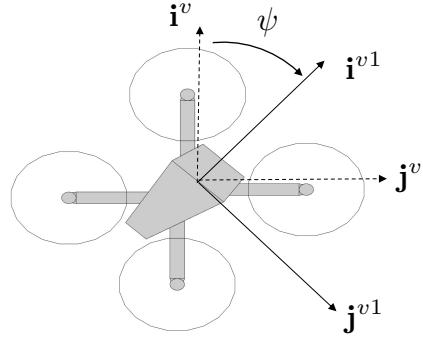
In recent years, quadrotors and other multicopter systems have become ubiquitous. Many of the principles outlined in this book can be applied to multicopter systems. The purpose of this chapter is to show how this can be done. We will favor straight-forward application of the principles already covered in the book, and not necessarily cover the state-of-the-art in multicopter estimation and control. The chapter is organized so that it follows the chapter structure of the book. Therefore, adaptation of Chapter 2 to multicopters is described in Section 14.2, and so forth. The high-level architecture that we describe in this chapter is identical to Figure 1.1 with the *unmanned aircraft* replaced by an *unmanned multicopter*.

**RWB:** Come back and fill this in when the rest of the chapter is completed.

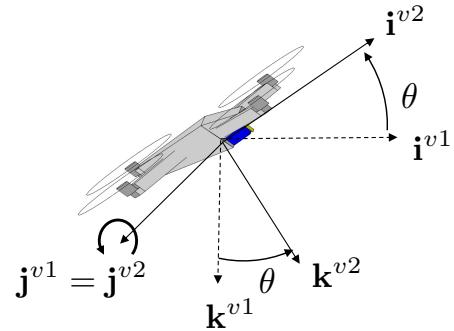
### 14.2 COORDINATE FRAMES

In this section we describe the adaptation of Chapter 2 to multicopter systems. While much of the multicopter literature, especially work appearing in the robotics community, uses a East-North-Up (ENU) inertial coordinate system for multicopters (see e.g., [146]), in this chapter we will continue to use North-East-Down (NED) inertial coordinate system to be consistent with the rest of the book. Similarly, the vehicle frame  $\mathcal{F}^v$  is located at the center of mass of the multicopter and oriented with the NED inertial frame  $\mathcal{F}^i$ . The vehicle-1 frame, as shown in Figure 14.1, is such that the  $\mathbf{i}^{v1}$ -axis points out the front of the multicopter, the  $\mathbf{k}^{v1}$ -axis points along the  $\mathbf{k}^v$ -axis, and  $\mathbf{j}^{v1}$  completes a right-handed coordinate system. The angle  $\psi$  is called the yaw angle, and for a multicopter, may not be linked to the heading or course angle. For multicopter systems, the vehicle-1 frame is often called the body-level frame and is denoted by  $\mathcal{F}^\ell = \{\mathbf{i}^\ell, \mathbf{j}^\ell, \mathbf{k}^\ell\}$ .

The vehicle-2 frame  $\mathcal{F}^{v2}$ , as shown in Figure 14.2, is obtained by rotating  $\mathcal{F}^{v1}$  by a rotation of  $\theta$ , called the pitch angle, about the  $\mathbf{j}^{v1}$ -axis. Similarly, the body frame  $\mathcal{F}^b$ , as shown in Figure 14.3, is obtained by rotating  $\mathcal{F}^{v2}$  by a rotation of  $\phi$ , called the roll angle, about  $\mathbf{i}^{v2}$ -axis.



**Figure 14.1:** The vehicle-1 frame for a multirotor.

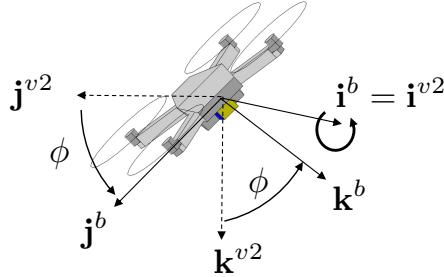


**Figure 14.2:** The vehicle-2 frame for a multirotor.

Therefore, the rotation matrix  $R_v^b$  from vehicle to body frame is still given by Equation (2.4).

### 14.3 KINEMATICS AND DYNAMICS

For fixed wing vehicles, the aerodynamic forces and moments are functions of the velocity vector expressed in the body or stability frame. However, for multirotor systems, the aerodynamics forces and moments that depend on the velocity are negligible, and therefore, it is more common in the multirotor literature to express the velocity vector in the inertial frame rather than the body frame. It is also common in the multirotor literature to use unit quaternions or rotation matrices to represent the attitude. In this chapter we will use vector quantities to simplify the notation. The inertial NED



**Figure 14.3:** The body frame for a multirotor.

position is given by  $\mathbf{p}_{b/i}^i \triangleq (p_n, p_e, p_d)^\top$ , where  $\mathbf{p}$  is a position vector, the subscript  $b/i$  means that the vector is the position of the body frame  $\mathcal{F}^b$  relative the inertial frame  $\mathcal{F}^i$ , and the superscript  $i$  means that the vector is expressed in the inertial frame. The inertial NED velocity is given by  $\mathbf{v}_{b/i}^i \triangleq (v_n, v_e, v_d)^\top$ . The attitude quaternion that transforms body coordinates to inertial coordinates is given by  $\bar{\mathbf{q}}_b^i \triangleq (e_0, \mathbf{q}^\top)^\top$  where  $e_0$  is the scalar part of the unit quaternion and  $\mathbf{q} = (e_x, e_y, e_z)^\top$  is the vector (imaginary) part. The angular velocity of the body relative to the inertial frame as expressed in the body frame is given by  $\boldsymbol{\omega}_{b/i}^b \triangleq (p, q, r)^\top$ . Therefore, the state variables for a multirotor are given in Table 14.1.

**Table 14.1:** State variables for Multirotor equations of motion.

Name	Description
$\mathbf{p}_{b/i}^i$	inertial NED position of the multirotor
$\mathbf{v}_{b/i}^i$	inertial NED velocity of the multirotor
$\bar{\mathbf{q}}_b^i$	attitude quaternion representing the rotation from the body to the inertial frame
$\boldsymbol{\omega}_{b/i}^b$	angular velocity expressed in the body frame.

Since the velocity vector is given in the inertial frame, the kinematics of the multirotor simplify to

$$\dot{\mathbf{p}}_{b/i}^i = \mathbf{v}_{b/i}^i \quad (14.1)$$

$$\dot{\bar{\mathbf{q}}}_b^i = \frac{1}{2}\Omega(\boldsymbol{\omega}_{b/i}^b)\bar{\mathbf{q}}_b^i \quad (14.2)$$

where

$$\Omega(\boldsymbol{\omega}_{b/i}^b) \triangleq \begin{pmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{pmatrix}.$$

Since the velocity of the multirotor is given in the inertial frame, the translational dynamics are given by Equation (3.4) as

$$\dot{\mathbf{v}}_{b/i}^i = \frac{1}{m} \mathbf{f}^i.$$

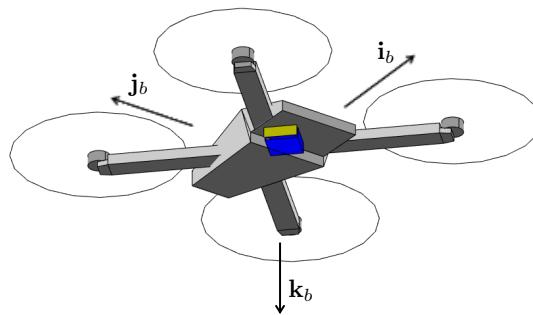
Similar to a fixed-wing aircraft, the rotational dynamics of a multirotor are given by Equation (3.11), i.e.,

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

where  $\mathbf{m}^b$  are control torques. Multirotor aircraft tend to be symmetric in their shape and mass distribution and this results in significant simplification in the inertia matrix. In particular, if the aircraft is symmetric with respect to the  $\mathbf{i}_b\text{-}\mathbf{j}_b$ ,  $\mathbf{i}_b\text{-}\mathbf{k}_b$ , and  $\mathbf{j}_b\text{-}\mathbf{k}_b$  planes, then the products of inertia,  $J_{xy}$ ,  $J_{xz}$ , and  $J_{yz}$ , are zero resulting in

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

Importantly, under these symmetry conditions,  $\mathbf{J}$  is easily inverted with its



**Figure 14.4:** Quadrotor with body-frame axes that coincide at the center of mass.

inverse expressed as

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

In this case we note that

$$\begin{aligned} \dot{\boldsymbol{\omega}}_{b/i}^b &= -\mathbf{J}^{-1} \left( \boldsymbol{\omega}_{b/i}^b \times (\mathbf{J} \boldsymbol{\omega}_{b/i}^b) \right) + \mathbf{J}^{-1} \mathbf{m}^b \\ \implies \begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} &= \begin{pmatrix} \left( \frac{J_y - J_z}{J_x} \right) qr \\ \left( \frac{J_z - J_x}{J_y} \right) pr \\ \left( \frac{J_x - J_y}{J_z} \right) pq \end{pmatrix} + \begin{pmatrix} \ell/J_x \\ m/J_y \\ n/J_z \end{pmatrix}. \end{aligned}$$

The rigid body dynamics of a multrotor can therefore be summarized as

$$\begin{aligned} \dot{\mathbf{p}}_{b/i}^i &= \mathbf{v}_{b/i}^i \\ \dot{\mathbf{v}}_{b/i}^i &= \frac{1}{\mathbf{m}} \mathbf{f}^i \\ \dot{\bar{\mathbf{q}}}_b^i &= \frac{1}{2} \boldsymbol{\Omega}(\boldsymbol{\omega}_{b/i}^b) \bar{\mathbf{q}}_b^i \\ \dot{\boldsymbol{\omega}}_{b/i}^b &= \mathbf{J}^{-1} \left[ -\boldsymbol{\omega}_{b/i}^b \times (\mathbf{J} \boldsymbol{\omega}_{b/i}^b) + \mathbf{m}^b \right], \end{aligned}$$

or in component form

$$\begin{aligned} \dot{p}_n &= v_n \\ \dot{p}_e &= v_e \\ \dot{p}_d &= v_d \\ \dot{v}_n &= \frac{1}{\mathbf{m}} f_n \\ \dot{v}_e &= \frac{1}{\mathbf{m}} f_e \\ \dot{v}_d &= \frac{1}{\mathbf{m}} f_d \\ \dot{e}_0 &= \frac{1}{2} (-pe_x - qe_y - re_z) \\ \dot{e}_x &= \frac{1}{2} (pe_0 + re_y - qe_z) \\ \dot{e}_y &= \frac{1}{2} (qe_0 - re_x + pe_z) \\ \dot{e}_z &= \frac{1}{2} (re_0 + qe_x - pe_y) \end{aligned}$$

$$\begin{aligned}\dot{p} &= \left( \frac{J_y - J_z}{J_x} \right) qr + \ell/J_x \\ \dot{q} &= \left( \frac{J_z - J_x}{J_y} \right) pr + m/J_y \\ \dot{r} &= \left( \frac{J_x - J_y}{J_z} \right) pq + n/J_z.\end{aligned}$$

If we define the skew operator as

$$\left[ \begin{pmatrix} a \\ b \\ c \end{pmatrix} \right]_{\times} \triangleq \begin{pmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{pmatrix},$$

then the rotation matrix that corresponds to the unit quaternion  $\bar{\mathbf{q}}$  is given by

$$R(\bar{\mathbf{q}}) = I + 2e_0 [\mathbf{q}]_{\times} + 2 [\mathbf{q}]_{\times}^2.$$

#### 14.4 FORCES AND MOMENTS

Multicopter systems do not have substantive lifting surfaces and so the primary forces and moments acting on a multicopter are due to gravity and the rotors. The total force acting on the multicopter is approximated by

$$\mathbf{f}^i = \mathbf{f}_g^i + \mathbf{f}_t^i + \mathbf{f}_d^i,$$

where  $\mathbf{f}_g^i$  is the force due to gravity,  $\mathbf{f}_t^i$  is the thrust force due to the propellers, and  $\mathbf{f}_d^i$  is the drag induced by the spinning propellers. The total moment acting on the multicopter is approximated by

$$\mathbf{m}^b = \mathbf{m}_p^b,$$

where  $\mathbf{m}_p^b$  is the moments produced by the propellers. We will discuss each of these terms in the paragraphs below.

##### 14.4.1 Gravitational Force

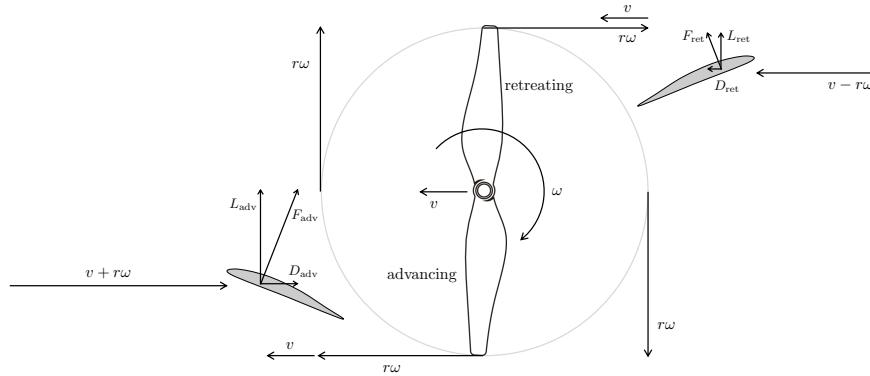
Since the translational equations of motion are expressed in the inertial frame, the force due to gravity is given by

$$\mathbf{f}_g^i = mg\mathbf{e}_3,$$

where  $g = 9.8 \text{ m/s}^2$  is the acceleration of a 1 kg mass at sea level, and  $\mathbf{e}_3 = (0, 0, 1)^T$  is a unit vector that points along the  $\mathbf{k}^i$  axis.

#### 14.4.2 Induced Drag

As with any lifting surface, induced drag forces are generated from the lift forces created by the rotating propellers. The direction of the drag force on a propeller blade at any instant is in the direction opposite of the velocity of the blade tip. Since the blade tip is moving much faster than the airspeed of the multirotor motion, when the blade is advancing into the oncoming apparent wind generated by the combination of multirotor motion and the wind, the induced drag is higher than when the blade is in the retreating portion of its circular motion and moving with the apparent wind. The net effect is that each of the rotors on the aircraft creates an induced drag force that is in the plane of the rotor and proportional to the projection of the lift force onto the plane of the rotor in magnitude and directed opposite to the direction of the air incident on the rotor plane [147, 146]. Accordingly, the



**Figure 14.5:** Induced drag on a propeller.

induced drag force expressed in the body frame can be represented as

$$\begin{aligned} \mathbf{f}_d^b &\approx -TC_d \operatorname{diag}(1, 1, 0) \mathbf{v}_{b/i}^b \\ &= \begin{pmatrix} -TC_d & 0 & 0 \\ 0 & -TC_d & 0 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{v}_{b/i}^b. \end{aligned}$$

For the purpose of calculating the induced drag, and to make it independent of the control variable  $T$ , we approximate the thrust force to be  $mg$ , the weight of the aircraft. Defining

$$D = \begin{pmatrix} gC_d & 0 & 0 \\ 0 & gC_d & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

the induced drag can be expressed in the inertial frame as

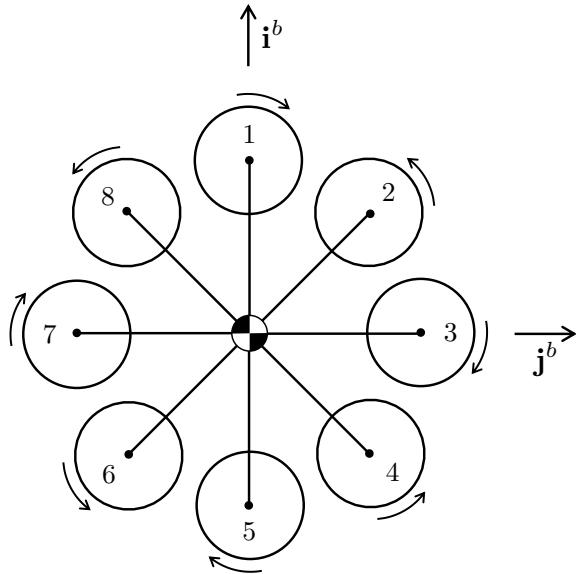
$$\mathbf{f}_d^i = -mR(\bar{\mathbf{q}}_b^i)DR^\top(\bar{\mathbf{q}}_b^i)\mathbf{v}_{b/i}^i.$$

The translational dynamics of a multirotor are therefore given by

$$\dot{\mathbf{v}}_{b/i}^i = g\mathbf{e}_3 - R(\bar{\mathbf{q}}_b^i)DR^\top(\bar{\mathbf{q}}_b^i)\mathbf{v}_{b/i}^i - \frac{T}{m}R^\top(\bar{\mathbf{q}}_b^i)\mathbf{e}_3. \quad (14.3)$$

#### 14.4.3 Propeller Forces and Moments

The aerodynamic forces and torques that enable the agile flight of a multirotor come from the dc-motor/propeller modules that we call the rotors. In hover, the rotors of a multirotor each provide a constant and equal thrust to overcome the effects of gravity. The rotors of a multirotor are usually configured in counter-rotating pairs, one rotating clockwise and the other counter-clockwise, so that the motor torques acting on the multirotor cancel out when in a non-yawing hover state. As the rotors rotate, the drag on the propellers produce a torque on the body of the aircraft opposite to the direction of rotation. Figure 14.6 depicts an octocopter with equally spaced counter-rotating rotors as an example of how a multirotor might be configured.



**Figure 14.6:** Octocopter example of a multirotor configuration.

To accelerate the multirotor up or down, the speed of the rotors can be

uniformly increased or decreased away from their nominal speed. To pitch up or down about the  $\mathbf{j}_b$  axis, the speed differential between the fore and aft rotors can be sped up or slowed down. Similarly, to roll the aircraft right or left about the  $\mathbf{i}_b$  axis the speed differential between the left and right rotors can be varied. To yaw the aircraft about the  $\mathbf{k}_b$  axis, on the other hand, the speed differential between the clockwise-rotating propellers and counter-clockwise-rotating propellers is varied.

In this chapter, we assume that the propellers are all mounted so that the direction of thrust is aligned along the body  $z$ -axis. If  $T$  is the total thrust from all of the rotors, then the force on the multirotor due to the thrust of the rotors is given in the body frame as

$$\mathbf{f}_p^b = -T\mathbf{e}_3,$$

where the negative sign is because the positive body  $z$ -axis points down, while the thrust vector will point in the opposite direction. Rotating the thrust vector into the inertial frame gives

$$\mathbf{f}_T^i = -TR^\top(\bar{\mathbf{q}}_b^i)\mathbf{e}_3.$$

In this section, we will develop the relationship between the thrust force  $T$ , the roll, pitch, and yaw torques  $\boldsymbol{\tau}^b = (\tau_x, \tau_y, \tau_z)^\top$ , and the individual rotor angular speeds that are commanded by the autopilot. For an  $N$ -rotor aircraft, these angular speeds will be given by  $\omega_1, \omega_2, \dots, \omega_N$  and specified in radians per second. From propeller theory, and as described in Chapter 4, the thrust and torque produced by a single rotor in hover can be modeled as

$$T_i = C_T \frac{\rho D^4}{4\pi^2} \omega_i^2 \quad (14.4)$$

$$Q_i = C_Q \frac{\rho D^5}{4\pi^2} \omega_i^2, \quad (14.5)$$

where  $\rho$  is the density of air,  $D$  is the propeller diameter, and  $C_T$  and  $C_Q$  are non-dimensional aerodynamic coefficients specific to the propeller.

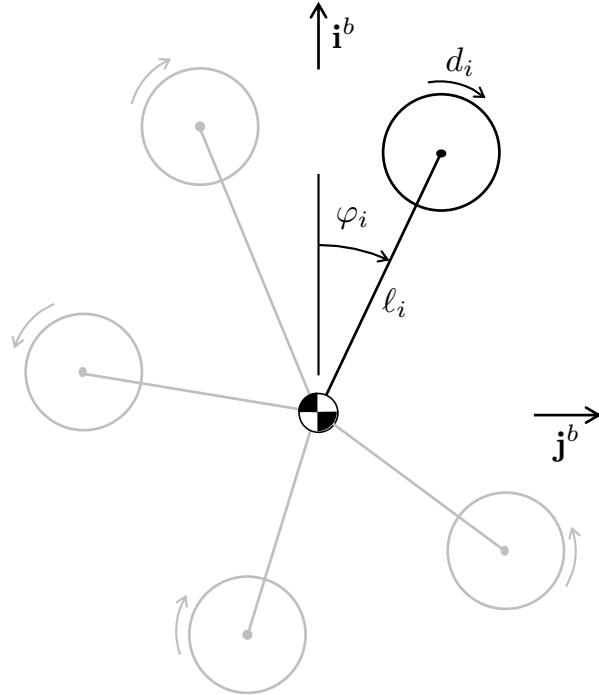
The total rotor thrust for the multirotor is given by

$$T = \sum_{i=1}^N T_i = C_T \frac{\rho D^4}{4\pi^2} \sum_{i=1}^N \omega_i^2,$$

where  $T_i$  is the thrust force produced by rotor  $i$ .

To define the torques produced by the rotors, we need to define the configuration of each rotor relative to the center of mass of the aircraft and its direction of rotation. Figure 14.7 highlights a single rotor  $i$  for an arbitrary multirotor configuration. The variable  $\varphi_i$  is the angle between the forward

direction of the multirotor, defined by the unit vector  $\mathbf{i}^b$ , and the radial direction of the  $i^{th}$  rotor. The variable  $\ell_i$  represents the radial distance from the center of mass to the  $i^{th}$  rotor. The variable  $d_i$  indicates the direction of rotation for each rotor as



**Figure 14.7:** Definition of rotor configuration variables.

$$d_i = \begin{cases} +1 & \text{for CCW rotation} \\ -1 & \text{for CW rotation.} \end{cases} \quad (14.6)$$

From the configuration geometry of Figure 14.7, the total roll torque about the  $\mathbf{i}_b$  axis is given by

$$\tau_x = - \sum_{i=1}^N (\ell_i \sin \varphi_i) T_i = -C_Q \frac{\rho D^5}{4\pi^2} \sum_{i=1}^N (\ell_i \sin \varphi_i) \omega_i^2. \quad (14.7)$$

Similarly, the total pitch torque about the  $\mathbf{j}_b$  axis can be calculated as

$$\tau_y = \sum_{i=1}^N (\ell_i \cos \varphi_i) T_i = C_Q \frac{\rho D^5}{4\pi^2} \sum_{i=1}^N (\ell_i \cos \varphi_i) \omega_i^2. \quad (14.8)$$

Finally, the total yaw torque about the  $\mathbf{k}_b$  axis, which depends on the direction of rotation of each rotor, is given by

$$\tau_z = \sum_{i=1}^N d_i Q_i = C_Q \frac{\rho D^5}{4\pi^2} \sum_{i=1}^N d_i \omega_i^2. \quad (14.9)$$

The relationship between thrust and torque on one hand, and angular velocity of the rotors on the other, can be summarized in matrix form as

$$\begin{pmatrix} T \\ \tau_x \\ \tau_y \\ \tau_z \end{pmatrix} = \mathbf{M} \begin{pmatrix} \omega_1^2 \\ \vdots \\ \omega_N^2 \end{pmatrix},$$

where

$$\mathbf{M} = \frac{\rho D^4}{4\pi^2} \begin{pmatrix} C_T, & C_T, & \dots & C_T \\ -C_Q D \ell_1 \sin \psi_1, & -C_Q D \ell_1 \sin \psi_2, & \dots & -C_Q D \ell_N \sin \psi_N \\ C_Q D \ell_1 \cos \psi_1, & C_Q D \ell_1 \cos \psi_2, & \dots & C_Q D \ell_N \cos \psi_N \\ C_Q D d_1, & C_Q D d_2, & \dots & C_Q D d_N \end{pmatrix}$$

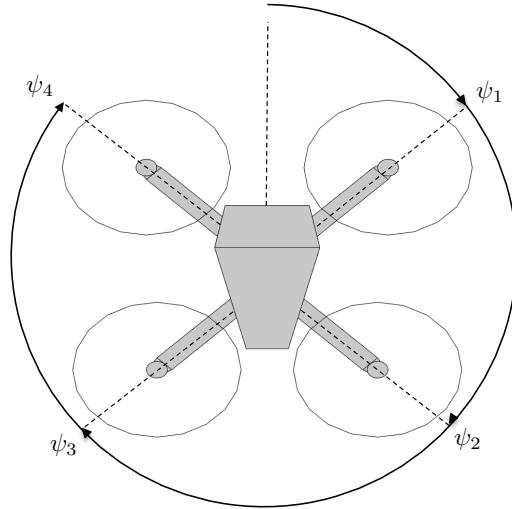
is called the *mixing matrix*. For all nontrivial configurations, the mixing matrix will be invertible, and the desired angular speeds of each motor can be determined from the desired thrust and torque through the (pre-computed and hard-coded) inverse of  $\mathbf{M}$ .

As an example, consider the quadrotor depicted in Figure 14.8. The length of each arm is given by  $\ell$ , and the arms are equally spaced at  $\psi_1 = 45^\circ$ ,  $\psi_2 = 135^\circ$ ,  $\psi_3 = 225^\circ$ ,  $\psi_4 = 315^\circ$ , and where the first and third propellers spin counter-clockwise, and the second and fourth propellers spin clockwise. In that case the mixing matrix is given by

$$\mathbf{M} = \frac{\rho D^4}{4\pi^2} \begin{pmatrix} C_T & C_T & C_T & C_T \\ -C_Q D \ell \frac{\sqrt{2}}{2} & -C_Q D \ell \frac{\sqrt{2}}{2} & C_Q D \ell \frac{\sqrt{2}}{2} & C_Q D \ell \frac{\sqrt{2}}{2} \\ C_Q D \ell \frac{\sqrt{2}}{2} & -C_Q D \ell \frac{\sqrt{2}}{2} & -C_Q D \ell \frac{\sqrt{2}}{2} & C_Q D \ell \frac{\sqrt{2}}{2} \\ C_Q D & -C_Q D & C_Q D & -C_Q D \end{pmatrix}.$$

It is clear (by observing the plus-minus patterns) that in this case, the mixing matrix  $\mathbf{M}$  is square and invertible. Therefore, the desired angular speeds of each rotor can be found from

$$\begin{pmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{pmatrix} = \mathbf{M}^{-1} \begin{pmatrix} T \\ \tau_x \\ \tau_y \\ \tau_z \end{pmatrix},$$



**Figure 14.8:** The top view of a quadrotor where the motor arms each of length  $\ell$  and are equally spaced angularly.

and then taking the positive square root of each term to find  $\omega_i, i = 1, 2, 3, 4$ . When there are more than four rotors, the mixing matrix will not be square, and so the Moore-Penrose pseudo-inverse is used to find the angular velocities, that are then sent to the speed controllers. Since there is a direct relationship between the thrust/torques and the angular velocities, and this relationship is platform dependent, we typically think of the control input to the multirotor as thrust and torque, similar to the way that we think of the control input to a fixed wing vehicle as the elevator/aileron/rudder/throttle, independent of how those inputs are actualized on a specific platform.

In summary, the dynamics of a multirotor, including forces and torques are given by

$$\dot{\mathbf{p}}_{b/i}^i = \mathbf{v}_{b/i}^i \quad (14.10)$$

$$\dot{\mathbf{v}}_{b/i}^i = g\mathbf{e}_3 - R(\bar{\mathbf{q}}_b^i)DR^\top(\bar{\mathbf{q}}_b^i)\mathbf{v}_{b/i}^i - \frac{T}{m}R^\top(\bar{\mathbf{q}}_b^i)\mathbf{e}_3 \quad (14.11)$$

$$\dot{\bar{\mathbf{q}}}_b^i = \frac{1}{2}\Omega(\omega_{b/i}^b)\bar{\mathbf{q}}_b^i \quad (14.12)$$

$$\dot{\omega}_{b/i}^b = \mathbf{J}^{-1} \left[ -\omega_{b/i}^b \times (\mathbf{J}\omega_{b/i}^b) + \boldsymbol{\tau}^b \right], \quad (14.13)$$

where the control inputs are the thrust  $T$  and the body torque  $\boldsymbol{\tau}^b$ .

### 14.5 LINEAR DESIGN MODELS

The linear design models are derived from Equations (14.10)–(14.13), by first using Euler angles to represent the attitude. In that case we have

$$\dot{\mathbf{p}}_{b/i}^i = \mathbf{v}_{b/i}^i \quad (14.14)$$

$$\dot{\mathbf{v}}_{b/i}^i = g\mathbf{e}_3 - R_b^i(\Theta)DR_b^{i\top}(\Theta)\mathbf{v}_{b/i}^i - \frac{T}{m}R_b^i(\Theta)\mathbf{e}_3 \quad (14.15)$$

$$\dot{\Theta} = S(\Theta)\omega_{b/i}^b \quad (14.16)$$

$$\dot{\omega}_{b/i}^b = \mathbf{J}^{-1} \left[ -\omega_{b/i}^b \times (\mathbf{J}\omega_{b/i}^b) + \boldsymbol{\tau}^b \right], \quad (14.17)$$

where  $\Theta = (\phi, \theta, \psi)^\top$ ,

$$R_b^i(\Theta) = \begin{pmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \theta & \sin \phi \sin \theta & \cos \phi \sin \theta \\ 0 & \cos \phi & -\sin \phi \\ -\sin \theta & \sin \phi \cos \theta & \cos \phi \cos \theta \end{pmatrix}$$

$$S(\Theta) = \begin{pmatrix} 1 & \sin \phi \tan \theta & \cos \phi \tan \theta \\ 0 & \cos \phi & -\sin \phi \\ 0 & \frac{\sin \phi}{\cos \theta} & \frac{\cos \phi}{\cos \theta} \end{pmatrix},$$

where we have isolated the dependence on the heading  $\psi$  since we are interested in linearizing around small roll and pitch angles, but arbitrary yaw angles. When the roll and pitch angles are small, we can approximate  $\cos \varphi \approx 1$  and  $\sin \varphi \approx \varphi$  to get

$$R_b^i(\Theta) \approx \begin{pmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & \phi\theta & \theta \\ 0 & 1 & -\phi \\ -\theta & \phi & 1 \end{pmatrix}$$

$$\approx \begin{pmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & \theta \\ 0 & 1 & -\phi \\ -\theta & \phi & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \cos \psi & -\sin \psi & \theta \cos \psi + \phi \sin \psi \\ \sin \psi & \cos \psi & \theta \sin \psi - \phi \cos \psi \\ -\theta & \phi & 1 \end{pmatrix}$$

$$S(\Theta) \approx \begin{pmatrix} 1 & \phi\theta & \theta \\ 0 & 1 & -\phi \\ 0 & \phi & 1 \end{pmatrix} \approx \begin{pmatrix} 1 & 0 & \theta \\ 0 & 1 & -\phi \\ 0 & \phi & 1 \end{pmatrix},$$

where the nonlinear cross terms are approximated as zero. Computing the drag and thrust terms, where nonlinear cross terms between velocity and

angles are approximated as zero gives, after some algebra

$$\begin{aligned} -RDR^\top \mathbf{v} &\approx -gC_d \begin{pmatrix} v_n \\ v_e \\ 0 \end{pmatrix} \\ -\frac{T}{m} R \mathbf{e}_3 &\approx -\frac{T}{m} \begin{pmatrix} \theta \cos \psi + \phi \sin \psi \\ \theta \sin \psi - \phi \cos \psi \\ 1 \end{pmatrix}. \end{aligned}$$

The thrust force has different effects in the lateral (side-to-side) plane and the longitudinal (up-and-down) direction. In the longitudinal direction, the thrust  $T/m$  directly impacts the down dynamics. In the lateral plane, if the thrust primarily counteracts gravity, then  $T \approx mg$ . In that case the lateral dynamics are primarily impacted by the roll and pitch angles. Accordingly, we further approximate the thrust term as

$$-\frac{T}{m} R \mathbf{e}_3 \approx \begin{pmatrix} -g(\theta \cos \psi + \phi \sin \psi) \\ g(\phi \cos \psi - \theta \sin \psi) \\ -\frac{T}{m} \end{pmatrix}.$$

Therefore, approximating all nonlinear cross terms as zero, Equations (14.10)–(14.13) become

$$\begin{aligned} \begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{p}_d \end{pmatrix} &= \begin{pmatrix} v_n \\ v_e \\ v_e \end{pmatrix} \\ \begin{pmatrix} \dot{v}_n \\ \dot{v}_e \\ \dot{v}_e \end{pmatrix} &= g \begin{pmatrix} -\theta \cos \psi - \phi \sin \psi \\ \phi \cos \psi - \theta \sin \psi \\ 1 \end{pmatrix} - gC_d \begin{pmatrix} v_n \\ v_e \\ 0 \end{pmatrix} - \frac{T}{m} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \\ \begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} &= \begin{pmatrix} p \\ q \\ r \end{pmatrix} \\ \begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} &= \begin{pmatrix} \frac{1}{J_x} \tau_x \\ \frac{1}{J_y} \tau_y \\ \frac{1}{J_z} \tau_z \end{pmatrix}. \end{aligned}$$

We can therefore write the linearized dynamics in second order form as

$$\begin{pmatrix} \ddot{p}_n \\ \ddot{p}_e \end{pmatrix} = -gC_d \begin{pmatrix} \dot{p}_n \\ \dot{p}_e \end{pmatrix} + g \begin{pmatrix} \cos \psi & -\sin \psi \\ \sin \psi & \cos \psi \end{pmatrix} \begin{pmatrix} -\theta \\ \phi \end{pmatrix} + \begin{pmatrix} d_n \\ d_e \end{pmatrix} \quad (14.18)$$

$$\ddot{p}_d = g - \frac{1}{m} T + d_d \quad (14.19)$$

$$\ddot{\phi} = \frac{1}{J_x} \tau_x + d_\phi \quad (14.20)$$

$$\ddot{\theta} = \frac{1}{J_y} \tau_y + d_\theta \quad (14.21)$$

$$\ddot{\psi} = \frac{1}{J_z} \tau_z + d_\psi, \quad (14.22)$$

where we have added the disturbance terms  $d_*$  to model the terms neglected during linearization.

Equation (14.18) is the translational dynamics that are driven by the pitch  $\theta$  and the roll  $\phi$ . For example, when the heading is  $\psi = 0$ , then a negative pitch down will cause the multirotor to accelerate in the north direction. Similarly, a positive roll to the right will cause the multirotor to accelerate in the east direction. Since the pitch and roll angles always cause acceleration in the body frame forward and right direction, the 2D rotation matrix  $\begin{pmatrix} \cos \psi & -\sin \psi \\ \sin \psi & \cos \psi \end{pmatrix}$  transforms the pitch and roll commands to the inertial frame. To obtain linear time-invariant equations, we use feedback linearization and let the commanded roll and pitch angles be given by

$$\begin{pmatrix} -\theta^c \\ \phi^c \end{pmatrix} = \frac{1}{g} \begin{pmatrix} \cos \psi & \sin \psi \\ -\sin \psi & \cos \psi \end{pmatrix} \begin{pmatrix} u_n \\ u_e \end{pmatrix} \quad (14.23)$$

where  $u_n$  and  $u_e$  are pseudo-control signals that are, in-essence, angle commands along the north and east coordinate axes. Equation (14.19) is the altitude dynamics and is affine in the input  $T$ . Hovering at a constant altitude requires zero acceleration, which requires  $T = mg$ . The altitude dynamics can be feedback-linearized by letting

$$T = m(g + u_d) \quad (14.24)$$

where again  $u_d$  is a pseudo-control signal. Equations (14.20)–(14.22) are the roll, pitch, and yaw dynamics.

Using Equations (14.23) and (14.24) in Equations (14.18)–(14.22) results in the linear time-invariant dynamics

$$\ddot{p}_n = -gC_d \dot{p}_n + u_n + d_n \quad (14.25)$$

$$\ddot{p}_e = -gC_d \dot{p}_e + u_e + d_e \quad (14.26)$$

$$\ddot{p}_d = u_d + d_d \quad (14.27)$$

$$\ddot{\phi} = \frac{1}{J_x} \tau_x + d_\phi \quad (14.28)$$

$$\ddot{\theta} = \frac{1}{J_y} \tau_y + d_\theta \quad (14.29)$$

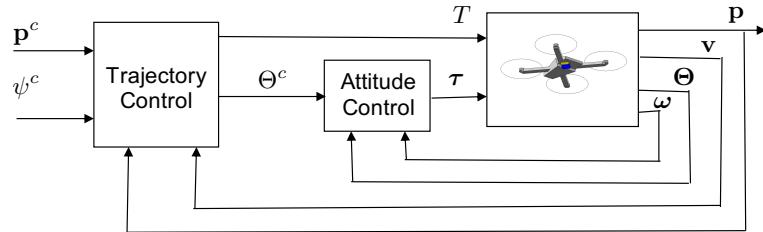
$$\ddot{\psi} = \frac{1}{J_z} \tau_z + d_\psi. \quad (14.30)$$

The associated transfer functions are

$$\begin{aligned} p_n(s) &= \left( \frac{1}{s(s + gC_d)} \right) (u_n(s) + d_n(s)) \\ p_e(s) &= \left( \frac{1}{s(s + gC_d)} \right) (u_e(s) + d_e(s)) \\ p_d(s) &= \left( \frac{1}{s^2} \right) (u_d(s) + d_d(s)) \\ \phi(s) &= \left( \frac{1/J_x}{s^2} \right) (\tau_x(s) + d_\phi(s)) \\ \theta(s) &= \left( \frac{1/J_y}{s^2} \right) (\tau_y(s) + d_\theta(s)) \\ \psi(s) &= \left( \frac{1/J_z}{s^2} \right) (\tau_z(s) + d_\psi(s)). \end{aligned}$$

#### 14.6 AUTOPILOT DESIGN

The high-level architecture for control of a multirotor is shown in Figure 14.9. As shown in the figure, the inputs to the control architecture are the



**Figure 14.9:** Architecture for the control of a multirotor UAV.

commanded position  $\mathbf{p}^c(t)$  in the inertial frame, and the commanded yaw angle  $\psi^c(t)$ , since a multirotor can move along a trajectory at any given yaw angle, and even at a time-varying yaw angle. The *trajectory control block* in Figure 14.9 uses the commanded and actual position, and the commanded yaw angle to produce a commanded thrust and commanded Euler angles. The *attitude control block* uses the commanded and actual Euler angles, and the angular velocity to produce the commanded body frame torque. The commanded torque and thrust are then passed to the multirotor. In practical implementation, the torque and thrust are mixed to produce commanded

angular speeds for each rotor. Attitude control and trajectory control are discussed in the sub-sections below.

#### 14.6.1 Attitude Control

The inner-loop attitude regulation is accomplished using PD control schemes for each of the Euler angles. In particular, we let

$$\tau_x = k_{p_\phi}(\phi^c - \phi) - k_{d_\phi}p \quad (14.31)$$

$$\tau_y = k_{p_\theta}(\theta^c - \theta) - k_{d_\theta}q \quad (14.32)$$

$$\tau_z = k_{p_\psi}(\psi^c - \psi) - k_{d_\psi}r + k_{i_\psi} \int_{-\infty}^t (\psi^c(\tau) - \psi(\tau)) d\tau, \quad (14.33)$$

where  $k_{p_*}$ ,  $k_{i_*}$  and  $k_{d_*}$  are positive proportional, integral, and derivative gains. To analyze the tracking and disturbance rejection properties of the inner-loop, note from Equations (14.28)–(14.30) and (14.31)–(14.33) that for the controlled roll angle we have

$$\begin{aligned} J_x \ddot{\phi} &= \tau_x + d_\phi \\ &= k_{p_\phi}(\phi^c - \phi) - k_{d_\phi}p + d_\phi \\ &= k_{p_\phi}(\phi^c - \phi) - k_{d_\phi}\dot{\phi} + d_\phi, \end{aligned}$$

where in the last line we have used the approximation that  $p \approx \dot{\phi}$ . Taking the Laplace transform and solving for  $\phi(s)$  gives

$$\phi(s) = \frac{k_{p_\phi}}{J_xs^2 + k_{d_\phi}s + k_{p_\phi}} \phi^c(s) + \frac{1}{J_xs^2 + k_{d_\phi}s + k_{p_\phi}} d_\phi(s).$$

Therefore, using the Final Value Theorem, it is clear to see that when  $d_\phi(s) = 0$ , and  $\phi^c(s) = A/s$ , i.e., a step of size  $A$ , that  $\phi(t) \rightarrow A$ . It is also clear to see that when  $\phi^c(s) = 0$  and  $d_\phi(s) = A/s$ , that  $\phi(t) \rightarrow A/k_{p_\phi}$ . Therefore, the size of the proportional gain  $k_{p_\phi}$  dictates the level of disturbance rejection. A similar arguments can be made for the pitch dynamics. Therefore, in tuning the gains, the proportional terms should be made as large as possible without saturating the torque signals. For the yaw dynamics the integrator ensures that constant disturbances can be rejected and the size of  $k_i$  dictates the level of disturbance rejection to a ramp.

#### 14.6.2 Trajectory Control

The outer-loop trajectory regulation is accomplished using PID control for both the longitudinal and lateral dynamics. In particular, we use

$$u_n(t) = \ddot{p}_n^c + gC_d\dot{p}_n(t) + k_{p_n}(p_n^c(t) - p_n(t))$$

$$+ k_{d_n}(\dot{p}_n^c(t) - \dot{p}_n(t)) + k_{i_n} \int_{-\infty}^t (p_n^c(\tau) - p_n(\tau)) d\tau \quad (14.34)$$

$$\begin{aligned} u_e(t) = & \dot{p}_e^c + gC_d \dot{p}_e(t) + k_{p_e}(p_e^c(t) - p_e(t)) \\ & + k_{d_e}(\dot{p}_e^c(t) - \dot{p}_e(t)) + k_{i_e} \int_{-\infty}^t (p_e^c(\tau) - p_e(\tau)) d\tau \end{aligned} \quad (14.35)$$

$$\begin{aligned} u_d(t) = & \ddot{p}_d^c + k_{p_d}(p_d^c(t) - p_d(t)) \\ & + k_{d_d}(\dot{p}_d^c(t) - \dot{p}_d(t)) + k_{i_d} \int_{-\infty}^t (p_d^c(\tau) - p_d(\tau)) d\tau, \end{aligned} \quad (14.36)$$

where  $k_{p_*}$ ,  $k_{i_*}$  and  $k_{d_*}$  are positive control gains, and where we have assumed the existence of feedforward trajectory velocities  $\dot{p}_n^c$ ,  $\dot{p}_e^c$ , and  $\dot{p}_d^c$ , and feedforward trajectory accelerations  $\ddot{p}_n^c$ ,  $\ddot{p}_e^c$ , and  $\ddot{p}_d^c$ . If the feedforward accelerations are unknown, then we set  $\ddot{p}_n^c = \ddot{p}_e^c = \ddot{p}_d^c = 0$ . If the feedforward velocities are unknown, then we set  $\dot{p}_n^c = \dot{p}_e^c = \dot{p}_d^c = 0$ . If the drag coefficient  $C_d$  is unknown, then the first two terms in Equations (14.34) and (14.35) are set to zero.

Given  $u_n$ ,  $u_e$ , and  $u_d$ , the commanded thrust and commanded roll and pitch angles are obtained from Equations (14.24) and (14.23) as

$$T^c = \text{sat}[m(g - u_d)] \quad (14.37)$$

$$\phi^c = \text{sat}\left[\frac{u_e}{g} \cos \psi - \frac{u_n}{g} \sin \psi\right] \quad (14.38)$$

$$\theta^c = \text{sat}\left[-\frac{u_n}{g} \cos \psi - \frac{u_e}{g} \sin \psi\right], \quad (14.39)$$

where  $\text{sat}[\cdot]$  is the saturation function with limits set by the designer.

To understand the behavior of the trajectory controller, we will analyze the north dynamics. To emphasize the different options available to the controller, we write  $u_n$  as

$$\begin{aligned} u_n(t) = & I_1 \ddot{p}_n^c + g\bar{C}_d \dot{p}_n(t) + k_{p_n}(p_n^c(t) - p_n(t)) \\ & + k_{d_n}(I_2 \dot{p}_n^c(t) - \dot{p}_n(t)) + k_{i_n} \int_{-\infty}^t (p_n^c(\tau) - p_n(\tau)) d\tau, \end{aligned}$$

where  $\bar{C}_d$  is the estimated value of  $C_d$  (which may be zero),  $I_1$  is the indicator function which is one if  $\ddot{p}_n^c$  is known and zero otherwise, and  $I_2$  is the indicator function for  $\dot{p}_n^c$ . Therefore, substituting into Equation (14.25) gives

$$\ddot{p}_n = -gC_d \dot{p}_n + I_1 \ddot{p}_n^c + g\bar{C}_d \dot{p}_n(t) + k_{p_n}(p_n^c(t) - p_n(t))$$

$$+ k_{d_n}(I_2 \dot{p}_n^c(t) - \dot{p}_n(t)) + k_{i_n} \int_{-\infty}^t (p_n^c(\tau) - p_n(\tau)) d\tau + d_n.$$

Defining  $\tilde{p}_n \triangleq p_n - p_n^c$  and  $\tilde{C}_d \triangleq C_d - \bar{C}_d$  we get, after some manipulation,

$$\begin{aligned} \ddot{\tilde{p}}_n + (k_{d_n} + g\tilde{C}_d)\ddot{p}_n + k_{p_n}\dot{\tilde{p}}_n + k_{i_n}\tilde{p}_n \\ = -g\tilde{C}_d\ddot{p}_n^c + (I_1 - 1)\ddot{p}_n^c + k_{d_n}(I_2 - 1)\dot{p}_n^c + \dot{d}_n. \end{aligned}$$

Therefore, in the Laplace domain we have

$$\tilde{p}_n(s) = \frac{-g\tilde{C}_d s^2 p_n^c(s) + (I_1 - 1)s^3 p_n^c(s) + k_{d_n}(I_2 - 1)p_n^c(s) + sd(s)}{s^3 + (k_{d_n} + g\tilde{C}_d)s^2 + k_{p_n}s + k_{i_n}}.$$

Using the final value theorem, we can draw the following conclusions.

- The system is stable if  $k_{d_n} + g(C_d - \bar{C}_d) > 0$ . Therefore, it is best to underestimate the drag coefficient.
- If  $C_d$  is known, the feedforward velocity and acceleration are used, and the disturbance is zero, then the tracking error will be zero.
- Constant disturbances are completely rejected and do not influence the tracking error.
- An error in modeling the drag coefficient does not impact the tracking error when  $p_n^c$  is a step or ramp.
- Removing the feedforward acceleration does not result in tracking error when  $p_n^c$  is a step, ramp, or parabola.
- Removing the feedforward velocity does not result in tracking error when  $p_n^c$  is a step or ramp.

#### 14.6.3 Go-to-point Trajectories

One particular application of the controller developed above is when the multirotor is commanded to transition to a particular position and then hold that position. In that case, the commanded trajectory is  $\mathbf{p}^c(t) \equiv \mathbf{p}^c$ , where  $\dot{\mathbf{p}}^c(t) \equiv \ddot{\mathbf{p}}^c(t) \equiv 0$ . In this case, the drag coefficient and the saturation limits on the commanded roll and pitch angles will determine how fast the multirotor transitions to the point.

To derive an approximate solution, assume that the yaw angle is zero and that the commanded position is north of the current position. Letting

$v_n = \dot{p}_n$  and assuming that the control saturates at  $\bar{\theta}$ , the equations of motion are given by

$$\dot{v}_n = -gC_d v_n + g\bar{\theta},$$

which implies that

$$v_n(s) = \left( \frac{g}{s + gC_d} \right) \frac{\bar{\theta}}{s},$$

where we have modeled the saturation as a step input of size  $\bar{\theta}$ . The final value theorem implies that the steady state velocity is  $v_{n,ss} = \bar{\theta}/C_d$ . Therefore, the approximate transit time between initial position  $\mathbf{p}(0)$  and the commanded point  $\mathbf{p}^c$  is

$$T_{transit} \approx \frac{C_d}{\bar{\theta}} \|\mathbf{p}^c - \mathbf{p}(0)\|.$$

## 14.7 SENSORS

Multicopters typically use the same sensor suite as fixed-wing aircraft with the exception of dynamic pressure sensor. The use of a pitot tube requires that the sensor be placed in the oncoming airstream. However, with multicopters it is difficult to place a pitot tube outside of the airflow produced by the propellers. Therefore, we do not assume the availability of an airspeed sensor. With the exception of the accelerometer equation (7.1), all other sensor models are valid for multicopters.

For multicopters, the accelerometer model deserves a more careful analysis. Recall that an accelerometer measures the specific acceleration resolved in the body frame, where the specific acceleration is the total acceleration minus the gravity term. Therefore, relating back to Equation (14.11) we have

$$\begin{aligned} \dot{\mathbf{v}}_{b/i}^i &= g\mathbf{e}_3 - R(\bar{\mathbf{q}}_b^i)DR^\top(\bar{\mathbf{q}}_b^i)\mathbf{v}_{b/i}^i - \frac{T}{m}R(\bar{\mathbf{q}}_b^i)\mathbf{e}_3 \\ &= g\mathbf{e}_3 + \mathbf{a}_s^i, \end{aligned}$$

where  $\mathbf{a}_s^i$  is the specific acceleration resolved in the inertial frame. Expressing for  $\mathbf{a}_s$  in the body frame, implies that the output of the accelerometer is

$$\mathbf{a}_s^b = -DR^\top(\bar{\mathbf{q}}_b^i)\mathbf{v}_{b/i}^i - \frac{T}{m}\mathbf{e}_3. \quad (14.40)$$

In component form with  $\mathbf{a}_s^b \triangleq (a_x, a_y, a_z)^\top$  we have

$$a_x = -gC_d \{\cos \theta [v_n \cos \psi + v_e \sin \psi] - v_d \sin \theta\}$$

$$a_y = -gC_d \left\{ \sin \phi \sin \theta [v_n \cos \psi + v_e \sin \psi] + \cos \phi [-v_n \sin \psi + v_e \cos \psi] + v_d \sin \phi \cos \theta \right\}$$

$$a_z = -\frac{T}{m}.$$

Therefore, we arrive at the seemingly counter-intuitive results that for a quadrotor, the gravity vector does not influence the accelerometer measurement. Rather, the  $z$ -axis accelerometer always measures the applied thrust, and the  $x$ -axis and  $y$ -axis accelerometers measure the induced drag, which is a function of roll and pitch angles. Using the notation of Chapter 7, the accelerometer measurements are given by

$$y_{\text{accel},x} = -gC_d \{ \cos \theta [v_n \cos \psi + v_e \sin \psi] - v_d \sin \theta \} + \eta_{\text{accel},x}$$

$$y_{\text{accel},y} = -gC_d \left\{ \sin \phi \sin \theta [v_n \cos \psi + v_e \sin \psi] + \cos \phi [-v_n \sin \psi + v_e \cos \psi] + v_d \sin \phi \cos \theta \right\} + \eta_{\text{accel},y}$$

$$y_{\text{accel},z} = -\frac{T}{m} + \eta_{\text{accel},z},$$

where  $\eta_*$  are zero mean Gaussian random variables that model the noise.

## 14.8 STATE ESTIMATION

In this section we will use an extended Kalman filter that simultaneously estimates the inertial position and velocity, the Euler angles, and the gyro biases. In that sense, it is similar to the full-state Kalman filter described in Section ???. From Equations (14.14)–(14.16), and Equation (14.40) we have that the mechanized dynamics are

$$\begin{aligned}\dot{\mathbf{p}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= g\mathbf{e}_3 + R(\Theta)\mathbf{a} \\ \dot{\Theta} &= S(\Theta)\boldsymbol{\omega} \\ \dot{\mathbf{b}} &= 0_{3 \times 1},\end{aligned}$$

where  $\mathbf{a}$  is the specific acceleration resolved in the body frame, and

$$R(\Theta) = \begin{pmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{pmatrix}$$

$$S(\Theta) = \begin{pmatrix} 1 & \sin \phi \tan \theta & \cos \phi \tan \theta \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi \sec \theta & \cos \phi \sec \theta \end{pmatrix}.$$

We will assume that the sensors that are available are three-axis gyroscopes and accelerometers, an absolute pressure sensor, a digital compass, GPS north and east measurements, and GPS ground-speed and course measurement. The mathematical models that relate the states to the sensor measurements are given by

$$\begin{aligned} \mathbf{y}_{\text{gyro}} &= \boldsymbol{\omega} + \mathbf{b} + \boldsymbol{\eta}_{\text{gyro}} \\ \mathbf{y}_{\text{accel}} &= \mathbf{a} + \boldsymbol{\eta}_{\text{accel}} \\ y_{\text{abs pres}} &= -\rho g p_d + b_h + \eta_{\text{abs pres}} \\ y_{\text{mag}} &= \psi + \eta_{\text{mag}} \\ y_{\text{GPS},n} &= p_n + \nu_n[n] \\ y_{\text{GPS},e} &= p_e + \nu_e[n] \\ y_{\text{GPS},V_g} &= \sqrt{v_n^2 + v_e^2} + \eta_V \\ y_{\text{GPS},\chi} &= \begin{cases} \psi, & \text{if } \sqrt{v_n^2 + v_e^2} > \epsilon \\ 0, & \text{otherwise} \end{cases} + \eta_\chi. \end{aligned}$$

#### 14.8.1 Propagation model for the EKF

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

$$\begin{aligned} \dot{\mathbf{p}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= g\mathbf{e}_3 + R(\Theta)(\mathbf{y}_{\text{accel}} - \boldsymbol{\eta}_{\text{accel}}) \\ \dot{\Theta} &= S(\Theta)(\mathbf{y}_{\text{gyro}} - \mathbf{b} - \boldsymbol{\eta}_{\text{gyro}}) \\ \dot{\mathbf{b}} &= 0_{3 \times 1}. \end{aligned}$$

Defining

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

we get

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

where

$$f(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} \mathbf{v} \\ g\mathbf{e}_3 + R(\Theta)\mathbf{y}_{\text{accel}} \\ S(\Theta)(\mathbf{y}_{\text{gyro}} - \mathbf{b}) \\ \mathbf{0}_{3 \times 1} \end{pmatrix}$$

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

$$G_a(\mathbf{x}) = \begin{pmatrix} \mathbf{0}_{3 \times 3} \\ -R(\Theta) \\ \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} \end{pmatrix},$$

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

The Jacobian of  $f$  is

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

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

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

Therefore

$$\frac{\partial [R(\Theta)\mathbf{y}_{\text{gyro}}]}{\partial \Theta} = \left( \frac{\partial R(\Theta)}{\partial \phi} \mathbf{y}_{\text{gyro}}, \quad \frac{\partial R(\Theta)}{\partial \theta} \mathbf{y}_{\text{gyro}}, \quad \frac{\partial R(\Theta)}{\partial \psi} \mathbf{y}_{\text{gyro}} \right)$$

$$\frac{\partial [S(\Theta)(\mathbf{y}_{\text{gyro}} - \mathbf{b})]}{\partial \Theta} = \left( \frac{\partial S(\Theta)}{\partial \phi} (\mathbf{y}_{\text{gyro}} - \mathbf{b}), \quad \frac{\partial S(\Theta)}{\partial \theta} (\mathbf{y}_{\text{gyro}} - \mathbf{b}), \quad \frac{\partial S(\Theta)}{\partial \psi} (\mathbf{y}_{\text{gyro}} - \mathbf{b}) \right),$$

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

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

$$Q = \text{diag}([\sigma_{p_n}^2, \sigma_{p_e}^2, \sigma_{p_d}^2, \sigma_u^2, \sigma_v^2, \sigma_w^2, \sigma_\phi^2, \sigma_\theta^2, \sigma_\psi^2, \sigma_{b_x}^2, \sigma_{b_y}^2, \sigma_{b_z}^2])$$

are tuning parameters.

### 14.8.2 Sensor Update Equations

#### 14.8.2.1 Static Pressure Sensor

The model for the static pressure sensor is

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

Therefore, the Jacobian is given by

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

#### 14.8.2.2 Digital Compass Sensor

The model for the static pressure sensor is

$$h_{\text{mag}}(\mathbf{x}) = \psi.$$

Therefore, the Jacobian is given by

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

#### 14.8.2.3 GPS North sensor

The model for the GPS north sensor is

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

with associated Jacobian

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

#### 14.8.2.4 GPS East sensor

The model for the GPS east sensor is

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

with associated Jacobian

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

#### 14.8.2.5 GPS ground-speed sensor

The model for the GPS ground speed is

$$h_{\text{GPS}, V_g}(\mathbf{x}) = \sqrt{v_n^2 + v_e^2}$$

with associated Jacobian

$$C_{\text{GPS}, V_g}(\mathbf{x}) = \left( 0, 0, 0, \frac{v_n}{\sqrt{v_n^2 + v_e^2}}, \frac{v_e}{\sqrt{v_n^2 + v_e^2}}, 0, 0, 0, 0, 0, 0, 0 \right).$$

#### 14.8.2.6 GPS course

The model for the GPS ground speed is

$$h_{\text{GPS}, \chi}(\mathbf{x}) = \psi$$

with associated Jacobian

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

In the case of GPS course, the measurement update should only be performed when the GPS ground speed is sufficiently high.

### 14.8.3 Algorithm for Direct Kalman Filter

We assume that the gyro, accelerometers, pressure sensor, and digital compass 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

$$\begin{aligned} \hat{\mathbf{x}} &= (p_n(0), p_e(0), p_d(0), 0, 0, 0, 0, 0, \psi(0), 0, 0, 0)^\top, \\ P &= \text{diag} \left( [e_{p_n}^2, e_{p_e}^2, e_{p_d}^2, e_{v_n}^2, e_{v_e}^2, e_{v_d}^2, e_\phi^2, e_\theta^2, e_\psi^2, e_{b_x}^2, e_{b_y}^2, e_{b_z}^2] \right), \end{aligned}$$

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

#### 1. At the fast sample rate $T_s$

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

$$\dot{\hat{\mathbf{x}}} = f(\hat{\mathbf{x}}, \mathbf{y})$$

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

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

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

1.c. Update  $\hat{\mathbf{x}}$  and  $P$  with the digital compass sensor according to

$$\begin{aligned} L &= P^- C_{\text{mag}}^\top(\hat{\mathbf{x}}^-) / (\sigma_{\text{mag}}^2 + C_{\text{mag}}(\hat{\mathbf{x}}^-) P^- C_{\text{mag}}^\top(\hat{\mathbf{x}}^-)) \\ P^+ &= (I - LC_{\text{mag}}(\hat{\mathbf{x}}^-))P^- \\ \hat{\mathbf{x}}^+ &= \hat{\mathbf{x}}^- + L(y_{\text{mag}} - h_{\text{mag}}(\hat{\mathbf{x}}^-)). \end{aligned}$$

## 2. When GPS measurements are received at $T_{\text{GPS}}$ :

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

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

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

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

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

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

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

$$\begin{aligned} L &= P^- C_{\text{GPS},\chi}^\top(\hat{\mathbf{x}}^-) / (\sigma_{\text{GPS},\chi}^2 + C_{\text{GPS},\chi}(\hat{\mathbf{x}}^-) P^- C_{\text{GPS},\chi}^\top(\hat{\mathbf{x}}^-)) \\ P^+ &= (I - LC_{\text{GPS},\chi}(\hat{\mathbf{x}}^-))P^- \\ \hat{\mathbf{x}}^+ &= \hat{\mathbf{x}}^- + L(y_{\text{GPS},\chi} - h_{\text{GPS},V_\chi}(\hat{\mathbf{x}}^-)). \end{aligned}$$

### 14.10 TRAJECTORY FOLLOWING

For multirotor systems, since the velocity of the aircraft is completely controllable, we use a trajectory following approach instead of a path following approach. The trajectory tracking strategy is given in Equations (14.31)–(14.39), and is summarized below in Algorithm 15 for convenience, where we use the notation  $\mathbf{p}^c(t) = (p_n^c(t), p_e^c(t), p_d^c(t))^\top$ .

---

#### Algorithm 15 Trajectory tracking algorithm

---

**Require:** The desired trajectory

$$\mathcal{T}(t) = \{\mathbf{p}^c(t), \dot{\mathbf{p}}^c(t), \ddot{\mathbf{p}}^c(t), \psi^c(t)\}$$

and the states  $\mathbf{p}_{b/i}^i, \mathbf{v}_{b/i}^i, R_b^i, \omega_{b/i}^b$ :

1: Compute the intermediate control signal  $\mathbf{u} = (u_n, u_e, u_d)^\top$ :

$$\begin{aligned} \mathbf{u}(t) = & \ddot{\mathbf{p}}^c + RDR^\top \mathbf{p}^c + K_p(\mathbf{p}^c - \mathbf{p}_{b/i}^i) + K_d(\dot{\mathbf{p}}^c - \mathbf{v}_{b/i}^i) \\ & + K_i \int_{-\infty}^t (\mathbf{p}^c(\tau) - \mathbf{p}_{b/i}^i(\tau)) d\tau. \end{aligned}$$

2: Compute the Euler angles  $\phi, \theta, \psi$  from  $R_b^i$ .

3: Compute the commanded thrust, roll angle, and pitch angle:

$$\begin{aligned} T^c &= \text{sat}[m(g - u_d)] \\ \phi^c &= \text{sat}\left[\frac{u_e}{g} \cos \psi - \frac{u_n}{g} \sin \psi\right] \\ \theta^c &= \text{sat}\left[-\frac{u_n}{g} \cos \psi - \frac{u_e}{g} \sin \psi\right]. \end{aligned}$$

4: Compute the commanded torques from  $\omega_{b/i}^b = (p, q, r)^\top$ :

$$\begin{aligned} \tau_x^c &= k_{p_\phi}(\phi^c - \phi) - k_{d_\phi}p \\ \tau_y^c &= k_{p_\theta}(\theta^c - \theta) - k_{d_\theta}q \\ \tau_z^c &= k_{p_\psi}(\psi^c - \psi) - k_{d_\psi}r + k_{i_\psi} \int_{-\infty}^t (\psi^c(\tau) - \psi(\tau)) d\tau. \end{aligned}$$

---

**return** The commanded thrust  $T^c$  and torque  $\boldsymbol{\tau}^c = (\tau_x^c, \tau_y^c, \tau_z^c)^\top$ .

---

### 14.11 PATH MANAGER

The trajectory tracking algorithm outlined in Algorithm 15 requires a desired time-parameterized trajectory of the form

$$\mathcal{T}(t) = \{\mathbf{p}^c(t), \dot{\mathbf{p}}^c(t), \ddot{\mathbf{p}}^c(t), \psi^c(t)\}.$$

There are a number of ways to compactly represent trajectories in this form. To be consistent with Chapter 11 we will restrict the discussion in this section to straight-line paths. In contrast to fixed-wing vehicles where the airspeed is constant, for multirotors we can prescribe an arbitrary speed profile along the waypoint path. In addition, the heading profile can also be arbitrarily defined. Therefore, we expand the definition of a waypoint given in Chapter 11 to

$$\hat{\mathbf{w}} = \begin{pmatrix} \mathbf{w} \\ s \\ \psi \end{pmatrix},$$

where  $\mathbf{w} \in \mathbb{R}^3$  is a desired inertial waypoint position,  $s \in \mathbb{R}^+$  is a desired positive speed at the waypoint, and  $\psi \in (-\pi, \pi]$  is a desired heading from North at the waypoint. A waypoint path is then specified as an ordered sequence of  $N$  extended waypoints:

$$\mathcal{W} = \{\hat{\mathbf{w}}_1, \hat{\mathbf{w}}_2, \dots, \hat{\mathbf{w}}_N\}.$$

In this section we will assume a linear progression in position, speed, and heading between waypoints  $\hat{\mathbf{w}}_{i-1}$  and  $\hat{\mathbf{w}}_i$  as

$$\mathbf{p}^c(t) = \sigma(t)\mathbf{w}_i + (1 - \sigma(t))\mathbf{w}_{i-1} \quad (14.41)$$

$$\psi^c(t) = \sigma(t)\psi_i + (1 - \sigma(t))\psi_{i-1} \quad (14.42)$$

where  $\sigma(t) \in [0, 1]$  is a monotonically non-decreasing function that reaches  $\sigma = 1$  in finite time. Therefore, the commanded velocity and acceleration profiles are

$$\dot{\mathbf{p}}^c(t) = \dot{\sigma}(\mathbf{w}_i - \mathbf{w}_{i-1}) \quad (14.43)$$

$$\ddot{\mathbf{p}}^c(t) = \ddot{\sigma}(\mathbf{w}_i - \mathbf{w}_{i-1}). \quad (14.44)$$

Accordingly, the path manager not only keeps track of the current waypoint segment, but also handles the time evolution of the parameter  $\sigma$ .

To derive an evolution law for  $\sigma$ , first note that changing from a fixed constant velocity at time  $t_{i-1}$  to a fixed constant velocity at time  $t_i$  will require a constant acceleration in the interval  $[t_{i-1}, t_i]$ . Therefore, referring to Equation (14.44), we expect  $\sigma$  to satisfy  $\ddot{\sigma} = a$ , where  $a$  is constant. Also

note that initial and final conditions can be derived from Equation (14.43) as

$$\begin{aligned}\dot{\sigma}(t_{i-1}^+) &= \frac{s_{i-1}}{\|\mathbf{w}_i - \mathbf{w}_{i-1}\|} \\ \dot{\sigma}(t_i^-) &= \frac{s_i}{\|\mathbf{w}_i - \mathbf{w}_{i-1}\|},\end{aligned}$$

from which we get, for  $t \in (t_{i-1}, t_i)$

$$\dot{\sigma}(t) = \dot{\sigma}(t_{i-1}^+) + a \int_{t_{i-1}}^t d\tau = \dot{\sigma}(t_{i-1}^+) + a(t - t_{i-1}).$$

Therefore, the required acceleration is

$$a = \frac{\dot{\sigma}(t_i^-) - \dot{\sigma}(t_{i-1}^+)}{t_i - t_{i-1}} = \frac{s_i - s_{i-1}}{\|\mathbf{w}_i - \mathbf{w}_{i-1}\| (t_i - t_{i-1})}. \quad (14.45)$$

Using the boundary conditions  $\sigma(t_{i-1}^+) = 0$  and  $\sigma(t_i^-) = 1$ , integrate  $\ddot{\sigma} = a$  to get

$$\begin{aligned}\sigma(t) &= \sigma(t_{i-1}^+) + \dot{\sigma}(t_{i-1}^+) (t - t_{i-1}) + a \int_{t_{i-1}}^t \int_{t_{i-1}}^s d\tau ds \\ &= \dot{\sigma}(t_{i-1}^+) (t - t_{i-1}) + \frac{a}{2} (t - t_{i-1})^2.\end{aligned}$$

Using the fact that  $\sigma(t_i^-) = 1$  gives

$$\begin{aligned}1 &= \dot{\sigma}(t_{i-1}^+) (t_i - t_{i-1}) + \frac{a}{2} (t_i - t_{i-1})^2 \\ \implies 1 &= \frac{s_{i-1}}{\|\mathbf{w}_i - \mathbf{w}_{i-1}\|} (t_i - t_{i-1}) + \frac{1}{2} \frac{s_i - s_{i-1}}{\|\mathbf{w}_i - \mathbf{w}_{i-1}\| (t_i - t_{i-1})} (t_i - t_{i-1})^2 \\ \implies 1 &= \frac{1}{2} \frac{s_i + s_{i-1}}{\|\mathbf{w}_i - \mathbf{w}_{i-1}\|} (t_i - t_{i-1}) \\ \implies (t_i - t_{i-1}) &= \frac{2 \|\mathbf{w}_i - \mathbf{w}_{i-1}\|}{s_i + s_{i-1}}.\end{aligned}$$

Substituting this expression into Equation (14.45) gives that the required acceleration between waypoints is

$$a = \frac{s_i^2 - s_{i-1}^2}{2 \|\mathbf{w}_i - \mathbf{w}_{i-1}\|^2},$$

from which we derive the commanded position, velocity, acceleration, and heading commands by integrating the second order equation

$$\ddot{\sigma} = \frac{s_i^2 - s_{i-1}^2}{2 \|\mathbf{w}_i - \mathbf{w}_{i-1}\|^2}, \quad \sigma(t_{i-1}^+) = 0, \quad \dot{\sigma}(t_{i-1}^+) = \frac{s_{i-1}}{\|\mathbf{w}_i - \mathbf{w}_{i-1}\|} \quad (14.46)$$

for  $t \geq t_{i-1}^+$  and using Equations (14.41), (14.43), (14.44), and (14.42) and then updating the waypoints when  $\sigma(t) = 1$ .

The path management algorithm is summarized in Algorithm 16.

---

**Algorithm 16** Multirotor Path Manager

---

**Require:** The current position  $\mathbf{p}$  and the ordered sequence of extended waypoints:

$$\mathcal{W} = \{\hat{\mathbf{w}}_1, \hat{\mathbf{w}}_2, \dots, \hat{\mathbf{w}}_N\}.$$

- 1: **Initialize**
- 2:    Waypoint pointer:  $i \leftarrow 2$
- 3:    Path parameter:  $x_1 \triangleq \sigma \leftarrow 0, \quad x_2 \triangleq \dot{\sigma} \leftarrow \frac{s_1}{\|\mathbf{w}_2 - \mathbf{w}_1\|}$
- 4: **End**
- 5: Integrate the path parameters (using for example RK4):

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{s_i^2 - s_{i-1}^2}{2 \|\mathbf{w}_i - \mathbf{w}_{i-1}\|^2}\end{aligned}$$

- 6: **if**  $x_1 \geq 1$  **then** update the path parameters as:
- 7:     $i \leftarrow i + 1, \quad x_1 \leftarrow 0, \quad x_2 \leftarrow \frac{s_{i-1}}{\|\mathbf{w}_i - \mathbf{w}_{i-1}\|}$
- 8: **end if**
- 9: Compute the commanded trajectory:

$$\begin{aligned}\mathbf{p}^c &\leftarrow x_1 \mathbf{w}_i + (1 - x_1) \mathbf{w}_{i-1} \\ \dot{\mathbf{p}}^c &\leftarrow x_2 (\mathbf{w}_i - \mathbf{w}_{i-1}) \\ \ddot{\mathbf{p}}^c &\leftarrow \dot{x}_2 (\mathbf{w}_i - \mathbf{w}_{i-1}) \\ \psi^c &\leftarrow x_1 \psi_i + (1 - x_1) \psi_{i-1}\end{aligned}$$

**return** The commanded trajectory

$$\mathcal{T} = \{\mathbf{p}^c, \dot{\mathbf{p}}^c, \ddot{\mathbf{p}}^c, \psi^c\}$$


---

### 14.12 PATH PLANNING

The literature contains many different strategies for path planning for quadro-tors. Rather than surveying that literature, in this section, we describe a simple strategy for adapting the RRT algorithm introduced in Chapter 12, recognizing that it will only be adequate in limited applications.

Suppose that the flight objectives allow for the quadrotor to take-off from the initial position at zero altitude, transition to a desired nominal altitude and then fly through the environment at a constant altitude and constant nominal speed until reaching the final waypoint, and then landing at zero altitude below that waypoint. Then the RRT algorithm in Section 12.1.2 can be adapted by planning from the start to the end location at the specified altitude, and then augmenting the path to include start and end waypoints at zero altitude. The speed of the start and end waypoint is set to zero, and the speed of the other waypoints is set to the nominal speed. The heading at each waypoint can be selected to accomplish other mission objectives, but in Algorithm 17 will be set to a nominal heading of  $\psi^{nom}$ .

Let  $\hat{\mathbf{w}}_s = (n_s, e_s, d_s, s_s, \psi_s)^\top$  be the start configuration, and let  $\hat{\mathbf{w}}_e = (n_e, e_e, d_e, s_e, \psi_e)^\top$  be the end configuration, where  $(n_*, e_*, d_*)^\top$  are the NED inertial coordinates,  $s_*$  is the speed, and  $\psi_*$  is the heading. Let  $h^{nom}$  be the nominal height,  $s^{nom}$  the nominal speed, and  $\psi^{nom}$  the nominal heading, then a possible RRT planning algorithm for multirotors is listed in Algorithm 17.

---

**Algorithm 17** RRT for Multirotors:

---

$$\mathcal{W} = \text{multirotorRRT}(\mathcal{T}, \hat{\mathbf{w}}_s, \hat{\mathbf{w}}_e, h^{nom}, s^{nom}, \psi^{nom})$$


---

**Input:** Terrain map  $\mathcal{T}$ , start configuration  $\hat{\mathbf{w}}_s$ , end configuration  $\hat{\mathbf{w}}_e$ , nominal altitude  $h^{nom}$ , speed  $s^{nom}$ , heading  $\psi^{nom}$

1: Let  $\mathbf{p}'_s = (n_s, e_s, -h^{nom})^\top$ , and  $\mathbf{p}'_e = (n_e, e_e, -h^{nom})^\top$

2: Use Algorithm 13 to plan straight-line 3D constant altitude paths through the environment:

$$\{\mathbf{p}_1, \dots, \mathbf{p}_N\} = \text{planRRT}(\mathcal{T}, \mathbf{p}'_s, \mathbf{p}'_e)$$

3: Create the associated waypoints

$$\hat{\mathbf{w}}_i = (\mathbf{p}_i^\top, s^{nom}, \psi^{nom})^\top, \quad i = 1 \dots N$$

**return** The extended waypoint path

---


$$\mathcal{W} = \{\hat{\mathbf{w}}_s, \hat{\mathbf{w}}_1, \dots, \hat{\mathbf{w}}_N, \hat{\mathbf{w}}_e\}$$


---

The extended waypoint path returned from Algorithm 17 includes the actual start  $\hat{\mathbf{w}}_s$  and end  $\hat{\mathbf{w}}_e$  configurations of the multirotor, the configuration  $\hat{\mathbf{w}}_1$  that is at the same north-east position as  $\hat{\mathbf{w}}_s$  but at the nominal height, speed, and heading, and the configuration  $\hat{\mathbf{w}}_N$  that is at the same north-east position as  $\hat{\mathbf{w}}_e$  but at the nominal height, speed, and heading, as well as the intermediate waypoints  $\hat{\mathbf{w}}_2, \dots, \hat{\mathbf{w}}_{N-1}$  that specify the path through the environment at the nominal height, speed, and heading.

### 14.13 VISION GUIDED NAVIGATION

In this section we will cover two different algorithms for vision-based navigation of a quadrotor: vision-based altitude hold over a target, and vision-based target following. In both cases we will assume that the camera is facing down and that the target remains in the camera field-of-view. The camera can either be gimbaled or not gimbaled, but our equations will assume a gimbal to cover the more general case.

We will assume that by some means, manual or automated, the pixel location and the pixel size of target at each time are known. Similar to Chapter 13,  $\epsilon_s$  denotes the pixel size of the target, and  $\check{\ell}^c$  denotes the normalized line-of-sight vector derived from the pixel location of the target using Equation (13.9). The line-of-sight vector in the inertial frame is given by

$$\check{\ell}^i = R_v^{b\top} R_b^{g\top} R_g^{c\top} \check{\ell}^c, \quad (14.47)$$

where  $R_v^b$  is the rotation matrix from vehicle or inertial frame to the body frame given in Equation (2.4),  $R_b^g$  is the rotation matrix from body to gimbal given in Equation (13.3), and  $R_g^c$  is the rotation matrix from gimbal to camera given in Equation (13.4). The vision-based target tracking problem is shown pictorially in Figure 14.10.

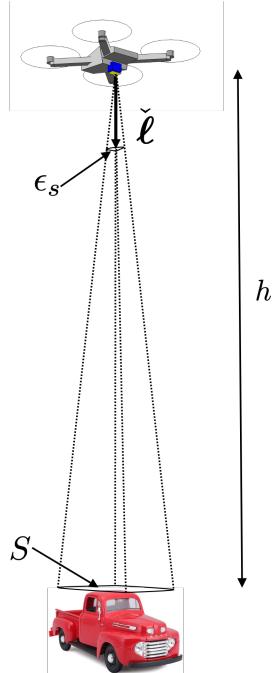
#### 14.13.1 Vision-Based Altitude Hold

From Equations (14.25)–(14.30) we see that the altitude of the aircraft is given by Equation (14.27) which we repeat here for convenience as

$$\ddot{p}_d = u_d, \quad (14.48)$$

where we have ignored the disturbance term. We assume that other control loops are holding the position and attitude of the aircraft at appropriate values. The objective is to derive a control law for  $u_d$  that maintains the multirotor at a constant altitude above the target.

Figure 14.10 shows a quadrotor hovering over a target with physical size  $S$  ( $m^2$ ). When the target is projected onto the image plane, the resulting



**Figure 14.10:** Vision-based target tracking from a quadrotor, where  $\check{\ell}$  is the measured normalized line-of-sight vector and  $\epsilon_s$  is the measured pixel area, and  $S$  is the size(area) of the target being tracked..

pixel area is  $\epsilon_s$  (pixels<sup>2</sup>). From geometry, we see that

$$\frac{h^2}{S} = \frac{f^2}{\epsilon_s}, \quad (14.49)$$

where  $h$  is the altitude of the multirotor, and  $f$  is the camera focal length in units of pixels. Since  $p_d = -h$  we have that

$$\epsilon_s = \frac{f^2 S}{p_d^2} \quad (14.50)$$

Differentiating gives

$$\dot{\epsilon}_s = f^2 S \left( \frac{-2p_d \dot{p}_d}{p_d^4} \right) = -2\epsilon_s \frac{\dot{p}_d}{p_d}.$$

Differentiating again and simplifying gives

$$\ddot{\epsilon}_s = -2\epsilon_s \left( \frac{p_d \ddot{p}_d - \dot{p}_d^2}{p_d^2} \right) - 2\dot{\epsilon}_s \frac{\dot{p}_d}{p_d}$$

$$\begin{aligned}
&= -2\epsilon_s \frac{u_d}{p_d} + 2\epsilon_s \left( -\frac{1}{2} \frac{\dot{\epsilon}_s}{\epsilon_s} \right)^2 - 2\dot{\epsilon}_s \left( -\frac{1}{2} \frac{\dot{\epsilon}_s}{\epsilon_s} \right) \\
&= -2\epsilon_s \frac{u_d}{p_d} + \frac{3}{2} \frac{\dot{\epsilon}_s^2}{\epsilon_s}.
\end{aligned}$$

From Equation (14.50) we have that  $p_d = f\sqrt{S}/\sqrt{\epsilon_s}$ , which implies that

$$\ddot{\epsilon}_s = - \left( \frac{2\epsilon_s^{3/2}}{f\sqrt{S}} \right) u_d + \frac{3}{2} \frac{\dot{\epsilon}_s^2}{\epsilon_s}. \quad (14.51)$$

Suppose that the objective is to track the target at a height of  $h^c = -p_d^c$ , then from Equation (14.50) the desired pixel size in the image is

$$\epsilon_s^c = \frac{f^2 S}{(h^c)^2}.$$

Setting

$$u_d = - \left( \frac{f\sqrt{S}}{2\epsilon_s^{3/2}} \right) \left[ -\frac{3}{2} \frac{\dot{\epsilon}_s^2}{\epsilon_s} - k'_d \dot{\epsilon}_s + k'_p (\epsilon_s^c - \epsilon_s) \right] \quad (14.52)$$

results in

$$\ddot{\epsilon}_s + k'_d \dot{\epsilon}_s + k'_p \epsilon_s = k'_p \epsilon_s^c$$

implying that  $\epsilon_s(t) \rightarrow \epsilon_s^c$ .

The downside to this equation is that it requires knowledge of the target size  $S$  and the focal length  $f$ . Noting that Equation (14.52) can be written as

$$\begin{aligned}
u_d &= - \left( \frac{1}{2\epsilon_s^{3/2}} \right) \left[ -\frac{3f\sqrt{S}}{2} \frac{\dot{\epsilon}_s^2}{\epsilon_s} - (k'_d f \sqrt{S}) \dot{\epsilon}_s + (k'_p f \sqrt{S})(\epsilon_s^c - \epsilon_s) \right] \\
&= - \left( \frac{1}{2\epsilon_s^{3/2}} \right) \left[ -(k'_d f \sqrt{S}) \left( 1 + \frac{3}{2k'_d} \frac{\dot{\epsilon}_s}{\epsilon_s} \right) \dot{\epsilon}_s + (k'_p f \sqrt{S})(\epsilon_s^c - \epsilon_s) \right],
\end{aligned}$$

and then implementing

$$u_d = - \left( \frac{1}{2\epsilon_s^{3/2}} \right) [-k_d \dot{\epsilon}_s + k_p (\epsilon_s^c - \epsilon_s)], \quad (14.53)$$

results in the closed loop system

$$\ddot{\epsilon}_s + \frac{k_d}{f\sqrt{S}} \left( 1 + \frac{3f\sqrt{S}}{2k_d} \frac{\dot{\epsilon}_s}{\epsilon_s} \right) \dot{\epsilon}_s + \frac{k_p}{f\sqrt{S}} \epsilon_s = \frac{k_p}{f\sqrt{S}} \epsilon_s^c.$$

We can therefore tune the gains  $k_p$  and  $k_d$  to get the desired dynamic response and to ensure that during the descent

$$\left| \frac{\dot{\epsilon}_s}{\epsilon_s} \right| << \frac{2k_d}{3f\sqrt{S}}.$$

#### 14.13.2 Vision Based Target Following

From Equations (14.25) and (14.26) we have, ignoring the disturbance terms, that

$$\ddot{p}_n = -gC_d\dot{p}_n + u_n \quad (14.54)$$

$$\ddot{p}_e = -gC_d\dot{p}_e + u_e. \quad (14.55)$$

If the target is not accelerating, then to track the target perfectly, we would like to implement

$$\begin{aligned} u_n^{des} &= gC_d\dot{p}_n + k_d\dot{\tilde{p}}_n + k_p\tilde{p}_n \\ u_e^{des} &= gC_d\dot{p}_e + k_d\dot{\tilde{p}}_e + k_p\tilde{p}_e, \end{aligned} \quad (14.56)$$

where  $\tilde{p}_n = p_{t,n} - p_n$  and  $\tilde{p}_e = p_{t,e} - p_e$  are the north and east position errors between the target and the multirotor, and where selecting  $k_p > 0$  and  $k_d > 0$  results in the stable tracking dynamics

$$\begin{aligned} \ddot{\tilde{p}}_n + k_d\dot{\tilde{p}}_n + k_p\tilde{p}_n &= 0 \\ \ddot{\tilde{p}}_e + k_d\dot{\tilde{p}}_e + k_p\tilde{p}_e &= 0. \end{aligned}$$

Unfortunately, we cannot directly measure  $\tilde{p}_n$ ,  $\dot{\tilde{p}}_n$ ,  $\tilde{p}_e$ , or  $\dot{\tilde{p}}_e$ .

To obtain estimates of  $\tilde{p}_n$ ,  $\dot{\tilde{p}}_n$ ,  $\tilde{p}_e$ , or  $\dot{\tilde{p}}_e$  from measurements note that from Equation (14.47) we measure the line-of-sight vector in the inertial frame as

$$\check{\ell}^i = \begin{pmatrix} \ell_n \\ \ell_e \\ \ell_d \end{pmatrix} = \frac{\mathbf{p}_{t/i}^i - \mathbf{p}_{b/i}^i}{\|\mathbf{p}_{t/i}^i - \mathbf{p}_{b/i}^i\|} = \frac{1}{\|\mathbf{p}_{t/i}^i - \mathbf{p}_{b/i}^i\|} \begin{pmatrix} \tilde{p}_n \\ \tilde{p}_e \\ \tilde{p}_d \end{pmatrix},$$

If we assume that the multirotor is approximately directly overhead the target, then  $\|\mathbf{p}_{t/i}^i - \mathbf{p}_{b/i}^i\| \approx h$ , where  $h$  is the height-above-ground, and can be approximated from Equation (14.49) as  $h = \sqrt{Sf^2/\epsilon_s}$ . Therefore we can approximate  $\tilde{p}_n$  and  $\tilde{p}_e$  as

$$\begin{aligned} \tilde{p}_n &\approx \ell_n \sqrt{\frac{\epsilon_s}{Sf^2}} \\ \tilde{p}_e &\approx \ell_e \sqrt{\frac{\epsilon_s}{Sf^2}}. \end{aligned}$$

Differentiating  $\check{\ell}^i$  gives

$$\dot{\check{\ell}}^i = \begin{pmatrix} \dot{\ell}_n \\ \dot{\ell}_e \\ \dot{\ell}_d \end{pmatrix} = \left( I - \check{\ell}^i \check{\ell}^{i\top} \right) \frac{\dot{\mathbf{p}}_{t/i}^i - \dot{\mathbf{p}}_{b/i}^i}{\| \mathbf{p}_{t/i}^i - \mathbf{p}_{b/i}^i \|} = \frac{\left( I - \check{\ell}^i \check{\ell}^{i\top} \right)}{\| \mathbf{p}_{t/i}^i - \mathbf{p}_{b/i}^i \|} \begin{pmatrix} \dot{\tilde{p}}_n \\ \dot{\tilde{p}}_e \\ \dot{\tilde{p}}_d \end{pmatrix}$$

Again assuming that the multirotor is approximately directly overhead the target, then  $\check{\ell}^i \approx (0, 0, 1)^\top$ , resulting in

$$\begin{aligned} \dot{\tilde{p}}_n &\approx \dot{\ell}_n \sqrt{\frac{\epsilon_s}{Sf^2}} \\ \dot{\tilde{p}}_e &\approx \dot{\ell}_e \sqrt{\frac{\epsilon_s}{Sf^2}}. \end{aligned}$$

From Equation (14.56) the target tracking algorithm is therefore given by

$$\begin{aligned} u_n &= gC_d \dot{p}_n + \sqrt{\frac{\epsilon_s}{Sf^2}} (k_d \dot{\ell}_n + k_p \ell_n) \\ u_e &= gC_d \dot{p}_e + \sqrt{\frac{\epsilon_s}{Sf^2}} (k_d \dot{\ell}_e + k_p \ell_e). \end{aligned}$$

#### 14.14 CHAPTER SUMMARY

#### NOTES AND REFERENCES

## 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  $\chi^c$  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  $\sigma$ .
- 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,  $\omega$  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,  $a_{\beta*}$  is used to denote  $a_{\beta_1}$  and  $a_{\beta_2}$ .

$a_{\beta*}$	Constants for transfer function associated with <a href="#">sideslip dynamics</a> ( <a href="#">chapter 5</a> )
$a_{\phi*}$	Constants for transfer function associated with roll dynamics ( <a href="#">chapter 5</a> )
$a_{\theta*}$	Constants for transfer function associated with pitch dynamics ( <a href="#">chapter 5</a> )
$a_{V*}$	Constants for transfer function associated with airspeed dynamics ( <a href="#">chapter 5</a> )
$A_R$	Wing aspect ratio ( <a href="#">chapter 4</a> )
$\alpha$	<a href="#">Angle-of-attack</a> ( <a href="#">chapter 2</a> )
$\alpha_{az}$	Gimbal azimuth angle ( <a href="#">chapter 13</a> )
$\alpha_{el}$	Gimbal elevation angle ( <a href="#">chapter 13</a> )
$b$	Wing span ( <a href="#">chapter 4</a> )
$b_*$	Coefficients for reduced order model of the autopilot ( <a href="#">chapter 9</a> )
$\beta$	sideslip angle ( <a href="#">chapter 2</a> )
$c$	Mean aerodynamic chord of the wing ( <a href="#">chapter 4</a> )
$C_D$	Aerodynamic drag coefficient ( <a href="#">chapter 4</a> )
$C_L$	Aerodynamic lift coefficient ( <a href="#">chapter 4</a> )
$C_{M_{x*}}$	Aerodynamic moment coefficient along the body frame $x$ -axis ( <a href="#">chapter 4</a> )
$C_{M_{y*}}$	Aerodynamic pitching moment coefficient along the body frame $y$ -axis ( <a href="#">chapter 4</a> )
$C_{M_{z*}}$	Aerodynamic moment coefficient along the body frame $z$ -axis ( <a href="#">chapter 4</a> )
$C_{p*}$	Aerodynamic moment coefficient along the body frame $x$ -axis ( <a href="#">chapter 5</a> )
$C_{prop}$	Aerodynamic coefficient for the propeller ( <a href="#">chapter 4</a> )

$C_{q_*}$	Aerodynamic moment coefficient along the body frame $y$ -axis ( <a href="#">chapter 5</a> )
$C_{r_*}$	Aerodynamic moment coefficient along the body frame $z$ -axis ( <a href="#">chapters 4, 5</a> )
$C_{X_*}$	Aerodynamic force coefficient along the body frame $x$ -axis ( <a href="#">chapters 4, 5</a> )
$C_{Y_*}$	Aerodynamic force coefficient along the body frame $y$ -axis ( <a href="#">chapter 4</a> )
$C_{Z_*}$	Aerodynamic force coefficient along the body frame $z$ -axis ( <a href="#">chapters 4, 5</a> )
$\chi$	Course angle ( <a href="#">chapter 2</a> )
$\chi_c$	Crab angle: $\chi_c = \chi - \psi$ ( <a href="#">chapter 2</a> )
$\chi_d(e_{py})$	Desired course to track straight line path ( <a href="#">chapter 10</a> )
$\chi^\infty$	Desired approach angle for tracking a straight line path ( <a href="#">chapter 10</a> )
$\chi^o$	Course angle of the orbit path $\mathcal{P}_{\text{orbit}}$ ( <a href="#">chapter 10</a> )
$\chi_q$	Course angle of the straight line path $\mathcal{P}_{\text{line}}$ ( <a href="#">chapter 10</a> )
$d$	Distance between center of orbit and the MAV ( <a href="#">chapter 10</a> )
$d_\beta$	Disturbance signal associated with reduced <a href="#">sideslip</a> model ( <a href="#">chapter 5</a> )
$d_\chi$	Disturbance signal associated with reduced course model ( <a href="#">chapter 5</a> )
$d_h$	Disturbance signal associated with reduced altitude model ( <a href="#">chapter 5</a> )
$d_{\phi_*}$	Disturbance signals associated with reduced roll model ( <a href="#">chapter 5</a> )
$d_{\theta_*}$	Disturbance signals associated with reduced pitch model ( <a href="#">chapter 5</a> )
$d_{V_*}$	Disturbance signals associated with reduced airspeed model ( <a href="#">chapter 5</a> )
$\delta_a$	Control signal denoting the aileron deflection ( <a href="#">chapter 4</a> )
$\delta_e$	Control signal denoting the elevator deflection ( <a href="#">chapter 4</a> )
$\delta_r$	Control signal denoting the rudder deflection ( <a href="#">chapter 4</a> )
$\delta_t$	Control signal denoting the throttle deflection ( <a href="#">chapter 4</a> )
$\mathbf{e}_p$	Path error for straight line path following ( <a href="#">chapter 10</a> )
$\epsilon_s$	Pixel size ( <a href="#">chapter 13</a> )
$\epsilon_x$	Pixel location along the camera $x$ -axis ( <a href="#">chapter 13</a> )
$\epsilon_y$	Pixel location along the camera $y$ -axis ( <a href="#">chapter 13</a> )
$\eta_*$	Zero-mean Gaussian sensor noise ( <a href="#">chapter 7</a> )
$f$	Camera focal length ( <a href="#">chapter 13</a> )

$\mathbf{f}$	External force applied to the airframe with body-frame components denoted as $f_x$ , $f_y$ , and $f_z$ ( <a href="#">chapters 3, 4</a> )
$F$	the distance to pixel location $(\epsilon_x, \epsilon_y)$ in pixels as in $F = \sqrt{f^2 + \epsilon_x^2 + \epsilon_y^2}$ , ( <a href="#">chapter 13</a> )
$F_{\text{drag}}$	Force due to aerodynamic drag ( <a href="#">chapter 4, 9</a> )
$F_{\text{lift}}$	Force due to aerodynamic lift ( <a href="#">chapter 4, 9</a> )
$F_{\text{thrust}}$	Force due to thrust ( <a href="#">chapter 9</a> )
$\mathcal{F}^b$	Body coordinate frame ( <a href="#">chapter 2</a> )
$\mathcal{F}^i$	Inertial coordinate frame ( <a href="#">chapter 2</a> )
$\mathcal{F}^s$	Stability coordinate frame ( <a href="#">chapter 2</a> )
$\mathcal{F}^v$	Vehicle coordinate frame ( <a href="#">chapter 2</a> )
$\mathcal{F}^w$	Wind coordinate frame ( <a href="#">chapter 2</a> )
$\mathcal{F}^{v1}$	Vehicle-1 frame ( <a href="#">chapter 2</a> )
$\mathcal{F}^{v2}$	Vehicle-2 frame ( <a href="#">chapter 2</a> )
$g$	Gravitational acceleration ( $9.81 \text{ m/s}^2$ ) ( <a href="#">chapter 4</a> )
$\gamma$	Inertial-referenced flight path angle ( <a href="#">chapter 2</a> )
$\gamma_a$	Air-mass-referenced flight path angle: $\gamma_a = \theta - \alpha$ ( <a href="#">chapter 2</a> )
$\Gamma_*$	Products of the inertia matrix, as in equation (??) ( <a href="#">chapter 3</a> )
$h$	Altitude: $h = -p_d$ ( <a href="#">chapter 5</a> )
$h_{\text{AGL}}$	Altitude above ground level ( <a href="#">chapter 7</a> )
$\mathcal{H}(\mathbf{r}, \mathbf{n})$	Half plane defined at position $\mathbf{w}$ , with normal vector $\mathbf{n}$ ( <a href="#">chapter 11</a> )
$(\mathbf{i}^b, \mathbf{j}^b, \mathbf{k}^b)$	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 ( <a href="#">chapter 2</a> )
$(\mathbf{i}^i, \mathbf{j}^i, \mathbf{k}^i)$	Unit vectors defining the inertial frame. $\mathbf{i}^i$ points north, $\mathbf{j}^i$ points east, and $\mathbf{k}^i$ points down ( <a href="#">chapter 2</a> )
$(\mathbf{i}^v, \mathbf{j}^v, \mathbf{k}^v)$	Unit vectors defining the vehicle frame. $\mathbf{i}^v$ points north, $\mathbf{j}^v$ points east, and $\mathbf{k}^v$ points down ( <a href="#">chapter 2</a> )
$\mathbf{J}$	The inertia matrix with elements of the inertia matrix denoted as $J_x$ , $J_y$ , $J_z$ , and $J_{xz}$ ( <a href="#">chapter 3</a> )
$k_{d*}$	PID derivative gain ( <a href="#">chapter 6</a> )
$k_{\text{GPS}}$	Inverse of the time constant for GPS bias ( <a href="#">chapter 7</a> )
$k_{i*}$	PID integral gain ( <a href="#">chapter 6</a> )
$k_{\text{motor}}$	Constant that specifies the efficiency of the motor ( <a href="#">chapter 4</a> )
$k_{\text{orbit}}$	Control gain for tracking orbital path ( <a href="#">chapter 10</a> )
$k_{p*}$	PID proportional gain ( <a href="#">chapter 6</a> )
$k_{\text{path}}$	Control gain for tracking straight-line path ( <a href="#">chapter 10</a> )

$K_{\theta_{DC}}$	DC gain of the transfer function from the elevator to the pitch angle ( <a href="#">chapter 6</a> )
$\ell$	Line-of-sight vector from the MAV to a target location $\ell = (\ell_x, \ell_y, \ell_z)^\top$ ( <a href="#">chapter 13</a> )
$\check{\ell}$	Unit vector in the direction of the line of sight: $\check{\ell} = \ell/\mathbb{L}$ ( <a href="#">chapter 13</a> )
$L_*$	State-space coefficients associated with lateral dynamics ( <a href="#">chapter 5</a> )
$\mathbb{L}$	Length of the line of sight vector: $\mathbb{L} = \ \ell\ $ ( <a href="#">chapter 13</a> )
$LPF(x)$	Low-pass filtered version of $x$ ( <a href="#">chapter 8</a> )
$\lambda$	Direction of orbital path. $\lambda = +1$ specifies a clockwise orbit; $\lambda = -1$ specifies a counterclockwise orbit ( <a href="#">chapter 10</a> )
$\lambda_{\text{dutch roll}}$	Poles of the Dutch-roll mode ( <a href="#">chapter 5</a> )
$\lambda_{\text{phugoid}}$	Poles of the phugoid mode ( <a href="#">chapter 5</a> )
$\lambda_{\text{rolling}}$	Pole of the rolling mode ( <a href="#">chapter 5</a> )
$\lambda_{\text{short}}$	Pole of the short-period mode ( <a href="#">chapter 5</a> )
$\lambda_{\text{spiral}}$	Pole of the spiral mode ( <a href="#">chapter 5</a> )
$m$	Mass of the airframe ( <a href="#">chapter 3</a> )
$M_x$	External moment applied to the airframe about the body frame $x$ -axis ( <a href="#">chapter 3</a> )
$M_y$	External moment applied to the airframe about the body frame $y$ -axis ( <a href="#">chapter 3</a> )
$M_z$	External moment applied to the airframe about the body frame $z$ -axis ( <a href="#">chapter 3</a> )
$\mathbf{m}$	External moments applied to the airframe. The body frame components are denoted as $\ell$ , $m$ , and $n$ . ( <a href="#">chapters 3, 4</a> )
$M$	Width of the camera pixel array ( <a href="#">chapter 13</a> )
$M_*$	State-space coefficients associated with longitudinal dynamics ( <a href="#">chapter 5</a> )
$n_{lf}$	Load factor ( <a href="#">chapter 9</a> )
$N_*$	State-space coefficients associated with lateral dynamics ( <a href="#">chapter 5</a> )
$\nu_*$	Gauss-Markov process that models GPS bias ( <a href="#">chapter 7</a> )
$\omega_{n_*}$	Natural frequency ( <a href="#">chapter 6</a> )
$\omega_{b/i}$	Angular velocity of the body frame with respect to the inertial frame ( <a href="#">chapter 2</a> )
$p$	Roll rate of the MAV along the body frame $x$ -axis ( <a href="#">chapter 3</a> )

$p_d$	Inertial down position of the MAV ( <a href="#">chapter 3</a> )
$p_e$	Inertial east position of the MAV ( <a href="#">chapter 3</a> )
$\mathcal{P}_{\text{line}}$	Set defining a straight line ( <a href="#">chapter 10</a> )
$\mathbf{p}_{\text{MAV}}$	Position of the MAV ( <a href="#">chapter 13</a> )
$p_n$	Inertial north position of the MAV ( <a href="#">chapter 3</a> )
$\mathbf{p}_{\text{obj}}$	Position of the object of interest ( <a href="#">chapter 13</a> )
$P$	Covariance of the estimation error associated with the Kalman filter ( <a href="#">chapter 8</a> )
$\mathcal{P}_{\text{orbit}}$	Set defining an orbit ( <a href="#">chapter 10</a> )
$\phi$	Roll angle ( <a href="#">chapter 2, 3</a> )
$\varphi$	Angle of the MAV relative to a desired orbit ( <a href="#">chapter 10</a> )
$\psi$	Heading angle ( <a href="#">chapters 2, 3</a> )
$q$	Pitch rate of the MAV along the body frame $y$ -axis ( <a href="#">chapter 3</a> )
$Q_*$	Process covariance noise ( <a href="#">chapter 8</a> )
$r$	Yaw rate of the MAV along the body frame $z$ -axis ( <a href="#">chapter 3</a> )
$\rho$	Density of air ( <a href="#">chapter 4</a> )
$\varrho$	Angle between waypoint path segments ( <a href="#">chapter 11</a> )
$R$	Turning radius ( <a href="#">chapter 5</a> )
$R_*$	Covariance matrix for sensor measurement noise ( <a href="#">chapter 8</a> )
$\mathcal{R}_a^b$	Rotation matrix from frame $a$ to frame $b$ ( <a href="#">chapters 2, 13</a> )
$S$	Surface area of the wing ( <a href="#">chapter 4</a> )
$S_{\text{prop}}$	Area of the propeller ( <a href="#">chapter 4</a> )
$\sigma_*$	Standard deviation of zero-mean white Gaussian noise ( <a href="#">chapter 7</a> )
$t_c$	Time to collision: $t_c = \mathbb{L}/\dot{\mathbb{L}}$ ( <a href="#">chapter 13</a> )
$T_s$	Sample rate of the autopilot ( <a href="#">chapters 6, 7, 8</a> )
$\tau$	Bandwidth of dirty differentiator ( <a href="#">chapter 13</a> )
$\mathcal{T}$	Terrain map ( <a href="#">chapter 12</a> )
$\theta$	Pitch angle ( <a href="#">chapter 2, 3</a> )
$u$	Inertial velocity of the airframe projected onto $\mathbf{i}^b$ , the body frame $x$ -axis ( <a href="#">chapter 2, 3</a> )
$u_{\text{lat}}$	Input vector associated with lateral dynamics: $u_{\text{lat}} = (\delta_a, \delta_r)^\top$ ( <a href="#">chapter 5</a> )
$u_{\text{lon}}$	Input vector associated with longitudinal dynamics: $u_{\text{lon}} = (\delta_e, \delta_t)^\top$ ( <a href="#">chapter 5</a> )
$u_r$	Airspeed vector projected onto the body frame $x$ -axis: $u_r = u - u_w$ ( <a href="#">chapters 2, 4</a> )

$u_w$	Inertial wind velocity projected onto $\mathbf{i}^b$ , the body frame $x$ -axis ( <a href="#">chapters 2, 4</a> )
$v$	Camera field of view ( <a href="#">chapter 13</a> )
$\Upsilon$	Return map used for path planning. The return map at position $i$ is given by $\Upsilon_i$ ( <a href="#">chapter 12</a> )
$v$	Inertial velocity of the airframe projected onto $\mathbf{j}^b$ , the body frame $y$ -axis ( <a href="#">chapters 2, 3</a> )
$v_r$	Airspeed vector projected onto the body frame $y$ -axis: $v_r = v - v_w$ ( <a href="#">chapters 2, 4</a> )
$v_w$	Inertial wind velocity projected onto $\mathbf{j}^b$ , the body frame $y$ -axis ( <a href="#">chapters 2, 4</a> )
$\mathbf{V}_a$	Airspeed vector defined as the velocity of the airframe with respect to the air mass ( <a href="#">chapter 2</a> )
$V_a$	Airspeed where $V_a = \ \mathbf{V}_a\ $ ( <a href="#">chapter 2</a> )
$\mathbf{V}_g$	Ground speed vector defined as the velocity of the airframe with respect to the inertial frame ( <a href="#">chapter 2</a> )
$V_g$	Ground speed where $V_g = \ \mathbf{V}_g\ $ ( <a href="#">chapter 2</a> )
$\mathbf{V}_w$	Wind speed vector defined as the velocity of the wind with respect to the inertial frame ( <a href="#">chapter 2</a> )
$V_w$	Wind speed where $V_w = \ \mathbf{V}_w\ $ ( <a href="#">chapter 2</a> )
$w$	Inertial velocity of the airframe projected onto $\mathbf{k}^b$ , the body frame $z$ -axis ( <a href="#">chapters 2, 3</a> )
$w_d$	Component of the wind in the down direction ( <a href="#">chapter 2</a> )
$w_e$	Component of the wind in the east direction ( <a href="#">chapter 2</a> )
$w_n$	Component of the wind in the north direction ( <a href="#">chapter 2</a> )
$\mathbf{w}_i$	Waypoint in $\mathbb{R}^3$ ( <a href="#">chapter 11</a> )
$w_r$	Airspeed vector projected onto the body frame $z$ -axis: $w_r = w - w_w$ ( <a href="#">chapters 2, 4</a> )
$w_w$	Inertial wind velocity projected onto $\mathbf{k}^b$ , the body frame $z$ -axis ( <a href="#">chapters 2, 4</a> )
$W_*$	Bandwidth separation ( <a href="#">chapter 6</a> )
$\mathcal{W}$	Set of waypoints ( <a href="#">chapter 11</a> )
$x$	State variables ( <a href="#">chapter 5</a> )
$x_{\text{lat}}$	State variables associated with lateral dynamics: $x_{\text{lat}} = (v, p, r, \phi, \psi)^\top$ ( <a href="#">chapter 5</a> )
$x_{\text{lon}}$	State variables associated with longitudinal dynamics: $x_{\text{lon}} = (u, w, q, \theta, h)^\top$ ( <a href="#">chapter 5</a> )
$X_*$	State-space coefficients associated with longitudinal dynamics ( <a href="#">chapter 5</a> )
$y_{\text{abs pres}}$	Absolute pressure measurement signal ( <a href="#">chapter 7</a> )
$y_{\text{accel},*}$	Accelerometer measurement signal ( <a href="#">chapter 7</a> )

$y_{\text{diff pres}}$	Differential pressure measurement signal ( <a href="#">chapter 7</a> )
$y_{\text{GPS,*}}$	GPS measurement signal. GPS measurements are available for north, east, altitude, course, and groundspeed ( <a href="#">chapter 7</a> )
$y_{\text{gyro,*}}$	Rate gyro measurement signal ( <a href="#">chapter 7</a> )
$y_{\text{mag}}$	Magnetometer measurement signal ( <a href="#">chapter 7</a> )
$Y_*$	State-space coefficients associated with lateral dynamics ( <a href="#">chapter 5</a> )
$Z_*$	State-space coefficients associated with longitudinal dynamics ( <a href="#">chapter 5</a> )
$Z(\epsilon)$	Transformation from pixel motion to motion of the line of sight vector in the camera frame ( <a href="#">chapter 13</a> )
$\zeta_*$	Damping coefficient ( <a href="#">chapter 6</a> )

## 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 worked 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  $\mathbf{z}$  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 matrix on 2-dimensional vectors.

## 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

$$\mathbf{qe} = M(\mathbf{q})\mathbf{e},$$

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

Just as complex numbers on the unit circle represent rotations in 2D, similarly, unit quaternions, i.e., quaternions where  $\|\mathbf{e}\| = 1$  can be used to represent rotations in 3D. As such, 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 rotational 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  $\theta$  is  $\pm 90$  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 acrobatic 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 equations. A thorough introduction to quaternions and rotation sequences is given by Kuipers [148]. An in-depth treatment of the use of quaternions for aircraft applications is given by Phillips [24].

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  $\mathbb{R}^4$  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,  $\|e\| = \sqrt{e_0^2 + e_x^2 + e_y^2 + e_z^2} = 1$ .

It is common to refer  $e_0$  as the scalar part of the unit quaternion and the vector defined by

$$\mathbf{e} = (e_x, e_y, e_z)^\top$$

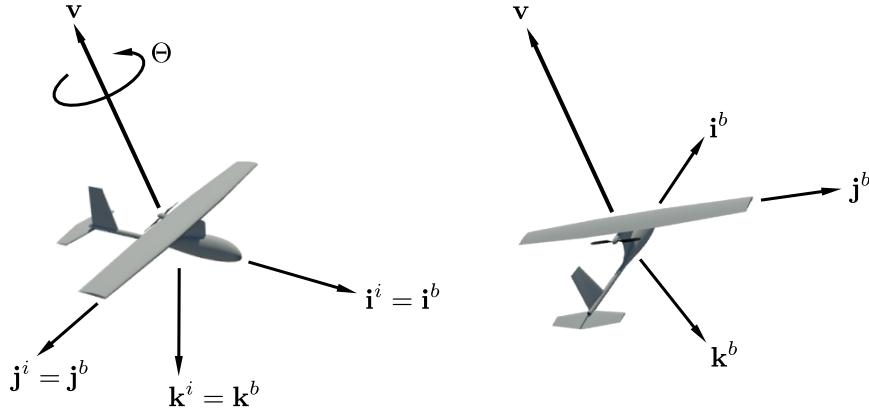
as the vector part. A unit quaternion can be interpreted as a single rotation about an axis in three-dimensional space. For a rotation of angle  $\Theta$  about the axis specified by the unit vector  $\mathbf{v}$ , the scalar part of the unit quaternion is related to the magnitude of the rotation by

$$e_0 = \cos\left(\frac{\Theta}{2}\right).$$

The vector part of the unit quaternion is related to the axis of rotation by

$$\begin{pmatrix} e_x \\ e_y \\ e_z \end{pmatrix} = \mathbf{v} \sin\left(\frac{\Theta}{2}\right).$$

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  $v$  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

$$\begin{aligned} \phi &= \text{atan2}\left(2(e_0 e_x + e_y e_z), (e_0^2 + e_z^2 - e_x^2 - e_y^2)\right) \\ \theta &= \text{asin}\left(2(e_0 e_y - e_x e_z)\right) \\ \psi &= \text{atan2}\left(2(e_0 e_z + e_x e_y), (e_0^2 + e_x^2 - e_y^2 - e_z^2)\right), \end{aligned} \quad (\text{B.1})$$

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  $\text{asin}$  operator since the pitch angle is only defined in the range  $[\pi/2, \pi/2]$ .

Alternatively, given the 3-2-1 Euler angles  $(\psi, \phi, \theta)$ , 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

$$R(\mathbf{e}_b^i) = \begin{pmatrix} e_0^2 + e_x^2 - e_y^2 - e_z^2 & 2(e_x e_y - e_0 e_z) & 2(e_x e_z + e_0 e_y) \\ 2(e_x e_y + e_0 e_z) & e_0^2 - e_x^2 + e_y^2 - e_z^2 & 2(e_y e_z - e_0 e_x) \\ 2(e_x e_z - e_0 e_y) & 2(e_y e_z + e_0 e_x) & e_0^2 - e_x^2 - e_y^2 + e_z^2 \end{pmatrix}.$$

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 [149]:

$$\begin{aligned} e_0 &= \begin{cases} \frac{1}{2}\sqrt{1+r_{11}+r_{22}+r_{33}}, & \text{if } r_{11}+r_{22}+r_{33} > 0 \\ \frac{1}{2}\sqrt{\frac{(r_{12}-r_{21})^2+(r_{13}-r_{31})^2+(r_{23}-r_{32})^2}{3-r_{11}-r_{22}-r_{33}}}, & \text{otherwise} \end{cases} \\ e_x &= \begin{cases} \frac{1}{2}\sqrt{1+r_{11}-r_{22}-r_{33}}, & \text{if } r_{11}-r_{22}-r_{33} > 0 \\ \frac{1}{2}\sqrt{\frac{(r_{12}+r_{21})^2+(r_{13}+r_{31})^2+(r_{23}-r_{32})^2}{3-r_{11}+r_{22}+r_{33}}}, & \text{otherwise} \end{cases} \\ e_y &= \begin{cases} \frac{1}{2}\sqrt{1-r_{11}+r_{22}-r_{33}}, & \text{if } -r_{11}+r_{22}-r_{33} > 0 \\ \frac{1}{2}\sqrt{\frac{(r_{12}+r_{21})^2+(r_{13}-r_{31})^2+(r_{23}+r_{32})^2}{3+r_{11}-r_{22}+r_{33}}}, & \text{otherwise} \end{cases} \\ e_z &= \begin{cases} \frac{1}{2}\sqrt{1-r_{11}-r_{22}+r_{33}}, & \text{if } -r_{11}-r_{22}+r_{33} > 0 \\ \frac{1}{2}\sqrt{\frac{(r_{12}-r_{21})^2+(r_{13}+r_{31})^2+(r_{23}+r_{32})^2}{3+r_{11}+r_{22}-r_{33}}}, & \text{otherwise} \end{cases}. \end{aligned}$$

## B.6 QUATERNION KINEMATICS

### MODIFIED MATERIAL:

Suppose that  $\mathbf{q}_\Delta$  represents a small rotation of angle  $\Delta\Theta$ , i.e.,

$$\mathbf{q}_\Delta = \begin{pmatrix} \cos(\Delta\Theta/2) \\ \sin(\Delta\Theta/2)u_x \\ \sin(\Delta\Theta/2)u_y \\ \sin(\Delta\Theta/2)u_z \end{pmatrix} \approx \begin{pmatrix} 1 \\ \frac{\omega_x \Delta t}{2} \\ \frac{\omega_y \Delta t}{2} \\ \frac{\omega_z \Delta t}{2} \end{pmatrix},$$

where  $(\omega_x, \omega_y, \omega_z)^\top$  is the instantaneous angular velocity, and  $\Delta t$  is a small time step. 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

$$\Omega(\boldsymbol{\omega}) = \begin{pmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{pmatrix}$$

If  $\mathbf{e}_b^i$  is the unit quaternion representing the rotation from the body frame to the inertial frame, and  $\boldsymbol{\omega}_{b/i}^b = (p, q, r)^\top$  is the angular velocity vector, then

$$\dot{\mathbf{e}}_b^i = \frac{1}{2} \Omega(\boldsymbol{\omega}_{b/i}^b) \mathbf{e}_b^i.$$

## B.7 AIRCRAFT KINEMATIC AND DYNAMIC EQUATIONS

Using a unit quaternion to represent the aircraft attitude, equations (3.13) through (3.16), which describe the MAV kinematics and dynamics, can be

reformulated as

$$\begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{p}_d \end{pmatrix} = R(\mathbf{e}_b^i) \begin{pmatrix} u \\ v \\ w \end{pmatrix} \quad (\text{B.3})$$

$$\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} = \begin{pmatrix} rv - qw \\ pw - ru \\ qu - pv \end{pmatrix} + \frac{1}{m} \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix}, \quad (\text{B.4})$$

$$\begin{pmatrix} \dot{e}_0 \\ \dot{e}_x \\ \dot{e}_y \\ \dot{e}_z \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{pmatrix} \begin{pmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{pmatrix} \quad (\text{B.5})$$

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} \Gamma_1 pq - \Gamma_2 qr \\ \Gamma_5 pr - \Gamma_6(p^2 - r^2) \\ \Gamma_7 pq - \Gamma_1 qr \end{pmatrix} + \begin{pmatrix} \Gamma_3 l + \Gamma_4 n \\ \frac{1}{J_y} m \\ \Gamma_4 l + \Gamma_8 n \end{pmatrix}. \quad (\text{B.6})$$

Note that the dynamic equations given by [equations \(B.4\)](#) and [\(B.6\)](#) are unchanged from [equations \(3.14\)](#) and [\(3.16\)](#) presented in the summary of [chapter 3](#). However, care must be taken when propagating [equation \(B.5\)](#) to ensure that  $\mathbf{e}$  remains a unit quaternion. If the dynamics are implemented using a Simulink s-function, then one possibility for maintaining  $\|\mathbf{e}\| = 1$  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  $J = \frac{1}{8}(1 - \|\mathbf{e}\|^2)^2$ . Since  $J$  is quadratic, we can use gradient descent to minimize  $J$ , and [equation \(B.5\)](#) becomes

$$\begin{pmatrix} \dot{e}_0 \\ \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{pmatrix} \begin{pmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{pmatrix} - \lambda \frac{\partial J}{\partial \mathbf{e}} \\ = \frac{1}{2} \begin{pmatrix} \lambda(1 - \|\mathbf{e}\|^2) & -p & -q & -r \\ p & \lambda(1 - \|\mathbf{e}\|^2) & r & -q \\ q & -r & \lambda(1 - \|\mathbf{e}\|^2) & p \\ r & q & -p & \lambda(1 - \|\mathbf{e}\|^2) \end{pmatrix} \begin{pmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{pmatrix},$$

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 [24, 150].

**NEW MATERIAL:** If the dynamics are propagated using the RK4 algo-

rithm in Python, then  $\mathbf{e}$  is normalized using

$$\mathbf{e} \leftarrow \frac{\mathbf{e}}{\|\mathbf{e}\|}$$

after every RK4 step.

RWB: Add this reference [151] about non-unit length quaternions. This is a very interesting paper that shows how to use non-unit length quaternions for attitude representation. The advantage of this representation is that there is no need to renormalize the quaternion after each step. A method to contain the magnitude of the unit quaternion is to add the additional dynamics of  $\lambda(1 - \mathbf{q}^\top \mathbf{q})$  to keep the length close to one, but there is no need to have a large  $\lambda$ , which makes the dynamics stiff. The paper shows that non-unit length quaternion, and the renormalization after each time step, results in essentially identical behavior. to use this method in the simulation, we would need to change quat2rot, and other function.

## Appendix C

---

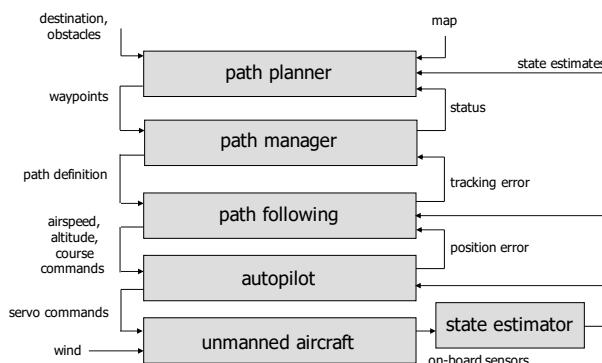
### Simulation Using Object Oriented Programming

RWB: Turn this appendix into a general introduction to digital implementation. Include implementation of transfer functions and PID loops. Needs to be totally re-written.

#### 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

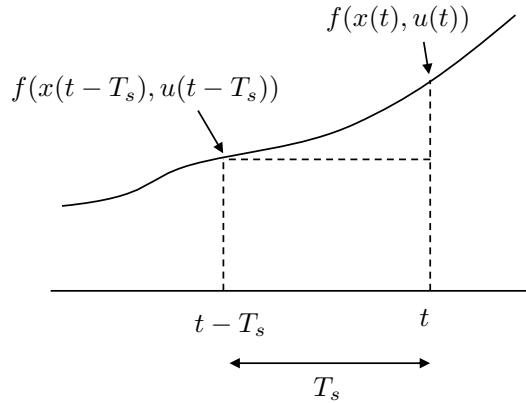
$$\int_{t-T_s}^t f(x(\tau), u(\tau)) d\tau \approx T_s f(x(t - T_s), u(t - T_s)).$$

Using the notation  $x_k \triangleq x(t_0 + kT_s)$  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  $f(\cdot, \cdot)$ .

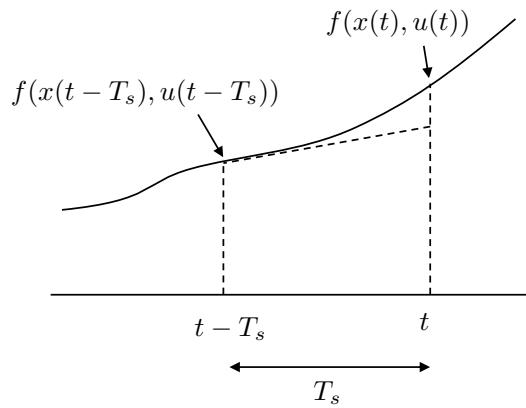


**Figure C.2:** Approximation of integral using the left endpoint method results in Runge-Kutta first order method (RK1).

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  $[t - T_s, t]$  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

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

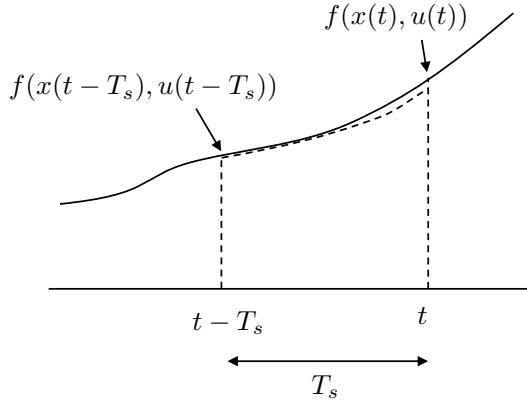
and then to use

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

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

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

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



**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')
plt.show()

```

### C.3 NUMERICAL IMPLEMENTATION OF TRANSFER FUNCTIONS

RWB: This was moved from Chapter 4 with Tim's comment. Maybe make this more general, and then use pseudo-code instead of Python. How do you do classes in pseudo-code?

TWM: I'm not sure that this is the right place to put implementation details. Plus, these details are pretty specific to Python. It seems like this would be more appropriate for an appendix or end of chapter implementation details. We need to be consistent throughout.

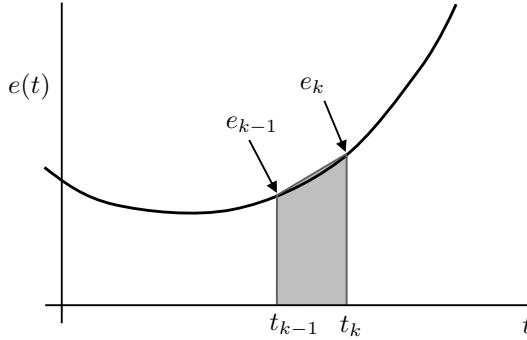
In this book, we have often utilized transfer functions to implement filters and controllers. In practice, we have designed these transfer functions in the continuous domain with the goal of implementing them on microcontrollers and computers in the discrete domain. In this section, we describe a general and widely used approach for discrete-time implementation of continuous-time transfer functions. The process first involves converting a continuous-domain transfer function  $G(s)$  to its equivalent transfer function in the discrete domain,  $G(z)$ . We then use the inverse  $z$ -transform to convert  $G(z)$  to a discrete-time difference equation that can be implemented in code on a computer. Before we proceed, we need to define the  $z$ -transform. For a signal  $e(k)$  having discrete values  $e_0, e_1, \dots, e_k, \dots$  the  $z$ -transform of the signal is defined as

$$\begin{aligned} E(z) &\triangleq \mathcal{Z}\{e(k)\} \\ &\triangleq \sum_{k=-\infty}^{\infty} e_k z^{-k} \end{aligned}$$

where  $z$  is a complex number.

To convert a continuous-domain transfer function to its discrete equivalent, we will develop an approximation for the continuous-time Laplace transform variable  $s$  expressed in terms of the discrete-time transform variable  $z$ . To do so, we will consider the discrete approximation to integration.

For a continuous signal,  $e(t)$ , as shown in figure C.5, we want to compute



**Figure C.5:** Discrete approximation of area under the curve using the trapezoidal rule.

an approximation to the integral

$$u(t) = \int_0^t e(t) dt$$

over the interval from time zero to  $t_k$  using samples of  $e(t)$  taken at times  $t = 0, t_1, t_2, \dots, t_k$ . Defining the approximation of the integral from zero to  $t_{k-1}$  as  $u_{k-1}$ , we can calculate  $u_k$  by approximating the area under the curve of  $e(t)$  from  $t_{k-1}$  to  $t_k$  and adding it to  $u_{k-1}$ . Approximating the area under the curve as a trapezoid, we can calculate  $u_k$  to be

$$u_k = u_{k-1} + \frac{T_s}{2} (e_{k-1} + e_k) \quad (\text{C.11})$$

where  $T_s = t_k - t_{k-1}$  is the sample period. Taking the  $z$ -transform and recalling that  $z^{-1}$  can be viewed as a one time-period delay we get

$$U(z) = z^{-1}U(z) + \frac{T_s}{2} (z^{-1}E(z) + E(z)). \quad (\text{C.12})$$

We can further manipulate this expression to define the transfer function for trapezoidal-rule integration as

$$G(z) = \frac{U(z)}{E(z)} \triangleq \frac{T_s}{2} \left( \frac{1+z^{-1}}{1-z^{-1}} \right). \quad (\text{C.13})$$

We can express the transfer function for integration in the continuous-time domain as

$$G(s) = \frac{U(s)}{E(s)} \triangleq \frac{1}{s}. \quad (\text{C.14})$$

By equating the transfer functions (C.13) and (C.14), we can formulate an approximate mapping between the Laplace transform variable  $s$  and the  $z$ -transform variable  $z$ :

$$\frac{1}{s} \mapsto \frac{T_s}{2} \left( \frac{1 - z^{-1}}{1 + z^{-1}} \right),$$

and thus

$$s \mapsto \frac{2}{T_s} \left( \frac{1 - z^{-1}}{1 + z^{-1}} \right). \quad (\text{C.15})$$

#### *Yaw Damper Implementation Example*

The compensation transfer function for the yaw damper of chapter 6 has the form

$$C(s) = \frac{\text{Delta}(s)}{R(s)} = \frac{as}{s + b}, \quad (\text{C.16})$$

where  $R(s)$  is the yaw rate of the aircraft and  $R_f(s)$  is the washout-filtered yaw rate. We can obtain the discrete equivalent transfer function  $C(z)$  by applying the mapping of equation (C.15) to give

$$\begin{aligned} C(z) &= \frac{as}{s + b} \Big|_{s=\frac{2}{T_s} \left( \frac{1-z^{-1}}{1+z^{-1}} \right)} \\ &= \frac{a \frac{2}{T_s} (1 - z^{-1})}{\frac{2}{T_s} (1 - z^{-1}) + b (1 + z^{-1})} \\ C(z) &= \frac{\frac{2a}{2+bT_s} - \left( \frac{2a}{2+bT_s} \right) z^{-1}}{1 - \left( \frac{2-bT_s}{2+bT_s} \right) z^{-1}} \end{aligned} \quad (\text{C.17})$$

or

$$C(z) = \frac{\left( \frac{2a}{2+bT_s} \right) z - \frac{2a}{2+bT_s}}{z - \frac{2-bT_s}{2+bT_s}} \quad (\text{C.18})$$

Notice that the discrete transfer function  $C(z)$  is a ratio of two polynomials in  $z$ . For reasons that will be apparent, it is customary to factor the transfer function so that the coefficient of the highest degree term of the denominator polynomial is 1.

With the discrete transfer function defined, we can apply the inverse  $z$  transform to obtain a discrete-time difference equation that can be used to

implement the yaw damper washout filter on the onboard computer. From equation C.17 and recalling that  $C(z) = \Delta(z)/R(z)$ , we can write

$$\frac{2a}{2+bT_s}R(z) - \left(\frac{2a}{2+bT_s}\right)z^{-1}R(z) = \Delta(z) - \left(\frac{2-bT_s}{2+bT_s}\right)z^{-1}\Delta(z).$$

Taking the inverse  $z$  transform, recognizing  $z^{-1}$  as a one time-step delay, gives

$$\frac{2a}{2+bT_s}r_k - \left(\frac{2a}{2+bT_s}\right)r_{k-1} = \delta_k - \left(\frac{2-bT_s}{2+bT_s}\right)\delta_{k-1}.$$

Solving for the filter output at the current time step results in

$$\delta_k = \frac{2a}{2+bT_s}r_k + \left(\frac{2-bT_s}{2+bT_s}\right)\delta_{k-1} - \left(\frac{2a}{2+bT_s}\right)r_{k-1}$$

**RWB: More moved stuff. Need to rewrite and fix this section / appendix.**

**RWB: Move implementation details to appendix about transfer functions**

#### *Yaw Damper Implementation*

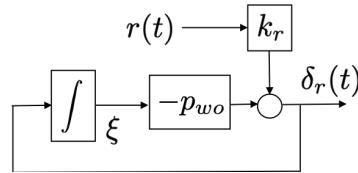
Converting to the time domain results in the differential equation

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

Integrating once gives

$$\delta_r(t) = -p_{wo} \int_{-\infty}^t \delta_r(\sigma) d\sigma + k_r r(t). \quad (\text{C.19})$$

A block diagram that corresponds to equation (C.19) is shown in figure C.6.



**Figure C.6:** Block diagram for yaw damper transfer function.

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

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

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

TWM: I think there is an error in the equations below. There needs to be a  $T_s$  value in there somewhere, as in the Python code. Also, I think it might be better to use the transfer function  $\delta_r(s)/r(s)$  and trapezoidal rule approximation to come up with the corresponding discrete-time difference equation. This is a more accurate representation of the frequency response of the transfer function than given by the Euler approximation. It's also an approach we could use with any of our implementations (e.g., PID, etc.) that require implementation of a transfer function compensator – it's general.)

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

The corresponding Python code is shown below.

```
class yawDamper:
    def __init__(self, k_r, p_wo, Ts):
        self.xi = 0.
        self.Ts = Ts
        self.k_r = k_r
        self.p_wo = p_wo

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

TWM: Not sure mid-chapter is best place for this material. I am more in favor of pseudo code than a Python-specific implementation. Appendix or uavbook website seems more appropriate.

### C.3.1 Digital Implementation of PID Loops

NEW MATERIAL:

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

```
import numpy as np

class pid_control:
    def __init__(self, kp=0.0, ki=0.0, kd=0.0, Ts=0.01,
                 sigma=0.05, limit=1.0):
        self.kp = kp
        self.ki = ki
```

```

        self.kd = kd
        self.Ts = Ts
        self.limit = limit
        self.integrator = 0.0
        self.error_delay_1 = 0.0
        self.error_dot_delay_1 = 0.0
# gains for differentiator
        self.a1 = (2.0 * sigma - Ts) / (2.0 * sigma + Ts)
        self.a2 = 2.0 / (2.0 * sigma + Ts)

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

def update_with_rate(self, y_ref, y, ydot,
                      reset_flag=False):
    if reset_flag == True:
        self.integrator = 0.0
        self.error_delay_1 = 0.0
# compute the error
    error = y_ref - y
# update the integrator using trapazoidal rule

```

```
    self.integrator = self.integrator \
        + (self.Ts/2) * (error + self.error_delay_1)
    # PID control
    u = self.kp * error \
        + self.ki * self.integrator \
        - self.kd * ydot
    # saturate PID control at limit
    u_sat = self._saturate(u)
    # integral anti-windup
    # adjust integrator to keep u out of saturation
    if np.abs(self.ki) > 0.0001:
        self.integrator = self.integrator \
            + (1.0 / self.ki) * (u_sat - u)
    self.error_delay_1 = error
    return u_sat

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

STOP NEW MATERIAL:

#### C.4 NUMERICAL IMPLEMENTATION OF PID LOOPS

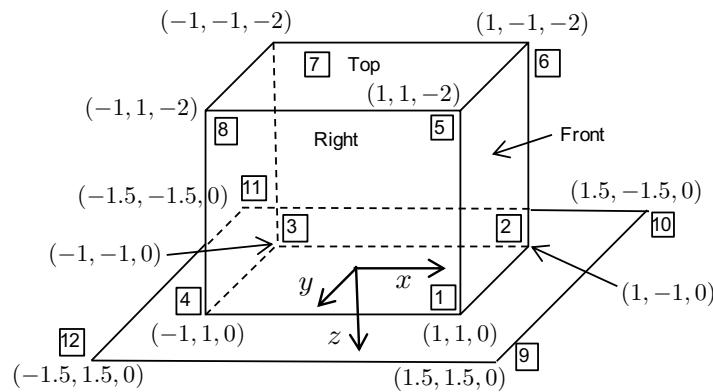


## Appendix D

---

### 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. As an example, we will animate the simple vertex-face spacecraft model shown in figure D.4.



**Figure D.1:** Simple spacecraft model defining vertices and faces.

This definition of the vertices and faces will be used render the spacecraft in Python, MATLAB, and Simulink as discussed in the following sections.

### D.1 3D ANIMATION USING PYTHON

See <http://uavbook.byu.edu> for sample Python code. When you download the project templates, the spacecraft simulation software is in the Appendix C (appC) folder.

## D.2 3D ANIMATION USING MATLAB

## D.3 3D ANIMATION USING SIMULINK

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

## D.4 HANDLE GRAPHICS IN MATLAB

When a graphics function like `plot` is called in Matlab, the function returns a *handle* to the plot. A graphics handle is similar to a pointer in C/C++ in the sense that all of the properties of the plot can be accessed through the handle. For example, the Matlab command

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

returns a pointer, or handle, to the plot of  $\sin(t)$ . Properties of the plot can be changed by using the handle, rather than reissuing the `plot` command. For example, the Matlab command

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

changes the plot to  $\cos(t)$  without redrawing the axes, title, label, or other objects that may be associated with the plot. If the plot contains drawings of several objects, a handle can be associated with each object. For example,

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

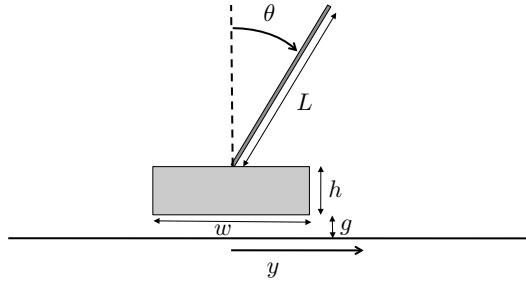
draws both  $\sin(t)$  and  $\cos(t)$  on the same plot, with a handle associated with each object. The objects can be manipulated separately without redrawing the other object. For example, to change  $\cos(t)$  to  $\cos(2t)$ , issue the command

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

We can exploit this property to animate simulations in Simulink by redrawing only the parts of the animation that change in time, and thereby significantly reducing the simulation time. To show how handle graphics can be used to produce animations in Simulink, we will provide three detailed examples. In [section D.5](#) we illustrate a 2-D animation of an inverted pendulum using the `fill` command. In [section D.6](#) we illustrate a 3-D animation of a spacecraft using lines to produce a stick figure. In [section D.7](#) we modify the spacecraft animation to use the vertices-faces data construction in Matlab.

## D.5 ANIMATION EXAMPLE: INVERTED PENDULUM

Consider the image of the inverted pendulum shown in [figure D.2](#), where the configuration is completely specified by the position of the cart  $y$ , and the angle of the rod from vertical  $\theta$ . The physical parameters of the system are the rod length  $L$ , the base width  $w$ , the base height  $h$ , and the gap between the base and the track  $g$ . The first step in developing the animation is to



**Figure D.2:** Drawing for inverted pendulum. The first step in developing an animation is to draw a figure of the object to be animated and identify all of the physical parameters.

determine the position of points that define the animation. For example, for the inverted pendulum in [figure D.2](#), the four corners of the base are

$$(y + w/2, g), (y + w/2, g + h), (y - w/2, g + h), \text{ and } (y - w/2, g),$$

and the two ends of the rod are given by

$$(y, g + h) \text{ and } (y + L \sin \theta, g + h + L \cos \theta).$$

Since the base and the rod can move independently, each will need its own figure handle. The `drawBase` command can be implemented with the following Matlab code.

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

Lines 2 and 3 define the X and Y locations of the corners of the base. Note that in line 1, `handle` is both an input and an output. If an empty array is passed into the function, then the `fill` command is used to plot the base in

line 5. On the other hand, if a valid handle is passed into the function, then the base is redrawn using the `set` command in line 7.

The Matlab code for drawing the rod is similar and is listed below.

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

The input `mode` is used to specify the `EraseMode` in Matlab. The `EraseMode` can be set to `normal`, `none`, `xor`, or `background`. A description of these different modes can be found by looking under `Image Properties` in the Matlab Helpdesk.

TWM: `EraseMode` is no longer supported in Matlab. This reference to `EraseMode` needs to be modified/deleted and updated code needs to be provided. The main routine for the pendulum animation is listed below.

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

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

    % define persistent variables
    persistent base_handle
    persistent rod_handle

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

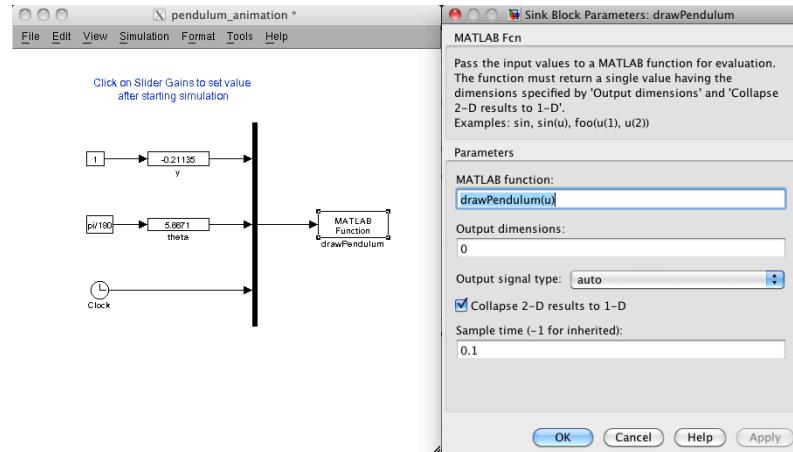
    % at every other time step, redraw base and rod
```

```

else
    drawBase(y, width, height, gap, base_handle);
    drawRod(y, theta, L, gap, height, rod_handle);
end

```

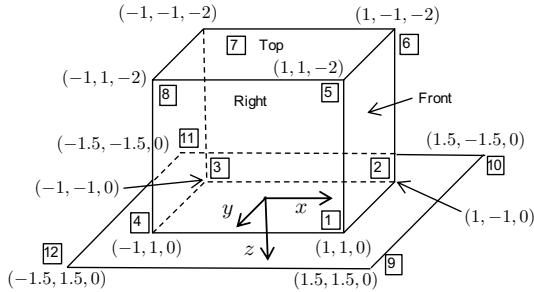
The routine `drawPendulum` is called from the Simulink file shown in [figure D.3](#), where there are three inputs: the position  $y$ , the angle  $\theta$ , and the time  $t$ . Lines 3–5 rename the inputs to  $y$ ,  $\theta$ , and  $t$ . Lines 8–11 define the drawing parameters. We require that the handle graphics persist between function calls to `drawPendulum`. Since a handle is needed for both the base and the rod, we define two persistent variables in lines 14 and 15. The `if` statement in lines 18–31 is used to produce the animation. Lines 19–25 are called once at the beginning of the simulation, and draw the initial animation. Line 19 brings the figure 1 window to the front and clears it. Lines 20 and 21 draw the ground along which the pendulum will move. Line 23 calls the `drawBase` routine with an empty handle as input, and returns the handle `base_handle` to the base. The `EraseMode` is set to `normal`. Line 24 calls the `drawRod` routine, and line 25 sets the axes of the figure. After the initial time step, all that needs to be changed are the locations of the base and rod. Therefore, in lines 29 and 30, the `drawBase` and `drawRod` routines are called with the figure handles as inputs.



**Figure D.3:** Simulink file for debugging the pendulum simulation. There are three inputs to the Matlab m-file `drawPendulum`: the position  $y$ , the angle  $\theta$ , and the time  $t$ . Slider gains for  $y$  and  $\theta$  are used to verify the animation.

## D.6 ANIMATION EXAMPLE: SPACECRAFT USING LINES

The previous section described a simple 2-D animation. In this section we discuss a 3-D animation of a spacecraft with six degrees of freedom. Figure D.4 shows a simple line drawing of a spacecraft, where the bottom is meant to denote a solar panel that should be oriented toward the sun.



**Figure D.4:** Drawing used to create spacecraft animation. Standard aeronautics body axes are used, where the  $x$ -axis points out the front of the spacecraft, the  $y$ -axis points to the right, and the  $z$ -axis point out the bottom of the body.

The first step in the animation process is to label the points on the spacecraft and to determine their coordinates in a body-fixed coordinate system. We will use standard aeronautics axes with  $X$  pointing out the front of the spacecraft,  $Y$  pointing to the right, and  $Z$  pointing out the bottom. The points 1 through 12 are labeled in figure D.4 and specify how coordinates are assigned to each label. To create a line drawing, we need to connect the points in a way that draws each of the desired line segments. To do this as one continuous line, some of the segments will need to be repeated. To draw the spacecraft shown in figure D.4, we will transition through the following nodes: 1 – 2 – 3 – 4 – 1 – 5 – 6 – 2 – 6 – 7 – 3 – 7 – 8 – 4 – 8 – 5 – 1 – 9 – 10 – 2 – 10 – 11 – 3 – 11 – 12 – 4 – 12 – 9. Matlab code that defines the local coordinates of the spacecraft is given below.

```
function XYZ=spacecraftPoints
% define points on the spacecraft in local NED coordinates
XYZ = [ ...
    1     1     0;... % point 1
    1    -1     0;... % point 2
   -1    -1     0;... % point 3
   -1     1     0;... % point 4
    1     1     0;... % point 1
    1     1    -2;... % point 5
    1    -1    -2;... % point 6
    1    -1     0;... % point 2
    1    -1    -2;... % point 6
```

```

-1    -1    -2;... % point 7
-1    -1    0;... % point 3
-1    -1    -2;... % point 7
-1    1    -2;... % point 8
-1    1    0;... % point 4
-1    1    -2;... % point 8
1     1    -2;... % point 5
1     1    0;... % point 1
1.5   1.5   0;... % point 9
1.5   -1.5  0;... % point 10
1     -1   0;... % point 2
1.5   -1.5  0;... % point 10
-1.5  -1.5  0;... % point 11
-1     -1   0;... % point 3
-1.5  -1.5  0;... % point 11
-1.5  1.5   0;... % point 12
-1     1   0;... % point 4
-1.5  1.5   0;... % point 12
1.5   1.5   0;... % point 9
];

```

The configuration of the spacecraft is given by the Euler angles  $\phi$ ,  $\theta$ , and  $\psi$ , which represent the roll, pitch, and yaw angles, respectively, and  $p_n$ ,  $p_e$ ,  $p_d$ , which represent the north, east, and down positions, respectively. The points on the spacecraft can be rotated and translated using the Matlab code listed below.

```

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

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

```

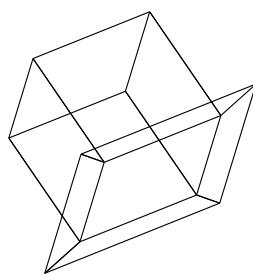
Drawing the spacecraft at the desired location is accomplished using the following Matlab code:

```

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

```

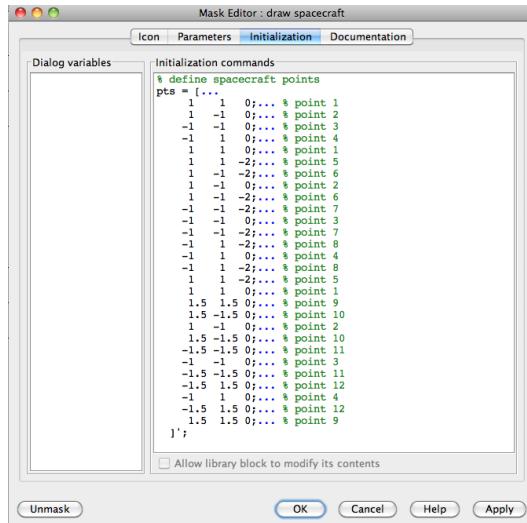
Lines 9–14 are used to transform the coordinates from the north-east-down (NED) coordinate frame to the drawing frame used by Matlab which has the  $x$ -axis to the viewer's right, the  $y$ -axis into the screen, and the  $z$ -axis up. The **plot3** command is used in Line 17 to render the original drawing, and the **set** command is used to change the XData, YData, and ZData in Line 19. A Simulink file that can be used to debug the animation is on the book website. A rendering of the spacecraft is shown in figure D.5.



**Figure D.5:** Rendering of the spacecraft using lines and the **plot3** command.

The disadvantage of implementing the animation using the function **spacecraftPoints** to define the spacecraft points is that this function is called each time the animation is updated. Since the points are static, they only need to be defined once. The Simulink mask function can be used to define the points

at the beginning of the simulation. Masking the drawSpacecraft m-file in Simulink, and then clicking on Edit Mask brings up a window like the one shown in figure D.6. The spacecraft points can be defined in the initialization window, as shown in figure D.6, and passed to the drawSpacecraft m-file as a parameter.



**Figure D.6:** The mask function in Simulink allows the spacecraft points to be initialized at the beginning of the simulation.

## D.7 ANIMATION EXAMPLE: SPACECRAFT USING VERTICES AND FACES■

The stick-figure drawing shown in figure D.5 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     1.5    0;... % 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
    1     1.5    0;... % point 13
    1.5  1.5    0;... % point 14
    1.5  1.5    0;... % point 15
    1.5  1.5    0;... % point 16
];

```

```

-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 D.4](#) 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*

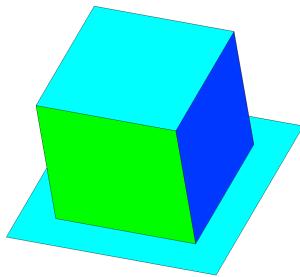
```

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.7](#). Additional



**Figure D.7:** 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

This appendix provides model parameters for the Aerosonde aircraft shown in figure E.1. Many of the physical and aerodynamic parameters specified below are reported in [152]. The original Aerosonde was created for scientific missions and gained notoriety for becoming the first UAV to fly across the Atlantic in 1998.



**Figure E.1:** The Aerosonde aircraft in flight.

The Aerosonde is powered by a small four-stroke engine. Our propulsion model is based on a DC motor and we have selected a motor-propeller combination that is suitable to power the Aerosonde. The motor is an AXI Motors model 5345/20 [153] and the propeller is an APC Propellers model 20x10E [154]. The propeller torque and thrust coefficients were obtained from performance data posted on the APC website using the methods described in chapter 4.

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

<b>Physical Param</b>	<b>Value</b>	<b>Motor Param</b>	<b>Value</b>
$m$	11.0 kg	$V_{\max}$	44.4 V
$J_x$	0.824 kg-m <sup>2</sup>	$D_{\text{prop}}$	0.508 m
$J_y$	1.135 kg-m <sup>2</sup>	$K_V$	0.0659 V-s/rad
$J_z$	1.759 kg-m <sup>2</sup>	$K_Q$	0.0659 N-m
$J_{xz}$	0.120 kg-m <sup>2</sup>	$R_{\text{motor}}$	0.042 Ω
$S$	0.55 m <sup>2</sup>	$\dot{i}_0$	1.5 A
$b$	2.9 m	$C_{Q_2}$	-0.01664
$c$	0.19 m	$C_{Q_1}$	0.004970
$\rho$	1.268 kg/m <sup>3</sup>	$C_{Q_0}$	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.

<b>Longitudinal Coef.</b>	<b>Value</b>	<b>Lateral Coef.</b>	<b>Value</b>
$C_{L_0}$	0.23	$C_{Y_0}$	0
$C_{D_0}$	0.043	$C_{l_0}$	0
$C_{m_0}$	0.0135	$C_{n_0}$	0
$C_{L_\alpha}$	5.61	$C_{Y_\beta}$	-0.83
$C_{D_\alpha}$	0.030	$C_{l_\beta}$	-0.13
$C_{m_\alpha}$	-2.74	$C_{n_\beta}$	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
$\alpha_0$	0.47	$C_{l_{\delta_a}}$	0.17
$C_{D_p}$	0.043	$C_{n_{\delta_a}}$	-0.011
		$C_{Y_{\delta_r}}$	0.19
		$C_{l_{\delta_r}}$	0.0024
		$C_{n_{\delta_r}}$	-0.069

## Appendix F

---

### Trim and Linearization

**NEW MATERIAL:** This appendix discussed specific methods for computing the trim conditions in simulation. Each method will require numerical computation of the Jacobian of a function, we discuss first in section F.1. Transformations between Euler angle models and quaternion models will be discussed in section F.2. Computing trim and linear models in Simulink will be discussed in section F.3. Computing trim and linear models using Python will be discussed in section F.5.

#### F.1 NUMERICAL COMPUTATION OF THE JACOBIAN

**NEW MATERIAL:** We will need to compute the Jacobian of a function  $w = f(z)$  where  $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ . Writing the function out component-wise gives

$$\begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{pmatrix} = \begin{pmatrix} f_1(x_1, x_2, \dots, x_n) \\ f_2(x_1, x_2, \dots, x_n) \\ \vdots \\ f_m(x_1, x_2, \dots, x_n) \end{pmatrix}.$$

The Jacobian of  $f$  with respect to  $x$  is defined as

$$\frac{\partial f}{\partial x} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & & & \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}.$$

Defining the column vectors

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

the Jacobian can be written as

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

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  $e_i \in \mathbb{R}^n$  is the column vector of all zeros except for a one in the  $i^{th}$  row. Therefore, letting  $\epsilon$  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  $f(x, z)$  at  $x$  is given below.

---

**Algorithm F.1:** Jacobian.

---

```

1  input: vector x, vector z
2  output: J
3  begin
4      eps = 0.001 # deviation
5      J ← m × n zero matrix
6      fx ← f(x, z)
7      for i = 1 to n do
8          x_eps ← column vector of zeros with eps in ith row
9          feps ← f(x_eps, z)
10         dfdx_i = (feps - fx) / eps
11         ith column of A ← = dfdx_i
12     end
13     return J
14 end

```

---

```

import numpy as np
def df_dx(x, z):
    # take partial of f(x, z) with respect to x
    eps = 0.001 # deviation
    A = np.zeros((m, n)) # Jacobian of f wrt x
    f_at_x = f(x, z)
    for i in range(0, n):
        x_eps = np.copy(x)
        x_eps[i][0] += eps # add eps to ith state
        f_at_x_eps = f(x_eps, z)
        df_dx_i = (f_at_x_eps - f_at_x) / eps
        A[:, i] = df_dx_i[:, 0]
    return A

```

## F.2 TRANSFORMATIONS BETWEEN EULER AND QUATERNION REPRESENTATIONS OF ATTITUDE

### NEW MATERIAL:

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

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

represent the state using Euler angles and let

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

represent the state using unit quaternions for attitude. We will use the shorthand notation  $\mathbf{p} = (p_n, p_e, p_d)^\top$ ,  $\mathbf{v} = (u, v, w)^\top$ ,  $\boldsymbol{\vartheta} = (\phi, \theta, \psi)^\top$ ,  $\mathbf{q} = (e_0, e_1, e_2, e_3)^\top$  and  $\boldsymbol{\omega} = (p, q, r)^\top$ , giving

$$\begin{aligned} x_e &= (\mathbf{p}^\top, \mathbf{v}^\top, \boldsymbol{\vartheta}^\top, \boldsymbol{\omega}^\top)^\top \\ x_q &= (\mathbf{p}^\top, \mathbf{v}^\top, \mathbf{q}^\top, \boldsymbol{\omega}^\top)^\top. \end{aligned}$$

Let  $\boldsymbol{\vartheta} = \Theta(\mathbf{q})$  denote the function given in Equation (B.1) that converts quaternions to Euler angles, and let  $\mathbf{q} = Q(\boldsymbol{\vartheta})$  be the function given in Equation (B.2) that converts Euler angles to quaternions. Using the methods described in the previous section, we can compute the Jacobians  $\frac{\partial \Theta}{\partial \mathbf{q}}$  and  $\frac{\partial Q}{\partial \boldsymbol{\vartheta}}$ .

Let  $T_e : \mathbb{R}^{13} \rightarrow \mathbb{R}^{12}$  be the mapping that converts quaternion states to Euler states so that  $x_e = T_e(x_q)$ , specifically,

$$T_e \begin{pmatrix} \mathbf{p} \\ \mathbf{v} \\ \mathbf{q} \\ \boldsymbol{\omega} \end{pmatrix} \triangleq \begin{pmatrix} \mathbf{p} \\ \mathbf{v} \\ \Theta(\mathbf{q}) \\ \boldsymbol{\omega} \end{pmatrix}.$$

Let  $T_q : \mathbb{R}^{12} \rightarrow \mathbb{R}^{13}$  be the mapping that converts Euler states to quaternion states so that  $x_q = T_q(x_e)$ , specifically,

$$T_q \begin{pmatrix} \mathbf{p} \\ \mathbf{v} \\ \boldsymbol{\vartheta} \\ \boldsymbol{\omega} \end{pmatrix} \triangleq \begin{pmatrix} \mathbf{p} \\ \mathbf{v} \\ Q(\boldsymbol{\vartheta}) \\ \boldsymbol{\omega} \end{pmatrix}.$$

The associated Jacobians are given by

$$\frac{\partial T_e}{\partial x_q}(x_q) = \begin{pmatrix} I_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 4} & 0_{3 \times 3} \\ 0_{3 \times 3} & I_{3 \times 3} & 0_{3 \times 4} & 0_{3 \times 3} \\ 0_{3 \times 3} & 0_{3 \times 3} & \frac{\partial \Theta}{\partial q}(x_q) & 0_{3 \times 3} \\ 0_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 4} & I_{3 \times 3} \end{pmatrix}$$

$$\frac{\partial T_q}{\partial x_e}(x_e) = \begin{pmatrix} I_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} \\ 0_{3 \times 3} & I_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} \\ 0_{4 \times 3} & 0_{4 \times 3} & \frac{\partial Q}{\partial \theta}(x_e) & 0_{4 \times 3} \\ 0_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} & I_{3 \times 3} \end{pmatrix}.$$

In the simulator, the equations of motion for the aircraft are given by

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

If the equations of motion had been written in terms of Euler states, then we would have

$$\begin{aligned} \dot{x}_e &= \frac{d}{dt} T_e(x_q) \\ &= \frac{\partial T_e}{\partial x_q}(x_q) \dot{x}_q \\ &= \frac{\partial T_e}{\partial x_q}(x_q) f_q(x_q, u) \\ &= \frac{\partial T_e}{\partial x_q}(T_q(x_e)) f_q(T_q(x_e), u), \end{aligned}$$

where we have used  $T_q$  to convert Euler states to quaternion states, and  $T_e$  to convert quaternion state derivatives into Euler state derivatives. The state space equations can now be computed numerically as

$$A = \frac{\partial}{\partial x_e} \left( \frac{\partial T_e}{\partial x_q}(T_q(x_e)) f_q(T_q(x_e), u) \right)$$

$$B = \frac{\partial}{\partial u} \left( \frac{\partial T_e}{\partial x_q}(T_q(x_e)) f_q(T_q(x_e), u) \right).$$

The resulting linear time-invariant equations of motion are

$$\dot{\tilde{x}}_e = A \tilde{x}_e + B \tilde{u},$$

where  $\tilde{x}_e = x_e - x_e^*$ , and  $\tilde{u} = u - u^*$ , where  $x^*$  and  $u^*$  are the trimmed state and input. Since  $x_e = (p_n, p_e, p_d, u, v, w, \phi, \theta, \psi, p, q, r)^\top$  and  $u = (\delta_t, \delta_e, \delta_a, \delta_r)^\top$  if we want the longitudinal state space equations where

$x_{lon} = (u, w, q, \theta, h)^\top$  and  $u_{lon} = (\delta_e, \delta_t)^\top$ , i.e.,

$$x_{lon} = \begin{pmatrix} u \\ w \\ q \\ \theta \\ h \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} p_n \\ p_e \\ p_d \\ u \\ v \\ w \\ \phi \\ \theta \\ \psi \\ p \\ q \\ r \end{pmatrix} \triangleq E_1 x_e$$

$$u_{lon} = \begin{pmatrix} \delta_e \\ \delta_t \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \delta_t \\ \delta_e \\ \delta_a \\ \delta_r \end{pmatrix} \triangleq E_2 u,$$

we have that

$$A_{lon} = E_1 A E_1^\top$$

$$B_{lon} = E_1 B E_2^\top.$$

Similarly, if we want the lateral state space equations where  $x_{lat} = (v, p, r, \phi, \psi)^\top$  and  $u_{lat} = (\delta_a, \delta_r)^\top$ , i.e.,

$$x_{lat} = \begin{pmatrix} v \\ p \\ r \\ \phi \\ \psi \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} p_n \\ p_e \\ p_d \\ u \\ v \\ w \\ \phi \\ \theta \\ \psi \\ p \\ q \\ r \end{pmatrix} \triangleq E_3 x_e$$

$$u_{lat} = \begin{pmatrix} \delta_a \\ \delta_r \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \delta_t \\ \delta_e \\ \delta_a \\ \delta_r \end{pmatrix} \triangleq E_4 u,$$

we have that

$$A_{lat} = E_3 A E_3^\top$$

$$\mathcal{B}_{lat} = E_3 \mathcal{B} E_4^\top.$$

### F.3 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 5.3, given the parameters  $V_a^*$ ,  $\gamma^*$ , 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 (5.17) and (5.18),  $\dot{x}^*$  is given by equation (5.20), and where  $f(x, u)$  is defined by the right-hand side of equations (5.1)–(5.12).

The format for the Simulink `trim` command is

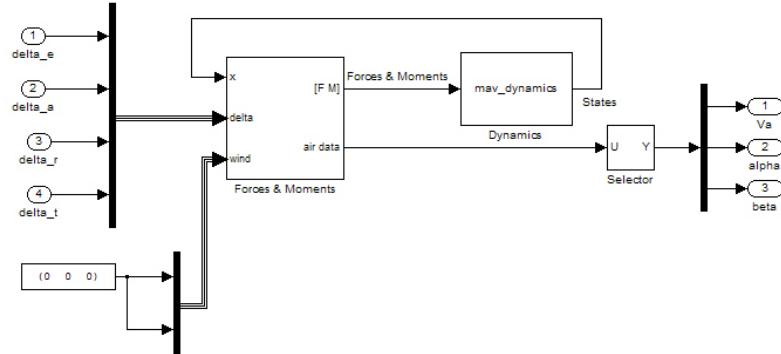
```
[X,U,Y,DX]=TRIM( 'SYS' ,X0,U0,Y0,IX,IU,IY,DX0,IDX),
```

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  $\alpha$ , and the sideslip angle  $\beta$ . Our purpose in specifying  $V_a$ ,  $\alpha$ , and  $\beta$  as outputs is that we wish to force the Simulink `trim` command to maintain  $V_a = V_a^*$  and  $\alpha^*$  is often a quantity of interest. If we have access to a rudder, then we can enforce a coordinated turn to force the trim command to maintain  $\beta^* = 0$ . If a rudder is not available, then  $\beta$  will not necessarily be zero in a turn.

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 `X0`, input `U0`, output `Y0`, and derivative of the state `DX0` 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`, `IY`, and `IDX`.



**Figure F.1:** Simulink diagram used to compute trim and linear state space models.

**MODIFIED MATERIAL:** For our situation we know that

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

therefore we let

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

Similarly, the initial state, inputs, and outputs can be specified as

```
X0 = [0; 0; 0; Va; 0; 0; 0; gamma; 0; 0; 0; 0]
IX0 = []
U0 = [0; 0; 0; 1]
IU0 = []
Y0 = [Va; 0; 0]
IY0 = [1, 3].
```

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 (5.42) and (5.49). To obtain equation (5.42) 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]);
```

#### F.4 TRIM AND LINEARIZATION IN MATLAB

##### NEW MATERIAL:

In Matlab, trim can be computed using the built in optimization function `fmincon`. In general `fmincon` can be configured to minimize nonlinear functions with general constraints. For example, to minimize the function  $J(x) = x^T x - \gamma$  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.5 TRIM AND LINEARIZATION IN PYTHON

##### NEW MATERIAL:

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

```
import numpy as np
from scipy.optimize import minimize
# initial guess for the optimum
x0 = np.array([[10; 5; -20; 8]]).T;
gam = 3; # parameter passed into objective function
# first constraint is equality constraint x1==x2
```

```
# second constraint is inequality constraint sin(x3)<=-0.5
cons = ({'type': 'eq', 'fun': lambda x: x[0]-x[1],},
         {'type': 'ineq', 'fun': lambda x: np.sin(x[2])+0.5})
# solve the minimization problem
res = minimize(objective, x0, method='SLSQP', args = (gam), constraints=cons,
               options={'ftol': 1e-10, 'disp': True})
# extract optimal state
xopt = np.array([res.x]).T

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



## Appendix G

---

### Essentials from Probability Theory

Let  $X = (x_1, \dots, x_n)^\top$  be a random vector whose elements are random variables. The mean, or expected value of  $X$ , is denoted by

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_n \end{pmatrix} = \begin{pmatrix} E\{x_1\} \\ \vdots \\ E\{x_n\} \end{pmatrix} = E\{X\},$$

where

$$E\{x_i\} = \int \xi f_i(\xi) d\xi$$

and  $f(\cdot)$  is the probability density function for  $x_i$ . Given any pair of components  $x_i$  and  $x_j$  of  $X$ , we denote their covariance as

$$\text{cov}(x_i, x_j) = \Sigma_{ij} = E\{(x_i - \mu_i)(x_j - \mu_j)\}.$$

The covariance of any component with itself is the variance, that is,

$$\text{var}(x_i) = \text{cov}(x_i, x_i) = \Sigma_{ii} = E\{(x_i - \mu_i)(\xi - \mu_i)\}.$$

The standard deviation of  $x_i$  is the square root of the variance:

$$\text{stdev}(x_i) = \sigma_i = \sqrt{\Sigma_{ii}}.$$

The covariances associated with a random vector  $X$  can be grouped into a matrix known as the covariance matrix:

$$\begin{aligned} \Sigma &= \begin{pmatrix} \Sigma_{11} & \Sigma_{12} & \cdots & \Sigma_{1n} \\ \Sigma_{21} & \Sigma_{22} & \cdots & \Sigma_{2n} \\ \vdots & & \ddots & \vdots \\ \Sigma_{n1} & \Sigma_{n2} & \cdots & \Sigma_{nn} \end{pmatrix} \\ &= E\{(X - \boldsymbol{\mu})(X - \boldsymbol{\mu})^\top\} \\ &= E\{XX^\top\} - \boldsymbol{\mu}\boldsymbol{\mu}^\top. \end{aligned}$$

Note that  $\Sigma = \Sigma^\top$  so that  $\Sigma$  is both symmetric and positive semi-definite, which implies that its eigenvalues are real and nonnegative.

The probability density function for a Gaussian random variable is given by

$$f_x(x) = \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{(x-\mu_x)^2}{\sigma_x^2}},$$

where  $\mu_x$  is the mean of  $x$  and  $\sigma_x$  is the standard deviation. The vector equivalent is given by

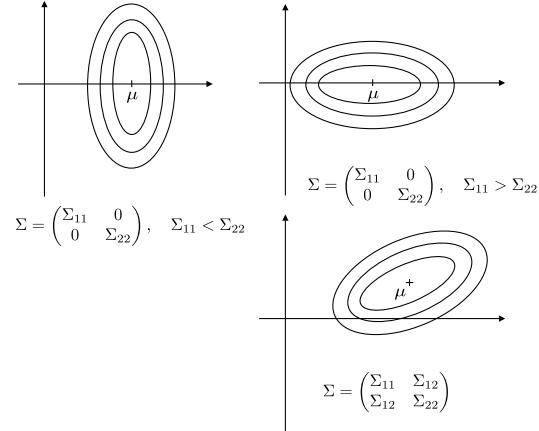
$$f_X(X) = \frac{1}{\sqrt{2\pi \det \Sigma}} \exp \left[ -\frac{1}{2}(X - \boldsymbol{\mu})^\top \Sigma^{-1}(X - \boldsymbol{\mu}) \right],$$

in which case we write

$$X \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$$

and say that  $X$  is normally distributed with mean  $\boldsymbol{\mu}$  and covariance  $\Sigma$ .

Figure G.1 shows the level curves for a 2-D Gaussian random variable with different covariance matrices.



**Figure G.1:** Level curves for the pdf of a 2-D Gaussian random variable.

## Appendix H

---

### Sensor Parameters

This appendix gives the noise and error parameters for several commercially available sensors typical of what would be used in a MAV autopilot. There are numerous error sources for each of the sensors. Some are random, like electrical noise. Others, such as nonlinearity, temperature sensitivity, and cross-axis sensitivity, are more deterministic in nature. We assume that these deterministic error sources can be mitigated through careful calibration and compensation, as is done by most autopilot manufacturers, with the majority of the remaining error having random characteristics.

Rate gyros, accelerometers, and pressure sensors are sampled each time through the autopilot control loop. A common sample rate is in the range of 50 to 100 Hz. We denote the control loop sample period as  $T_s$ . Digital compasses and GPS receivers are both digital devices with slower update rates that depend on the specific model of the sensor. We denote these sample periods as  $T_{s,\text{compass}}$  and  $T_{s,\text{GPS}}$  and specify values for typical sensors below.

**TWM:** We need to update this section to reflect current (2021) sensor performance capabilities.

#### H.1 RATE GYROS

An example of a MEMS rate gyro suitable for a MAV autopilot is the Analog Devices ADXRS450. It has a range of  $\pm 350$  deg/s, a bandwidth of 80 Hz, and a noise density of  $0.015 \text{ deg/s}/\sqrt{\text{Hz}}$ . The standard deviation of the measurement error due to sensor noise is

$$\sigma_{\text{gyro},*} = N\sqrt{B},$$

which for the ADXRS450 results in  $\sigma_{\text{gyro},*} = 0.13$  deg/s. Sources of deterministic error include cross-axis sensitivity ( $\pm 3$  percent), nonlinearity (0.05 percent full-scale range RMS), and acceleration sensitivity of (0.03 deg/s/g).

## H.2 ACCELEROMETERS

An example of a MEMS accelerometer that could be used in an autopilot is the Analog Devices ADXL325. It has a range of  $\pm 6 g$  and a variable bandwidth of 0.5 to 550 Hz. For an autopilot, a typical bandwidth would be 100 Hz. It has a noise density of  $250 \mu\text{g}/\sqrt{\text{Hz}}$ . The standard deviation of the measurement error due to sensor noise is

$$\sigma_{\text{accel},*} = N\sqrt{B},$$

which for the ADXL325 results in  $\sigma_{\text{accel},*} = 0.0025g$ . Sources of deterministic error include cross-axis sensitivity ( $\pm 1$  percent) and nonlinearity ( $\pm 0.2$  percent of full-scale range).

## H.3 PRESSURE SENSORS

An example of an absolute pressure sensor that can be used for altitude measurement is the Freescale Semiconductor MP3H6115A. It has a range of 15 to 115 kPa, and its maximum error is characterized as having a bound of 1.5 percent of the full scale, or  $\pm 1.5$  kPa. The accuracy of the sensor, as indicated by the maximum error, is limited due to linearity errors, temperature sensitivity, and pressure hysteresis. Practical experience with MEMS absolute pressure sensors has shown that careful calibration can reduce these errors to about 0.125 kPa of temperature-related bias drift and about 0.01 kPa of sensor noise. Therefore, we have  $\beta_{\text{abs pres}} = 0.125 \text{ kPa}$  and  $\sigma_{\text{abs pres}} = 0.01 \text{ kPa}$ .

The Freescale Semiconductor MPXV5004G is an example of a differential pressure sensor appropriate for use as an airspeed sensor. It has a range of 0 to 4 kPa, and its maximum error has a bound of 2.5 percent of the full-scale range, or 0.1 kPa. The accuracy of the sensor is limited by linearity errors, temperature sensitivity, and pressure hysteresis. Practical experience with this type of differential pressure sensor has shown that careful calibration can reduce these errors to about 0.02 kPa of temperature-related bias drift and about 0.002 kPa of sensor noise. Therefore, we have  $\beta_{\text{diff pres}} = 0.020 \text{ kPa}$  and  $\sigma_{\text{diff pres}} = 0.002 \text{ kPa}$ .

## H.4 DIGITAL COMPASS/MAGNETOMETER

An example of a digital compass suitable for a small unmanned aircraft is the Honeywell HMR3300. It is a three-axis, tilt-compensated device with a built-in microcontroller for signal conditioning. When level it has an accu-

racy of  $\pm 1$  degree and  $\pm 3$  degrees of accuracy up to  $\pm 30$  degrees of tilt. It has a 0.5 degree repeatability and an 8 Hz update rate ( $T_{s,\text{compass}} = 0.125$  s). Assuming that a portion of the accuracy and repeatability errors are due to uncertainty in the declination angle and that a portion is due to electromagnetic interference, reasonable parameters for the sensor noise standard deviation and bias error are  $\sigma_{\text{mag}} = 0.3$  degrees and  $\beta_{\text{mag}} = 1$  degree.

## H.5 GPS

Sources of GPS measurement error and modeling of GPS measurement error are discussed in detail in section 7.5. Sample periods for GPS can vary between 0.2 and 2 seconds. For our purposes we will assume that the sample period for GPS is given by  $T_{s,\text{GPS}} = 1.0$  s.

TWM: The GPS model parameters are not independent of the sample rate. Need to update the model parameters to current values.



## Appendix I

---

### Useful Formulas and Other Information

#### I.1 CONVERSION FROM KNOTS TO MPH

The formula for conversion from knots to miles per hour is

$$1 \text{ mph} = 1.15 \text{ knots.}$$

Example values are shown in the table below.

knots	10	20	30	40	50	60	70	80	90
mph	11.5	23.0	34.6	46.1	57.6	69.1	80.6	92.2	103.7

#### I.2 DENSITY OF AIR

The density of air is dependent on the air pressure  $P_s$  and the temperature  $T$ . Air density is given by the ideal gas law as

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

where  $R = 287.05[\text{J/kg-K}]$  is the specific gas constant for dry air. Notice that in this formula, temperature is expressed in units of Kelvin. The conversion from Fahrenheit to Kelvin is given by

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

Pressure is expressed in  $\text{N/m}^2$ . Typical weather data reports pressure in inches of mercury. The conversion from  $\text{N/m}^2$  to inches of mercury is given by

$$P_s[\text{N/m}^2] = 3376.85P_s[\text{inHg}].$$



---

## Bibliography

- [1] B. L. Stevens and F. L. Lewis, *Aircraft Control and Simulation*. Hoboken, NJ: John Wiley & Sons, Inc., 2nd ed., 2003.
- [2] M. D. Shuster, “A survey of attitude representations,” *The Journal of the Astronautical Sciences*, vol. 41, no. 4, pp. 439–517, 1993.
- [3] D. T. Greenwood, *Principles of Dynamics*. Englewood Cliffs, NJ: Prentice Hall, 2nd ed., 1988.
- [4] T. R. Kane and D. A. Levinson, *Dynamics: Theory and Applications*. New York: McGraw Hill, 1985.
- [5] M. W. Spong and M. Vidyasagar, *Robot Dynamics and Control*. New York: John Wiley & Sons, Inc., 1989.
- [6] J. H. Blakelock, *Automatic Control of Aircraft and Missiles*. New York: John Wiley & Sons, 2nd ed., 1991.
- [7] R. C. Nelson, *Flight Stability and Automatic Control*. Boston, MA: McGraw-Hill, 2nd ed., 1998.
- [8] 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.
- [9] H. Goldstein, *Classical Mechanics*. Addison-Wesley, 1951.
- [10] A. V. Rao, *Dynamics of Particles and Rigid Bodies: A Systematic Approach*. Cambridge: Cambridge University Press, 2006.
- [11] M. J. Sidi, *Spacecraft Dynamics and Control*. Cambridge Aerospace Series, New York: Cambridge University Press, 1997.
- [12] S. S. Patankar, D. E. Schinstock, and R. M. Caplinger, “Application of pendulum method to UAV momental ellipsoid estimation,” in *6th AIAA Aviation Technology, Integration and Operations Conference (ATIO)*, AIAA 2006-7820, September 2006.

- [13] 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*, AIAA 2007-6822, August 2007.
- [14] J. B. Marion, *Classical Dynamics of Particles and Systems*. New York: Academic Press, 2nd ed., 1970.
- [15] W. E. Wiesel, *Spaceflight Dynamics*. New York: McGraw Hill, 2nd ed., 1997.
- [16] J. R. Wertz, ed., *Spacecraft Attitude Determination and Control*. Dordrecht, Neth.: Kluwer Academic Publishers, 1978.
- [17] J. Roskam, *Airplane Flight Dynamics and Automatic Flight Controls, Parts I & II*. Lawrence, KS: DARcorporation, 1998.
- [18] M. V. Cook, *Flight Dynamics Principles*. New York: John Wiley & Sons, 1997.
- [19] R. F. Stengel, *Flight Dynamics*. Princeton, N.J.: Princeton University Press, 2004.
- [20] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*. McGraw-Hill, 1987.
- [21] D. O. Dommasch, *Elements of Propeller and Helicopter Aerodynamics*. Pitman Publishing Corporation, 2053.
- [22] “Uiuc propeller data site.”
- [23] J. W. Langelaan, N. Alley, and J. Niedhoefer, “Wind field estimation for small unmanned aerial vehicles,” in *AIAA Guidance, Navigation, and Control Conference*, August 2010. AIAA 2010-8177.
- [24] W. F. Phillips, *Mechanics of Flight*. Hoboken, NJ: Wiley, 2nd ed., 2010.
- [25] R. Rysdyk, “UAV path following for constant line-of-sight,” in *Proceedings of the AIAA 2nd Unmanned Unlimited Conference*, AIAA, September 2003. AIAA 2003-6626.
- [26] J. Osborne and R. Rysdyk, “Waypoint guidance for small UAVs in wind,” in *Proceedings of the AIAA Infotech@Aerospace Conference*, September 2005.

- [27] B. Etkin and L. D. Reid, *Dynamics of Flight: Stability and Control*. New York: John Wiley & Sons, 1996.
- [28] L. F. Faleiro and A. A. Lambregts, “Analysis and tuning of a total energy control system control law using eigenstructure assignment,” *Aerospace Science and Technology*, no. 3, pp. 127–140, 1999.
- [29] A. A. Lambregts, “Integrated systems design for flight and propulsion control using total energy principles,” in *AIAA Aircraft Design, Systems and Technology Meeting*, 1983.
- [30] J. H. Blakelock, *Automatic Control of Aircraft and Missiles*. New York: John Wiley & Sons, 1965.
- [31] A. A. Lambregts, “Vertical flight path and speed control autopilot design using total energy principles,” in *Guidance and Control Conference*, 1983.
- [32] A. A. Lambregts, “Operational Aspects of the Integrated Vertical Flight Path and Speed Control System,” tech. rep., SAE Technical Paper 831420, 1983.
- [33] A. A. Lambregts, “Integrated system design for flight and propulsion control using total energy principles,” in *Aircraft Design, Systems and Technology Meeting*, 1983.
- [34] K. R. Bruce, J. R. Kelly, and L. H. J. Person, “NASA B737 Flight test Results of the Total Energy Control System,” in *AIAA Conference on Guidance, Navigation, and Control*, (Williamsburg, Virginia), 1986.
- [35] K. R. Bruce, “Flight Test Results of the Total Energy Control System,” tech. rep., NASA CR-178285, 1987.
- [36] C. Voth and U.-L. Ly, “Total Energy Control System Autopilot Design with Constrained Parameter Optimization,” in *American Control Conference*, pp. 1332–1337, 1990.
- [37] M. A. Bruzzini, *Development of a TECS Control-Law for the Lateral Directional Axis of the McDonnell Douglas F-15 Eagle*. Master’s, University of Washington, 1994.
- [38] K. E. Ji, W. Wei, L. I. Ai-jun, and W. Chang-qing, “Lateral Directional Axis Control of Aircraft Based on TECS,” in *2nd International Congress on Image and Signal Processing*, (Tianjin), pp. 1–5, 2009.

- [39] S.-W. Chen, P.-C. Chen, C.-D. Yang, and Y.-F. Jeng, “Total Energy Control System for Helicopter Flight/Propulsion Integrated Controller Design,” *Journal of Guidance, Control, and Dynamics*, vol. 30, pp. 1030–1039, July 2007.
- [40] M. E. Argyle and R. W. Beard, “Nonlinear total energy control for the longitudinal dynamics of an aircraft,” in *2016 American Control Conference (ACC)*, pp. 6741–6746, 2016.
- [41] A. E. Bryson and Y. C. Ho, *Applied Optimal Control*. New York, New York: Hemisphere, 1975.
- [42] B. D. O. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods*. Englewood Cliffs, New Jersey: Prentice Hall, 1990.
- [43] F. L. Lewis, *Optimal Control*. New York: John Wiley and Sons, 1986.
- [44] P. Dorato, C. Abdallah, and V. Cerone, *Linear-Quadratic Control: An Introduction*. Prentice Hall, 1995.
- [45] E. Lavretsky and K. A. Wise, *Robust and Adaptive Control*. Springer, 2013.
- [46] “U.S. standard atmosphere, 1976.” U.S. Government Printing Office, Washington, D.C., 1976.
- [47] National Oceanic and Atmospheric Administration, “The world magnetic model.” <http://www.ngdc.noaa.gov/geomag/WMM/>, 2011.
- [48] J.-M. Zogg, *GPS: Essentials of Satellite Navigation*. <http://www.u-blox.com>: u-blox AG, 2009.
- [49] B. W. Parkinson, J. J. Spilker, P. Axelrad, and P. Enge, eds., *Global Positioning System: Theory and Applications*. American Institute for Aeronautics and Astronautics, 1996.
- [50] E. D. Kaplan and C. J. Hegarty, eds., *Understanding GPS/GNSS: Principles and Applications*. Boston: Artech House, third ed., 1996.
- [51] M. S. Grewal, A. P. Andrews, and C. Bartone, *Global Navigation Satellite Systems, Inertial Navigation, and Integration*. Hoboken, NJ: John Wiley & Sons, fourth ed., 2020.

- [52] J. Rankin, “GPS and differential GPS: An error model for sensor simulation,” in *Position, Location, and Navigation Symposium*, pp. 260–266, 1994.
- [53] R. Figliola and D. Beasley, *Theory and Design for Mechanical Measurements*. Hoboken, NJ: John Wiley & Sons, Inc., seventh ed., 2109.
- [54] S. D. Senturia, *Microsystem Design*. New York: Springer, 2001.
- [55] J. W. Gardner, V. K. Varadan, and O. O. Awadelkarim, *Microsensors, MEMS, and Smart Devices*. West Sussex, England: John Wiley & Sons Ltd, 2001.
- [56] V. Kaajakari, *Practical MEMS: Design of Microsystems, Accelerometers, Gyroscopes, RF MEMS, Optical MEMS, and Microfluidic Systems*. Small Gear Publishing, 2009.
- [57] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Transactions of the ASME, Journal of Basic Engineering*, vol. 82, pp. 35–45, 1960.
- [58] F. L. Lewis, *Optimal Estimation: With an Introduction to Stochastic Control Theory*. New York: John Wiley & Sons, 1986.
- [59] A. Gelb, ed., *Applied Optimal Estimation*. Cambridge, MA: The M.I.T. Press, 1974.
- [60] B. Anderson and J. B. Moore, *Linear Optimal Control*. Englewood Cliffs, NJ: Prentice Hall, 1971.
- [61] R. G. Brown and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*. Hoboken, NJ: John Wiley & Sons, Inc., 2012.
- [62] A. M. Eldredge, “Improved state estimation for miniature air vehicles,” Master’s thesis, Brigham Young University, 2006.
- [63] R. W. Beard, “State estimation for micro air vehicles,” in *Innovations in Intelligent Machines I* (J. S. Chahl, L. C. Jain, A. Mizutani, and M. Sato-Ilic, eds.), pp. 173–199, Berlin Heidelberg: Springer Verlag, 2007.
- [64] 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, no. 9, pp. 348–360, 2005.

- [65] 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, no. 3, 2007.
- [66] S. Ettinger, M. Nechyba, P. Ifju, and M. Waszak, “Vision-guided flight stability and control for micro air vehicles,” *Advanced Robotics*, vol. 17, no. 7, pp. 617–640, 2003.
- [67] H. Chitsaz and S. M. LaValle, “Time-optimal paths for a dubins airplane,” in *2007 46th IEEE Conference on Decision and Control*, pp. 2379–2384, 2007.
- [68] J. D. Anderson, *Introduction to Flight*. Eighth, New York: McGraw Hill, 2016.
- [69] M. Owen, R. W. Beard, and T. W. McLain, “Implementing dubins airplane paths on fixed-wing uavs,” in *Handbook of Unmanned Aerial Vehicles* (G. J. V. Kimon P. Valavanis, ed.), ch. 68, pp. 1677–1702, Springer, 2014.
- [70] V. M. Goncalves, L. C. A. Pimenta, C. A. Maia, B. C. O. Dutra, 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.
- [71] 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, no. 3, pp. 519–529, 2007.
- [72] 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*, pp. 5788–5794, June 2006.
- [73] 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.
- [74] D. A. Lawrence, E. W. Frew, and W. J. Pisano, “Lyapunov vector fields for autonomous unmanned aircraft flight control,” *AIAA Journal of Guidance, Control, and Dynamics*, vol. 31, no. 5, pp. 1220–1229, 2008.
- [75] 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, March 1985.

- [76] K. Sigurd and J. P. How, “UAV trajectory design using total field collision avoidance,” in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, August 2003.
- [77] S. Park, J. Deyst, and J. How, “A new nonlinear guidance logic for trajectory tracking,” in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, pp. AIAA 2004–4900, August 2004.
- [78] I. Kaminer, A. Pascoal, E. Hallberg, and C. Silvestre, “Trajectory tracking for autonomous vehicles: An integrated approach to guidance and control,” *AIAA Journal of Guidance, Control and Dynamics*, vol. 21, no. 1, pp. 29–38, 1998.
- [79] T. W. McLain and R. W. Beard, “Coordination variables, coordination functions, and cooperative timing missions,” *AIAA Journal of Guidance, Control and Dynamics*, vol. 28, no. 1, pp. 150–161, 2005.
- [80] L. E. Dubins, “On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents,” *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.
- [81] 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, no. 3, pp. 471–477, 2005.
- [82] 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*, pp. 1301–1306, December 2002.
- [83] P. Chandler, S. Rasmussen, and M. Pachter, “UAV cooperative path planning,” in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, pp. AIAA–2000–4370, August 2000.
- [84] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, “Randomized kinodynamic motion planning with moving obstacles,” in *Algorithmic and Computational Robotics: New Directions*, pp. 247–264, Natick, MA: A. K. Peters, 2001.
- [85] F. Lamiraux, S. Sekhavat, and J.-P. Laumond, “Motion planning and control for Hilare pulling a trailer,” *IEEE Transactions on Robotics and Automation*, vol. 15, no. 4, pp. 640–652, 1999.

- [86] R. M. Murray and S. S. Sastry, “Nonholonomic motion planning: Steering using sinusoids,” *IEEE Transactions on Automatic Control*, vol. 38, no. 5, pp. 700–716, 1993.
- [87] T. Balch and R. C. Arkin, “Behavior-based formation control for multirobot teams,” *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 926–939, 1998.
- [88] R. C. Arkin, *Behavior-based Robotics*. Cambridge, MA: MIT Press, 1998.
- [89] R. Sedgewick and K. Wayne, *Algorithms*. Upper Saddle River, NJ: Addison-Wesley, 4th ed., 2011.
- [90] F. Aurenhammer, “Voronoi diagrams – a survey of fundamental geometric data structure,” *ACM Computing Surveys*, vol. 23, no. 3, pp. 345–405, 1991.
- [91] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA: The MIT Press, third ed., 2009.
- [92] T. K. Moon and W. C. Stirling, *Mathematical Methods and Algorithms*. Englewood Cliffs, NJ: Prentice Hall, 2000.
- [93] S. M. LaValle and J. J. Kuffner, “Randomized kinodynamic planning,” *International Journal of Robotic Research*, vol. 20, no. 5, pp. 378–400, 2001.
- [94] J.-C. Latombe, *Robot Motion Planning*. Dordrecht, Neth.: Kluwer Academic Publishers, 1991.
- [95] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementation*. Cambridge, MA: MIT Press, 2005.
- [96] S. M. LaValle, *Planning Algorithms*. Cambridge: Cambridge University Press, 2006.
- [97] T. McLain and R. Beard, “Cooperative rendezvous of multiple unmanned air vehicles,” in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, pp. AIAA 2000–4369, August 2000.
- [98] T. W. McLain, P. R. Chandler, S. Rasmussen, and M. Pachter, “Cooperative control of UAV rendezvous,” in *Proceedings of the American Control Conference*, pp. 2309–2314, June 2001.

- [99] R. W. Beard, T. W. McLain, M. Goodrich, and E. P. Anderson, “Coordinated target assignment and intercept for unmanned air vehicles,” *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 911–922, December 2002.
- [100] H. Choset and J. Burdick, “Sensor-based exploration: The hierarchical generalized Voronoi graph,” *The International Journal of Robotic Research*, vol. 19, no. 2, pp. 96–125, 2000.
- [101] H. Choset, S. Walker, K. Eiamsa-Ard, and J. Burdick, “Sensor-based exploration: Incremental construction of the hierarchical generalized Voronoi graph,” *The International Journal of Robotics Research*, vol. 19, no. 2, pp. 126–148, 2000.
- [102] D. Eppstein, “Finding the  $k$  shortest paths,” *SIAM Journal of Computing*, vol. 28, no. 2, pp. 652–673, 1999.
- [103] S. M. LaValle, “Rapidly-exploring random trees: A new tool for path planning.” TR 98-11, Computer Science Dept., Iowa State University, October 1998.
- [104] 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*, pp. 995–1001, April 2000.
- [105] M. Zucker, J. Kuffner, and M. Branicky, “Multipartite RRTs for rapid replanning in dynamic environments,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, April 2007.
- [106] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *International Journal of Robotic Research*, vol. 30, no. 7, pp. 846–894, 2011.
- [107] A. Ladd and L. E. Kavraki, “Generalizing the analysis of PRM,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2120–2125, May 2002.
- [108] E. Frazzoli, M. A. Dahleh, and E. Feron, “Real-time motion planning for agile autonomous vehicles,” *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 1, pp. 116–129, 2002.
- [109] E. U. Acar, H. Choset, and J. Y. Lee, “Sensor-based coverage with extended range detectors,” *IEEE Transactions on Robotics*, vol. 22, no. 1, pp. 189–198, 2006.

- [110] 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*, pp. 612–617, May 2002.
- [111] 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*, pp. 2722–2727, April 2000.
- [112] 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*, pp. 1327–1332, May 2002.
- [113] M. Schwager, J.-J. Slotine, and D. Rus, “Consensus learning for distributed coverage control,” in *Proceedings of the International Conference on Robotics and Automation*, pp. 1042–1048, May 2008.
- [114] 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*, NATO Research and Technology Organization AVT-146, May 2007.
- [115] 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, no. 4, pp. 361–382, 2006.
- [116] G. F. Franklin, J. D. Powell, and M. Workman, *Digital Control of Dynamic Systems*. Addison Wesley, 3rd ed., 1998.
- [117] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*. New York: Springer-Verlag, 2004.
- [118] P. Zarchan, *Tactical and Strategic Missile Guidance*, vol. 124 of *Progress in Astronautics and Aeronautics*. Washington, DC: American Institute of Aeronautics and Astronautics, 1990.
- [119] M. Guelman, M. Idan, and O. M. Golan, “Three-dimensional minimum energy guidance,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 31, no. 2, pp. 835–841, 1995.
- [120] 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*, pp. 991–996, August 1999.

- [121] E. Frew and S. Rock, "Trajectory generation for monocular-vision based tracking of a constant-velocity target," in *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, vol. 3, pp. 3479–3484, September 2003.
- [122] R. Kumar, S. Samarasekera, S. Hsu, and K. Hanna, "Registration of highly-oblique and zoomed in aerial video to reference imagery," in *Proceedings of the IEEE Computer Society Computer Vision and Pattern Recognition Conference*, vol. 4, pp. 303–307, September 2000.
- [123] K. Lillywhite, D.-J. Lee, B. Tippets, S. Fowers, A. Dennis, B. Nelson, and J. Archibald, "An embedded vision system for an unmanned four-rotor helicopter," in *SPIE Intelligent Robots and Computer Vision XXIV: Algorithms, Techniques, and Active Vision*, vol. 6384, October 2006.
- [124] 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. 929–932, September 2003.
- [125] 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*, December 2007.
- [126] I. Wang, V. Dobrokhodov, I. Kaminer, and K. Jones, "On vision-based target tracking and range estimation for small UAVs," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, pp. 1–11, August 2005.
- [127] 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*, pp. AIAA 2006–6608, August 2006.
- [128] 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*, pp. AIAA 2004–4779, American Institute of Aeronautics and Astronautics, August 2004.
- [129] I. H. Wang, V. N. Dobrokhodov, 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*, August 2005.

- [130] R. W. Beard, D. Lee, M. Quigley, S. Thakoor, and S. Zornetzer, “A new approach to observation of descent and landing of future Mars mission using bioinspired technology innovations,” *AIAA Journal of Aerospace Computing, Information, and Communication*, vol. 2, no. 1, pp. 65–91, 2005.
- [131] 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*, pp. AIAA 2006–6246, August 2006.
- [132] V. N. Dobrokhodov, I. I. Kaminer, and K. D. Jones, “Vision-based tracking and motion estimation for moving targets using small UAVs,” in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, pp. AIAA 2006–6606, August 2006.
- [133] E. W. Frew, “Sensitivity of cooperative target geolocation to orbit coordination,” *Journal of Guidance, Control, and Dynamics*, vol. 31, no. 4, pp. 1028–1040, 2008.
- [134] D. Murray and A. Basu, “Motion tracking with an active camera,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 449–459, 1994.
- [135] S. Hutchinson, G. D. Hager, and P. I. Corke, “A tutorial on visual servo control,” *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, 1996.
- [136] J. Oliensis, “A critique of structure-from-motion algorithms,” *Computer Vision and Image Understanding (CVIU)*, vol. 80, no. 2, pp. 172–214, 2000.
- [137] J. Santos-Victor and G. Sandini, “Uncalibrated obstacle detection using normal flow,” *Machine Vision and Applications*, vol. 9, no. 3, pp. 130–137, 1996.
- [138] L. Loriog, R. Brooks, and W. Grimson, “Visually guided obstacle avoidance in unstructured environments,” in *Proceedings of IROS '97*, September 1997.
- [139] R. Nelson and Y. Aloimonos, “Obstacle avoidance using flow field divergence,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 10, pp. 1102–1106, 1989.

- [140] F. Gabbiani, H. Krapp, and G. Laurent, “Computation of object approach by a wide field visual neuron,” *Journal of Neuroscience*, vol. 19, no. 3, pp. 1122–1141, 1999.
- [141] 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*, pp. AIAA 2007–6609, American Institute of Aeronautics and Astronautics, August 2007.
- [142] A. E. Bryson and Y. C. Ho, *Applied Optimal Control*. Waltham, MA: Blaisdell Publishing Company, 1969.
- [143] M. Guelman, “Proportional navigation with a maneuvering target,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 8, no. 3, pp. 364–371, 1972.
- [144] C. F. Lin, *Modern Navigation, Guidance, and Control Processing*. Englewood Cliffs, NJ: Prentice Hall, 1991.
- [145] J. Waldmann, “Line-of-sight rate estimation and linearizing control by an imaging seeker in a tactical missile guided by proportional navigation,” *IEEE Transactions on Control Systems Technology*, vol. 10, no. 4, pp. 556–567, 2002.
- [146] R. Mahony, V. Kumar, and P. Corke, “Multirotor aerial vehicles: Modeling, estimation, and control of a quadrotor,” *IEEE Robotics & Automation Magazine*, aug 2012.
- [147] P. Martin and E. Salaun, “The true role of accelerometer feedback in quadrotor control,” in *Proceedings of the IEEE International Conference on Robotics & Automation*, (Anchorage, Alaska), pp. 1623–1629, may 2010.
- [148] J. B. Kuipers, *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace, and Virtual Reality*. Princeton, NJ: Princeton University Press, 1999.
- [149] S. Sarabandi and F. Thomas, “Accurate computation of quaternions from rotation matrices,” in *Advances in Robot Kinematics* (J. Lenarctic and V. Parenti-Castelli, eds.), pp. 39–46, Springer, 2019.
- [150] J. P. Corbett and F. B. Wright, “Stabilization of computer circuits,” in *WADC TR 57-25* (E. Hochfeld, ed.), (Wright-Patterson Air Force Base, OH), 1957.

- [151] C. Rucker, “Integrating rotations using nonunit quaternions,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2979–2986, 2018.
- [152] M. Burston, R. Sabatini, A. Gardi, and R. Clotheir, “Reverse engineering of a fixed wing unmanned aircraft 6-dof model based on laser scanner measurements,” in *IEEE Metrology for Aerospace*, (Benevento, Italy), pp. 144–149, 2014.
- [153] AXI Motors. <https://www.modelmotors.cz/product/detail/516/>, 2023.
- [154] APC Propellers. <https://www.apcprop.com/product/20x10e/>, 2023.

---

## Index

- [angle-of-attack](#), 20
- [angle-of-attack](#), 16
- absolute pressure sensor, 123, 144
- accelerometers, 117, 145, 157
- aerodynamic coefficients, 51
- aileron, 42
- airspeed, 19, 20, 23, 57, 59
- altitude, 65
- angle-of-attack, 41, 59
- autopilot
  - airspeed hold using throttle, 107
  - altitude hold using pitch, 106
  - course hold, 99
  - pitch attitude hold, 104
  - yaw damper, 101
- bandwidth separation, 101
- coordinate frames, 9, 247
  - body frame, 15, 248
  - camera frame, 248
  - gimbal frame, 248
  - gimbal-1 frame, 247
  - inertial frame, 13
  - stability frame, 16
  - vehicle frame, 13
  - vehicle-1 frame, 14
  - vehicle-2 frame, 14
  - wind frame, 17
- coordinated turn, 66, 72, 177
- course angle, 21, 22, 138
- crab angle, 21, 22
- design models, 4, 63, 175
- differential pressure sensor, 126, 144
- Dubins airplane, 181
- Dubins path
  - computation, 217
  - definition, 216
- RRT paths through obstacle field, 242
- tracking, 222
- Dutch-roll mode, 90
- dynamics
  - rotational motion, 34
  - translational motion, 33
- ego motion, 255
- elevator, 42
- estimation
  - airspeed, 144
  - altitude, 144
  - angular rates, 143
  - course, 146, 159
  - ground speed, 146
  - heading, 159
  - position, 146, 159
  - roll and pitch angles, 145, 157
  - wind, 159
- Euler angles, 12, 30
  - heading (yaw), 14
  - pitch, 14
  - roll, 15
- fillet path, 211
- flat-earth model, 252, 260
- flight-path angle
  - air-mass referenced, 23, 24
  - inertial referenced, 21, 176, 178
- focal length, 249
- forces
  - drag, 40, 44–46
  - gravitation, 40
  - lift, 40, 44, 46
- gimbal
  - azimuth angle, 248, 252
  - dynamics, 250
  - elevation angle, 248, 252
- GPS, 131, 159

- ground speed, 19, 23, 24, 57, 138
- half plane, 210, 213, 223
- heading angle, 14, 21, 24
- inertia matrix, 35
- Kalman filter
  - basic explanation, 147
  - derivation, 149
  - extended Kalman filter, 154
  - geolocation, 253
  - position, course, wind, heading, 159
  - roll and pitch, 157
- kinematics
  - guidance model, 179
  - position, 31, 64, 176, 177
  - rotation, 32, 64
- lateral-directional motion, 50, 70, 80, 97
- linear quadratic regulator, 110
- linearization, 79
- load factor, 176
- longitudinal motion, 43, 74, 83, 104
- low-pass filter, 142, 255
- minimum turning radius, 68
- mixing matrix, 277
- normalized line of sight vector, 250
- path following, 189
  - 3-D, 199
  - 3-D helical, 202
  - 3-D straight-line, 201
  - orbit, 195
  - straight-line, 190
- phugoid Mode, 88
- PID: digital implementation, 329
- pitch angle, 14
- pitching moment, 44, 45
- pitot tube, 127
- precision landing, 261
- proportional navigation, 261, 263
- rapidly exploring random trees, 236
  - 3-D terrain, 241
- point-to-point algorithm, 238, 297
- smoothing algorithm, 238
- using Dubins paths, 242
- rate gyros, 120, 143, 157
- right-handed rotation, 11
- roll angle, 15
- roll Mode, 89
- rotation matrices, 10
  - body to gimbal, 248
  - body to gimbal-1, 248
  - body to stability, 17
  - body to wind, 18
  - gimbal to camera, 249
  - gimbal-1 to gimbal, 248
  - stability to wind, 18
  - vehicle to body, 16
  - vehicle to vehicle-1, 14
  - vehicle-1 to vehicle-2, 15
  - vehicle-2 to body, 16
  - rudder, 42
- short-period mode, 87
- side-slip angle, 73
- sideslip angle, 17, 20, 41, 59
- simulation model, 4, 64, 347
- spiral-divergence mode, 90
- stall, 45, 46
- State Variables, 269
- state variables, 29
- state-space models
  - lateral-directional, 80
  - longitudinal, 83
- successive loop closure, 95
- time to collision, 258
- total energy control, 109
- transfer functions
  - aileron to roll, 71
  - airspeed to altitude, 76
  - elevator to pitch, 75, 104
  - pitch to altitude, 76, 106
  - roll to course, 72, 99
  - roll to heading, 73
  - rudder to sideslip, 73
  - throttle to airspeed, 79, 107
  - trim, 68, 79
- vector field
  - orbit, 198
  - straight-line, 194
- Voronoi path planning, 230

waypoint configuration, 223  
waypoint path, 209  
wind gusts, 57  
    Dryden model, 58  
wind speed, 19, 23, 24, 57  
wind triangle, 21, 23, 162  
  
yaw-damper, 91