Navigation, Guidance, and Control of

Small and Miniature Air Vehicles

Randal W. Beard Timothy W. McLain
Brigham Young University

Contents

Preface **vii**

1 Introduction **1**
1.1 System Architecture 1
1.2 Nomenclature 4
1.3 Notation 4
1.4 Design Project 5

2 Coordinate Frames **7**
2.1 Rotation Matrices 8
2.2 MAV Coordinate Frames 11
2.3 Differentiation of a Vector 20
2.4 Chapter Summary 22
2.5 Design Project 22

3 Kinematics and Dynamics **27**
3.1 State Variables 27
3.2 Kinematics 29
3.3 Rigid Body Dynamics 30
3.4 Chapter Summary 35
3.5 Design Project 35

4 Forces and Moments **37**
4.1 Gravitational Forces 37
4.2 Aerodynamic Forces 38
4.3 Propulsion Forces 47
4.4 Atmospheric Disturbances 49

4.5 Chapter Summary 52
4.6 Design Project 53

5 Linear Design Models **55**
5.1 Summary of Nonlinear Equations of Motion 55
5.2 Trim Conditions 58
5.3 Transfer Functions 65
5.4 Linear State Space Models 72
5.5 Chapter Summary 84
5.6 Design Project 84

6 Autopilot Design Using Successive Loop Closure **89**
6.1 Successive Loop Closure 89
6.2 Saturation Constraints and Performance 91
6.3 Lateral-directional Autopilot 93
6.4 Longitudinal Autopilot 100
6.5 PID Loop Implementation 108
6.6 Chapter Summary 111
6.7 Design Project 113

7 Nonlinear Design Models **115**
7.1 Autopilot Model 115
7.2 Kinematic Model of Controlled Flight 116
7.3 Kinematic Guidance Models 119
7.4 Dynamic Guidance Model 122
7.5 Chapter Summary 123
7.6 Design Project 124

8 Sensors for MAVs **125**
8.1 Accelerometers 126
8.2 Rate Gyros 128
8.3 Pressure Sensors 131
8.4 Magnetometers 135
8.5 GPS 136
8.6 Chapter Summary 138
8.7 Design Project 138

9 State Estimation **139**
9.1 Base Maneuver . . . 139
9.2 Low Pass Filters . . . 140
9.3 State Estimation by Inverting the Sensor Model . . . 141
9.4 Dynamic Observer Theory . . . 146
9.5 Derivation of the Kalman Filter . . . 148
9.6 Attitude Estimation . . . 156
9.7 GPS Smooting . . . 159
9.8 Chapter Summary . . . 161
9.9 Design Project . . . 163

10 Straight-Line and Orbit Following **165**
10.1 Straight-Line Path Following . . . 166
10.2 Orbit Following . . . 171
10.3 Chapter Summary . . . 174
10.4 Design Project . . . 177

11 Path Manager **179**
11.1 Transitions between waypoints . . . 179
11.2 Dubins Paths . . . 187
11.3 Chapter Summary . . . 198
11.4 Design Project . . . 199

12 Path Planning **201**
12.1 Point-to-Point Algorithms . . . 202
12.2 Coverage Algorithms . . . 219
12.3 Chapter Summary . . . 223
12.4 Design Project . . . 223

13 Vision Guided Navigation **225**
13.1 Gimbal and Camera Frames and Projective Geometry . . . 226
13.2 Gimbal Pointing . . . 228
13.3 Geolocation . . . 230
13.4 Estimating Target Motion in the Image Plane . . . 233
13.5 Time to Collision . . . 237
13.6 Precision Landing . . . 239

13.7 Chapter Summary 245
13.8 Design Project 246

A Animations in Simulink **247**
A.1 Handle Graphics in Matlab 247
A.2 Animation Example: Inverted Pendulum 248
A.3 Animation Example: Spacecraft using lines 251
A.4 Animation Example: Spacecraft using vertices and faces 256

B Modeling in Simulink using S-functions **259**
B.1 Example: Second Order Differential Equation 259

C Airframe Parameters **265**
C.1 Zagi 265

D Trim and Linearization in Simulink **269**

E Essentials from Probability Theory **271**

F Sensor Parameters **273**
F.1 Rate Gyros 273
F.2 Accelerometers 273
F.3 Pressure Sensors 274
F.4 Magnetometers 274
F.5 GPS 274

G Useful Formulas and other Information **277**
G.1 Conversion from knots to mph 277
G.2 Density of Air 277

Bibliography **288**

Preface

Unmanned air systems (UASs) are playing increasingly prominent roles in defense programs and defense strategy around the world. Technology advancements have enabled the development of large UASs (e.g., Global Hawk, Predator) and the creation of smaller, increasingly capable UASs. As recent conflicts have demonstrated, there are numerous military applications for UASs including reconnaissance, surveillance, battle damage assessment, and communications relays.

Civil and commercial applications are not as well developed, although potential applications are extremely broad in scope. Possible applications for UAS technology include environmental monitoring (e.g., pollution, weather, and scientific applications), forest fire monitoring, homeland security, border patrol, drug interdiction, aerial surveillance and mapping, traffic monitoring, precision agriculture, disaster relief, ad-hoc communications networks, and rural search and rescue. For many of these applications to develop to maturity, the reliability of UASs will need to increase, their capabilities will need to be extended further, their ease of use will need to be improved, and their cost will have to come down. In addition to these technical and economic challenges, the regulatory challenge of integrating UASs into the national and international air space needs to be overcome.

UASs can generally be divided into two categories: fixed-wing aircraft, and hovercraft. Both types of aircraft have distinctive characteristics that make autonomous behavior difficult to design. In this book we focus exclusively on fixed-wing aircraft which can be roughly categorized by size. We use the term *small UAS (sUAS)* to refer to the class of fixed wing aircraft with wing span between 4-10 feet. Small UASs are usually gas powered and typically require a runway for take-off and landing, although the Boeing ScanEagle, which uses a catapult for take-off and a skyhook for recovery, is a notable exception. Small UASs are typically designed to operate on the order of 10-12 hours, with payloads of approximately 10-50 pounds.

The term *miniature air vehicle (MAV)* will be used to refer to the class of fixed-wing aircraft with wingspan less than 5 feet. Miniature UASs are typically battery powered, hand launched, and belly landed, and therefore do not require a runway for take-off or landing. They are designed to operate for times that range from 20 minutes to a couple of hours. Payloads range from several

ounces to 2 pounds. The small payload severely restricts the sensor suite that can be placed on MAVs, and also restricts the computer that can be put on-board. These restrictions pose interesting challenges for the design of autonomous modes of operation. While many of the concepts described in this book are also applicable to larger UASs, the primary focus of the book is on the challenges that are inherent with limited-payload sUASs and MAVs.

This textbook was inspired by our desire to teach a course to our graduate students that prepared them to do work in the area of cooperative control for UASs. Most of our students come from a background in Electrical and Computer Engineering, Mechanical Engineering, and Computer Science. Very few of them have had courses in aerodynamics, and the ECE and CS students have not generally had courses in kinematics, dynamics, or fluids. However, most of our students have had courses in signals and systems, feedback control, robotics, and computer vision.

There are a large number of textbooks that cover aircraft dynamics and control, however most of them assume a background in aeronautics and assume that the student has not had exposure to feedback control. Therefore, textbooks like [1, 2, 3, 4, 5, 6] present the discussion on aerodynamic forces without discussing basic ideas in fluid mechanics and aeronautics. On the other hand, they typically include a detailed introduction to introductory feedback control concepts. The textbook [7] is much more in line with what we wanted to teach our students, but the focus of that text is on stability augmentation systems as opposed to autonomous operations. Autonomous operations require more than a simple autopilot; they require autonomous take-off and landing, path planning, and path following operations, integrated with higher level decision making processes. To our knowledge, another textbook covering aircraft dynamic models, low level autopilot design, state estimation, and high-level path planning is not available. Our hope is that this book fits that gap. Our target audience is Electrical and Computer Engineering, Mechanical Engineering, and Computer Science students who have had a course in introductory feedback control or robotics. We hope that the textbook is also interesting to Aeronautical Engineers seeking a background in autonomous systems.

In writing this book, we have been influenced by the exceptional textbook by Rugh [8]. In the spirit of [8], our objective was to have 14 chapters, each of which could be covered in three, one-hour lectures. While some of the chapters are longer than we originally hoped, we feel that the book can be covered in a one semester course. The additional material will allow the instructor some flexibility in the topics that are covered.

One of the unique features of the book is the associated design project. As we have taught this course, we have evolved from assigning pencil and paper homework assignments, to computer simulation assignments. We have found that students are more engaged in the material and tend to understand it better when they implement the concepts in a computer simulation.

When we teach the course, we have our students develop a complete end-to-end flight simulator that includes realistic flight dynamics, sensor models, autopilot design, and path planning. By the end of the course, they have implemented each piece of the puzzle and therefore understand how each piece fits together. In addition, they understand the inner workings of a fairly sophisticated flight simulation package that can be used in their future research projects.

RWB: Fix this statement when the design exercises are complete.

Unfortunately, developing an end-to-end simulator requires the use of a compiled language like C/C++. For our course at BYU, knowledge of C/C++ is a stated prerequisite. However we understand that this requirement would not be welcome by many instructors who would otherwise want to use the textbook. While Matlab/Simulink cannot be used effectively to develop an end-to-end simulator, it can be used to implement the pieces separately with increasing levels of abstraction as the book progresses. Accordingly, we have written the project exercises so that they can be done in either C/C++ or Matlab/Simulink. The accompanying CD-ROM contains the appropriate files for both methods. The ideal approach may be to work the project in Matlab/Simulink, and then port the solution to C/C++. The Appendices describe both simulation packages in detail.

Chapter 1

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 need to be familiar with a wide range of topics that include 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 (MAVs).

In the development of the topics, we have in mind the software architecture shown in Figure 1.1. The block labeled *Unmanned Air System* in Figure 1.1 is the six degree-of-freedom physical airframe that responds to servo command inputs (elevator, aileron, rudder, and throttle) and wind and other disturbances. The mathematical models that are required to understand fixed-wing flight are complicated and are covered in Chapters 2–**??**. 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 sensors measure information with respect to the body frame, and the actuators exert forces and torques in the body frame. In Chapter 3 we describe basic rigid body kinematics and dynamics. The mathematical models developed in Chapter 3 are applicable to any physical object. In Chapter 4 we describe the aerodynamic forces and moments that act on fixed-wing aircraft. Chapter **??** completes the development of the six DOF model by combining the results of Chapters 3 and 4. The six DOF model is complicated and cumbersome to work with, and the coefficients of the model are often not explicitly known. In addition, the level of fidelity offered by the six DOF

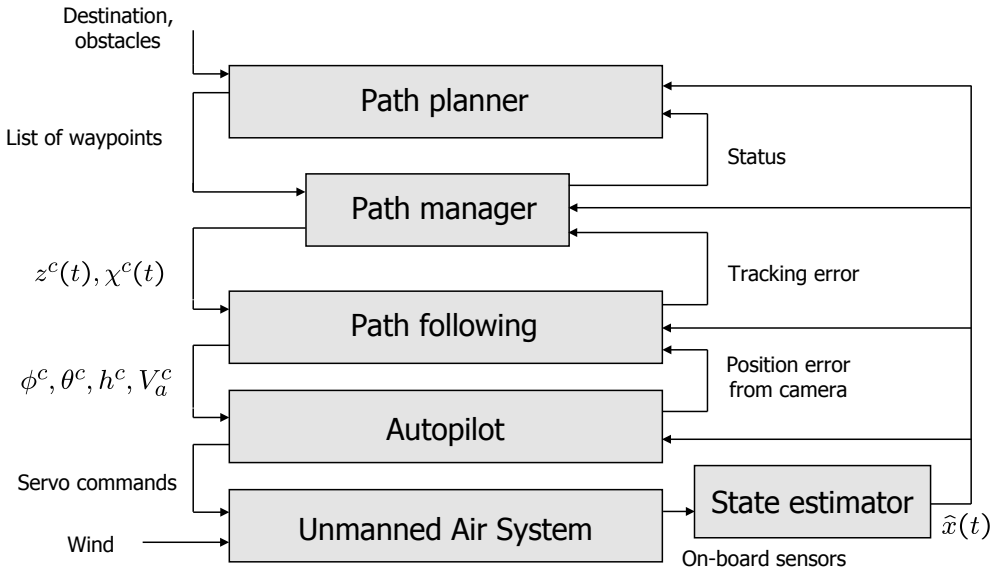


Figure 1.1: The system architecture that will be assumed throughout the book. The path planner produces straight line paths through obstacle fields. The path manager switches between orbit following and straight line path following to maneuver along the waypoint paths. The path following block produces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

model is often not needed to design higher level path planning and trajectory following algorithms. In fact, the requirement for computational efficiency precludes the use of the full six DOF model. Therefore, Chapter **??** also describes several nonlinear lower fidelity models that can be used to effectively design higher levels of behavior. Linear models that describe small deviations from trim can also be derived. In Chapter D we describe trim conditions, and how they can be computed numerically from the six DOF model. Chapter **??** derives the linear models in both transfer function and state space forms.

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 heading. Chapter 6 introduces the standard technique of successive loop closure to design the autopilots. 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 on-board sensors which include accelerometers, rate gyros, pressure sensors, and GPS. A mathematical model of the sensors is given in Chapter 8. A description of several state estimation techniques that are effective for MAVs is described in Chapter 9.

One of the primary challenges with MAVs is flight in wind conditions. Since airspeeds in the range of 20-40 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. Airframes are designed to fly efficiently at a particular airspeed. Therefore, one of the control objectives will typically be to maintain a constant airspeed. Therefore, for a constant airspeed, head winds will decrease the ground speed, and tail winds will increase the ground speed. Since the ground speed is constantly changing, we have found that traditional trajectory tracking methods used in robotics do not work well for MAVs. The primary difficulty is the time dependance of the trajectory which does not take into account the variations in ground speed. On the other hand, path following methods that simply maintain the vehicle on a desired path have proven to be very effective in flight tests. In Chapter 10 we describe a path following method which is labeled *Path following* in Figure 1.1. We will focus exclusively on straight-line paths and circular orbits.

The block labeled *Path manager* in Figure 1.1 is a finite state machine that converts straight-line waypoint paths into a sequence of straight-line paths and circular orbits, making it possible to significantly simplify the path planning problem at the highest level. The *Path planner* produces a sequence of waypoints that maneuver the MAV through obstacle fields. Chapter **??** describes the *Path planner*, while Chapter 11 describes the *Path manager*. 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 with a sensor footprint, with a set of potential no-go regions.

Almost all applications involving MAVs require the use of an on-board electro-optical (E/O) video camera. The objective of the camera is to provide visual information to the end user. However, since payload is limited, 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 geo-location and vision-based landing. The geo-location problem is to use a sequence of images as well as the on-board sensors to estimate the world coordinates of objects on the ground. The vision-based landing problem uses the video sequence to guide the MAV to a target identified in the image plane. We feel that these problems will enable further investigations in vision based control of MAVs.

In Chapter 13 we use the software architecture shown in Figure 1.2, where the *Path planner* 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. Modularity of the architecture is one of its most appealing features.

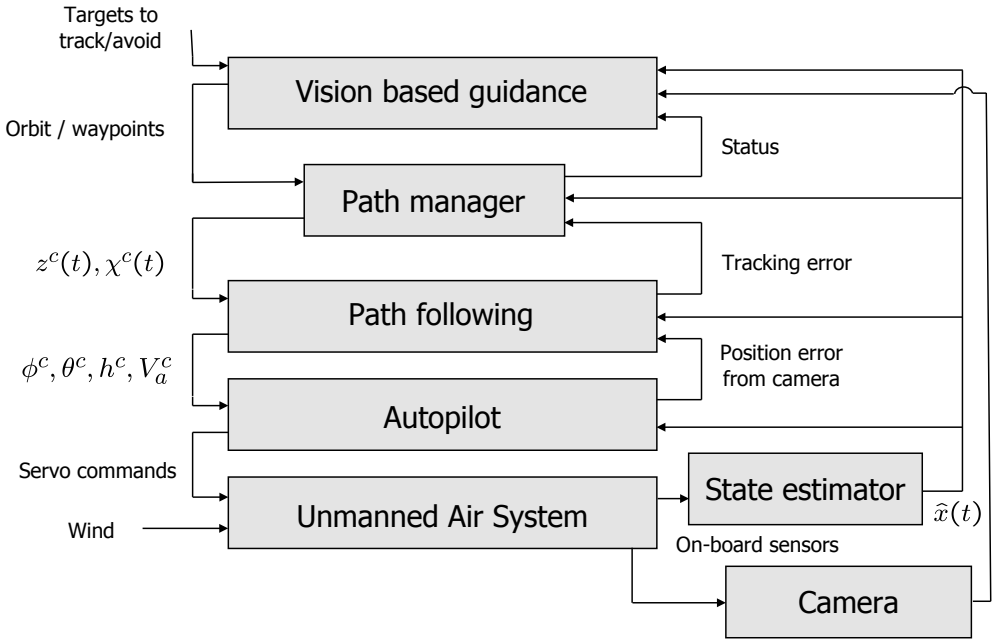


Figure 1.2: System architecture for vision based navigation, guidance, and control. A video camera is added as an additional sensor and the path planner has been replaced with a vision based guidance block.

1.2 Nomenclature

RWB:

Define:

MAV - Miniature Air Vehicle - we will use MAV as a generic term for small and miniature aircraft - an alternative use of the MAV acronym is micro air vehicle, meaning a vehicles with wingspan less then 12 inches. Many of the concepts discussed in this book will certainly be applicable to micro air vehicles, but there are differences as well (spell out a few).

UAS - Unmanned Air Systems (include ground station, sensors, etc)

UAV - typically only refers to the vehicle - may or may not be autonomous

1.3 Notation

RWB: - use an arrow for unit vectors

- use a subscript like u_* to denote a trim condition

- use $\bar{x} = x - x_*$ to denote deviation from trim. This makes the expression of linear dynamics

clean (it is currently this way)

- use $\hat{x}$ to denote state estimate of x
- use $\tilde{x}$ to denote estimation error.
- denote control error by e_x
- unit vectors are denoted by a bold typeface

1.4 Design Project

RWB:

Describe the design project and how it fits with the rest of the book. Describe how every chapter is designed to allow the students to implement one piece of the project.

Refer the reader to the Appendixes for tutorials on the different simulation packages.

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 you, the reader, to internalize the material that is presented.

The design project involves building a MAV flight simulator from the ground up. Two approaches are suggested: 1) Using the public domain flight simulator *Aviones* available for download on Sourceforge.net, or 2) Using Matlab/Simulink. At the conclusion of each chapter, assignments are given that will direct you in the completion of the various components of the simulator.

Chapter 2

Coordinate Frames

In the course of 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, and 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 that are used to describe the position and orientation of the aircraft and its sensors, and the transformation between these coordinate systems. It is necessary to use several different coordinate systems for the following reasons:

- Newton's equations of motion are derived relative to a fixed, inertial reference frame. However, motion is most easily described in a body-fixed frame.
- Aerodynamics 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: rotations and translations. Section 2.1 describes rotation matrices and their use in transforming between coordinate frames. Section 2.2 describes the specific coordinate frames used for micro air vehicle systems. In Section 2.3 we derive and expression for differentiating a vector in a rotating and translating frame.

2.1 Rotation Matrices

We begin by considering the two coordinate frames shown in Figure 2.1. The vector $\mathbf{p}$ can

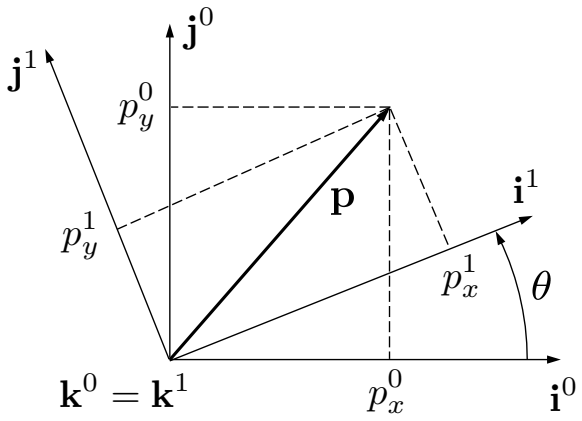


Figure 2.1: Rotation in 2D

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.

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

$$\mathbf{p}^1 = R_0^1 \mathbf{p}^0, \tag{2.1}$$

where

$$R_0^1 \triangleq \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The notation 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

$$R_0^1 \triangleq \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix},$$

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

$$R_0^1 \triangleq \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{pmatrix}.$$

As pointed out in [7], the negative sign on the $\sin$ term appears above the line with only ones and zeros.

The matrix R_0^1 in the above equations are examples of a more general class of *orthonormal* rotation matrices that have the following properties:

P.1. $(R_a^b)^{-1} = (R_a^b)^T = R_b^a$.

P.2. $R_b^c R_a^b = R_a^c$.

P.3. $\det\left(R_a^b\right) = 1$,

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

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

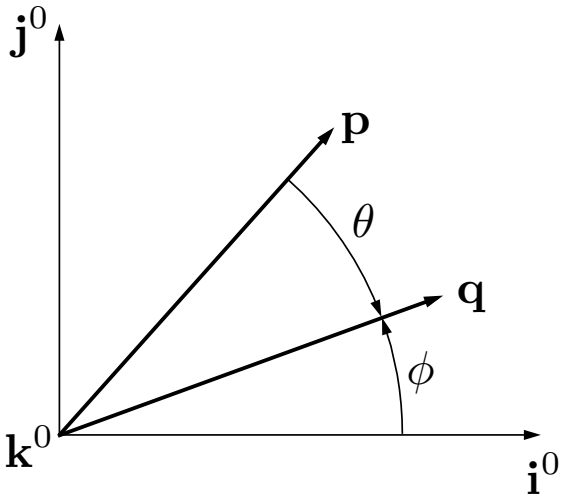


Figure 2.2: Rotation of $\mathbf{p}$ about the $\mathbf{k}^0$-axis.

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

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

and

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

where $p \triangleq |\mathbf{p}| = q \triangleq |\mathbf{q}|$. Combining (2.2) and (2.3) gives

$$\begin{aligned}\mathbf{p} &= \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \mathbf{q} \\ &= (R_0^1)^T\mathbf{q} \end{aligned}$$

and

$$\mathbf{q} = R_0^1\mathbf{p}.$$

In this case, the rotation matrix R_0^1 can be interpreted as a left-handed rotation of the vector $\mathbf{p}$ through the angle θ to a new vector $\mathbf{q}$ in the same reference frame. Notice that a right-handed

rotation of a vector (in this case from $\mathbf{q}$ to $\mathbf{p}$) can be obtained by using $(R_0^1)^T$. 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. In our treatment of MAVs, we will use rotation matrices primarily to rotate coordinate frames as described in the following section.

2.2 MAV Coordinate Frames

For MAVs, there are several coordinate systems that 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. 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 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.

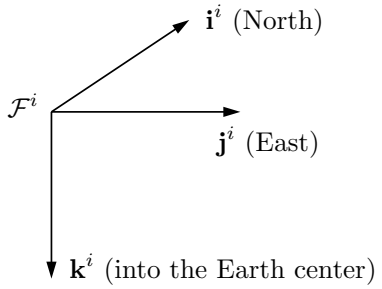


Figure 2.3: The inertial coordinate frame. The x-axis points North, the y-axis points East, and the z-axis points into the Earth.

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

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

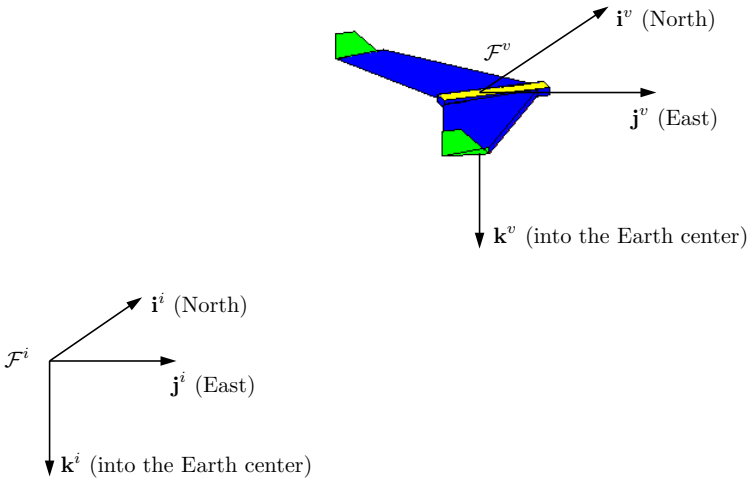


Figure 2.4: The vehicle coordinate frame. The x-axis points North, the y-axis points East, and the z-axis points into the Earth.

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

The origin of the vehicle-1 frame is identical to the vehicle frame, i.e, the the center of mass. However, $\mathcal{F}^{v1}$ is positively rotated about $\mathbf{k}^v$ by the yaw angle ψ so that if the airframe is not rolling or pitching, then $\mathbf{i}^{v1}$ would point 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 2.5.

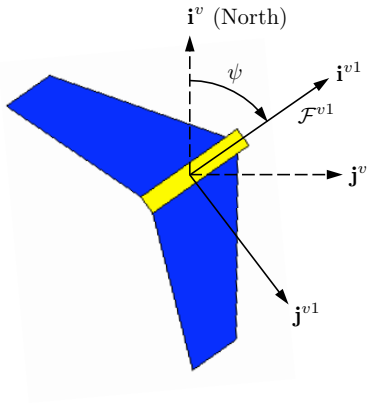


Figure 2.5: The vehicle-1 frame. If the roll and pitch angles are zero, then the x-axis points out the nose of the airframe, the y-axis points out the right wing, and the z-axis points into the Earth.

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

$$\mathbf{p}^{v1} = R_v^{v1}(\psi)\mathbf{p}^v,$$

where

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

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

The origin of the vehicle-2 frame is again the center of mass and is obtained by rotating the vehicle-1 frame in a right-handed rotation about the $\mathbf{j}^{v1}$ axis by the pitch angle θ. If the roll angle is zero, then $\mathbf{i}^{v2}$ points out the nose of the airframe, $\mathbf{j}^{v2}$ points out the right wing, and $\mathbf{k}^{v2}$ points out the belly, a shown in Figure 2.6.

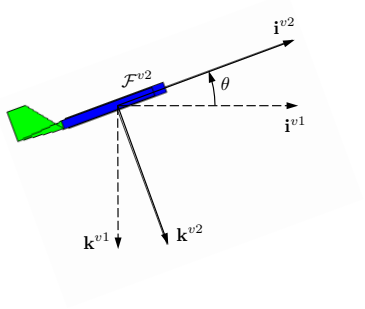


Figure 2.6: The vehicle-2 frame. If the roll angel is zero, then the x-axis points out the nose of the airframe, the y-axis points out the right wing, and the z-axis points out the belly.

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

$$\mathbf{p}^{v2} = R_{v1}^{v2}(\theta)\mathbf{p}^{v1},$$

where

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

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 ϕ. Therefore, the origin is the center-of-gravity, $\mathbf{i}^{b}$ 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 2.7.

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

$$\mathbf{p}^{b} = R_{v2}^{b}(\phi)\mathbf{p}^{v2},$$

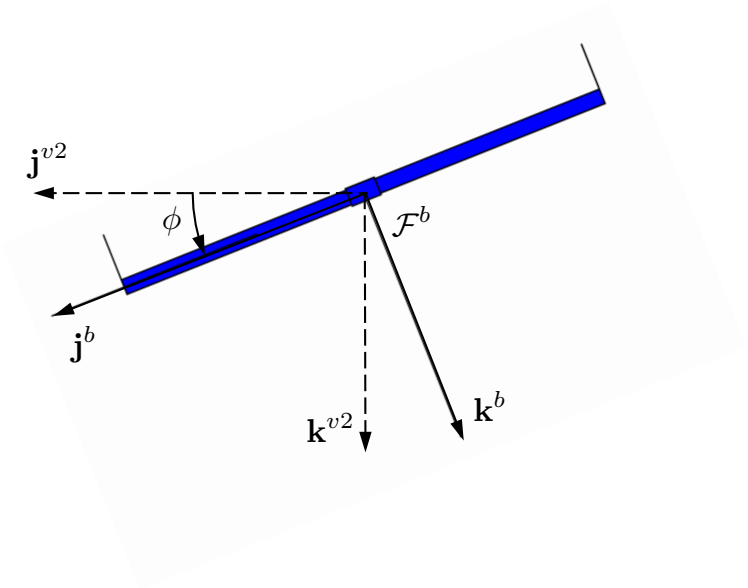


Figure 2.7: The body frame. The x-axis points out the nose of the airframe, the y-axis points out the right wing, and the z-axis points out the belly.

where

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

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

$$\begin{aligned} R_{v}^{b}(\phi,\theta,\psi) &= R_{v2}^{b}(\phi)R_{v1}^{v2}(\theta)R_{v}^{v1}(\psi) \qquad (2.4)\\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} c\theta c\psi & c\theta s\psi & -s\theta \\ s\phi s\theta c\psi - c\phi s\psi & s\phi s\theta s\psi + c\phi c\psi & s\phi c\theta \\ c\phi s\theta c\psi + s\phi s\psi & c\phi s\theta s\psi - s\phi c\psi & c\phi c\theta \end{pmatrix}, \qquad (2.5) \end{aligned}$$

where $c\phi \triangleq \cos\phi$ and $s\phi \triangleq \sin\phi$. The angles ϕ, θ, and ψ are commonly referred to as Euler angles. The use of Euler angles is one approach for representing the orientation of a body in three-dimensional space. The rotation sequence ψ-θ-ϕ is commonly used for aircraft and is just one of several Euler angle systems in use [9].

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 airframe relative to the surrounding air as the relative wind or the relative wind vector, denoted $\mathbf{V}_a$. The magnitude of the relative wind vector is typically 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 wind vector. This angle is called the angle-of-attack and is denoted by α. As shown in Figure 2.8, the angle of attack is defined as a **left**-handed rotation about $\mathbf{j}^b$ and is such that $\mathbf{i}^s$ aligns with the projection of $\mathbf{V}_a$, onto the plane spanned by $\mathbf{i}^b$ and $\mathbf{k}^b$. The need for a left handed rotation is caused by the definition of positive angle of attack, which is positive for a right-handed rotation from the stability frame $\mathbf{i}^s$ direction to the body frame $\mathbf{i}^b$ direction.

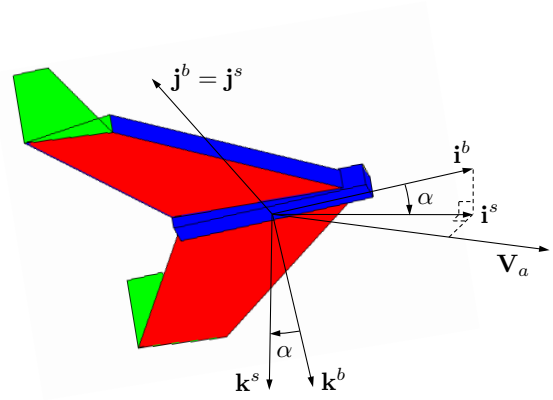


Figure 2.8: The stability frame. The x-axis points along the projection of the relative wind vector onto the $x-z$ plane of the body frame, the y-axis is identical to the y-axis of the body frame, and the z-axis is constructed to make a right handed coordinate system. Note that angle-of-attack is defined as a **left**-handed rotation about the body y-axis.

Since α is given by a left-handed rotation, the transformation from $\mathcal{F}^b$ to $\mathcal{F}^s$ is given by

$$\mathbf{p}^s = R_b^s(\alpha)\mathbf{p}^b,$$

where

$$R_b^s(\alpha) = \begin{pmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{pmatrix}.$$

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

When the airframe rolls, gravity will cause it to fall, which implies that there will be a component of the velocity vector that does not lie in the $\mathbf{i}^b - \mathbf{k}^b$ plane. The angle between the velocity vector and the $\mathbf{i}^b - \mathbf{k}^b$ plane is called the side-slip angle and is denoted by β. As shown in Figure 2.9, the wind frame is obtained by rotating the stability frame by a right-handed rotation of β about $\mathbf{k}^s$. The unit vector $\mathbf{i}^w$ is aligned with the relative wind vector $\mathbf{V}_a$.

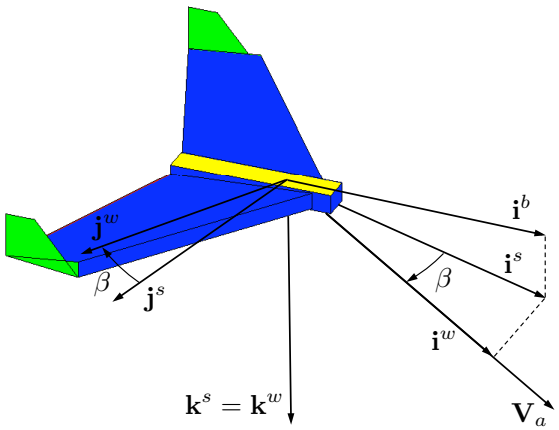


Figure 2.9: The wind frame. The x-axis points along the relative wind vector.

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

$$\mathbf{p}^w = R_s^w(\beta)\mathbf{p}^s,$$

where

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

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

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

Alternatively, we have

$$R_w^b(\alpha,\beta) = (R_b^w)^T(\alpha,\beta) = \begin{pmatrix} \cos\beta\cos\alpha & -\sin\beta\cos\alpha & -\sin\alpha \\ \sin\beta & \cos\beta & 0 \\ \cos\beta\sin\alpha & -\sin\beta\sin\alpha & \cos\alpha \end{pmatrix}.$$

When developing the dynamic equations of motion for the 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, are dependent on the velocity of the airframe relative to the surrounding air. When wind is not present, these velocities are the same. Unfortunately, wind is almost always present with MAVs and we must carefully distinguish between airspeed, represented by the relative wind vector $\mathbf{V}_a$, and the ground speed, represented by the velocity with respect to the inertial frame $\mathbf{V}_g$. These velocities are related by the expression

$$\mathbf{V}_a = \mathbf{V}_g - \mathbf{V}_w. \tag{2.6}$$

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

The MAV velocity, $\mathbf{V}_g$, can be expressed in the body frame in terms of components along the $\mathbf{i}^b$, $\mathbf{j}^b$, 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, we can write an expression for the wind velocity in the body-frame

$$\mathbf{V}_w^b = \begin{pmatrix} u_w \\ v_w \\ w_w \end{pmatrix}.$$

Noting that the relative wind vector $\mathbf{V}_a$ (keep in mind that this is the velocity of the MAV relative to the wind) in the wind frame can be written as

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

and that in the body frame it is given by

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

the following expression results

$$\mathbf{V}_a^b = \begin{pmatrix} u_r \\ v_r \\ w_r \end{pmatrix} = R_w^b \begin{pmatrix} V_a \\ 0 \\ 0 \end{pmatrix}$$
$$= \begin{pmatrix} \cos\beta\cos\alpha & -\sin\beta\cos\alpha & -\sin\alpha \\ \sin\beta & \cos\beta & -\sin\beta\sin\alpha \\ \cos\beta\sin\alpha & 0 & \cos\alpha \end{pmatrix} \begin{pmatrix} V_a \\ 0 \\ 0 \end{pmatrix}.$$

This implies that

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

Inverting this relationship gives

$$\begin{aligned} V_a &= \sqrt{u_r^2 + v_r^2 + w_r^2} \\ \alpha &= \tan^{-1}\left(\frac{w_r}{u_r}\right) \\ \beta &= \tan^{-1}\left(\frac{v_r}{\sqrt{u_r^2 + w_r^2}}\right). \end{aligned} \tag{2.8}$$

These expressions for V_a, α, and β are commonly used to calculate the aerodynamic forces and moments acting on the vehicle.

2.2.8 Heading, Course, and Crab Angles

For small UAVs, 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 larger than the wind speed. Having introduced the concepts of reference frames, airframe velocity, wind velocity, and the relative wind vector, we can discuss some important definitions relating to the navigation of MAVs.

Figure 2.10 shows a MAV following a ground track represented by the dashed line. The North direction is indicated by the $\mathbf{i}^i$ vector and the direction that the MAV is pointed is shown by the $\mathbf{i}^b$ vector, which is fixed in the direction of the body x-axis. For level flight, the heading, or yaw angle ψ, is the angle between $\mathbf{i}^i$ and $\mathbf{i}^b$ and defines the direction the MAV is pointed. The direction that the MAV is traveling with respect to the surrounding air mass is given by the relative wind vector $\mathbf{V}_a$. In steady, level flight, $\mathbf{V}_a$ is typically aligned with $\mathbf{i}^b$ meaning the sideslip angle β is zero.

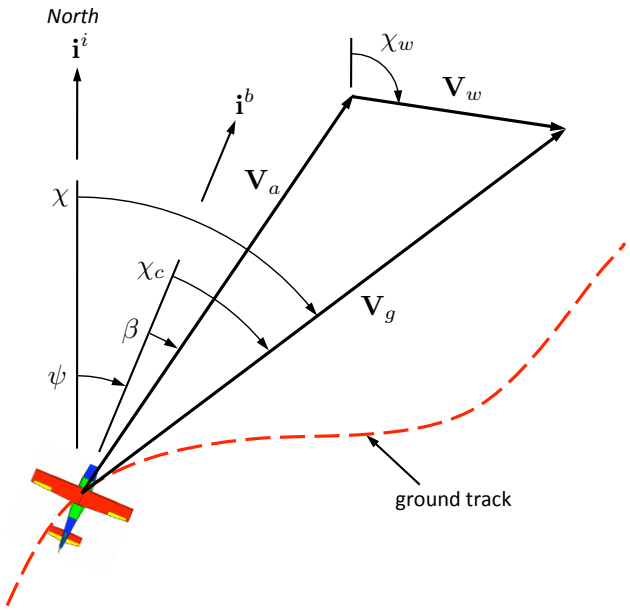


Figure 2.10: Heading is the direction that the MAV is pointed. Course is the direction of travel relative to the earth's surface. The crab angle is the difference between course and heading. In the absence of wind, the crab angle is zero.

The direction that the MAV 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 is called the course, χ. The definition of course is the line of flight taken by an aircraft. The ground track is the projection of the line of flight onto the surface of the earth. If there is a constant ambient wind, the aircraft will need to crab into the wind to follow a ground track that is not aligned with the wind. The *crab angle* χ_c is defined as the difference between the course angle and the heading

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

To do: Work on figure and equations in estimation chapter to bring them into agreement. Note that those equations assume $\beta = 0$.

Keeping in mind that $|\mathbf{V}_a| = V_a$, $|\mathbf{V}_g| = V_g$, and $|\mathbf{V}_w| = V_w$, the relationship between windspeed, groundspeed, and airspeed can be obtained from the geometry shown in Figure 2.11. Let V_w is the windspeed and V_g be the groundspeed, and let χ_w be the direction of the wind, then from the law of cosines we have

$$V_g^2 = V_a^2 + V_w^2 + 2V_aV_w\cos(\psi + \beta - \chi_w).$$

Using the law of cosines, we can also express the groundspeed in terms of the crab angle as

$$V_w^2 = V_a^2 + V_g^2 - 2V_aV_g\cos(\chi_c - \beta),$$

or in terms of the course and wind direction as

$$V_a^2 = V_g^2 + V_w^2 - V_gV_w\cos(\chi_w - \chi).$$

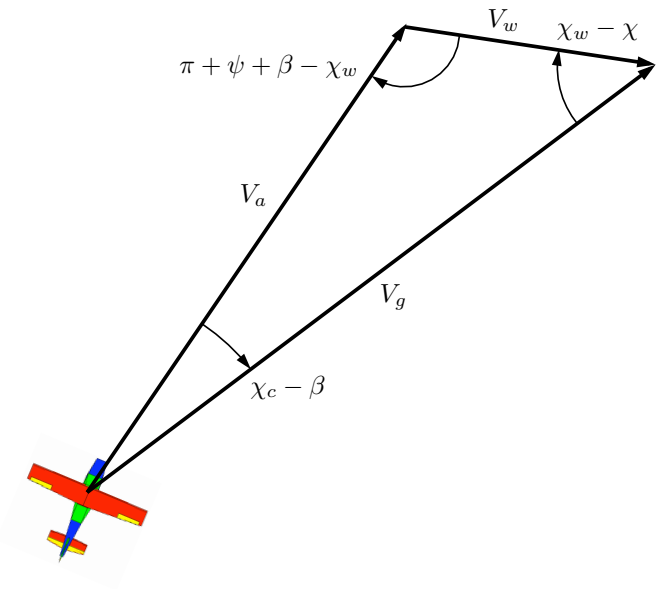


Figure 2.11: The wind triangle showing the relationship between airspeed V_a, windspeed V_w, groundspeed V_g, course χ, wind direction χ_w, crab angle χ_c, heading ψ, and sideslip angle β.

2.3 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 $\mathcal{F}^i$. 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

$$\mathbf{p} = p_x\mathbf{i}^b + p_y\mathbf{j}^b + p_z\mathbf{k}^b. \tag{2.9}$$

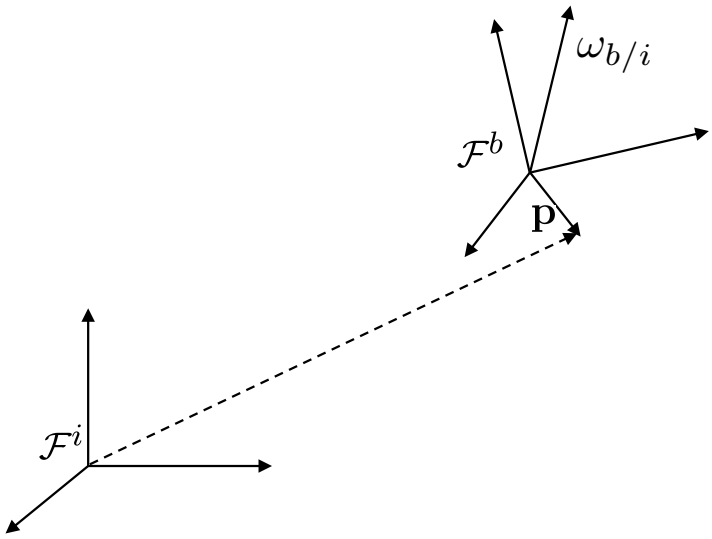


Figure 2.12: A vector in a rotating reference frame.

The time derivative of $\mathbf{p}$ with respect to frame $\mathcal{F}^i$ can be found by differentiating (2.9) as

$$\frac{d}{dt_i}\mathbf{p} = \dot{p}_x\mathbf{i}^b + \dot{p}_y\mathbf{j}^b + \dot{p}_z\mathbf{k}^b + p_x\dot{\mathbf{i}}^b + p_y\dot{\mathbf{j}}^b + p_z\dot{\mathbf{k}}^b, \tag{2.10}$$

where d/dt_i represents time differentiation with respect to the inertial frame. The first three terms on the right-hand side of (2.10) 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

$$\frac{d}{dt_b}\mathbf{p} = \dot{p}_x\mathbf{i}^b + \dot{p}_y\mathbf{j}^b + \dot{p}_z\mathbf{k}^b. \tag{2.11}$$

The next three terms on the right-hand side of (2.10) 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 [10]

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

We can rewrite the last three terms of (2.10) as

$$\begin{aligned}p_x\dot{\mathbf{i}}^b + p_y\dot{\mathbf{j}}^b + p_z\dot{\mathbf{k}}^b &= p_x(\boldsymbol{\omega}_{b/i} \times \mathbf{i}^b) + p_y(\boldsymbol{\omega}_{b/i} \times \mathbf{j}^b) + p_z(\boldsymbol{\omega}_{b/i} \times \mathbf{k}^b) \\ &= \boldsymbol{\omega}_{b/i} \times \mathbf{p}.\end{aligned} \tag{2.12}$$

Combining results from (2.10), (2.11), and (2.12) we obtain the desired relation

$$\frac{d}{dt_i}\mathbf{p} = \frac{d}{dt_b}\mathbf{p} + \boldsymbol{\omega}_{b/i} \times \mathbf{p}, \tag{2.13}$$

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.

2.4 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 Euler angles, ψ, θ, and ϕ, as a means to rotate from the inertial coordinate frame to the body frame fixed in the orientation of the MAV. We have also introduced the angle of attack α and the sideslip angle β to describe the relative orientation of the body frame, the stability frame, and the wind frame. An understanding of these orientations is essential to the derivation of equations of motion and the modeling of aerodynamic forces involved in MAV flight.

Notes and References

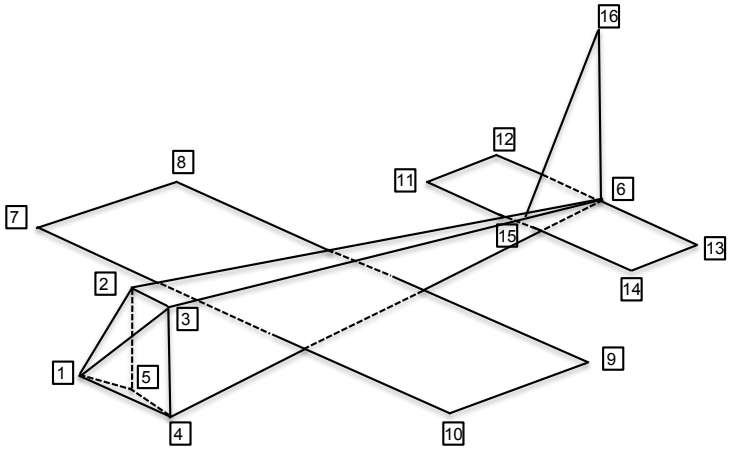
There are many references on coordinate frames and rotations matrices. A particularly good overview of rotation matrices is [11]. An overview of attitude representations is [12, 9]. The definition of the different aircraft frames can be found in [4, 1, 7, 13]. A particularly good source is [14]. Vector differentiation is discussed in most textbooks on mechanics [15, 16, 17, 10].

2.5 Design Project

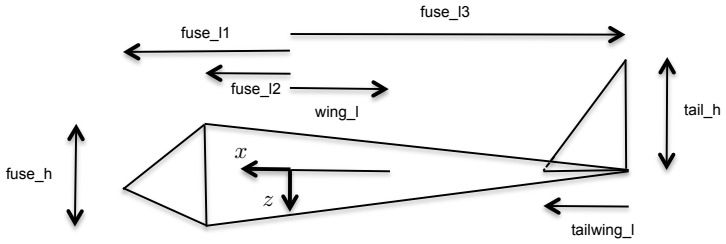
The objective of this assignment is to create a 3D graphic of a MAV that is correctly rotated and translated to the desired configuration. Creating animations in Simulink is described in Appendix A and example files are contained on the accompanying CD and on the textbook web site.

2.1 Read Appendix A and study carefully the spacecraft animation using vertices and faces given on the accompanying CD or at the textbook web site.

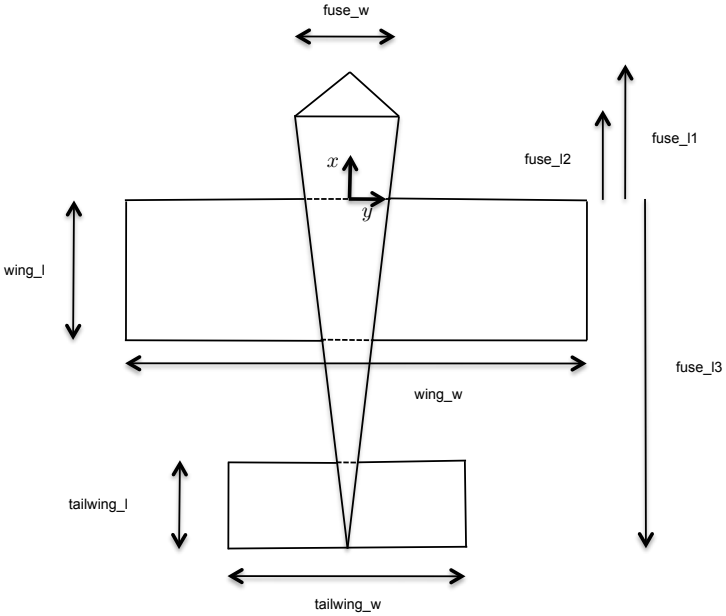
2.2 Create an animation drawing of the aircraft shown in Figure 2.13.



(a) Airframe



(b) Side View



(c) Top View

Figure 2.13: Specifications for animation of aircraft for the project.

2.3 Using a Simulink model like the one shown in Figure 2.14 verify that the aircraft is correctly rotated and translated in the animation.

2.4 In the animation file, switch the order of rotation and translation so that the aircraft is first translated and then rotated, and observe the effect.

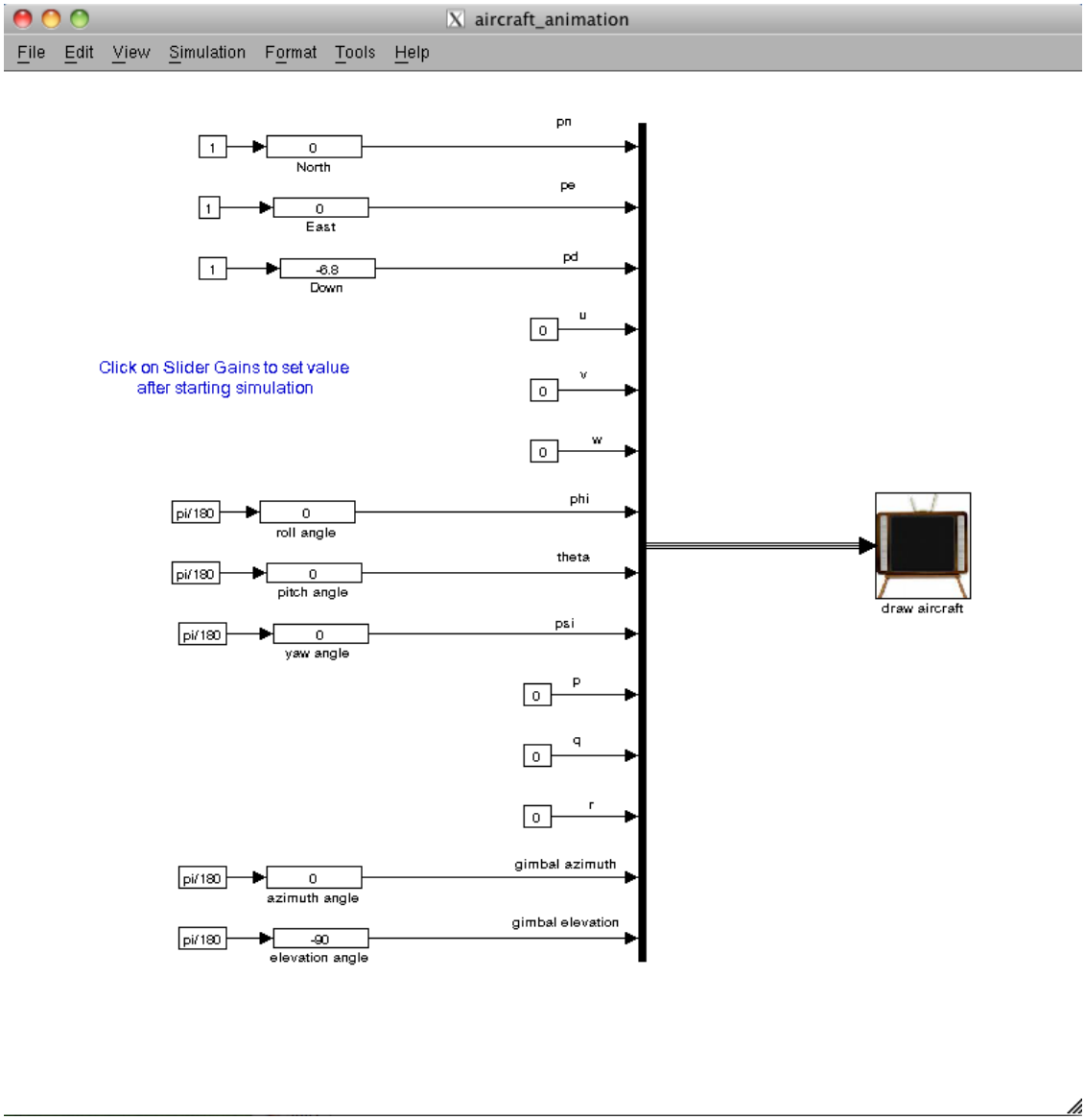


Figure 2.14: Simulink file used to debug the animation. Slider gains are used to specify position and attitude. The input to the animation is designed to be the state of the aircraft.

Chapter 3

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, 4, and **??**. In Chapter **??** 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 relations for $m\dot{\mathbf{v}}$ (the acceleration portion of the dynamics) and relations between positions and velocities (the kinematics). In Chapter 4, we will concentrate on the definition of the forces involved, particularly the aerodynamic forces. In Chapter **??**, 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 the state variables of a MAV. In Section 3.2 we derive the expressions for 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 translation motion of the MAV. Similarly, there are three angular position and three angular velocity states associated with

the rotation motion. The state variables of the MAV are the following twelve quantities

$p_n =$ the inertial (North) position of the MAV along $\mathbf{i}^i$ in $\mathcal{F}^i$,
$p_e =$ the inertial (East) position of the MAV along $\mathbf{j}^i$ in $\mathcal{F}^i$,
$p_d =$ the inertial down position (negative of altitude) of the aircraft measured along $\mathbf{k}^i$ in $\mathcal{F}^i$,
$u =$ the body frame velocity measured along $\mathbf{i}^b$ in $\mathcal{F}^b$,
$v =$ the body frame velocity measured along $\mathbf{j}^b$ in $\mathcal{F}^b$,
$w =$ the body frame velocity measured along $\mathbf{k}^b$ in $\mathcal{F}^b$,
$\phi =$ the roll angle defined with respect to $\mathcal{F}^{v2}$,
$\theta =$ the pitch angle defined with respect to $\mathcal{F}^{v1}$,
$\psi =$ the yaw angle defined with respect to $\mathcal{F}^v$,
$p =$ the roll rate measured along $\mathbf{i}^b$ in $\mathcal{F}^b$,
$q =$ the pitch rate measured along $\mathbf{j}^b$ in $\mathcal{F}^b$,
$r =$ the yaw rate measured along $\mathbf{k}^b$ in $\mathcal{F}^b$.

The state variables are shown schematically in Figure 3.1. The position (p_n, p_e, p_d) of the MAV is

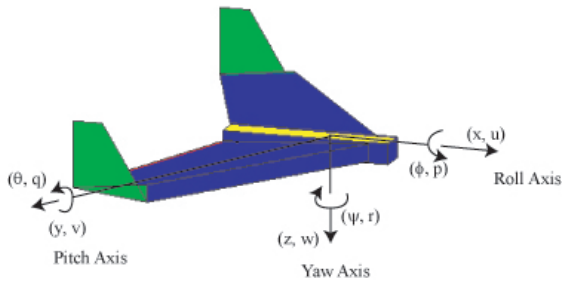


Figure 3.1: Definition of Axes

given in the inertial frame. We will sometimes use $h = -p_d$ to denote the altitude. The velocity (u, v, w) and the angular velocity (p, q, r) of the MAV are given with respect to the body frame. The Euler angles (roll ϕ, pitch θ, and yaw ψ) are given with respect to the vehicle 2-frame, the vehicle 1-frame, and the vehicle frame respectively.

3.2 Kinematics

The translational velocity of the MAV is commonly expressed in terms of the velocity components along each of axes in a body-fixed coordinate frame. The components u, v, and w correspond to the velocities along the $\mathbf{i}^b$, $\mathbf{j}^b$, and $\mathbf{k}^b$ axes respectively. We will calculate angle of attack, sideslip angle, and aerodynamic forces based on these velocity components. On the other hand, the translational position of the MAV is usually measured and expressed in an inertial reference frame. Relating the translational velocity and position requires differentiation and a rotational transformation

$$\begin{aligned}\frac{d}{dt}\begin{pmatrix} p_n \\ p_e \\ p_d \end{pmatrix} &= R_b^v \begin{pmatrix} u \\ v \\ w \end{pmatrix} = (R_v^b)^T \begin{pmatrix} u \\ v \\ w \end{pmatrix} \\ &= \begin{pmatrix} \mathrm{c}_\theta \mathrm{c}_\psi & \mathrm{s}_\phi \mathrm{s}_\theta \mathrm{c}_\psi - \mathrm{c}_\phi \mathrm{s}_\psi & \mathrm{c}_\phi \mathrm{s}_\theta \mathrm{c}_\psi + \mathrm{s}_\phi \mathrm{s}_\psi \\ \mathrm{c}_\theta \mathrm{s}_\psi & \mathrm{s}_\phi \mathrm{s}_\theta \mathrm{s}_\psi + \mathrm{c}_\phi \mathrm{c}_\psi & \mathrm{c}_\phi \mathrm{s}_\theta \mathrm{s}_\psi - \mathrm{s}_\phi \mathrm{c}_\psi \\ -\mathrm{s}_\theta & \mathrm{s}_\phi \mathrm{c}_\theta & \mathrm{c}_\phi \mathrm{c}_\theta \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix}. \end{aligned} \tag{3.1}$$

This is a kinematic relation in that it relates the derivative of position to velocity: forces or accelerations are not considered.

The relationship between angular positions ϕ, θ, and ψ, and the angular rates p, q, and r is also complicated by the fact that these quantities are defined in different coordinate frames. The angular rates are defined in the body frame $\mathcal{F}^b$. The angular positions (Euler angles) are defined in three different coordinate frames. The roll angle ϕ is a rotation from $\mathcal{F}^{v2}$ to $\mathcal{F}^b$ about the $\mathbf{i}^{v2} = \mathbf{i}^b$ axis. The pitch angle θ is a rotation from $\mathcal{F}^{v1}$ to $\mathcal{F}^{v2}$ about the $\mathbf{j}^{v1} = \mathbf{j}^{v2}$ axis. The yaw angle ψ is a rotation from $\mathcal{F}^v$ to $\mathcal{F}^{v1}$ about the $\mathbf{k}^v = \mathbf{k}^{v1}$ axis.

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

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

Inverting we get

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

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 micro 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 $\mathbf{v}$, which for convenience, is most commonly expressed in the body frame as $\mathbf{v}^b = (u, v, w)^T$. $\mathbf{v}^b$ 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

$$\mathsf{m}\frac{d\mathbf{v}}{dt_i} = \mathbf{f}, \tag{3.4}$$

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

The derivative of velocity taken in the inertial frame can be written in terms of the derivative in the body frame and the angular velocity according to (2.13) as

$$\frac{d\mathbf{v}}{dt_i} = \frac{d\mathbf{v}}{dt_b} + \boldsymbol{\omega}_{b/i} \times \mathbf{v} \tag{3.5}$$

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

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

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

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

$$\mathsf{m}\left(\frac{d\mathbf{v}^b}{dt_b} + \boldsymbol{\omega}^b_{b/i} \times \mathbf{v}^b\right) = \mathbf{f}^b \tag{3.6}$$

where $\mathbf{v}^b = (u, v, w)^T$, $\boldsymbol{\omega}^b_{b/i} = (p, q, r)^T$, and $\mathbf{f}^b \triangleq (f_x, f_y, f_z)^T$.

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

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

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

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

These are the three translational dynamic equations that are part of the twelve state equations used to model the MAV.

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 moment taken in the inertial frame can be expanded using (2.13) 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, we can require that the angular momentum and velocity be expressed in the body frame, giving

$$\frac{d\mathbf{h}^b}{dt_b} + \boldsymbol{\omega}^b_{b/i} \times \mathbf{h}^b = \mathbf{m}^b. \tag{3.8}$$

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

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

The diagonal terms of $\mathbf{J}$ are called the *moments of inertia*, while the off-diagonal terms are called the *products of inertia*. Because the volume integrals in (3.9) are calculated with respect to the $\mathbf{i}^b$, $\mathbf{j}^b$, and $\mathbf{k}^b$ axes fixed in the body, $\mathbf{J}$ is constant when viewed from the body frame, hence $\frac{d\mathbf{J}}{dt_b} = 0$. Taking derivatives and substituting into (3.8), we get

$$\mathbf{J}\frac{d\boldsymbol{\omega}_{b/i}^b}{dt_b} + \boldsymbol{\omega}_{b/i}^b \times \left(\mathbf{J}\boldsymbol{\omega}_{b/i}^b\right) = \mathbf{m}^b. \tag{3.10}$$

The expression $\frac{d\boldsymbol{\omega}_{b/i}^b}{dt_b}$ 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 $\boldsymbol{\omega}_{b/i}^b$ onto the $\mathbf{i}^b$, $\mathbf{j}^b$, and $\mathbf{k}^b$ axes, it follows that

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

Rearranging (3.10), we get

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

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

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

Therefore

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

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

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

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

where

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

Equation (3.12) represents the rotational dynamics of the system and provides the equations of motion for the final three states in our twelve-state model. The six degree-of-freedom, twelve-state model for the MAV kinematics and dynamics can be summarized as

$$\begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{p}_d \end{pmatrix} = \begin{pmatrix} \mathsf{c}_\theta \mathsf{c}_\psi & \mathsf{s}_\phi \mathsf{s}_\theta \mathsf{c}_\psi - \mathsf{c}_\phi \mathsf{s}_\psi & \mathsf{c}_\phi \mathsf{s}_\theta \mathsf{c}_\psi + \mathsf{s}_\phi \mathsf{s}_\psi \\ \mathsf{c}_\theta \mathsf{s}_\psi & \mathsf{s}_\phi \mathsf{s}_\theta \mathsf{s}_\psi + \mathsf{c}_\phi \mathsf{c}_\psi & \mathsf{c}_\phi \mathsf{s}_\theta \mathsf{s}_\psi - \mathsf{s}_\phi \mathsf{c}_\psi \\ -\mathsf{s}_\theta & \mathsf{s}_\phi \mathsf{c}_\theta & \mathsf{c}_\phi \mathsf{c}_\theta \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix} \tag{3.14}$$

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

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

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

Equations (3.14)-(3.17) represent the dynamics of the MAV. They are not complete in that the externally applied forces and moments are not yet defined. Models for forces and moments due to gravity, aerodynamics, propulsion, and disturbances will be considered in Chapter 4.

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 [15, 18, 16], space dynamics [19, 20], flight dynamics [13, 21, 1, 2, 5, 7] and robotics [22, 11].

3.5 Design Project

3.1 Read the appendix on building s-functions in Simulink, and also the Matlab documentation on s-functions.

3.2 Implement the MAV equations of motion given in Equations (3.14)–(3.17) using a C-file s-function. Assume that the inputs to the block are the forces and moments applied to the MAV in the body frame. Block parameters should include the mass, the moments and products of inertia, and the initial conditions for each state. Use the parameters given in Appendix C. Simulink templates are on the accompanying CDROM.

3.3 Connect the equations of motion to the animation block developed in the previous chapter. Verify that the equations of motion are correct by individually setting the forces and moments along each axis to a nonzero value and convincing yourself that the motion is appropriate.

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

Chapter 4

Forces and Moments

The objective of this chapter is to describe the forces and moments that act on the UAV. Following [5], we will assume that the forces and moments are primarily due to four sources, namely gravity, aerodynamics, propulsion and atmospheric disturbances. Letting $\mathbf{f}_g$ be the force due to gravity, $(\mathbf{f}_a, \mathbf{m}_a)$ be the forces and moments due to aerodynamics, $(\mathbf{f}_p, \mathbf{m}_p)$ be the forces and moments due to propulsion, and $(\mathbf{f}_d, \mathbf{m}_d)$ be the effects of atmospheric disturbances, we have

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

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

In this section we derive expressions for each of the forces and moments. Gravitational forces are discussed in Section 4.1, the aerodynamic forces and torques are described in Section 4.2, the forces and torques due to propulsion are described in Section 4.3, and atmospheric effects are discussed in Section 4.4.

4.1 Gravitational Forces

The effect of the earth's gravitational field on the MAV can be modeled as a force proportional to the mass acting on 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

$$\mathbf{f}_g^v = \begin{pmatrix} 0 \\ 0 \\ \mathsf{m}g \end{pmatrix}.$$

When applying Newton's 2nd 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

$$\mathbf{f}_g^b = R_v^b \begin{pmatrix} 0 \\ 0 \\ \mathsf{m}g \end{pmatrix} = \begin{pmatrix} -\mathsf{m}g\sin\theta \\ \mathsf{m}g\cos\theta\sin\phi \\ \mathsf{m}g\cos\theta\cos\phi \end{pmatrix}.$$

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

4.2 Aerodynamic Forces

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

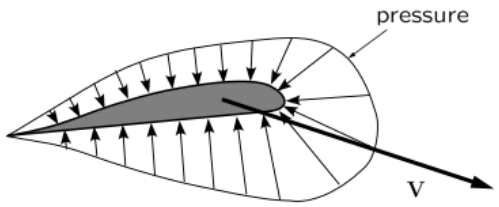


Figure 4.1: Pressure distribution around airfoil.

Instead of attempting to characterize the pressure distribution around the wing, the common approach is to capture the effect of the pressure with a combination of forces and a moment. For example, if we consider the longitudinal ($\mathbf{i}^b$-$\mathbf{k}^b$) plane, the effect of the pressure acting on the MAV body can be modeled using a lift force, a drag force, and a moment. As shown in Figure 4.2, the lift and drag forces are applied at the quarter-chord point, also known as the aerodynamic center.

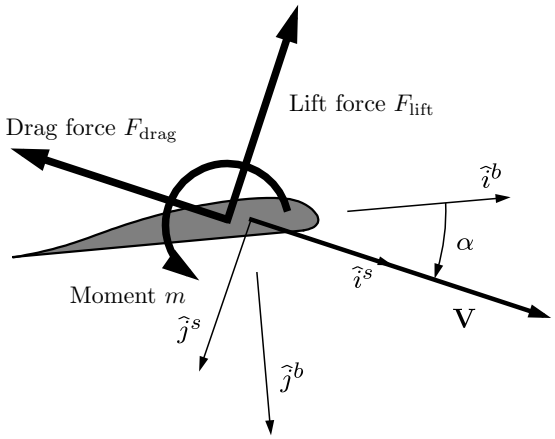


Figure 4.2: Effect of pressure distribution can be modeled using a lift force, a draft force, and a moment.

The lift, drag, and moment are commonly expressed as

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

where C_L, C_D, and C_m are nondimensional aerodynamic coefficients, S is the planform area of the MAV wing, and c is the mean chord of the MAV wing. For airfoils generally, the lift, drag, and pitching moment coefficients are significantly influenced by the airfoil shape, Reynolds number, Mach number, and the angle of attack. For the range of airspeeds flown by small and micro MAVs, the Reynolds number and Mach number effects are not significant. We will consider the effects of the angles α and β, the angular rates p, q, and r, and the deflection of control surfaces, on the aerodynamic coefficients.

It is common to decompose the aerodynamic forces and moments into two groups: longitudinal and lateral-directional. 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-directional 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.

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

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

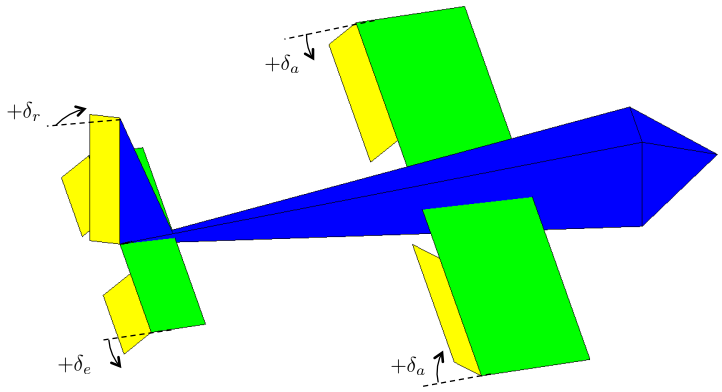


Figure 4.3: Control surfaces for a standard aircraft configuration. The ailerons, denoted by δ_a, are used to control the roll angle ϕ. The elevators, denoted by δ_e, are used to control the pitch angle θ. The rudder, denoted by δ_r, directly effects the yaw angle ψ.

For small aircraft, there are two other standard configurations. The first is the v-tail configuration as shown in Figure 4.4. The control surfaces for a v-tail are called ruddervators. The angle deflection of the right ruddervator is denoted as δ_{rr}, and the angle deflection of the left ruddervator

is denoted as δ_{rl}. Driving the ruddervators differentially has the same effect as a rudder, i.e., torque about $\mathbf{k}^b$, and driving the ruddervators together has the same effect as an elevator, i.e., torque about $\mathbf{j}^b$. Mathematically we can convert between ruddervators and rudder-elevator signals as

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

Therefore the mathematical model for forces and torques for v-tail aircraft can be expressed in terms of standard (rudder-elevator) notation.

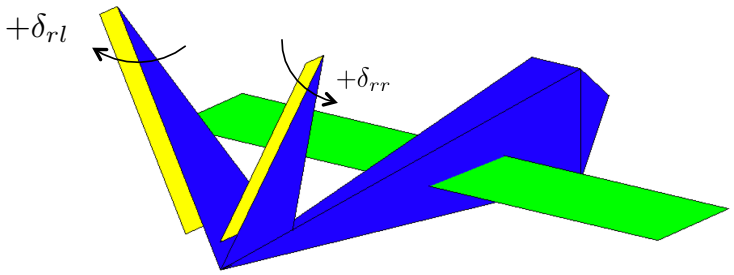


Figure 4.4: Ruddervators are used to control a v-tail aircraft. The ruddervators replace the rudder and the elevator. Driving the ruddervators together has the same effect as an elevator, and driving them differentially has the same effect as a rudder.

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 angle deflection of the right elevon is denoted as δ_{er}, and the angle deflection of the left elevon is denoted as δ_{el}. Driving the elevons differentially has the same effect as ailerons, i.e., torque about $\mathbf{i}^b$, and driving the elevons together has the same effect as and elevator, i.e., torque about $\mathbf{j}^b$. Mathematically we can convert between elevons and aileron-elevator signals a

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

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 $x - z$ plane. They are the aerodynamic forces and moments with which we are perhaps most familiar: lift, drag, and

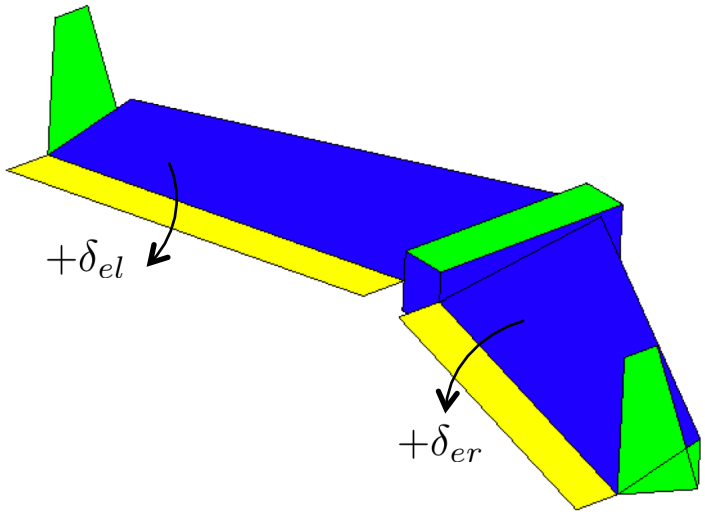


Figure 4.5: Elevons are used to control a flying-wing aircraft. The elevons replace the aileron and the elevator. Driving the elevons together has the same effect as an elevator, and driving them differentially has the same effect as ailerons.

pitching moment. By definition, the lift and drag force 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. Each of these forces/moments is considered to be a strong function of the angle of attack. The pitch rate q and the elevator deflection δ_e also influence the longitudinal forces/moments. The general expression is given by Equation (4.1) where C_L, C_D, and C_m are functions of the angle of attack α, the pitch rate q, and the elevator deflection δ_e. While lift, drag, and pitching moment are nonlinear functions of α, they are approximately linear in q, and δ_e [21]. Therefore we have

$$F_{\text{lift}} = \frac{1}{2}\rho V_a^2 S \left[C_L(\alpha) + C_{L_q}\frac{c}{2V_a}q + C_{L_{\delta_e}}\delta_e \right] \tag{4.2}$$

$$F_{\text{drag}} = \frac{1}{2}\rho V_a^2 S \left[C_D(\alpha) + C_{D_q}\frac{c}{2V_a}q + C_{D_{\delta_e}}\delta_e \right] \tag{4.3}$$

$$m = \frac{1}{2}\rho V_a^2 S c \left[C_M(\alpha) + C_{M_q}\frac{c}{2V_a}q + C_{M_{\delta_e}}\delta_e \right] \tag{4.4}$$

For a cambered wing at small angle of attack, the flow of air over the wing is laminar and is attached to the wing. Under these conditions, the lift coefficient can be modeled quite accurately as being linear in α:

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

Under most flight conditions, this simple linear model is satisfactory. For extreme maneuvers involving large angles of attack, however, the flow over the wing separates producing a sudden reduction in lift known as stall. For angles of attack that are beyond the stall conditions, the wing acts roughly like a flat plate, whose lift coefficient is given by [21]

$$C_{L,\text{flat plate}} = 2\text{sign}(\alpha)\sin^2\alpha\cos\alpha. \tag{4.5}$$

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 important aerodynamic effects such as stall. A lift model that incorporates the common linear lift behavior and the effects of stall is given by

$$C_L(\alpha) = (1-\sigma(\alpha))\left[C_{L_0} + C_{L_\alpha}\alpha\right] + \sigma(\alpha)\left[2\text{sign}(\alpha)\sin^2\alpha\cos\alpha\right], \tag{4.6}$$

where

$$\sigma(\alpha) = \frac{1 + e^{-M(\alpha-\alpha_0)} + e^{M(\alpha+\alpha_0)}}{\left(1+e^{-M(\alpha-\alpha_0)}\right)\left(1+e^{M(\alpha+\alpha_0)}\right)} \tag{4.7}$$

where M and α_0 are positive constants. The sigmoid functions (4.7) is a blending function with cut-off at $\pm\alpha_0$ and transition rate M. Figure 4.6 shows the lift coefficient in (4.6) as a blended function of the linear term $C_{L_0} + C_{L_\alpha}\alpha$ and the flat plate term (4.5). For small aircraft, the linear lift coefficient can be reasonably approximated as

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

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

The drag coefficient C_D is also a nonlinear function of the angle of attack. There are two contributions to the drag coefficient namely induced drag and parasitic drag [21]. The parasitic drag, generated by the shear stress of air moving over the wing and other effects, is roughly constant and will be denoted by C_{D_0}. For small angles of attack, the induced drag is proportional to the square of the list

$$C_{D,\text{small }\alpha}(\alpha) = \epsilon C_L^2(\alpha), \tag{4.8}$$

where $\epsilon = \pi/AR$ for wings with elliptical lift distribution [21] and is slightly larger for other wings. For large angles of attack, the induced drag is roughly equal to that of a flat plat which is given by [21]

$$C_{D,\text{flat plate}}(\alpha) = 2\sin^3\alpha. \tag{4.9}$$

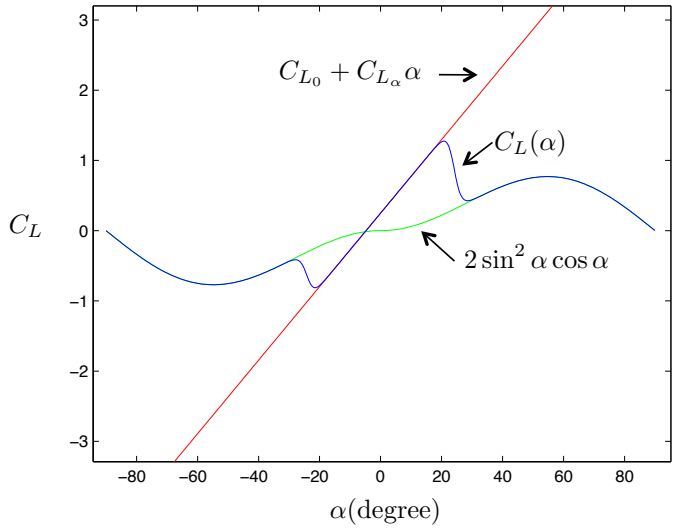


Figure 4.6: The lift coefficient as a function of α (blue) can be a approximated by blending a linear function of alpha (red), with the lift coefficient of a flat plate (green).

For simulation purposes, we will model the drag coefficient as a blending of (4.8) and (4.9), where

$$C_D(\alpha) = C_{D_0} + (1 - \sigma(\alpha))\,\epsilon\left[C_{L_0} + C_{L_\alpha}\alpha\right]^2 + \sigma(\alpha)\left[2\text{sign}(\alpha)\sin^3\alpha\right]. \tag{4.10}$$

Figure 4.7 shows the drag coefficient in (4.10) as a blended function of (4.8) and (4.9).

The lift and drag forces expressed in Equations (4.2) and (4.3) are expressed in the stability frame. To express lift and drag in the body frame requires a rotation by the angle of attack:

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

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

The pitching moment of the aircraft will in general be a nonlinear function of angle of attack and must be determined by wind tunnel or flight experiments. For the purposes of simulation, we will use the linear model

$$C_M(\alpha) = C_{M_0} + C_{M_\alpha}\alpha,$$

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

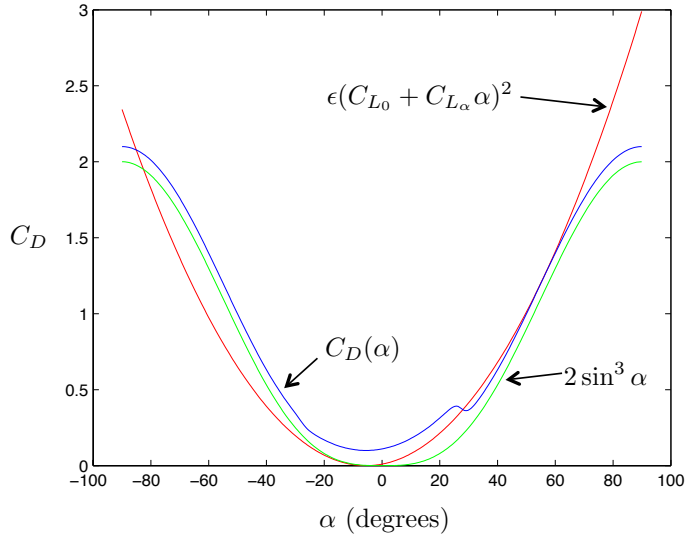


Figure 4.7: The drag coefficient as a function of α (blue) can be a approximated by blending the square of the lift coefficient at low angle-of-attach (red), with the drag coefficient of a flat plate (green).

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 β. They are also influenced by the roll rate p, the yaw rate r, the deflection of the aileron δ_a, and the deflection of the rudder δ_r. Denoting the lateral force as f_x and the roll and yaw moments as l and n respectively, we have

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

where C_Y, C_l, and C_n are nondimensional aerodynamic coefficients. The aerodynamic coefficient C_Y is a nonlinear function of β and nonlinear equations can be derived using the same line of reasoning as in Section 4.2.2. However, since the side slip angle is typically small we will use a

linear approximation. The linear relationship is given by

$$f_y = \frac{1}{2}\rho V_a^2 S \left[C_{Y_0} + C_{Y_\beta}\beta + C_{Y_p}\frac{b}{2V_a}p + C_{Y_r}\frac{b}{2V_a}r + C_{Y_{\delta_a}}\delta_a + C_{Y_{\delta_r}}\delta_r \right] \tag{4.11}$$

$$l = \frac{1}{2}\rho V_a^2 S b \left[C_{l_0} + C_{l_\beta}\beta + C_{l_p}\frac{b}{2V_a}p + C_{l_r}\frac{b}{2V_a}r + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r \right] \tag{4.12}$$

$$n = \frac{1}{2}\rho V_a^2 S b \left[C_{n_0} + C_{n_\beta}\beta + C_{n_p}\frac{b}{2V_a}p + C_{n_r}\frac{b}{2V_a}r + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r \right] \tag{4.13}$$

where b is the wingspan of the MAV.

4.2.4 Aerodynamic Coefficients

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

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

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

C_{n_β} is referred to as the yaw static stability derivative. It is sometimes called the weathercock stability derivative. For the MAV to be stable in yaw, C_{n_β} must be positive. This simply implies that for a positive sideslip angle, a positive yawing moment will be induced. This yawing moment will yaw the MAV into the direction of the relative wind, driving the sideslip angle to zero.

Dynamic stability deals with the dynamic behavior of the plane in response to disturbances. If a disturbance is applied to the MAV, the MAV is said to by dynamically stable if the response of the MAV damps out in a finite amount of time. If we use a second-order mass-spring-damper analogy to analyze the MAV, the stability derivatives C_{m_α}, C_{l_β}, and C_{n_β} model torsional springs, while the derivatives C_{m_q}, C_{l_p}, and C_{n_r} model torsional dampers. The moments of inertia of the MAV

body provide the mass. As we will see in Chapter **??** when we linearize the dynamic equations of motion for the MAV, the signs of the stability derivatives must be consistent to ensure that the characteristic roots of the MAV lie in the left half of the s-plane.

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

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

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

4.3 Propulsion Forces

A simple model for the thrust generated by a propeller can be developed by applying Bernoulli's principle to calculate the pressure ahead of and behind the propeller and applying the pressure difference to the propeller area. This approach will yield a model that is correct for a perfectly efficient propeller. While overly optimistic in its thrust predictions, this model will provide a reasonable starting point for a MAV simulation.

Using Bernoulli's equation, the total pressure upstream of the propeller can be written as

$$P_{\text{upstream}} = P_0 + \frac{1}{2}\rho V_a^2$$

where P_0 is the static pressure and ρ is the air density. The pressure downstream of the propeller can be expressed as

$$P_{\text{downstream}} = P_0 + \frac{1}{2}\rho V_{\text{exit}}^2.$$

where V_{exit} is the speed of the air as it leaves the propeller. Ignoring the transients in the motor, there is a linear relationship between the pulse-width-modulation command δ_t and the angular

velocity of the propeller. The propeller in turn creates an exit air speed of

$$V_{\text{exit}} = k_{\text{motor}}\delta_t.$$

If S_{prop} is the area swept out by the propeller, then the thrust produced by the motor is given by

$$\begin{aligned} F_{x_p} &= S_{\text{prop}}C_{\text{prop}}(P_{\text{downstream}} - P_{\text{upstream}}) \\ &= \frac{1}{2}\rho S_{\text{prop}}C_{\text{prop}}\left[(k_{\text{motor}}\delta_t)^2 - V_a^2\right]. \end{aligned}$$

Therefore

$$\mathbf{f}_p = \frac{1}{2}\rho S_{\text{prop}}C_{\text{prop}}\begin{pmatrix}(k_{\text{motor}}\delta_t)^2 - V_a^2 \\ 0 \\ 0\end{pmatrix}.$$

Most MAVs are designed so that the thrust acts directly along the $\mathbf{i}^b$ body-axis of the aircraft. Therefore the thrust does not produce any moments about the center of mass of the MAV.

RWB: Perhaps we should be more general here. It would be easy to show how moment is derived using the cross product.

Propeller Torque

As the MAV propeller spins, it applies force to the air that passes through the propeller, increasing the momentum of the air while generating a thrust force on the MAV. Equal and opposite forces are applied by the air on the propeller. The net effect of these forces is a torque about the propeller axis of rotation applied to the MAV. The torque applied by the motor to the propeller (and then to the air) results in an equal and opposite torque applied by the propeller to the motor which is fixed to the MAV body. This torque is opposite to the direction of the propeller rotation and proportional to the square of the propeller angular velocity

$$T_p = -k_{T_p}(k_\Omega\delta_t)^2$$

where $\Omega = k_\Omega\delta_t$ is the propeller speed and k_{T_p} is a constant determined by experiment. The moments due to the propulsion system are therefore

$$\mathbf{m}_p = \begin{pmatrix}-k_{T_p}(k_\Omega\delta_t)^2 \\ 0 \\ 0\end{pmatrix}.$$

The effects of this propeller torque are usually relatively minor. If unaccounted for, it will cause a slow rolling motion in the direction opposite the propeller rotation. It is easily corrected by applying a small aileron deflection, which generates a rolling moment to counteract the propeller torque.

4.4 Atmospheric Disturbances

In this section, we will discuss atmospheric disturbances, such as wind, and describe how these disturbances enter in the dynamics of the aircraft. In Chapter 2, we defined $\mathbf{V}_g$ as the velocity of the airframe relative to the ground, $\mathbf{V}_a$ as the velocity of the airframe relative to the surrounding air mass, and $\mathbf{V}_w$ as the velocity of the air mass relative to the ground, or in other words, the wind velocity. As shown in Equation (2.6), the relationship between ground velocity, air velocity, and wind velocity is given by

$$\mathbf{V}_g = \mathbf{V}_a + \mathbf{V}_w. \tag{4.14}$$

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

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

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

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

where w_{n_s} is the speed of the steady wind in the North direction, w_{e_s} is the speed of the steady wind in the East direction, and w_{d_s} is the speed of the steady wind in the Down direction. The stochastic (gust) component of the wind is typically expressed in the aircraft *body* frame as

$$\mathbf{V}^b_{w_g} = \begin{pmatrix} u_{w_g} \\ v_{w_g} \\ w_{w_g} \end{pmatrix}$$

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 given in [21]. Unfortunately, the von Karmen spectrum does not result in a rational transfer function. A suitable approximation of the von Karmen model is given by the

Dryden transfer functions

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

where σ_u, σ_v, and σ_w are the standard deviation of the noise along the vehicle frame axes, and L_u, L_v, L_w are spatial wavelengths, and V_a is the airspeed of the vehicle. According to [21], spatial wavelengths that correspond with experimentally collected data at high altitude are

$$L_u = 1250 \text{ m}$$
$$L_v = 1750 \text{ m}$$
$$L_w = 1750 \text{ m}.$$

Figure 4.8 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

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

where R_v^b is the rotation matrix from the vehicle to the body frame given in Equation (2.5). From the components of the wind velocity $\mathbf{V}_w^b$ and the ground velocity $\mathbf{V}_g^b$ we can calculate the body-frame components of the air velocity

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

From the body-frame components of the air velocity, we can calculate airspeed, the angle of attack,

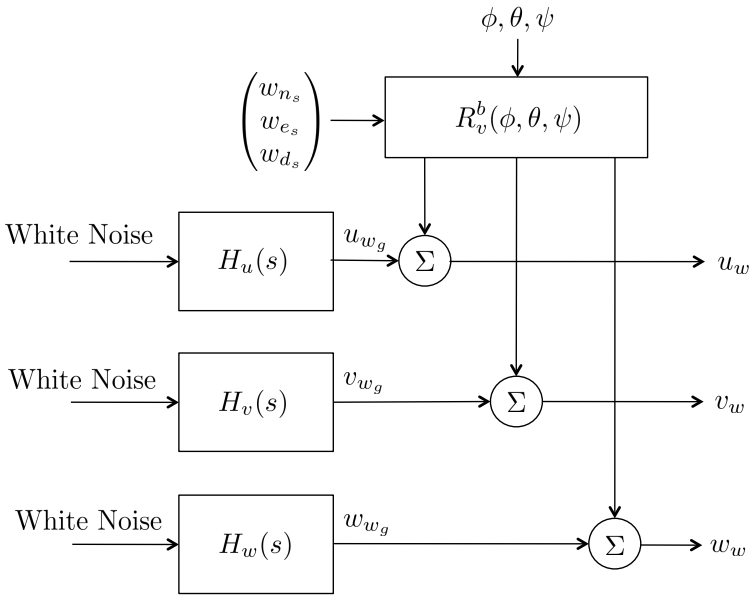


Figure 4.8: The wind is modeled as a constant wind field plus turbulence. The turbulence is generated by filtering white noise with a Dryden model.

and the sideslip angle according to Equation (2.8) as

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

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

4.5 Chapter Summary

The total forces on the MAV are be summarized as follows:

$$\begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} = \begin{pmatrix} -\mathsf{m}g\sin\theta \\ \mathsf{m}g\cos\theta\sin\phi \\ \mathsf{m}g\cos\theta\cos\phi \end{pmatrix} + \frac{1}{2}\rho V_a^2 S \begin{pmatrix} C_X(\alpha) + C_{X_q}(\alpha)\frac{c}{2V_a}q + C_{X_{\delta_e}}(\alpha)\delta_e \\ C_{Y_0} + C_{Y_\beta}\beta + C_{Y_p}\frac{b}{2V_a}p + C_{Y_r}\frac{b}{2V_a}r + C_{Y_{\delta_a}}\delta_a + C_{Y_{\delta_r}}\delta_r \\ C_Z(\alpha) + C_{Z_q}(\alpha)\frac{c}{2V_a}q + C_{Z_{\delta_e}}(\alpha)\delta_e \end{pmatrix} + \frac{1}{2}\rho S_{\text{prop}} C_{\text{prop}} \begin{pmatrix} (k_{\text{motor}}\delta_t)^2 - V_a^2 \\ 0 \\ 0 \end{pmatrix} \tag{4.15}$$

where

$$\begin{aligned} C_X(\alpha) &\triangleq -C_D(\alpha)\cos\alpha + C_L(\alpha)\sin\alpha \\ C_{X_q}(\alpha) &\triangleq -C_{D_q}\cos\alpha + C_{L_q}\sin\alpha \\ C_{X_{\delta_e}}(\alpha) &\triangleq -C_{D_{\delta_e}}\cos\alpha + C_{L_{\delta_e}}\sin\alpha \\ C_Z(\alpha) &\triangleq -C_D(\alpha)\sin\alpha - C_L(\alpha)\cos\alpha \\ C_{Z_q}(\alpha) &\triangleq -C_{D_q}\sin\alpha - C_{L_q}\cos\alpha \\ C_{Z_{\delta_e}}(\alpha) &\triangleq -C_{D_{\delta_e}}\sin\alpha - C_{L_{\delta_e}}\cos\alpha, \end{aligned} \tag{4.16}$$

and where $C_L(\alpha)$ is given by Equation (4.6) and $C_D(\alpha)$ is given by Equation (4.10).

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

$$\begin{pmatrix} l \\ m \\ n \end{pmatrix} = \frac{1}{2}\rho V_a^2 S \begin{pmatrix} b\left[C_{l_0} + C_{l_\beta}\beta + C_{l_p}\frac{b}{2V_a}p + C_{l_r}\frac{b}{2V_a}r + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r\right] \\ c\left[C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{c}{2V_a}q + C_{m_{\delta_e}}\delta_e\right] \\ b\left[C_{n_0} + C_{n_\beta}\beta + C_{n_p}\frac{b}{2V_a}p + C_{n_r}\frac{b}{2V_a}r + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r\right] \end{pmatrix} + \begin{pmatrix} -k_{T_p}(k_\Omega\delta_t)^2 \\ 0 \\ 0 \end{pmatrix}. \tag{4.17}$$

Notes and References

The material in this chapter can be found in most textbooks on flight dynamics including [13, 21, 1, 2, 5, 7, 23]. Our discussion on lift, drag, and moment coefficients is drawn primarily from [21]. The discussion breaking the wind vector into a constant and a random term follows [21].

4.6 Design Project

4.1 Develop a block in Simulink that implements the gravity, aerodynamic, and propulsion forces and torques described in this chapter. Use the parameters given in Appendix C.

4.2 Develop blocks in Simulink that outputs $(w_{n_s}, w_{e_s}, w_{d_s})^T$ and $(u_{w_g}, v_{w_g}, w_{w_g})^T$. Connect these blocks to the forces and moments block as shown in Figure 4.9

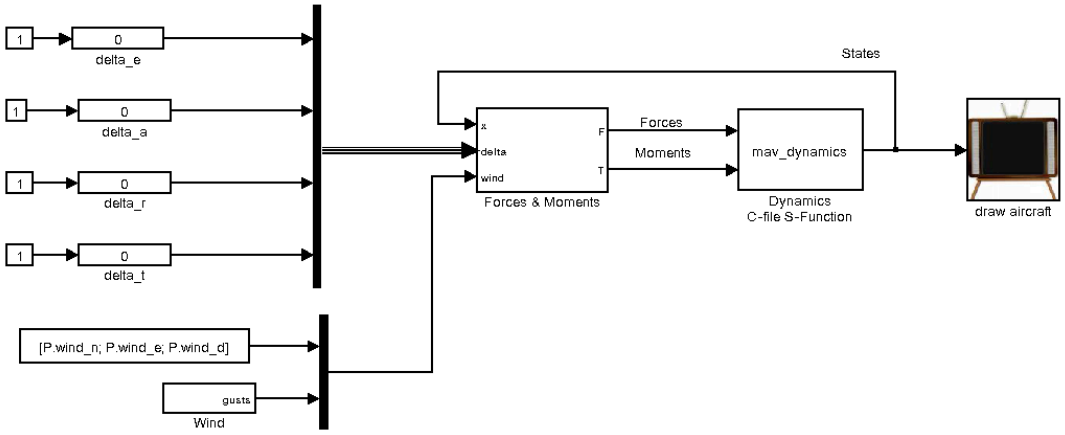


Figure 4.9: The aircraft simulation is decomposed into a block for the dynamics, a block for the forces and moments, and two blocks for the wind.

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

Chapter 5

Linear Design Models

The low-level autopilot will be designed based on linear design models that capture the approximate motion of the system under specific scenarios. 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 included airspeed, pitch angle, and altitude, and into lateral motion, which include roll and yaw angles, and heading. While there is coupling between longitudinal and lateral motion, for most airframes the coupling is small and can be handled 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 are given with respect to equilibrium conditions. In flight dynamics, force and moment equilibrium is called trim, which is discussed in Section 5.2. Transfer function for both the lateral and longitudinal dynamics are derived in Section 5.3. State space models are derived in Section 5.4.

5.1 Summary of Nonlinear Equations of Motion

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 present 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. Neglecting atmospheric disturbances and incorporating the aerodynamic and propulsion models described in Chapter 4 into Equations (**??**)–(**??**) we get the

following equations of motion:

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

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

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

$$\dot{u} = rv - qw - g\sin\theta + \frac{\rho V_a^2 S}{2\mathsf{m}}\left[C_X(\alpha) + C_{X_q}(\alpha)\frac{\bar{c}q}{2V_a} + C_{X_{\delta_e}}(\alpha)\delta_e\right] + \frac{\rho S_{\text{prop}} C_{\text{prop}}}{2\mathsf{m}}\left[(k_{\text{motor}}\delta_t)^2 - V_a^2\right] \tag{5.4}$$

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

$$\dot{w} = qu - pv + g\cos\theta\cos\phi + \frac{\rho V_a^2 S}{2\mathsf{m}}\left[C_Z(\alpha) + C_{Z_q}(\alpha)\frac{\bar{c}q}{2V_a} + C_{Z_{\delta_e}}(\alpha)\delta_e\right] \tag{5.6}$$

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

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

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

$$\dot{p} = \Gamma_1 pq - \Gamma_2 qr + \frac{1}{2}\rho V_a^2 Sb\left[C_{p_0} + C_{p_\beta}\beta + C_{p_p}\frac{bp}{2V_a} + C_{p_r}\frac{br}{2V_a} + C_{p_{\delta_a}}\delta_a + C_{p_{\delta_r}}\delta_r\right] \tag{5.10}$$

$$\dot{q} = \Gamma_5 pr - \Gamma_6(p^2 - r^2) + \frac{\rho V_a^2 S\bar{c}}{2J_y}\left[C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{\bar{c}q}{2V_a} + C_{m_{\delta_e}}\delta_e\right] \tag{5.11}$$

$$\dot{r} = \Gamma_7 pq - \Gamma_1 qr + \frac{1}{2}\rho V_a^2 Sb\left[C_{r_0} + C_{r_\beta}\beta + C_{r_p}\frac{bp}{2V_a} + C_{r_r}\frac{br}{2V_a} + C_{r_{\delta_a}}\delta_a + C_{r_{\delta_r}}\delta_r\right], \tag{5.12}$$

where

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

The inertia parameters specified by Γ_1, Γ_2, ... , Γ_8 are defined in (3.13). As shown in Chapter 4 the aerodynamic force coefficients in the 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 and drag coefficients, we can model them as

$$
\begin{aligned}
C_L(\alpha) &= (1-\sigma(\alpha))\left[C_{L_0} + C_{L_\alpha}\alpha\right] + \sigma(\alpha)\left[2\text{sign}(\alpha)\sin^2\alpha\cos\alpha\right] \\
C_D(\alpha) &= C_{D_0} + (1-\sigma(\alpha))\,\epsilon\left[C_{L_0} + C_{L_\alpha}\alpha\right]^2 + \sigma(\alpha)\left[2\text{sign}(\alpha)\sin^3\alpha\right],
\end{aligned}
$$

where

$$
\sigma(\alpha) = \frac{1 + e^{-M(\alpha-\alpha_0)} + e^{M(\alpha+\alpha_0)}}{\left(1+e^{-M(\alpha-\alpha_0)}\right)\left(1+e^{M(\alpha+\alpha_0)}\right)} \tag{5.13}
$$

where M and α_0 are positive constants.

If we are interested in modeling MAV flight under fairly gentle flight conditions (i.e., no high-angle-of-attack maneuvers), a simpler, linear model for the lift and drag coefficients can be used

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

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 will be the core of the MAV simulation environment that you develop as part of the project exercises at the end of each chapter.

5.2 Trim Conditions

Given a nonlinear system described by the differential equations

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

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

$$f(x^*, u^*) = 0.$$

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 ϕ, θ, ψ, 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, $\dot{h}$ is constant and h grows linearly. Also, in a constant turn $\dot{\psi}$ is constant and ψ has linear growth. Therefore, in general, the trim condition is given by

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

The objective of this section is describe how to compute trim states and inputs for several common flight conditions. In particular, we will focus on the following three flight conditions.

1. **Wings-level flight.** In wings-level flight, airspeed is assumed to be constant. The wings are level but the flight path angle may not be zero. In other words, we allow for constant rate climbs and descents. Wings-level flight is a pure longitudinal motion.

2. **Constant altitude turn.** In a constant-altitude turn, the velocity is assumed constant and the altitude is held constant. To maintain constant altitude, both longitudinal and lateral controls are required.

3. **Turn with a constant climb rate.** This flight condition is a combination of wings-level flight and constant altitude turn maneuvers. Wings level flight and constant altitude turns are special cases of this flight condition. It is particularly useful for autonomous take-off and landing of MAVs.

Since the third condition, turn with a constant climb rate, subsumes the the first two conditions, we will focus on turns with a constant climb rate. Since these trim conditions are independent of inertial position, p_n and p_e do not factor into trim calculations.

5.2.1 Turn with a Constant Climb Rate

Let $x \triangleq (h, u, v, w, \phi, \theta, \psi, p, q, r)^T$ and $u \triangleq (\delta_e, \delta_t, \delta_a, \delta_r)^T$. The objective is to find the trim states x^* and the trim inputs u^* that guarantee that

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

where f is given by Equations (**??**)–(**??**).

Consider the desired flight condition with a constant airspeed of V_a, a flight path angle of γ, and a turning radius of R. The climb rate will therefore be constant and is given by

$$\dot{h}^* = V_a \sin\gamma.$$

The turn rate is also constant and is given by

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

The rest of the states will be constant, therefore

$$\dot{x}^* = \begin{pmatrix} \dot{h}^* = V_a \sin\gamma \\ \dot{u}^* = 0 \\ \dot{v}^* = 0 \\ \dot{w}^* = 0 \\ \dot{\phi}^* = 0 \\ \dot{\theta}^* = 0 \\ \dot{\psi}^* = \frac{V_a}{R} \\ \dot{p}^* = 0 \\ \dot{q}^* = 0 \\ \dot{r}^* = 0 \end{pmatrix}. \tag{5.14}$$

The first objective is to show that the state variables and the input commands can be expressed in terms of V_a, γ, R, α, β, and ϕ. Since V_a, γ, and R are inputs to the algorithm, computing the trim states will then consist of an optimization algorithm over α, β, and ϕ.

Altitude h^*: Since $\dot{h}^* = V_a \sin\gamma$ where V_a and γ are constant, altitude is given by

$$h^* = h^*(t_0) + V_a \sin\gamma (t - t_0).$$

However, since the right hand side of Equations (**??**)–(**??**) are independent of h, this expression does not enter into the trim calculation.

Body frame velocities u^*, v^*, w^*: From Equation (2.7), the body frame velocities can be expressed in terms of V_a, α and β as

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

Euler angles θ^*, ψ^*: By definition of the flight path angle we have

$$\theta^* = \alpha^* + \gamma.$$

Since $\dot{\psi}^* = \frac{V_a}{R}$ where V_a and R are constant, the yaw angle is given by

$$\psi^* = \psi^*(t_0) + \frac{V_a}{R}(t - t_0).$$

However, since the right hand side of Equations (**??**)–(**??**) are independent of ψ, this expression does not enter into the trim calculation.

Angular rates p, q, r: The angular rates can be expressed in terms of the Euler angles by using Equation (3.2). Therefore

$$\begin{pmatrix} p^* \\ q^* \\ r^* \end{pmatrix} = \begin{pmatrix} 1 & 0 & -\sin\theta^* \\ 0 & \cos\phi^* & \sin\phi^* \cos\theta^* \\ 0 & -\sin\phi^* & \cos\phi^* \cos\theta^* \end{pmatrix} \begin{pmatrix} \dot{\phi}^* = 0 \\ \dot{\theta}^* = 0 \\ \dot{\psi}^* = \frac{V_a}{R} \end{pmatrix} = \frac{V_a}{R} \begin{pmatrix} -\sin\theta^* \\ \sin\phi^* \cos\theta^* \\ \cos\phi^* \cos\theta^* \end{pmatrix},$$

where θ^* has already been expressed in terms of γ and α^*.

Elevator δ_e: Given p^*, q^*, and r^* we can solve Equation (**??**) for δ_e giving

$$\delta_e^* = \frac{\left[\frac{J_{xz}(p^{*2}-r^{*2})+(J_x-J_z)p^*r^*)}{\frac{1}{2}\rho V_a^2 \bar{c} S}\right] - C_{m_0} - C_{m_\alpha}\alpha^* - C_{m_q}\frac{\bar{c}q^*}{2V_a}}{C_{m_{\delta_e}}}. \tag{5.15}$$

Throttle δ_t: Equation (**??**) can be solved for δ_t giving

$$\delta_t^* = \sqrt{\frac{-r^*v^* + q^*w^* + g\sin\theta^* - \frac{\rho V_a^2 S}{2\mathsf{m}}\left[C_X(\alpha^*) + C_{X_q}(\alpha^*)\frac{\bar{c}q^*}{2V_a^*} + C_{X_{\delta_e}}(\alpha^*)\delta_e^*\right]}{\frac{\rho S_{\text{prop}} C_{\text{prop}} k_{\text{motor}}^2}{2\mathsf{m}}} + \frac{V_a^2}{k_{\text{motor}}^2}}. \tag{5.16}$$

Aileron δ_a and Rudder δ_r: The aileron and rudder commands are found by solving Equations (**??**) and (**??**):

$$\begin{pmatrix}\delta_a^* \\ \delta_r^*\end{pmatrix} = \begin{pmatrix}C_{p_{\delta_a}} & C_{p_{\delta_r}} \\ C_{r_{\delta_a}} & C_{r_{\delta_r}}\end{pmatrix}^{-1} \begin{pmatrix}\frac{-\Gamma_1 p^*q^* + \Gamma_2 q^*r^*}{\frac{1}{2}\rho V_a^2 S b} - C_{p_0} - C_{p_\beta}\beta^* - C_{p_p}\frac{bp^*}{2V_a} - C_{p_r}\frac{br^*}{2V_a} \\ \frac{-\Gamma_7 p^*q^* + \Gamma_1 q^*r^*}{\frac{1}{2}\rho V_a^2 S b} - C_{r_0} - C_{r_\beta}\beta^* - C_{r_p}\frac{bp^*}{2V_a} - C_{r_r}\frac{br^*}{2V_a}\end{pmatrix}. \tag{5.17}$$

5.2.2 Trim Algorithm

All of the state variables of interest and the control inputs have been expressed in terms of V_a, γ, R, α, β and ϕ. The inputs to the trim algorithm are V, γ, and R. To find α, β, and ϕ we need to solve the following optimization problem:

$$(\alpha^*, \phi^*, \beta^*) = \arg\min \left\| \dot{x}^* - f(x^*, u^*) \right\|^2.$$

This can be performed numerically using a gradient descent algorithm which will be described in the next section. The trim algorithm is summarized in Algorithm 1.

5.2.3 Numerical implementation of Gradient Descent

The objective of this section is to describe a simple gradient descent algorithm that solves the optimization problem

$$\min_{\xi} J(\xi),$$

where $J : \mathbb{R}^m \to \mathbb{R}$ is assumed to be continuously differentiable with well defined local minima. The basic idea is to follow the negative gradient of the function given an initial starting location $\xi^{(0)}$. In other words, we let

$$\dot{\xi} = -\kappa \frac{\partial J}{\partial \xi}(\xi), \tag{5.18}$$

Algorithm 1 Trim for Climbing Turns

1: Input: Desired airspeed V_a, desired flight path angle γ, and desired turn radius R.

2: Compute: $(\alpha^*, \beta^*, \phi^*) = \arg\min \|\dot{x}^* - f(x^*, u^*)\|^2$.

3: Compute trimmed states:

$$\begin{pmatrix} u^* = V_a \cos\alpha^* \cos\beta^* \\ v^* = V_a \sin\beta^* \\ w^* = V_a \sin\alpha^* \cos\beta^* \\ \theta^* = \alpha^* + \gamma \\ p^* = -\frac{V_a}{R} \sin\theta^* \\ q^* = \frac{V_a}{R} \sin\phi^* \cos\theta^* \\ r^* = \frac{V_a}{R} \cos\phi^* \cos\theta^* \end{pmatrix}$$

4: Compute trimmed input:

$$\begin{pmatrix} \delta_e^* = \text{[Equation (5.15)]} \\ \delta_t^* = \text{[Equation (5.16)]} \\ \begin{pmatrix} \delta_a^* \\ \delta_r^* \end{pmatrix} = \text{[Equation (5.17)]} \end{pmatrix}.$$

where κ is a positive constant that defines the descent rate. A discrete approximation of (5.18) is given by

$$\xi^{(k+1)} = \xi^{(k)} - \kappa \frac{\partial J}{\partial \xi}(\xi^{(k)}).$$

For the trim calculation $\frac{\partial J}{\partial \xi}$ is difficult to determine analytically. However, it can be efficiently computed numerically. By definition we have

$$\frac{\partial J}{\partial \xi} = \begin{pmatrix} \frac{\partial J}{\partial \xi_1} \\ \vdots \\ \frac{\partial J}{\partial \xi_m} \end{pmatrix},$$

where

$$\frac{\partial J}{\partial \xi_i} = \lim_{\epsilon \to 0} \frac{J(\xi_1, \cdots, \xi_i + \epsilon, \cdots, \xi_m) - J(\xi_1, \cdots, \xi_i, \cdots, \xi_m)}{\epsilon},$$

which can be numerically approximated as

$$\frac{\partial J}{\partial \xi_i} \approx \frac{J(\xi_1, \cdots, \xi_i + \epsilon, \cdots, \xi_m) - J(\xi_1, \cdots, \xi_i, \cdots, \xi_m)}{\epsilon},$$

where ϵ is a small constant.

For the trim algorithm, $J(\alpha, \beta, \phi) = \|\dot{x}^* - f(x^*, u^*)\|^2$ which is computed using Algorithm 2. The gradient descent optimization algorithm is summarized in Algorithm 3.

Algorithm 2 Computation of $J = \|\dot{x}^* - f(x^*, u^*)\|^2$

1: Input: α, β, ϕ, V_a, R, γ.

2: Compute $\dot{x}^*$:

$$\dot{x}^* = \text{[Equation (5.14)]}.$$

3: Compute trimmed states:

$$\begin{pmatrix} u^* = V_a \cos\alpha^* \cos\beta^* \\ v^* = V_a \sin\beta^* \\ w^* = V_a \sin\alpha^* \cos\beta^* \\ \theta^* = \alpha^* + \gamma \\ p^* = -\frac{V_a}{R} \sin\theta^* \\ q^* = \frac{V_a}{R} \sin\phi^* \cos\theta^* \\ r^* = \frac{V_a}{R} \cos\phi^* \cos\theta^* \end{pmatrix}$$

4: Compute trimmed input:

$$\begin{pmatrix} \delta_e^* = \text{[Equation (5.15)]} \\ \delta_t^* = \text{[Equation (5.16)]} \\ \begin{pmatrix} \delta_a^* \\ \delta_r^* \end{pmatrix} = \text{[Equation (5.17)]} \end{pmatrix}.$$

5: Compute $f(x^*, u^*)$:

$$f(x^*, u^*) = \text{[Equation (??)–(??)]}$$

6: Compute J:

$$J = \|\dot{x}^* - f(x^*, u^*)\|^2.$$

Algorithm 3 Minimize $J(\xi)$

Input: $\alpha^{(0)}$, $\beta^{(0)}$, $\phi^{(0)}$, V_a, R, γ.
for $k = 1$ to N **do**
  $\alpha^{+} = \alpha^{(k-1)} + \epsilon$
  $\beta^{+} = \beta^{(k-1)} + \epsilon$
  $\phi^{+} = \phi^{(k-1)} + \epsilon$
  $\frac{\partial J}{\partial \alpha} = \frac{J(\alpha^{+},\beta^{(k-1)},\phi^{(k-1)}) - J(\alpha^{(k-1)},\beta^{(k-1)},phi^{(k-1)})}{\epsilon}$
  $\frac{\partial J}{\partial \beta} = \frac{J(\alpha^{(k-1)},\beta^{+},\phi^{(k-1)}) - J(\alpha^{(k-1)},\beta^{(k-1)},phi^{(k-1)})}{\epsilon}$
  $\frac{\partial J}{\partial \phi} = \frac{J(\alpha^{(k-1)},\beta^{(k-1)},\phi^{+}) - J(\alpha^{(k-1)},\beta^{(k-1)},phi^{(k-1)})}{\epsilon}$
  $\alpha^{(k)} = \alpha^{(k-1)} - \kappa \frac{\partial J}{\partial \alpha}$
  $\beta^{(k)} = \beta^{(k-1)} - \kappa \frac{\partial J}{\partial \beta}$
  $\phi^{(k)} = \phi^{(k-1)} - \kappa \frac{\partial J}{\partial \phi}$
end for

5.3 Transfer Functions

Transfer functions for the lateral dynamics are derived in Section 5.3.1 and transfer functions for the longitudinal dynamics are derived in Section 5.3.2.

5.3.1 Lateral Transfer Functions

In this section we will derive transfer function models for the lateral dynamics. The variables of interest are the roll angle ϕ, the roll rate p, the heading angle ψ and the yaw rate r. The control surfaces used to influence the lateral dynamics are the ailerons δ_a, and the rudder δ_r. The ailerons are primarily used to influence the roll rate p, while the rudder is primarily used to maintain a coordinated turn condition.

Our first task is to derive a transfer function from the the ailerons δ_a to the roll angle ϕ. From Equation (**??**) we have

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

Since in most flight condition θ 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_{ϕ_1} as a disturbance gives

$$\dot{\phi} = p + d_{\phi_1}. \tag{5.19}$$

Differentiating (5.19) and using Equation (**??**) we get

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

where

$$a_{\phi_1} \triangleq -\frac{1}{2}\rho V_a^2 S b C_{p_p} \frac{b}{2V_a} \tag{5.20}$$

$$a_{\phi_2} \triangleq \frac{1}{2}\rho V_a^2 S b C_{p_{\delta_a}} \tag{5.21}$$

$$d_{\phi_2} \triangleq \Gamma_1 pq - \Gamma_2 qr + \frac{1}{2}\rho V_a^2 S b \left[C_{p_0} + C_{p_\beta}\beta - C_{p_p}\frac{b}{2V_a}(d_{\phi_1}) + C_{p_r}\frac{br}{2V_a} + C_{p_{\delta_r}}\delta_r \right] + \dot{d}_{\phi_1}, \tag{5.22}$$

where d_{ϕ_2} is considered a disturbance on the system.

In the Laplace domain we have

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

A block diagram is shown in Figure 5.1, where the inputs to the block diagram are the ailerons δ_a and the disturbance d_{ϕ_2}.

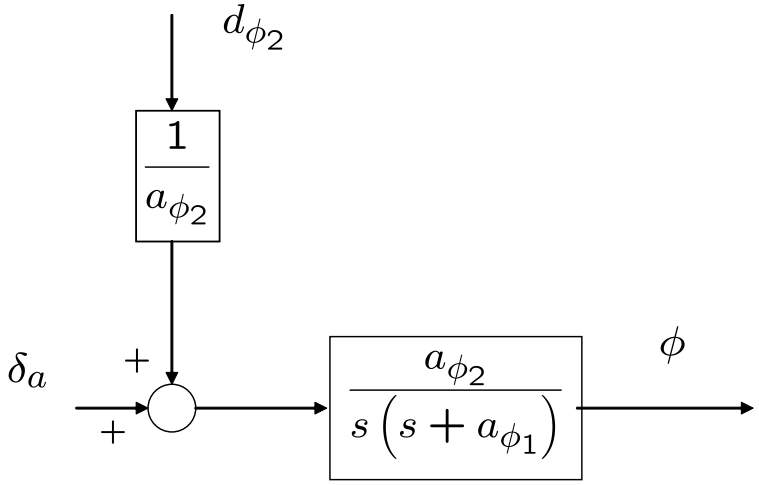


Figure 5.1: Block diagram for roll dynamics. The inputs are the ailerons δ_a and the disturbance d_{ϕ_2}.

We can also derive a transfer function from the roll angle ϕ to the heading angle ψ. From Equation (**??**) we have that in a coordinated turn

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

This equation can be rewritten as

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

where

$$d_\psi = \tan\phi - \phi$$

is a disturbance. In the Laplace domain we have

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

Therefore the block diagram for the lateral dynamics is shown in Figure 5.2.

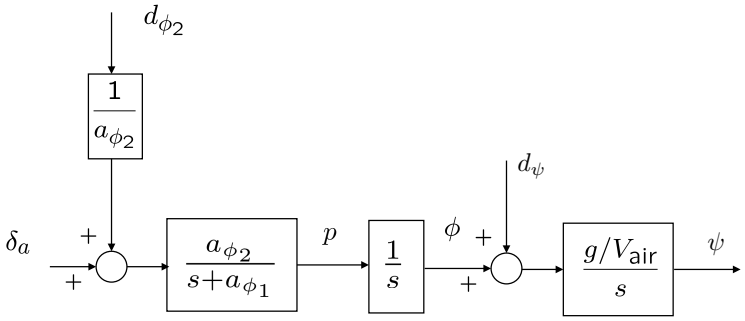


Figure 5.2: Block diagram for lateral dynamics. The roll rate p is shown explicitly because it can be obtained directly from the rate gyros and will be used as a feedback signal in Chapter 6.

An important part of the autopilot design is to maintain a zero side slip velocity, or zero lateral velocity relative to the airmass. To derive the relevant transfer function we will assume that the ambient wind is constant, which implies that $\dot{v}_r = \dot{v} - \dot{v}_w = \dot{v}$. Therefore, from Equation (**??**) we have

$$\begin{aligned}\dot{v}_r &= pw - ru + g\cos\theta\sin\phi + \frac{\rho V_a^2 S}{2\mathsf{m}}\left[C_{Y_0} + C_{Y_\beta}\beta + C_{Y_p}\frac{bp}{2V_a} + C_{Y_r}\frac{br}{2V_a} + C_{Y_{\delta_a}}\delta_a + C_{Y_{\delta_r}}\delta_r\right] \\ &= -a_{\beta_1}v_r + a_{\beta_2}\delta_r + d_v,\end{aligned}$$

where

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

In the Laplace domain we have

$$v_r(s) = \frac{a_{\beta_2}}{s + a_{\beta_1}} \left(\delta_r(s) + d_\beta(s)\right). \tag{5.23}$$

5.3.2 Longitudinal Transfer Functions

In this section we will derive transfer function models for the longitudinal dynamics. The variables of interest are the pitch angle θ, 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 δ_e, and the throttle δ_t. The elevator will be used to directly influence the pitch angle θ. 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 trottle 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 δ_e and the pitch angle θ. From Equation (**??**) 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_{θ_1} is small for small roll angles ϕ. Differentiating we get

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

Using Equation (**??**) and the relationship $\theta = \alpha + \gamma$, where γ is the flight path angle, we get

$$\begin{aligned}
\ddot{\theta} &= \Gamma_6(r^2 - p^2) + \Gamma_5 pr + \frac{\rho V_a^2 \bar{c} S}{2J_y}\left[C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{\bar{c}q}{2V_a} + C_{m_{\delta_e}}\delta_e\right] + \dot{d}_{\theta_1} \\
&= \Gamma_6(r^2 - p^2) + \Gamma_5 pr + \frac{\rho V_a^2 \bar{c} S}{2J_y}\left[C_{m_0} + C_{m_\alpha}(\theta - \gamma) + C_{m_q}\frac{\bar{c}}{2V_a}(\dot{\theta} - d_{\theta 1}) + C_{m_{\delta_e}}\delta_e\right] + \dot{d}_{\theta_1} \\
&= \left(\frac{\rho V_a^2 \bar{c} S}{2J_y} C_{m_q}\frac{\bar{c}}{2V_a}\right)\dot{\theta} + \left(\frac{\rho V_a^2 \bar{c} S}{2J_y} C_{m_\alpha}\right)\theta + \left(\frac{\rho V_a^2 \bar{c} S}{2J_y} C_{m_{\delta_e}}\right)\delta_e + \Big\{\Gamma_6(r^2 - p^2) \\
&\qquad + \Gamma_5 pr + \frac{\rho V_a^2 \bar{c} S}{J_y}\left[C_{m_0} - C_{m_\alpha}\gamma - C_{m_q}\frac{\bar{c}}{2V_a}d_{\theta_1}\right] + \dot{d}_{\theta_1}\Big\} \\
&= -a_{\theta_1}\dot{\theta} - a_{\theta_2}\theta + a_{\theta_3}\delta_e + d_{\theta_2},
\end{aligned}$$

where

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

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

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

Note that in straight and level flight, $r = p = \phi = \gamma = 0$ In addition, airframes are usually designed such 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 5.3. The model shown in Figure 5.3 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.

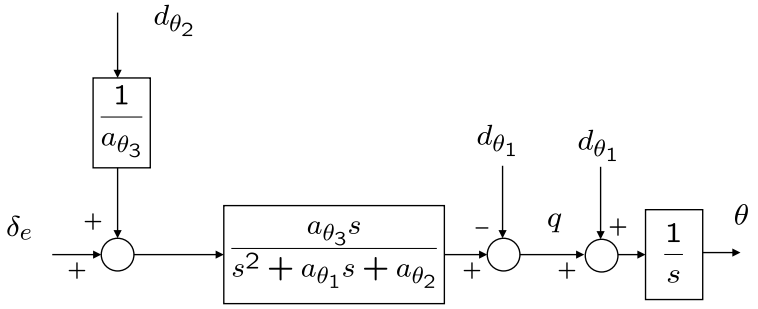


Figure 5.3: Block diagram for the transfer function from the elevator to the pitch angle. The pitch rate q is shown explicitly because it is available from the rate gyros and will be used as a feedback signal in Chapter 6.

For a constant airspeed, the pitch angle directly influences the climb rate of the airframe. Therefore, we can develop a transfer function from pitch angle to altitude. From Equation (**??**) we have

$$
\begin{aligned}
\dot{h} &= u \sin\theta - v \sin\phi \cos\theta - w \cos\phi \cos\theta \\
&= V_a \theta + (u \sin\theta - V_a \theta) - v \sin\phi \cos\theta - w \cos\phi \cos\theta \\
&= V_a \theta + d_h,
\end{aligned} \tag{5.24}
$$

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 $v \approx 0$, $w \approx 0$, $u \approx V_a$, $\phi \approx 0$ and θ is small we have $d_h \approx 0$.

If we assume that V_a is constant and that θ is the input, then in the Laplace domain Equation (5.24) becomes

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

and the resulting block diagram for the longitudinal dynamics from the elevator to the altitude is shown in Figure 5.4.

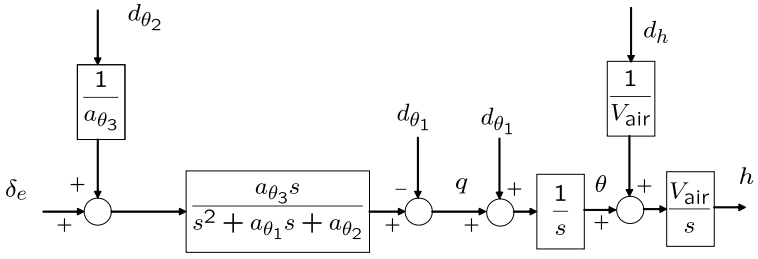


Figure 5.4: Block diagram for longitudinal dynamics.

Alternatively, if the pitch angle is 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 θ constant in Equation (5.24) and consider V_a as the input to obtain

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

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. Alternatively, if the throttle is held constant, then pitching the nose down will decrease the lift causing the aircraft to accelerate downward and thereby increasing its airspeed.

To complete longitudinal models, we derive the transfer functions from throttle and pitch angle to airspeed. Toward that objective note that $V_a = \sqrt{u_r^2 + v_r^2 + w_r^2}$ which implies that

$$\dot{V}_a = \frac{u_r \dot{u}_r + v_r \dot{v}_r + w_r \dot{w}_r}{V_a}.$$

Using Equation (2.7) we get

$$\begin{aligned}\dot{V}_a &= \dot{u}_r \cos\alpha\cos\beta + \dot{v}_r \sin\beta + \dot{w}_r \sin\alpha\cos\beta \\ &= \dot{u}_r \cos\alpha + \dot{w}_r \sin\alpha + d_{V_1},\end{aligned} \tag{5.25}$$

where

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

Note that when $\beta = 0$, we have $d_{V1} = 0$. Substituting Equations (**??**) and (**??**) in Equation (5.25) we obtain

$$\begin{aligned}\dot{V}_a = \cos\alpha\Bigg\{ rv_r - qw + r - g\sin\theta + \frac{\rho V_a^2 S}{2\mathsf{m}}\Big[-C_D\cos\alpha + C_L\sin\alpha + (-C_{D_q}\cos\alpha + C_{L_q}\sin\alpha)\frac{\bar{c}q}{2V_a} \\ + (-C_{D_{\delta_e}}\cos\alpha + C_{L_{\delta_e}}\sin\alpha)\delta_e\Big]\Bigg\} + \frac{\rho S_{\text{prop}} C_{\text{prop}}}{2\mathsf{m}}\left[(k\delta_t)^2 - V_a^2\right] \\ + \sin\alpha\Bigg\{ qu_r - pv_r + g\cos\theta\cos\phi + \frac{\rho V_a^2 S}{2\mathsf{m}}\Big[-C_D\sin\alpha - C_L\cos\alpha + (-C_{D_q}\sin\alpha - C_{L_q}\cos\alpha)\frac{\bar{c}q}{2V_a} \\ + (-C_{D_{\delta_e}}\sin\alpha - C_{L_{\delta_e}}\cos\alpha)\delta_e\Big]\Bigg\} + d_{V_1}.\end{aligned}$$

Using Equations (2.7) and the linear approximation $C_D(\alpha) \approx C_{D_0} + C_{D_\alpha}\alpha$ and simplifying we get

$$\begin{aligned}\dot{V}_a &= rV_a\cos\alpha\sin\beta - pV_a\sin\alpha\sin\beta \\ &\quad - g\cos\alpha\sin\theta + g\sin\alpha\cos\theta\cos\phi \\ &\quad + \frac{\rho V_a^2 S}{2\mathsf{m}}\left[-C_D(\alpha) - C_{D_\alpha}\alpha - C_{D_q}\frac{\bar{c}q}{2V_a} - C_{D_{\delta_e}}\delta_e\right] \\ &\quad + \frac{\rho S_{\text{prop}} C_{\text{prop}}}{2\mathsf{m}}\left[(k\delta_t)^2 - V_a^2\right] + d_{V_1} \\ &= (rV_a\cos\alpha - pV_a\sin\alpha)\sin\beta \\ &\quad - g\sin(\theta-\alpha) - g\sin\alpha\cos\theta(1-\cos\phi) \\ &\quad + \frac{\rho V_a^2 S}{2\mathsf{m}}\left[-C_{D_0} - C_{D_\alpha}\alpha - C_{D_q}\frac{\bar{c}q}{2V_a} - C_{D_{\delta_e}}\delta_e\right] \\ &\quad + \frac{\rho S_{\text{prop}} C_{\text{prop}}}{2\mathsf{m}}\left[(k\delta_t)^2 - V_a^2\right] + d_{V_1} \\ &= -g\sin\gamma + \frac{\rho V_a^2 S}{2\mathsf{m}}\left[-C_{D_0} - C_{D_\alpha}\alpha - C_{D_q}\frac{\bar{c}q}{2V_a} - C_{D_{\delta_e}}\delta_e\right] \\ &\quad + \frac{\rho S_{\text{prop}} C_{\text{prop}}}{2\mathsf{m}}\left[(k\delta_t)^2 - V_a^2\right] + d_{V_2},\end{aligned} \tag{5.26}$$

where

$$d_{V_2} = (rV_a \cos\alpha - pV_a \sin\alpha)\sin\beta - g\sin\alpha\cos\theta(1-\cos\phi) + d_{V_1}.$$

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

When considering airspeed V_a, there are two inputs of interest: the throttle setting δ_t and the pitch angle θ. Since Equation (5.26) is nonlinear in V_a and δ_t, we must first linearize Equation (5.26) before we can find the desired transfer functions.

Letting $\bar{V}_a \stackrel{\triangle}{=} V_a - V_a^*$ be the deviation of V_a from trim, $\bar{\theta} \stackrel{\triangle}{=} \theta - \theta^*$ be the deviation of θ from trim, and $\bar{\delta}_t \stackrel{\triangle}{=} \delta_t - \delta_t^*$ be the deviation of the throttle from trim, then Equation (5.26) can be linearized around the wings-level, constant-altitude ($\gamma^* = 0$) trim condition to give

$$\begin{aligned}\dot{\bar{V}}_a &= -g\cos(\theta^* - \alpha^*)\bar{\theta} + \left\{\frac{\rho V_a^* S}{\mathsf{m}}\left[-C_{D_0} - C_{D_\alpha}\alpha^* - C_{D_{\delta_e}}\delta_e^*\right] - \frac{\rho S_{\text{prop}}}{\mathsf{m}}C_{\text{prop}}V_a^*\right\}\bar{V}_a \\ &\qquad + \left[\frac{\rho S_{\text{prop}}}{\mathsf{m}}C_{\text{prop}}k^2\delta_t^*\right]\bar{\delta}_t + d_V \\ &= -a_{V_1}\bar{V}_a + a_{V_2}\bar{\delta}_t - a_{V_3}\bar{\theta} + d_V, \end{aligned} \tag{5.27}$$

where

$$\begin{aligned} a_{V_1} &= \frac{\rho V_a^* S}{\mathsf{m}}\left[C_{D_0} + C_{D_\alpha}\alpha^* + C_{D_{\delta_e}}\delta_e^*\right] + \frac{\rho S_{\text{prop}}}{\mathsf{m}}C_{\text{prop}}V_a^* \\ a_{V_2} &= \frac{\rho S_{\text{prop}}}{\mathsf{m}}C_{\text{prop}}k^2\delta_t^*, \\ a_{V_3} &= g\cos(\theta^* - \alpha^*), \end{aligned}$$

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

$$\bar{V}_a(s) = \frac{1}{s + a_{V_1}}\left(a_{V_2}\bar{\delta}_t(s) - a_{V_3}\bar{\theta}(s) + d_V(s)\right) \tag{5.28}$$

and the associated block diagram is shown in Figure 5.5.

5.4 Linear State Space Models

In this section we will derive linear state space models for both longitudinal and lateral motion by linearizing Equations (**??**)–(**??**) about trim conditions. Section 5.4.1 discusses general linearization techniques. Section 5.4.2 derives the state space equations for the lateral dynamics and Section 5.4.3 derives the state space equations for the longitudinal dynamics. Section 5.4.4 describes numerical techniques for computing the state space equations from a nonlinear simulation model. Finally Section 5.4.5 describes the reduced order modes including the short period mode, the phugoid mode, the dutch roll mode, and the spiral divergence mode.

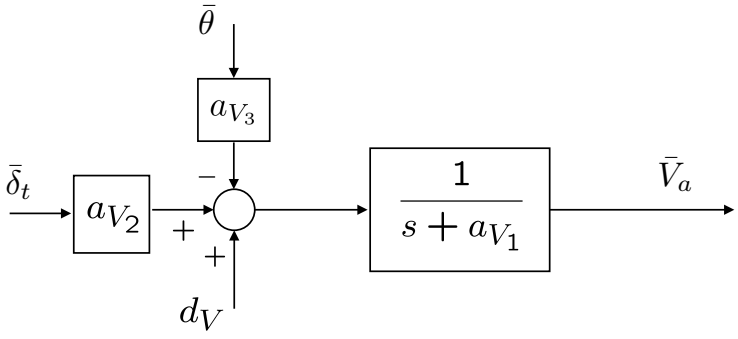


Figure 5.5: Block diagram for airspeed dynamics linearized about trim conditions. The inputs are either the deviation of the pitch angle from trim, or the deviation of the throttle from trim.

5.4.1 Linearization

The UAV dynamics can be represented by the general nonlinear system of equations

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

where $x \in \mathbb{R}^{12}$ is the state, $u \in \mathbb{R}^4$ is the control vector, and f is given by Equations (**??**)–(**??**). Suppose that using the techniques discussed in Section 5.2, a trim input u^* and state x^* are found such that

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

Letting $\bar{x} \stackrel{\triangle}{=} x - x^*$ 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

$$\begin{aligned}
\dot{\bar{x}} &= f(x^*, u^*) + \frac{\partial f(x^*, u^*)}{\partial x}\bar{x} + \frac{\partial f(x^*, u^*)}{\partial u}\bar{u} + H.O.T - f(x^*, u^*) \\
&\approx \frac{\partial f(x^*, u^*)}{\partial x}\bar{x} + \frac{\partial f(x^*, u^*)}{\partial u}\bar{u}.
\end{aligned} \tag{5.29}$$

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

5.4.2 Lateral State-space Equation

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

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

and the input vector is defined as

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

Expressing Equations (**??**), (**??**), (**??**), (**??**), and (**??**) in terms of x_{lat} and u_{lat} we get

$$\begin{aligned}
\dot{v} &= pw - ru + g\cos\theta\sin\phi + \frac{\rho\sqrt{u^2+v^2+w^2}S}{2\mathsf{m}}\frac{b}{2}\Big[C_{Y_p}p + C_{Y_r}r\Big] \\
&\quad + \frac{\rho(u^2+v^2+w^2)S}{2\mathsf{m}}\Big[C_{Y_0} + C_{Y_\beta}\tan^{-1}\left(\frac{v}{\sqrt{u^2+w^2}}\right) + C_{Y_{\delta_a}}\delta_a + C_{Y_{\delta_r}}\delta_r\Big] \qquad (5.30)\\
\dot{p} &= \Gamma_1 pq - \Gamma_2 qr + \frac{\rho\sqrt{u^2+v^2+w^2}S}{2}\frac{b^2}{2}\Big[C_{p_p}p + C_{p_r}r\Big] \\
&\quad + \frac{1}{2}\rho(u^2+v^2+w^2)Sb\Big[C_{p_0} + C_{p_\beta}\tan^{-1}\left(\frac{v}{\sqrt{u^2+w^2}}\right) + C_{p_{\delta_a}}\delta_a + C_{p_{\delta_r}}\delta_r\Big] \qquad (5.31)\\
\dot{r} &= \Gamma_7 pq - \Gamma_1 qr + \frac{\rho\sqrt{u^2+v^2+w^2}S}{2}\frac{b^2}{2}\Big[C_{r_p}p + C_{r_r}r\Big] \\
&\quad + \frac{1}{2}\rho(u^2+v^2+w^2)Sb\Big[C_{r_0} + C_{r_\beta}\tan^{-1}\left(\frac{v}{\sqrt{u^2+w^2}}\right) + C_{r_{\delta_a}}\delta_a + C_{r_{\delta_r}}\delta_r\Big] \qquad (5.32)\\
\dot{\phi} &= p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \qquad (5.33)\\
\dot{\psi} &= q\sin\phi\sec\theta + r\cos\phi\sec\theta, \qquad (5.34)
\end{aligned}$$

where we have used the 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 (5.30)–(5.34) are given by

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

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

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

$$\begin{pmatrix} \dot{\bar{v}} \\ \dot{\bar{p}} \\ \dot{\bar{r}} \\ \dot{\bar{\phi}} \\ \dot{\bar{\psi}} \end{pmatrix} = \begin{pmatrix} Y_v & Y_p & Y_r & g\cos\theta^*\cos\phi^* & 0 \\ L_v & L_p & L_r & 0 & 0 \\ N_v & N_p & N_r & 0 & 0 \\ 0 & 1 & \cos\phi^*\tan\theta^* & q^*\cos\phi^*\tan\theta^* - r^*\sin\phi^*\tan\theta^* & 0 \\ 0 & 0 & \cos\phi^*\sec\theta^* & p^*\cos\phi^*\sec\theta^* - r^*\sin\phi^*\sec\theta^* & 0 \end{pmatrix} \begin{pmatrix} \bar{v} \\ \bar{p} \\ \bar{r} \\ \bar{\phi} \\ \bar{\psi} \end{pmatrix} + \begin{pmatrix} Y_{\delta_a} & Y_{\delta_r} \\ L_{\delta_a} & L_{\delta_r} \\ N_{\delta_a} & N_{\delta_r} \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \bar{\delta}_a \\ \bar{\delta}_r, \end{pmatrix} \tag{5.35}$$

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 (2.7) we have

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

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

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

which implies that

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

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

Table 5.1: Lateral Coefficients State Space Models.

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

$$\begin{pmatrix} \dot{\bar{\beta}} \\ \dot{\bar{p}} \\ \dot{\bar{r}} \\ \dot{\bar{\phi}} \\ \dot{\bar{\psi}} \end{pmatrix} = \begin{pmatrix} Y_v & \frac{Y_p}{V_a^* \cos\beta^*} & \frac{Y_r}{V_a^* \cos\beta^*} & \frac{g\cos\theta^*\cos\phi^*}{V_a^*\cos\beta^*} & 0 \\ L_v V_a^* \cos\beta^* & L_p & L_r & 0 & 0 \\ N_v V_a^* \cos\beta^* & N_p & N_r & 0 & 0 \\ 0 & 1 & \cos\phi^*\tan\theta^* & q^*\cos\phi^*\tan\theta^* & \\ & & & -r^*\sin\phi^*\tan\theta^* & 0 \\ 0 & 0 & \cos\phi^*\sec\theta^* & p^*\cos\phi^*\sec\theta^* & \\ & & & -r^*\sin\phi^*\sec\theta^* & 0 \end{pmatrix} \begin{pmatrix} \bar{\beta} \\ \bar{p} \\ \bar{r} \\ \bar{\phi} \\ \bar{\psi} \end{pmatrix} + \begin{pmatrix} \frac{Y_{\delta_a}}{V_a^*\cos\beta^*} & \frac{Y_{\delta_r}}{V_a^*\cos\beta^*} \\ L_{\delta_a} & L_{\delta_r} \\ N_{\delta_a} & N_{\delta_r} \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \bar{\delta}_a \\ \bar{\delta}_r \end{pmatrix}. \tag{5.36}$$

5.4.3 Longitudinal State-space Equation

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

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

and the input vector is defined as

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

Expressing Equations (**??**), (**??**), (**??**), (**??**), and (**??**) in terms of x_{lon} and u_{lon} we get

$$\dot{u} = rv - qw - g\sin\theta + \frac{\rho V_a^2 S}{2\mathsf{m}}\left[C_{X_0} + C_{X_\alpha}\alpha + C_{X_q}\frac{\bar{c}q}{2V_a} + C_{X_{\delta_e}}\delta_e\right] + \frac{\rho S_{\text{prop}} C_{\text{prop}}}{2\mathsf{m}}\left[(k\delta_t)^2 - V_a^2\right]$$

$$\dot{w} = qu - pv + g\cos\theta\cos\phi + \frac{\rho V_a^2 S}{2\mathsf{m}}\left[C_{Z_0} + C_{Z_\alpha}\alpha + C_{Z_q}\frac{\bar{c}q}{2V_a} + C_{Z_{\delta_e}}\delta_e\right]$$

$$\dot{q} = \frac{J_{xz}}{J_y}(r^2 - p^2) + \frac{J_z - J_x}{J_y}pr + \frac{1}{2J_y}\rho V_a^2 \bar{c} S\left[C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{\bar{c}q}{2V_a} + C_{m_{\delta_e}}\delta_e\right]$$

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

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

Assuming that the lateral states are zero, i.e., set $\phi = p = r = \beta = v = 0$ and substituting

$$\alpha = \tan^{-1}\left(\frac{w}{u}\right)$$

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

from Equation (2.7) gives

$$\dot{u} = -qw - g\sin\theta + \frac{\rho(u^2+w^2)S}{2\mathsf{m}}\left[C_{X_0} + C_{X_\alpha}\tan^{-1}\left(\frac{w}{u}\right) + C_{X_{\delta_e}}\delta_e\right] + \frac{\rho\sqrt{u^2+w^2}S}{4\mathsf{m}}C_{X_q}\bar{c}q + \frac{\rho S_{\text{prop}}}{2\mathsf{m}}C_{\text{prop}}\left[(k\delta_t)^2 - (u^2+w^2)\right] \tag{5.37}$$

$$\dot{w} = qu + g\cos\theta + \frac{\rho(u^2+w^2)S}{2\mathsf{m}}\left[C_{Z_0} + C_{Z_\alpha}\tan^{-1}\left(\frac{w}{u}\right) + C_{Z_{\delta_e}}\delta_e\right] + \frac{\rho\sqrt{u^2+w^2}S}{4\mathsf{m}}C_{Z_q}\bar{c}q \tag{5.38}$$

$$\dot{q} = \frac{1}{2J_y}\rho(u^2+w^2)\bar{c}S\left[C_{m_0} + C_{m_\alpha}\tan^{-1}\left(\frac{w}{u}\right) + C_{m_{\delta_e}}\delta_e\right] + \frac{1}{4J_y}\rho\sqrt{u^2+w^2}SC_{m_q}\bar{c}^2q \tag{5.39}$$

$$\dot{\theta} = q \tag{5.40}$$

$$\dot{h} = u\sin\theta - w\cos\theta. \tag{5.41}$$

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

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

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

Note that

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

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

Longitudinal	Formula
X_u	$\frac{u^*\rho S}{\mathsf{m}}\left[C_{X_0}+C_{X_\alpha}\alpha^*+C_{X_{\delta_e}}\delta_e^*\right]-\frac{\rho S w^* C_{X_\alpha}}{2\mathsf{m}}+\frac{\rho S\bar{c}C_{X_q}u^*q^*}{4mV_a^*}-\frac{\rho S_{\text{prop}}C_{\text{prop}}u^*}{\mathsf{m}}$
X_w	$-q^*+\frac{w^*\rho S}{\mathsf{m}}\left[C_{X_0}+C_{X_\alpha}\alpha^*+C_{X_{\delta_e}}\delta_e^*\right]+\frac{\rho S\bar{c}C_{X_q}w^*q^*}{4mV_a^*}+\frac{\rho SC_{X_\alpha}u^*}{2\mathsf{m}}-\frac{\rho S_{\text{prop}}C_{\text{prop}}w^*}{\mathsf{m}}$
X_q	$-w^*+\frac{\rho V_a^* SC_{X_q}\bar{c}}{4\mathsf{m}}$
X_{δ_e}	$\frac{\rho V_a^{*2}SC_{X_{\delta_e}}}{2\mathsf{m}}$
X_{δ_t}	$\frac{\rho S_{prop}C_{prop}k^2\delta_t^*}{\mathsf{m}}$
Z_u	$q^*+\frac{u^*\rho S}{\mathsf{m}}\left[C_{Z_0}+C_{Z_\alpha}\alpha^*+C_{Z_{\delta_e}}\delta_e^*\right]-\frac{\rho SC_{Z_\alpha}w^*}{2\mathsf{m}}+\frac{u^*\rho SC_{Z_q}\bar{c}q^*}{4mV_a^*}$
Z_w	$\frac{w^*\rho S}{\mathsf{m}}\left[C_{Z_0}+C_{Z_\alpha}\alpha^*+C_{Z_{\delta_e}}\delta_e^*\right]+\frac{\rho SC_{Z_\alpha}u^*}{2\mathsf{m}}+\frac{\rho w^* S\bar{c}C_{Z_q}q^*}{4mV_a^*}$
Z_q	$u^*+\frac{\rho V_a^* SC_{Z_q}\bar{c}}{4\mathsf{m}}$
Z_{δ_e}	$\frac{\rho V_a^{*2}SC_{Z_{\delta_e}}}{2\mathsf{m}}$
M_u	$\frac{u^*\rho S\bar{c}}{J_y}\left[C_{m_0}+C_{m_\alpha}\alpha^*+C_{m_{\delta_e}}\delta_e^*\right]-\frac{\rho S\bar{c}C_{m_\alpha}w^*}{2J_y}+\frac{\rho S\bar{c}^2C_{m_q}q^*u^*}{4J_yV_a^*}$
M_w	$\frac{w^*\rho S\bar{c}}{J_y}\left[C_{m_0}+C_{m_\alpha}\alpha^*+C_{m_{\delta_e}}\delta_e^*\right]+\frac{\rho S\bar{c}C_{m_\alpha}u^*}{2J_y}+\frac{\rho S\bar{c}^2C_{m_q}q^*w^*}{4J_yV_a^*}$
M_q	$\frac{\rho V_a^* S\bar{c}^2C_{m_q}}{4J_y}$
M_{δ_e}	$\frac{\rho V_a^{*2}S\bar{c}C_{m_{\delta_e}}}{2J_y}$

Table 5.2: Longitudinal Coefficients State Space Models.

where we have used Equation (2.8) and the fact that $v=0$. Working out the derivatives we get that the linearized state space equations are

$$\begin{pmatrix}\dot{\bar{u}}\\ \dot{\bar{w}}\\ \dot{\bar{q}}\\ \dot{\bar{\theta}}\\ \dot{\bar{h}}\end{pmatrix}=\begin{pmatrix}X_u & X_w & X_q & -g\cos\theta^* & 0\\ Z_u & Z_w & Z_q & -g\sin\theta^* & 0\\ M_u & M_w & M_q & 0 & 0\\ 0 & 0 & 1 & 0 & 0\\ \sin\theta^* & -\cos\theta^* & 0 & u^*\cos\theta^*+w^*\sin\theta^* & 0\end{pmatrix}\begin{pmatrix}\bar{u}\\ \bar{w}\\ \bar{q}\\ \bar{\theta}\\ \bar{h}\end{pmatrix}+\begin{pmatrix}X_{\delta_e} & X_{\delta_t}\\ Z_{\delta_e} & 0\\ M_{\delta_e} & 0\\ 0 & 0\\ 0 & 0\end{pmatrix}\begin{pmatrix}\bar{\delta}_e\\ \bar{\delta}_t,\end{pmatrix} \tag{5.42}$$

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 (2.7) 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

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

Therefore we can write the the state space equations in terms of $\bar{\alpha}$ instead of $\bar{w}$:

$$\begin{pmatrix} \dot{\bar{u}} \\ \dot{\bar{\alpha}} \\ \dot{\bar{q}} \\ \dot{\bar{\theta}} \\ \dot{\bar{h}} \end{pmatrix} = \begin{pmatrix} X_u & X_w V_a^* \cos\alpha^* & X_q & -g\cos\theta^* & 0 \\ \frac{Z_u}{V_a^*\cos\alpha^*} & Z_w & \frac{Z_q}{V_a^*\cos\alpha^*} & \frac{-g\sin\theta^*}{V_a^*\cos\alpha^*} & 0 \\ M_u & M_w V_a^* \cos\alpha^* & M_q & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \sin\theta^* & -V_a^*\cos\theta^*\cos\alpha^* & 0 & u^*\cos\theta^* + w^*\sin\theta^* & 0 \end{pmatrix} \begin{pmatrix} \bar{u} \\ \bar{\alpha} \\ \bar{q} \\ \bar{\theta} \\ \bar{h} \end{pmatrix} + \begin{pmatrix} X_{\delta_e} & X_{\delta_t} \\ \frac{Z_{\delta_e}}{V_a^*\cos\alpha^*} & 0 \\ M_{\delta_e} & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \bar{\delta}_e \\ \bar{\delta}_t \end{pmatrix}. \tag{5.43}$$

5.4.4 Numerical Approximations of A and B.

Perhaps the best way to find A and B is to approximate $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial u}$ numerically. The i^{th} column of $\frac{\partial f}{\partial x}$ can be approximated as

$$\begin{pmatrix} \frac{\partial f_1}{\partial x_i} \\ \frac{\partial f_2}{\partial x_i} \\ \vdots \\ \frac{\partial f_n}{\partial x_i} \end{pmatrix} (x^*, u^*) \approx \frac{f(x^* + \epsilon e_i, u^*) - f(x^*, u^*)}{\epsilon},$$

where e_i has a one in the i^{th} element and zeros elsewhere. Similarly, the i^{th} column of $\frac{\partial f}{\partial u}$ can be approximated as

$$\begin{pmatrix} \frac{\partial f_1}{\partial u_i} \\ \frac{\partial f_2}{\partial u_i} \\ \vdots \\ \frac{\partial f_n}{\partial u_i} \end{pmatrix} (x^*, u^*) \approx \frac{f(x^*, u^* + \epsilon e_i) - f(x^*, u^*)}{\epsilon}.$$

5.4.5 Reduced Order Modes

The traditional literature on aircraft dynamics and control define several open loop aircraft modes. These include the *short period* mode, the *phugoid* mode, the *rolling* mode, the *spiral* 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 (5.43) to

$$\begin{pmatrix} \dot{\bar{u}} \\ \dot{\bar{\alpha}} \\ \dot{\bar{q}} \\ \dot{\bar{\theta}} \end{pmatrix} = \begin{pmatrix} X_u & X_w V_a^* \cos\alpha^* & X_q & -g\cos\theta^* \\ \frac{Z_u}{V_a^*\cos\alpha^*} & Z_w & \frac{Z_q}{V_a^*\cos\alpha^*} & \frac{-g\sin\theta^*}{V_a^*\cos\alpha^*} \\ M_u & M_w V_a^* \cos\alpha & M_q & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \bar{u} \\ \bar{\alpha} \\ \bar{q} \\ \bar{\theta} \end{pmatrix} + \begin{pmatrix} X_{\delta_e} \\ \frac{Z_{\delta_e}}{V_a^*\cos\alpha^*} \\ M_{\delta_e} \\ 0 \end{pmatrix} \bar{\delta}_e. \tag{5.44}$$

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. In the next two subsections, we will approximate the poles due to these modes.

For the short period mode, we will assume that u is constant, i.e., $\bar{u} = \dot{\bar{u}} = 0$. Therefore the state space equations in Equation (5.44) 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

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

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

$$s\left(s^2 + (-Z_w - M_q)\, s + M_q Z_w - M_w Z_q\right) = 0.$$

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

Assume that α is constant, i.e., $\bar{\alpha} = \dot{\bar{\alpha}} = 0$, then $\alpha = \alpha^*$ and Equation (5.44) becomes

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

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

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

Rolling Mode

If we ignore the heading dynamics, and assume a constant pitch angle, i.e., $\bar{\theta} = 0$, then Equation (5.36) becomes

$$\begin{pmatrix} \dot{\bar{\beta}} \\ \dot{\bar{p}} \\ \dot{\bar{r}} \\ \dot{\bar{\phi}} \end{pmatrix} = \begin{pmatrix} Y_v & \frac{Y_p}{V_a^*\cos\beta^*} & \frac{Y_r}{V_a^*\cos\beta^*} & \frac{g\cos\theta^*\cos\phi^*}{V_a^*\cos\beta^*} \\ L_v V_a^* \cos\beta^* & L_p & L_r & 0 \\ N_v V_a^* \cos\beta^* & N_p & N_r & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \bar{\beta} \\ \bar{p} \\ \bar{r} \\ \bar{\phi} \end{pmatrix} + \begin{pmatrix} \frac{Y_{\delta_a}}{V_a^*\cos\beta^*} & \frac{Y_{\delta_r}}{V_a^*\cos\beta^*} \\ L_{\delta_a} & L_{\delta_r} \\ N_{\delta_a} & N_{\delta_r} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \bar{\delta}_a \\ \bar{\delta}_r \end{pmatrix}. \tag{5.45}$$

The dynamics for $\bar{p}$ are obtained from Equation (5.45) 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 Mode

For the spiral mode we assume that $\dot{\bar{p}} = \bar{p} = 0$, and that the rudder command is negligible. Therefore, from the second and third equations in (5.45) we get

$$0 = L_v V_a^* \cos\beta^* \bar{\beta} + L_r \bar{r} + L_{\delta_a}\bar{\delta}_a \tag{5.46}$$

$$\dot{\bar{r}} = N_v V_a^* \cos\beta^* \bar{\beta} + N_r \bar{r} + N_{\delta_a}\bar{\delta}_a. \tag{5.47}$$

Solving (5.46) for $\bar{\beta}$ and substituting into (5.47) 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

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

Therefore the pole of the spiral mode is approximately

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

Dutch Roll Mode

For the dutch roll mode, we neglect the rolling motions and focus on the equations for sideslip and yaw. From Equation (5.45) we have

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

The characteristic equation is given by

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

Therefore, the poles so the dutch roll mode are approximated by

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

Example

RWB: Insert examples, and provide some physical intuition about what these modes are, and how they might feel when riding in an aircraft. Also, how these modes might be excited.

5.5 Chapter Summary

The transfer functions for the lateral dynamics are given by

$$\begin{aligned}\phi(s) &= \left(\frac{a_{\phi_2}}{s(s+a_{\phi_1})}\right)\delta_a(s)\\ \psi(s) &= \frac{g/V_a}{s}\phi(s)\\ v_r(s) &= \frac{a_{\beta_2}}{s+a_{\beta_1}}\delta_r(s).\end{aligned}$$

The transfer functions for the longitudinal dynamics are given by

$$\begin{aligned}\theta(s) &= \left(\frac{a_{\theta_3}}{s^2+a_{\theta_1}s+a_{\theta_2}}\right)\delta_e(s)\\ h(s) &= \left(\frac{V_a}{s}\right)\theta(s)\\ h(s) &= \left(\frac{\theta}{s}\right)V_a(s)\\ \bar{V}_a(s) &= \left(\frac{a_{V_2}}{s+a_{V_1}}\right)\bar{\delta}_t(s)\\ \bar{V}_a(s) &= \left(\frac{-a_{V_3}}{s+a_{V_1}}\right)\bar{\theta}(s).\end{aligned}$$

Notes and References

5.6 Design Project

5.1 The Simulink product comes with two functions that can be used to compute trim conditions and to extract linear equations from a Simulink diagram. Using the Matlab `helpdesk`, read the documentation on the `trim` and `linmod` commands.

5.2 To use the `trim` and `linmod` commands you will need to modify your Simulink diagram to have input ports for the four inputs to the system, and an output port for the states. For

example, the Simulink diagram should look like that shown in Figure 5.6, where the function `Create Output` block converts the state $(p_n, p_e, p_d, u, v, w, \phi, \theta, \psi, p, q, r)$ and the wind vector (w_n, w_e, w_d) to the outputs $(V_a, \alpha, \beta, \phi, \theta, \psi, p, q, r)$. Copy and rename your current

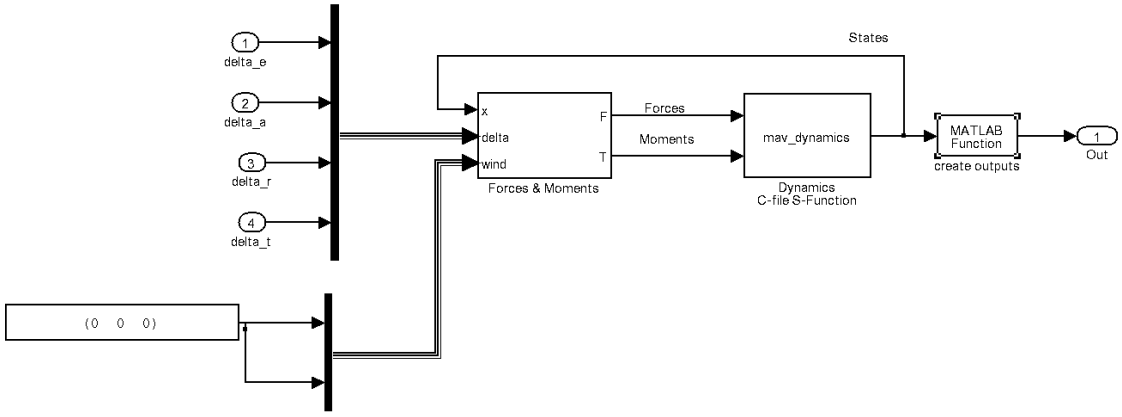


Figure 5.6: Simulink diagram used in the `trim` and `linmod` commands. The inputs must be specified with in-ports, and the output must be specified with out-ports.

Simulink diagram. Modify it so that it has the proper input-output structure as in Figure 5.6.

5.3 Create a Matlab script that computes the trim values for the Simulink simulation developed in Chapters 2–4. The input to the Matlab script should be the desired airspeed V_a, the desired path angle $\pm\gamma$, and the desired turn radius $\pm R$, where $+R$ indicates a right handed turn and $-R$ indicates a left handed turn. The Matlab script should use the command

```
[x_t,u_t,y_t,dx_t] = trim(filename,x0,u0,y0,ix,iu,iy,dx0,idx)
```

where `filename` is the name of the Simulink model. The variable $dx0$ is set according to Equation (5.14). You should also enforce that the output for airspeed remains fixed at V_a and that the output β remains fixed at zero.

5.4 Use the Matlab script to compute the trimmed state and controls for wings-level flight with $V_a = 10$ m/s and $\gamma = 0$ rad. Set the initial states in your original Simulink simulation to the trim state, and the inputs to the trim controls. If the trim algorithm is correct, the MAV states will remain constant during the simulation. Run the trim algorithm for various values of γ. The only variable that should change is the altitude h. Convince yourself that the climb rate is correct.

5.5 Use the Matlab script to compute the trimmed state and controls for constant turns with with $V = 10$ m/s and $R = 50$ m. et the initial states in your original Simulink simulation to the trim state, and the inputs to the trim controls. If the trim algorithm is correct, the UAV states will remain constant during the simulation except for the heading ψ.

5.6 Create a Matlab script that uses the trim values computed in the previous problem to create the transfer functions listed in Section 5.5.

5.7 Create a Matlab script that uses the trim values and the `linmod` command to linearize the Simulink model about the trim condition. The result should be an A matrix that is 12×12, and a B matrix that is 12×4. Extract the lateral state space model given in Equation (5.35) using the commands

```
1      >> A_lat = E_lat A E_lat'
2      >> B_lat = E_lat B E_lat2'
```

where `E_lat` is a 5×12 matrix of ones and zeros that extracts the relevant states, and `E_lat2` is a 2×4 matrix of ones and zeros that extract the relevant inputs. Extract the longitudinal state space model given in Equation (5.42) using a similar method.

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

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

extract ω_n and ζ the two complex conjugate pairs of poles. The pair with the larger ω_n correspond to the short period mode, and the pair with the smaller ω_n correspond to the phugoid mode. The phugoid and short period modes can be excited by starting the simulation in a wings-level, constant-altitude trim condition, and placing an impulse on the elevator. A Simulink diagram to accomplish this is shown in Figure 5.7 Using Figure 5.8 convince yourself that the eigenvalues of `A_lon` adequately predict the short period and phugoid modes.

5.9 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 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

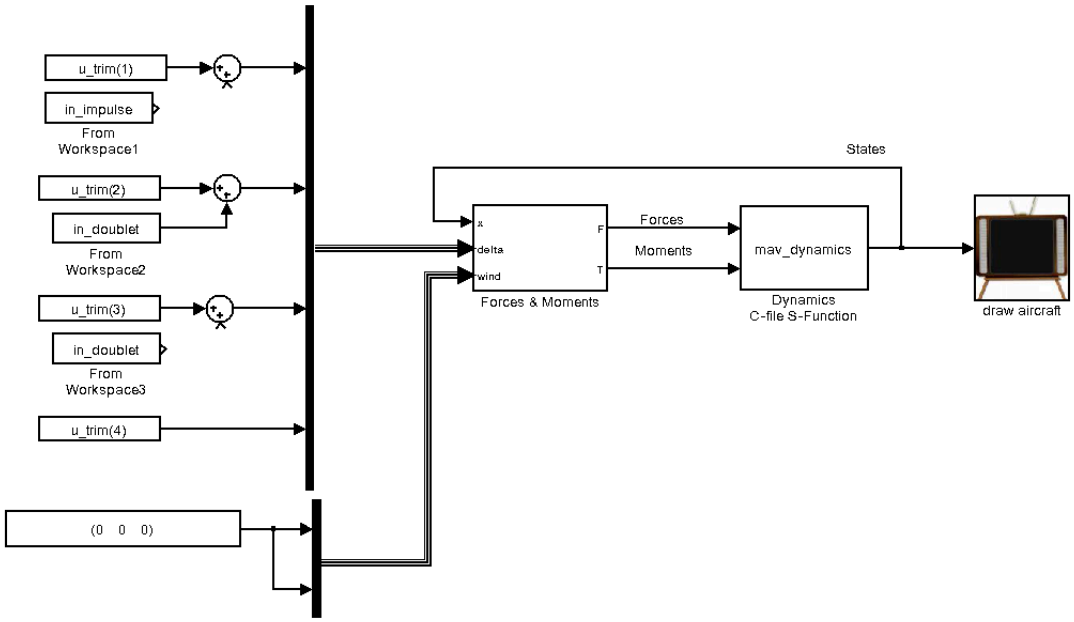


Figure 5.7: Simulink diagram to place an impulse on the elevator. The manual switch is changed quickly from 0 to 1 to simulate the impulse.

trim condition, and placing a unit doublet on the aileron or on the rudder. Using Figure 5.8 convince yourself that the eigenvalues of `A_lat` adequately predict the roll, spiral, and dutch-roll modes.

RWB: Add impulse and doublet to simulation blocks made available on web. Many students did not know how to implement these on their own.

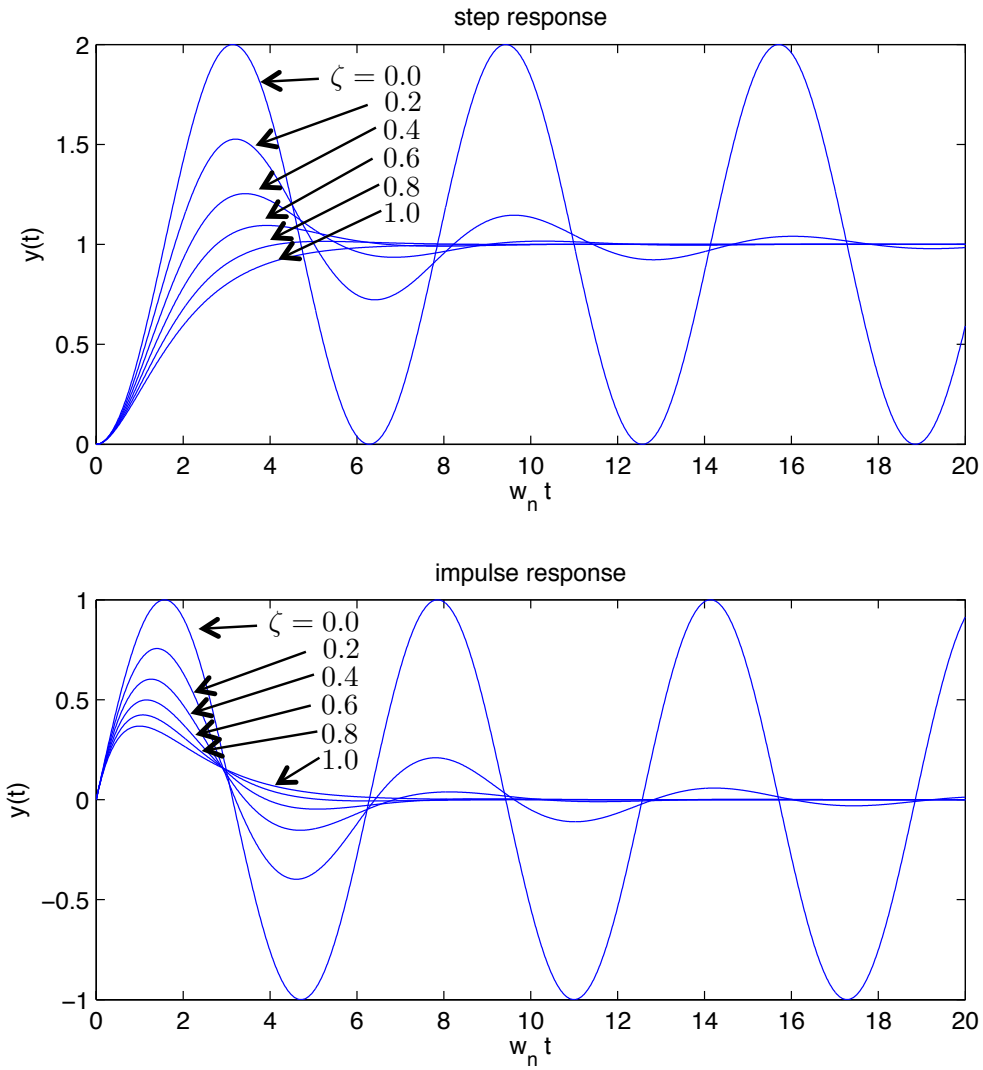


Figure 5.8: Step and Impulse response for a second order system with transfer function equal to $T(s) = \omega_n^2/(s^2 + 2\zeta\omega_n s + \omega_n^2)$.

Chapter 6

Autopilot Design Using Successive Loop Closure

6.1 Successive Loop Closure

In general terms, an autopilot is defined as 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 complicated enough to control position (altitude, latitude, longitude) and attitude (roll, pitch, yaw) during the various phases of flight (e.g., take-off, ascent, level flight, descent, approach, and 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 is the portion of the MAV control system that resides on board the MAV.

The primary goal in autopilot design is to control the inertial position (p_n, p_e, h) and attitude (ϕ, θ, ψ) of the MAV. For most flight maneuvers of interest, autopilots designed based on the assumption of decoupled dynamics yield good performance. In the autopilot design discussion that follows, we will assume that the longitudinal dynamics (forward speed, pitching, climbing/descending motions) are decoupled from the lateral-directional 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 each successive loop bandwidth a factor of 5 to 10 lower in frequency.

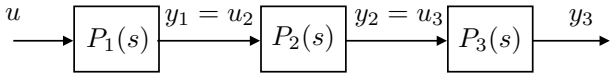


Figure 6.1: Open-loop transfer function modeled as a cascade of three transfer functions.

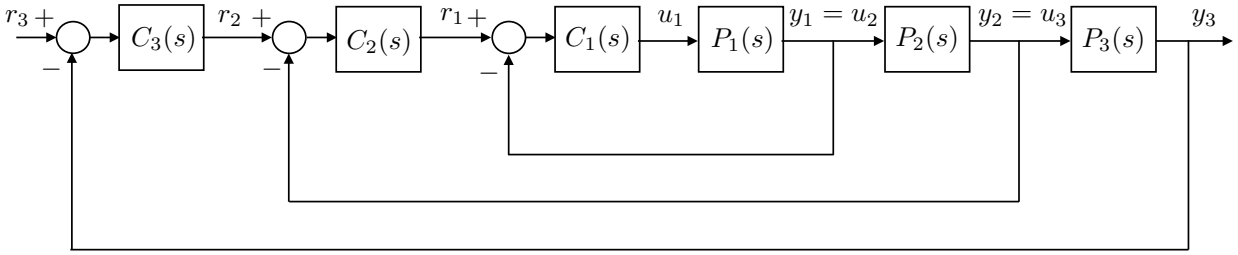


Figure 6.2: Three stage successive loop closure design.

Examining the inner loop shown in Figure 6.2, the goal is to design a closed-loop system from r_1 to y_1 having a bandwidth ω_{BW1}. The key assumption that we will make is that for frequencies well below ω_{BW1}, the closed-loop transfer function $y_1(s)/r_1(s)$ can be modeled as a gain of 1. This is depicted schematically in Figure 6.3. With the inner-loop transfer function modeled as a gain of 1, design of the second loop is simplified because it only includes the plant transfer function $P_2(s)$ and the compensator $C_2(s)$. The critical design step in closing the loops successively is to design the bandwidth of the next loop so that it is a factor of 5 to 10 slower than the preceding loop. In this case, we require $\omega_{\text{BW2}} < \frac{1}{5}\omega_{\text{BW1}}$. This ensures that our unity gain assumption on the inner loop is not violated over the range of frequencies that the middle loop operates.

With the two inner loops operating as designed, $y_2(s)/r_2(s) \approx 1$ and the transfer function from $r_2(s)$ to $y_2(s)$ can be replaced with a gain of 1 for the design of the outermost loop, as show in Figure 6.4. Again, there is a bandwidth constraint on the design of the outer loop: $\omega_{\text{BW3}} < \frac{1}{5}\omega_{\text{BW2}}$. Because each of the plant models $P_1(s)$, $P_2(s)$, $P_3(s)$ are 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.

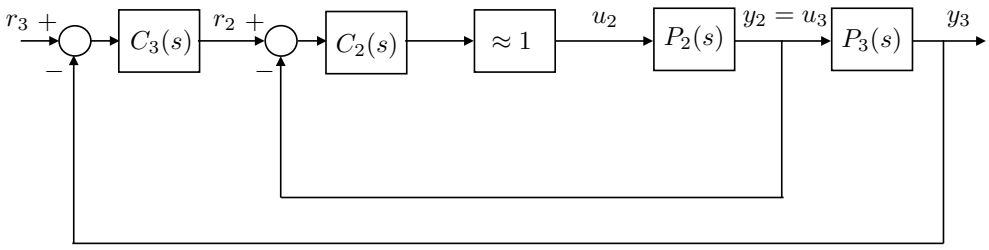


Figure 6.3: Successive loop closure design with inner loop modeled as a unity gain.

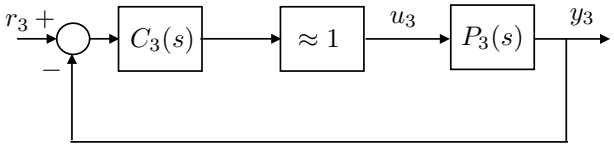


Figure 6.4: Successive loop closure design with two inner loops modeled as a unity gain.

The following sections discuss the design of a lateral-directional autopilot and a longitudinal autopilot. Transfer functions modeling the lateral-directional and longitudinal dynamics were developed in Section 5.3 and will be used to design the autopilots in this chapter.

6.2 Saturation Constraints and Performance

The successive loop closure design process implies that performance of the system is limited by the performance of the inner-most loop. The performance of the inner-most loop is often limited by saturation constraints. For example, in the design of the lateral autopilot, the fact that the ailerons are bounded by a maximum angle implies that the roll rate of the aircraft is limited. The design process is therefore to design the bandwidth of the inner loop to be as large as possible, without violating the saturation constraints, and then design the outer loops to ensure bandwidth separation of the successive loops. In this section, we briefly describe how knowledge of the system transfer function and the saturation constraints can be used to develop performance specifications for second order systems.

The canonical second order linear differential equation with no zeros is given by the standard form

$$\ddot{y} + 2\zeta\omega_n\dot{y} + \omega_n^2 y = \omega_n^2 y^c \tag{6.1}$$

where y^c is the commanded value, ζ is the damping ratio, and ω_n is the natural frequency. The

poles of this system are given by

$$\text{poles} = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2}. \tag{6.2}$$

If $0 \leq \zeta < 1$, then the poles are complex and the system is said to be *under damped*. If $\zeta = 1$, then there are two poles at $-\omega_n$ and the system is said to be *critically dampled*. If $\zeta > 1$, then there are two real poles and the system is said to be *over damped*. In these notes, we will assume that the system is underdamped. Solving Equation (6.1) for a step in y^c of magnitude $|y^c|$ we get

$$y(t) = |y^c| \left[1 - \frac{1}{\sqrt{1-\zeta^2}} e^{-\zeta\omega_n t} \cos\left(\omega_n\sqrt{1-\zeta^2}t - \frac{\zeta}{1-\zeta^2}\right)\right]. \tag{6.3}$$

Differentiating twice the desired closed-loop response (6.3) gives

$$\ddot{y}(t) = |y^c| \frac{\omega_n^2}{\sqrt{1-\zeta^2}} e^{-\zeta\omega_n t} \cos(\text{stuff}).$$

Therefore the acceleration of the desired closed-loop response is bounded by

$$|\ddot{y}| \leq |y^c| \frac{\omega_n^2}{\sqrt{1-\zeta^2}}. \tag{6.4}$$

Now suppose that the linear design model of the plant is second order and described by the equation

$$m\ddot{q} + b\dot{q} + kq = au,$$

where u is physically constrained as $|u| \leq u^{\max}$. The acceleration is bounded by

$$|\ddot{q}| \leq \left|-\frac{b}{m}\dot{q} - \frac{k}{m}q + \frac{a}{m}u\right|.$$

If b, k, and m are all non-negative constants (true for most physical systems), then the potential acceleration is bounded:

$$|\ddot{q}| \leq \frac{a}{m}u^{\max}. \tag{6.5}$$

For a physically realizable design, the acceleration of the desired closed-loop response must be bounded by the physically available acceleration. Therefore from Equations (6.5) and (6.4) we have that

$$|y^c| \frac{\omega_n^2}{\sqrt{1-\zeta^2}} \leq \frac{a}{m}u^{\max}. \tag{6.6}$$

Rearranging, we see that the limit on the natural frequency is

$$\omega_n \leq \sqrt{\frac{au^{\max}\sqrt{1-\zeta^2}}{m\,|y^c|}}. \tag{6.7}$$

6.3 Lateral-directional Autopilot

Figure 6.5 shows the block diagram for a lateral-directional autopilot using successive loop closure. There are five gains associated with the lateral-directional autopilot. The derivative gain k_{d_ϕ} provides roll rate damping at the innermost loop. The roll attitude is regulated with the proportional and integral gains k_{p_ϕ} and k_{i_ϕ}. The yaw attitude is regulated with the proportional and integral gains k_{p_ψ} and k_{i_ψ}. The idea with successive loop closure, is that the gains are successively chosen beginning with the inner loop and working outward. In particular, k_{d_ϕ} and k_{d_ϕ} are usually selected first, k_{i_ϕ} second, and finally k_{p_ψ} and k_{i_ψ}.

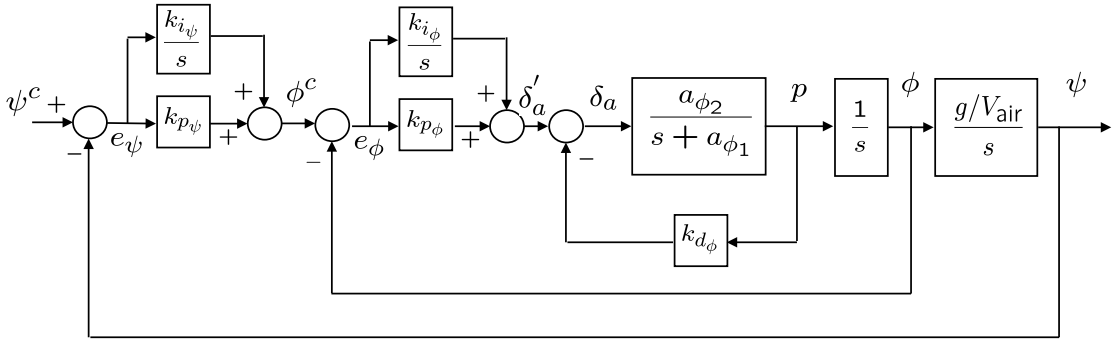


Figure 6.5: Autopilot for lateral-directional control using successive loop closure.

6.3.1 Roll Attitude Loop Design

The two inner loops of the lateral-directional autopilot are used to control roll angle and roll rate as shown in Figure 6.6. If the transfer function coefficients, a_{ϕ_1} and a_{ϕ_2}, are known, then there is

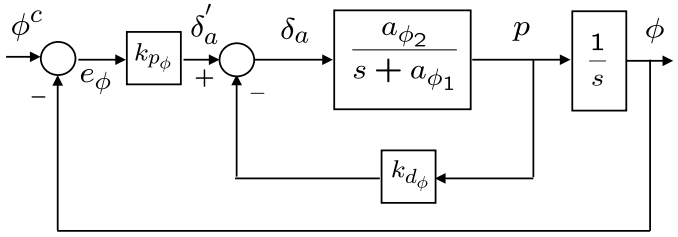


Figure 6.6: Roll attitude hold control loops.

a systematic method for selecting the control gains k_{d_ϕ} and k_{p_ϕ} based on the desired response of

closed-loop dynamics. From Figure 6.6, the transfer function from ϕ^c to ϕ is given by

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

Note that the DC gain is equal to one, and that the closed loop poles are located at

$$s_{1,2} = -\left(\frac{a_{\phi_1} + a_{\phi_2} k_{d_\phi}}{2}\right) \pm \sqrt{\left(\frac{a_{\phi_1} + a_{\phi_2} k_{d_\phi}}{2}\right)^2 - k_{p_\phi} a_{\phi_2}}.$$

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

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

then equating denominator polynomial coefficients, we get

$$\begin{aligned} \omega_{n_\phi}^2 &= k_{p_\phi} a_{\phi_2} \\ 2\zeta_\phi \omega_{n_\phi} &= a_{\phi_1} + a_{\phi_2} k_{d_\phi}. \end{aligned}$$

Solving these expressions for k_{p_ϕ} and k_{d_ϕ} gives

$$k_{p_\phi} = \frac{\omega_{n_\phi}^2}{a_{\phi_2}} \tag{6.8}$$

$$k_{d_\phi} = \frac{2\zeta_\phi \omega_{n_\phi} - a_{\phi_1}}{a_{\phi_2}}. \tag{6.9}$$

Since the roll attitude dynamics are given by

$$\ddot{\phi} + a_{\phi_1} \dot{\phi} = a_{\phi_2} \delta_a,$$

if the saturation constraint on δ_a is $\delta_a^{\max}$, and if we desired maximum control input for a roll angle command of $\phi^{\max}$, then from Equation (6.7), the maximum allowable bandwidth for the system is given by

$$\omega_{n_\phi} = \sqrt{\frac{a_{\phi_2} \delta_a^{\max} \sqrt{1 - \zeta_\phi^2}}{\phi^{\max}}}. \tag{6.10}$$

Integrator on Roll

Note that the open loop transfer function in Figure 6.6 is a type one system, which implies that zero steady-state tracking error in roll should be achievable without an integrator. However, from

Figure 5.1 we see that there is a disturbance that enters at the summing junction before δ_a. This disturbance represents the terms in the dynamics that were neglected in the process of creating the linear, reduced-order model of the roll dynamics. It can also represent physical perturbations to the system, such as those from gusts or turbulence. Figure 6.7 shows the roll loop with the disturbance. Solving for $\phi(s)$ in Figure 6.7 we get

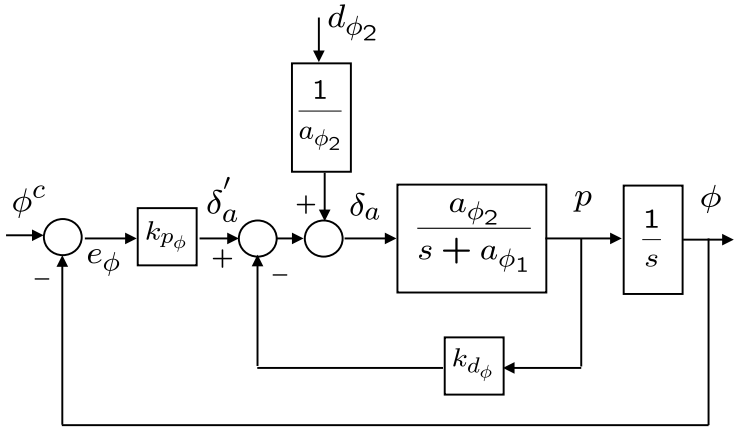


Figure 6.7: Roll attitude hold loop with input disturbance.

$$\phi = \left(\frac{1}{s^2 + (a_{\phi_1} + a_{\phi_2}k_{d_\phi})s + a_{\phi_2}k_{p_\phi}}\right) d_{\phi_2} + \left(\frac{a_{\phi_2}k_{d_\phi}}{s^2 + (a_{\phi_1} + a_{\phi_2}k_{d_\phi})s + a_{\phi_2}k_{p_\phi}}\right) \phi^c.$$

Note that if d_{ϕ_2} is a constant disturbance, i.e., $d_{\phi_2} = A/s$, then from the final value theorem, the steady state error due to d_{ϕ_2} is $\frac{A}{a_{\phi_2}k_{p_\phi}}$. In a constant orbit, p, q, and r will be constants and so d_{ϕ_2} will also be constant, as can be seen from Equation (5.22). Therefore, it is desirable to remove the steady-state error using an integrator.

Figure 6.8 shows the roll attitude hold loop with an integrator added to reject a constant d_{ϕ_2}. Solving for $\phi(s)$ in Figure 6.8 we get

$$\phi = \left(\frac{s}{s^3 + (a_{\phi_1} + a_{\phi_2}k_{d_\phi})s^2 + a_{\phi_2}k_{p_\phi}s + a_{\phi_2}k_{i_\phi}}\right) d_{\phi_2} + \left(\frac{a_{\phi_2}k_{p_\phi}\left(s + \frac{k_{i_\phi}}{k_{p_\phi}}\right)}{s^3 + (a_{\phi_1} + a_{\phi_2}k_{d_\phi})s^2 + a_{\phi_2}k_{p_\phi}s + a_{\phi_2}k_{i_\phi}}\right) \phi^c.$$

Note that in this case, the final value theorem predicts zero steady state error for a constant d_{ϕ_2}. If d_{ϕ_2} is a ramp, i.e., $d_{\phi_2} = A/s^2$, then the steady state error is given by $\frac{A}{a_{\phi_2}k_{i_\phi}}$. If a_{ϕ_1} and a_{ϕ_2} are known, then k_{i_ϕ} can be effectively selected using root locus techniques. The closed loop poles of the system are given by

$$s^3 + (a_{\phi_1} + a_{\phi_2}k_{d_\phi})s^2 + a_{\phi_2}k_{d_\phi}s + a_{\phi_2}k_{i_\phi} = 0,$$

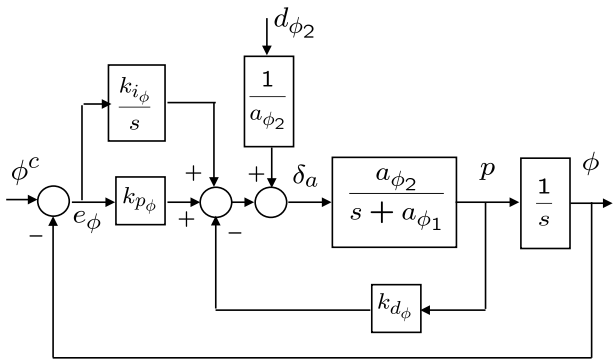


Figure 6.8: Integrator for roll attitude hold.

which can be placed in Evans form as

$$1 + k_{i_\phi} \left(\frac{a_{\phi 2}}{s\left(s^2 + (a_{\phi 1} + a_{\phi 2} k_{d_\phi})s + a_{\phi 2} k_{d_\phi}\right)} \right) = 0.$$

Figure 6.9 shows the root locus of the characteristic equation plotted as a function of k_{i_ϕ}. For small values of gain (less than 4 approximately for this system), the system remains stable. To maintain desirable transient response characteristics for the system, we will choose $k_{i_\phi} = 0.1$. The response of the system shown in Figure 6.10.

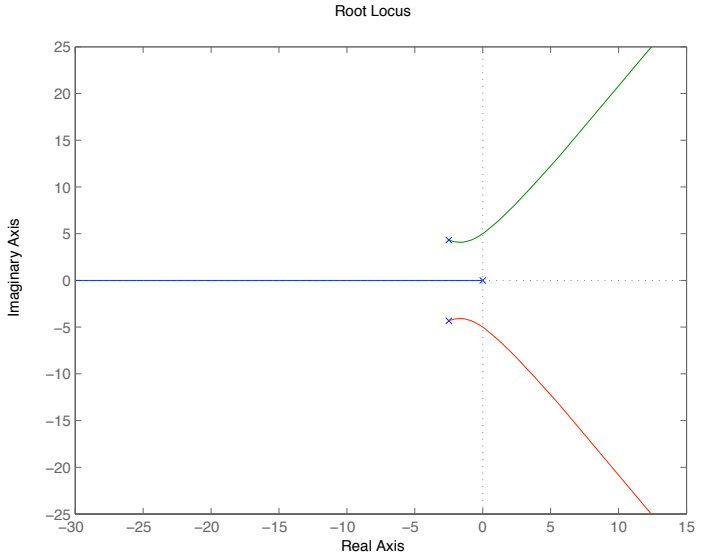


Figure 6.9: Roll loop root locus as a function of the integral gain k_{i_ϕ}.

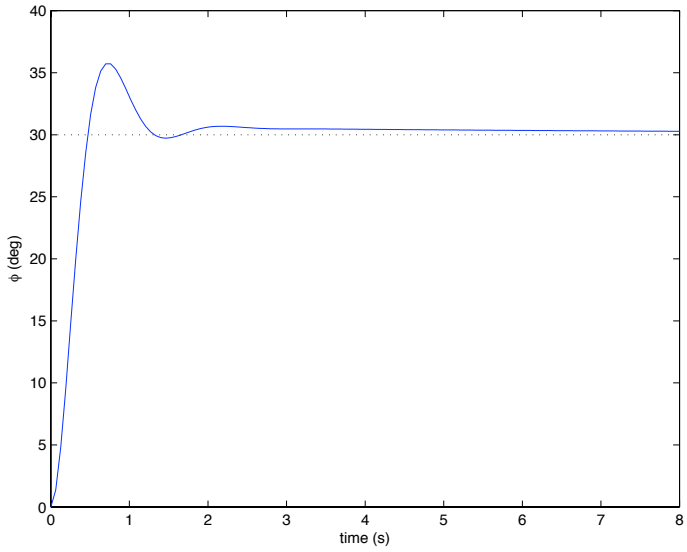


Figure 6.10: Roll loop response with proportional, derivative, and integral feedback.

We can see that the rise time and overshoot response have not been significantly degraded by the addition of the integral gain, however, the settling time response has been altered significantly. This is due to the introduction of the low-frequecy closed-loop pole near the origin as a result of the integral feedback. With the integral feedback, there is a trade-off between the speed of the settling-time (and disturbance-rejection) response and the rise time and overshoot characteristics. The response of the system to a unit-step disturbance d_{ϕ_2} is shown in Figure 6.11. The disturbance rejection response is somewhat slow because of the low value of the integral gain used. Better disturbance rejection (faster, lower magnitude disturbance response) could be obtained with higher integral gain, but at the expense of damping and overshoot in the transient response to a commanded input ϕ^c.

6.3.2 Heading Hold

The next step in the successive-loop-closure design of the lateral-directional autopilot is to design the heading-hold outer loop. If the inner loop from ϕ^c to ϕ has been adequately tuned, then $H_{\phi/\phi^c} \approx 1$ over the range of frequencies from 0 to ω_{n_ϕ}. Under this condition, the block diagram of Figure 6.5 can be simplified to the block diagram in Figure 6.12 for the purposes of designing the outer loop.

The objective of the yaw attitude hold design is to select k_{p_ψ} and k_{i_ψ} in Figure 6.5 so that the yaw angle ψ asymptotically tracks steps in the commanded yaw angle ψ^c. From the simplified

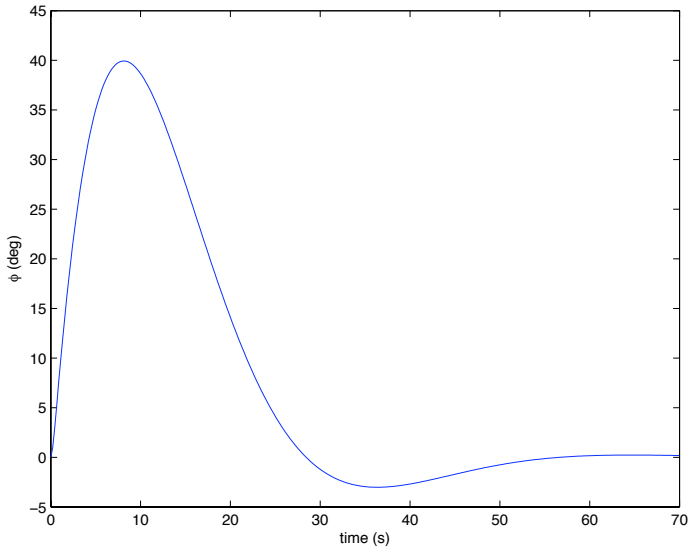


Figure 6.11: Roll loop disturbance rejection response with PID feedback.

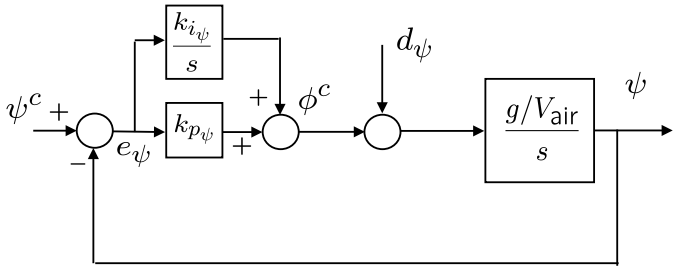


Figure 6.12: Heading hold outer feedback loop.

block diagram, the transfer functions from the inputs ψ^c and d_ψ to the output ψ are given by

$$\psi = \frac{g/V_a s}{s^2 + k_{p_\psi} g/V_a s + k_{i_\psi} g/V_a} d_\psi + \frac{k_{p_\psi} g/V_a s + k_{i_\psi} g/V_a}{s^2 + k_{p_\psi} g/V_a s + k_{i_\psi} g/V_a} \psi^c. \tag{6.11}$$

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

$$H_\psi = \frac{2\zeta_\psi \omega_{n_\psi} s + \omega_{n_\psi}^2}{s^2 + 2\zeta_\psi \omega_{n_\psi} s + \omega_{n_\psi}^2}. \tag{6.12}$$

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_ψ} and k_{i_ψ}. Figure 6.13 shows the fre-

quency response and the step response for H_ψ. Note that because of the numerator zero the standard intuition for the selection of ζ does not hold for this transfer function. Larger ζ results in larger bandwidth and smaller overshoot.

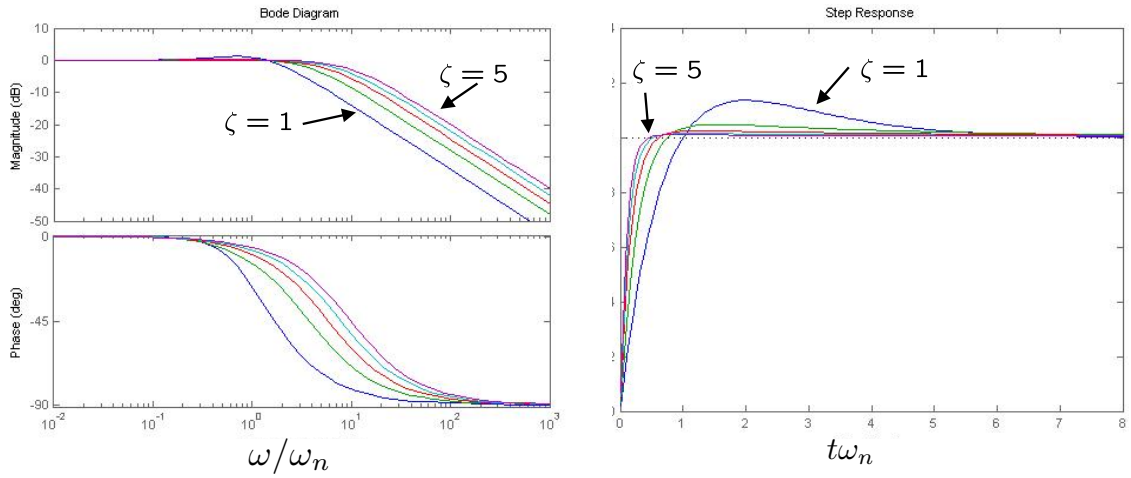


Figure 6.13: Frequency and step response for second order system.

Comparing coefficients in Equations (6.11) and (6.12), we find

$$\omega_{n_\psi}^2 = g/V_a k_{i_\psi}$$
$$2\zeta_\psi \omega_{n_\psi} = g/V_a k_{p_\psi}.$$

Solving these expressions for k_{p_ψ} and k_{i_ψ} we get

$$k_{p_\psi} = 2\zeta_\psi \omega_{n_\psi} V_a/g \tag{6.13}$$
$$k_{i_\psi} = \omega_{n_\psi}^2 V_a/g. \tag{6.14}$$

To ensure proper function of this successive-loop-closure design, it is essential that there be sufficient bandwidth separation between the inner (ϕ) and outer (ψ) feedback loops. Adequate separation can be achieved using the rule of thumb

$$\omega_{n_\phi} > 5\omega_{n_\psi}.$$

Generally, more bandwidth separation is better. More bandwidth separation requires either slower response in the ψ loop (lower ω_{n_ψ}), or faster response in the ϕ loop (higher ω_{n_ϕ}). Faster response usually comes at the cost of requiring more actuator control authority, which may not be possible given the physical constraints of the actuators.

6.3.3 Sideslip Hold

If the airframe is equipped with a rudder, then the rudder is used to maintain zero sideslip angle ($\beta(t) = 0$), or equivalently, a zero relative side-to-side velocity ($v_r(t) = 0$). The sideslip hold loop is shown in Figure 6.14, and the transfer function from v_r^c to v_r is given by

$$H_{v_r/v_r^c}(s) = \frac{a_{\beta_2} k_{p_\beta} s + a_{\beta_2} k_{i_\beta}}{s^2 + (a_{\beta_1} + a_{\beta_2} k_{p_\beta}) s + a_{\beta_2} k_{i_\beta}}.$$

Note that the DC gain is equal to one. If the desired closed pole are the roots of $s^2 + 2\zeta_\beta \omega_{n_\beta} s +$

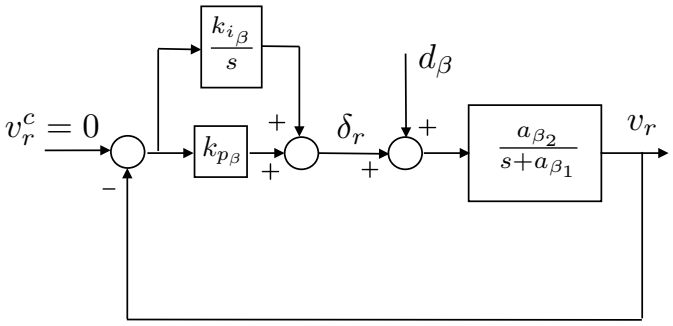


Figure 6.14: Sideslip hold control loop.

$\omega_{n_\beta}^2 = 0$, then solving for the control gains gives

$$k_{i_\beta} = \frac{\omega_{n_\beta}^2}{a_{\beta_2}} \tag{6.15}$$

$$k_{p_\beta} = \frac{2\zeta_\beta \omega_{n_\beta} - a_{\beta_1}}{a_{\beta_2}}. \tag{6.16}$$

Suppose now that there is a step on v_r^c of size $|v_r^c|$, then the maximum rudder command will be $\delta_r^{\max} = k_{p_\beta} |v_r^c|$. Therefore, ω_{n_β} is found from Equation (6.16) by setting $k_{p_\beta} = \delta_r^{\max} / |v_r^c|$.

6.4 Longitudinal Autopilot

The longitudinal autopilot is more complicated than the lateral autopilot because airspeed plays a significant role in the longitudinal dynamics. Our objective in designing the longitudinal autopilot will be to regulate airspeed and altitude using the throttle and the elevator as actuators. The method used to regulate altitude and airspeed depends on the altitude error. The flight regimes are shown in Figure 6.15

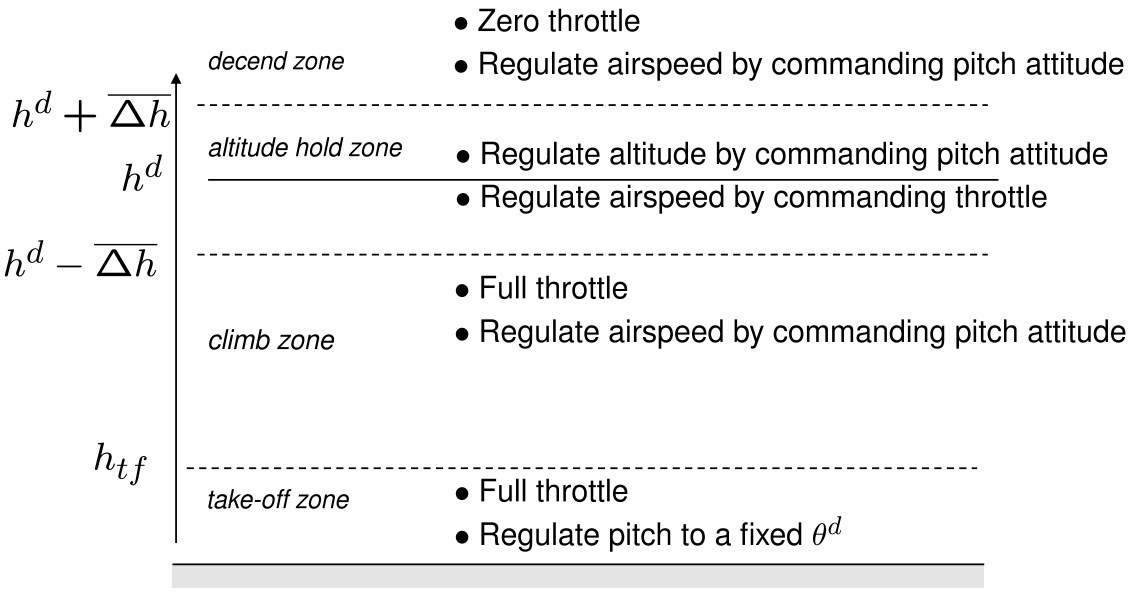


Figure 6.15: Flight regimes for the longitudinal autopilot

In the take-off zone, full throttle is commanded and the pitch attitude is regulated to a fixed pitch angle θ^c using the elevator. The objective in the climb zone is to maximize the climb rate given the current atmospheric conditions. To maximize the climb rate, full throttle is commanded and the airspeed is regulated using the pitch angle. If the airspeed increases above its nominal value, then the aircraft if caused to pitch up which results in an increased climb rate and a decrease in airspeed. Similarly, if the airspeed drops below the nominal value, the airframe is pitched down thereby increasing the airspeed but also decreasing the climb rate. Regulating the airspeed using pitch attitude effectively keeps the airframe away from stall conditions. Note however, that we would not want to regulate airspeed with pitch attitude immediately after take-off because after take-off the airframe is always trying to gain airspeed, however, pitching down will drive the aircraft into the ground.

The descend zone is similar to the climb zone except that the throttle is commanded to zero. Again, stall conditions are avoided by regulating airspeed using the pitch angle thus maximizing the descent rate at a given airspeed.

In the altitude hold zone, the airspeed is regulated by adjusting the throttle, and the altitude is regulated by commanding the pitch attitude.

To implement the longitudinal autopilot shown in Figure 6.15 we need the following feedback loops: (1) pitch attitude hold using elevator, (2) airspeed hold using throttle, (3) airspeed hold using pitch attitude, and (4) altitude hold using pitch attitude. The design of each of these loops will be discussed in the next four subsections. Finally, the complete longitudinal autopilot will be

presented in Section 6.4.5

6.4.1 Pitch Attitude Hold

The pitch attitude hold loop is similar to the roll attitude hold loop and we will follow a similar line of reasoning in its development. From Figure 6.16, the transfer function from θ^c to θ is given by is given by

$$H_{\theta/\theta^c}(s) = \frac{k_{p_\theta} a_{\theta_3}}{s^2 + (a_{\theta_1} + k_{d_\theta} a_{\theta_3})s + (a_{\theta_2} + k_{p_\theta} a_{\theta_3})}. \tag{6.17}$$

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

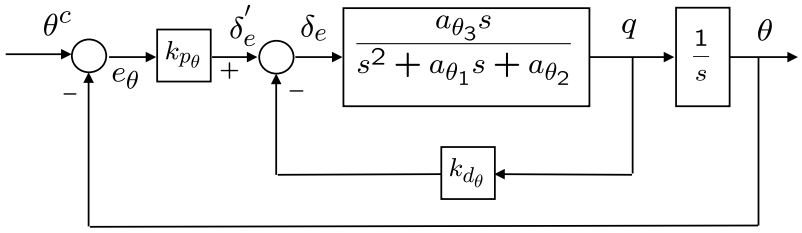


Figure 6.16: Pitch attitude hold feedback loops.

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

$$\frac{K_{\theta_{\text{DC}}} \omega_{n_\theta}^2}{s^2 + 2\zeta_\theta \omega_{n_\theta} s + \omega_{n_\theta}^2},$$

then equating denominator coefficients we get

$$\begin{aligned} \omega_{n_\theta}^2 &= a_{\theta_2} + k_{p_\theta} a_{\theta_3} \\ 2\zeta_\theta \omega_{n_\theta} &= a_{\theta_1} + k_{d_\theta} a_{\theta_3}. \end{aligned}$$

Solving these expressions for k_{p_θ} and k_{d_θ} we get

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

Given the saturation constraint on the elevator, the natural frequency of the pitch loop can be computed from Equation (6.7) as

$$\omega_{n_\theta} = \sqrt{\frac{|a_{\theta_3}|\, \delta_e^{\max} \sqrt{1 - \zeta_\theta^2}}{\theta^{max}}}. \tag{6.18}$$

Therefore, selecting the desired damping ratio ζ_θ and the pitch angle limit $\theta^{\max}$ fixes the value for k_{p_θ} and k_{d_θ}.

The DC gain of this inner-loop transfer function approaches one as the $k_{p_\theta} \to \infty$. The DC gain is given by

$$K_{\theta_{\text{DC}}} = \frac{k_{p_\theta} a_{\theta_3}}{(a_{\theta_2} + k_{p_\theta} a_{\theta_3})} \tag{6.19}$$

which for typical gain values is significantly less than one. For the design of the outer loop, we will use this DC gain to represent the gain of the inner loop over its full bandwidth. 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. For this reason, we have chosen not to use integral control on the pitch loop.

6.4.2 Altitude Hold Using Commanded Pitch

The altitude hold autopilot utilizes a successive-loop-closure strategy with the pitch attitude hold autopilot as an inner loop as shown in Figure 6.17. Assuming that the pitch loops function as de-

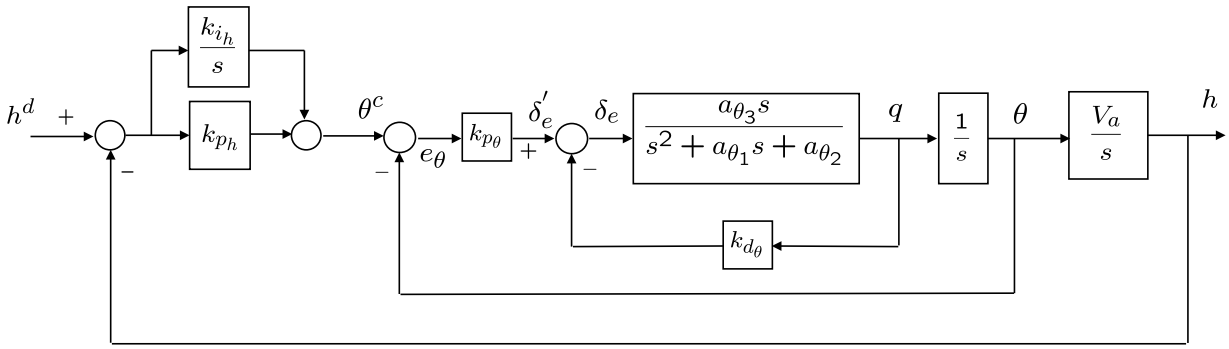


Figure 6.17: Successive loop feedback structure for altitude hold autopilot.

signed and $\theta \approx K_{\theta_{\text{DC}}} \theta^d$, the altitude hold loop using the commanded pitch is shown in Figure 6.18

In the Laplace domain we have

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

where again we see that the DC gain is equal to one, and constant disturbances are rejected. The closed loop transfer function is again independent of airframe parameters and is only dependent

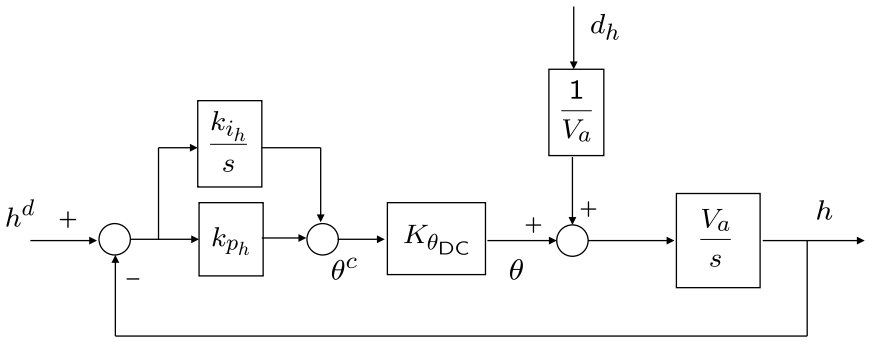


Figure 6.18: The altitude hold loop using the commanded pitch angle.

on the known airspeed. The gains k_{p_h} and k_{i_h} should be chosen such that the bandwidth of the altitude using pitch loop is less than the bandwidth of the pitch attitude hold loop. A good rule of thumb is that it at least is five times less.

If the desired response of the altitude hold loop is given by the canonical second order transfer function

$$\frac{\omega_{n_h}^2}{s^2 + 2\zeta_h \omega_{n_h} s + \omega_{n_h}^2},$$

then equating denominator coefficients we get

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

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

$$k_{i_h} = \frac{\omega_{n_h}^2}{K_{\theta_{\text{DC}}} V_a} \tag{6.20}$$

$$k_{p_h} = \frac{2\zeta_h \omega_{n_h}}{K_{\theta_{\text{DC}}} V_a}. \tag{6.21}$$

Therefore, selecting the desired damping ratio and natural frequency fixes the value for k_{p_h} and k_{i_h}.

6.4.3 Airspeed Hold Using Commanded Pitch

The dynamic model for airspeed using pitch angle is shown in Figure 5.5. Disturbance rejection again requires a PI controller. The resulting block diagram is shown in Figure 6.19.

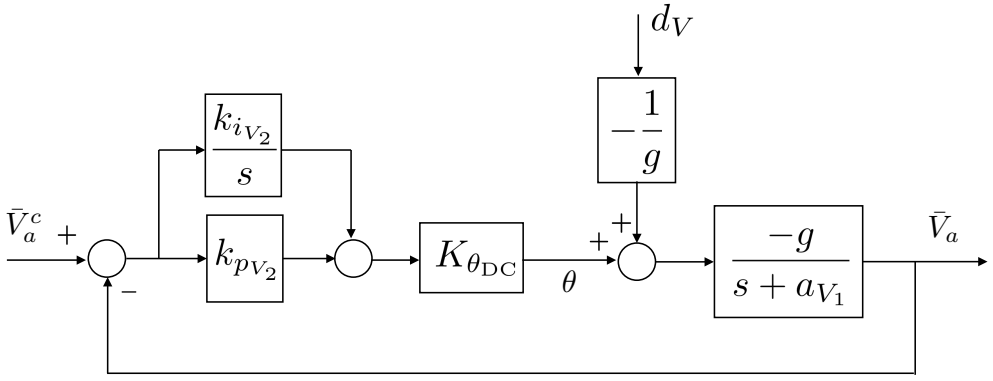


Figure 6.19: PI Controller to regulate airspeed using the pitch angle.

In the Laplace domain we have

$$\bar{V}_a(s) = \left(\frac{(-K_{\theta_{\rm DC}} g k_{p_{V_2}})(s + \frac{k_{i_{V_2}}}{k_{p_{V_2}}})}{s^2 + (a_{V_1} - K_{\theta_{\rm DC}} g k_{p_{V_2}})s - K_{\theta_{\rm DC}} g k_{i_{V_2}}} \right) \bar{V}_a^d(s) + \left(\frac{s}{s^2 + (a_{V_1} - K_{\theta_{\rm DC}} g k_{p_{V_2}})s - K_{\theta_{\rm DC}} g k_{i_{V_2}}} \right) d_V(s). \tag{6.22}$$

Note that the DC gain is equal to one and that step disturbances are rejected. The angle of attack α enters into the block diagram at the same location as d_V. If we assume that the angle of attack is constant, then the PI controller winds up the integrator to reject α. Therefore, the output of the PI controller can be used as the input to the pitch attitude hold loop.

The gains $k_{p_{V_2}}$ and $k_{i_{V_2}}$ should be chosen such that the bandwidth of the airspeed to pitch loop is less than the bandwidth of the pitch attitude hold loop. Following a similar procedure to what we've done previously, we can determine values for the feedback gains by matching denominator coefficients in Equation (6.22) with those of a canonical second-order transfer function. Denoting the desired natural frequency and damping ratio we seek to achieve with feedback as $\omega_{n_{V_2}}^2$ and ζ_{V_2}, matching coefficients gives

$$\begin{aligned} \omega_{n_{V_2}}^2 &= -K_{\theta_{\rm DC}} g k_{i_{V_2}} \\ 2\zeta_{V_2}\omega_{n_{V_2}} &= a_{V_1} - K_{\theta_{\rm DC}} g k_{p_{V_2}}. \end{aligned}$$

Solving for the control gains gives

$$k_{i_{V_2}} = -\frac{\omega_{n_{V_2}}^2}{K_{\theta_{\rm DC}} g} \tag{6.23}$$

$$k_{p_{V_2}} = \frac{a_{V_1} - 2\zeta_{V_2}\omega_{n_{V_2}}}{K_{\theta_{\rm DC}} g}. \tag{6.24}$$

6.4.4 Airspeed Hold Using Throttle

The dynamic model for airspeed using the throttle as an input is shown in Figure 5.5. Therefore, the associated closed loop system is shown in Figure 6.20

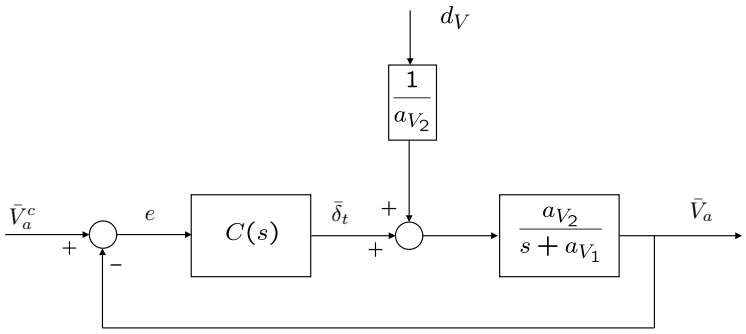


Figure 6.20: Airspeed hold using throttle.

If we use a proportional control, i.e. $C(s) = k_{p_V}$, then

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

Note that the DC gain is not equal to one, and that step disturbances are not rejected. If on the other hand we use proportional-integral control, i.e., $C(s) = k_{p_V} + \frac{k_{i_V}}{s}$, then

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

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

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

Inverting these expressions gives the control gains

$$k_{i_V} = \frac{\omega_{n_V}^2}{a_{V_2}} \tag{6.25}$$

$$k_{p_V} = \frac{2\zeta_V\omega_{n_V} - a_{V_1}}{a_{V_2}}. \tag{6.26}$$

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 6.20 is

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

Therefore, the control loop shown in Figure 6.20 can be implemented without knowledge of the trim velocity V_a^*. Similarly, since the integrator will wind up to reject step disturbances, and a constant error in δ_t^* can be thought of as a step disturbance, we can set

$$\delta_t = \bar{\delta}_t.$$

6.4.5 Altitude Control State Machine

The longitudinal autopilot deal with the control of the longitudinal motions in the body x-z plane: pitch angle, altitude, and airspeed. Up to this point, we have described four different longitudinal autopilot modes: 1) pitch attitude hold, 2) altitude hold using commanded pitch, 3) airspeed hold using commanded pitch, and 4) airspeed hold using throttle. These longitudinal control modes can be combined to create the altitude control state machine shown in Figure 6.21. In the climb zone, the throttle is set to its maximum value ($\delta_t = 1$) and the airspeed hold from commanded pitch mode is used to control the airspeed, thus ensuring that the airframe avoids stall conditions. In simple terms, this causes the MAV to climb at its maximum possible climb rate until it is close to the altitude set point. Similarly, in the descend zone, the throttle is set to its minimum value ($\delta_t = 0$) and the airspeed hold from commanded pitch mode is again used to control airspeed. In this way, the MAV descends at a steady descent rate until it reaches the altitude hold zone. In the altitude hold zone, the airspeed from throttle mode is used to regulate the airspeed around V_a^d, and the altitude from pitch mode is used to regulate the altitude around h^d.

Example An example of the altitude state machine response is shown in Figure 6.22. The MAV takes off from an altitude of 5 m. The initial altitude command is 80 m. Below 25 m is the take-off zone. In this altitude zone, the MAV is given a pitch angle command of 20 deg and a full throttle command. Once in the climb zone, full throttle is held and the pitch angle is controlled so that the desired airspeed is achieved. The airspeed command is set to a value lower than what is achieved in steady-state level flight at full throttle. In this way, the MAV must have a positive climb angle to achieve the airspeed command. In this example, we have define the altitude hold zone as being the range of altitudes from 10 m below to 10 m above the altitude command – in this example from 70 to 90 m. Once in the altitude hold zone, throttle is used to control the airspeed and pitch angle commands are used to zero any altitude errors. In this example a new altitude command of 100 m is given at 30 s, followed by a descent to 60 m altitude at 50 s. In the descend zone, the throttle

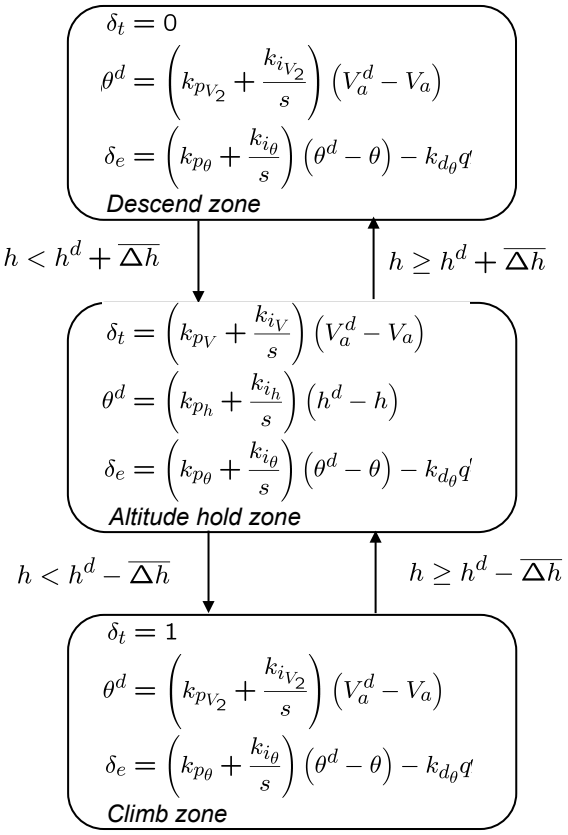


Figure 6.21: Altitude control state machine.

command is zeroed and the pitch angle is used to control the airspeed. A final altitude command of 70 m is given at 75 s.

6.5 PID Loop Implementation

The longitudinal and lateral control strategies presented in this chapter consist of several proportional-integral-derivative (PID) control loops. In this section we briefly describe how PID loops can be implemented in discrete-time code. A general PID control signal is given by

$$u(t) = k_p e(t) + k_i \int_{-\infty}^{t} e(\tau) d\tau + k_d \frac{de}{dt}(t),$$

where $e(t) = y^c(t) - y(t)$ is the error between the commanded output $y^c(t)$ and the current output $y(t)$. In the Laplace domain we have

$$U(s) = k_p E(s) + k_i \frac{E(s)}{s} + k_d s E(s).$$

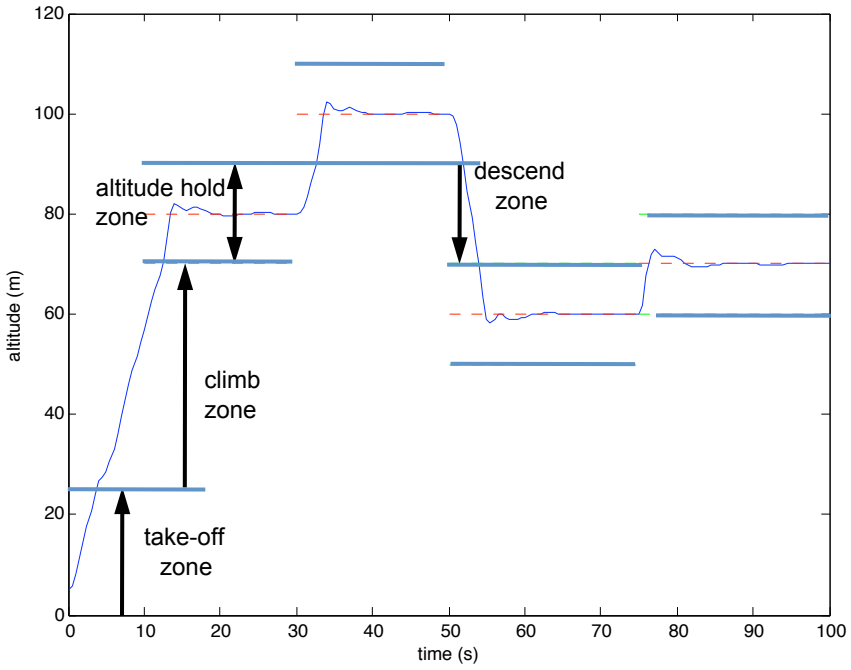


Figure 6.22: Simulation of MAV response using altitude control state machine.

Since a pure differentiator is not causal, the standard approach is to use a band limited, or dirty, differentiator so that

$$U(s) = k_p E(s) + k_i \frac{E(s)}{s} + k_d \frac{s}{\tau s + 1} E(s).$$

To convert to discrete time, we use the Tustin or trapazoidal rule where the Laplace variable s is replaced with the Z-transform approximation

$$s \mapsto \frac{2}{T_s} \left(\frac{1 - z^{-1}}{1 + z^{-1}} \right),$$

where T_s is the sample period [**?**]. Letting $I(s) \triangleq E(s)/s$, the integrator in the Z-domain becomes

$$I(z) = \frac{T_s}{2} \left(\frac{1 + z^{-1}}{1 - z^{-1}} \right) E(z).$$

Transforming to the time domain we have

$$I_k = I_{k-1} + \frac{Ts}{2} \left(E_k + E_{k-1} \right). \tag{6.27}$$

A formula for a discrete implementation of a differentiator can be derived in a similar manner.

Letting $D(s) \triangleq (s/(\tau s+1))E(s)$, the differentiator in the Z-domain is

$$\begin{aligned} D(z) &= \frac{\frac{2}{Ts}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)}{\frac{2\tau}{T_s}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)+1} E(z) \\ &= \frac{\left(\frac{2}{2\tau+T}\right)(1-z^{-1})}{1-\left(\frac{2\tau-T}{2\tau+T}\right)z^{-1}} E(z). \end{aligned}$$

Transforming to the time domain we have

$$D_k = \left(\frac{2\tau-T}{2\tau+T}\right) D_{k-1} + \left(\frac{2}{2\tau+T}\right)(E_k - E_{k-1}). \tag{6.28}$$

Matlab code that implements a general PID loop is shown below.

```
function u = pidloop(y_c, y, flag, kp, ki, kd, limit, Ts, tau)
  persistent integrator;
  persistent differentiator;
  persistent error_d1;
  if flag==1,  % reset (initialize) persistent variables when flag==1
      integrator = 0;
      differentiator = 0;
      error_d1   = 0; % _d1 means delayed by one time step
  end
  error = y_c - y; % compute the current error
  integrator = integrator + (Ts/2)*(error + error_d1); % update integrator
  differentiator = (2*tau-Ts)/(2*tau+Ts)*differentiator...
      + 2/(2*tau+Ts)*(error - error_d1);  % update differentiator
  u = sat(...                   % implement PID control
      kp * error +...           % proportional term
      ki * integrator +...      % integral term
      kd * differentiator,...   % derivative term
      limit...                  % ensure abs(u)<=limit
      );
  error_d1 = error;  % update persistent variables

function out = sat(in, limit)
  if in > limit,        out = limit;
  elseif in < -limit;   out = -limit;
  else                  out = in;
  end
```

The inputs on Line 1 are the commanded output y_c, the current output y, a `flag` used to reset the integrator, the PID gain k_p, k_i, and k_d, the `limit` of the saturation command, the sample time T_s, and the dirty derivative gain τ. Line 11 implements Equation (6.27) and Lines 12-13 implement Equation (6.28).

6.6 Chapter Summary

In this chapter, we utilized the technique of successive loop closure to develop lateral-directional and longitudinal autopilots for a MAV. Following the direction of Chapter 5, we assumed that the lateral-directional and longitudinal dynamics were sufficiently decoupled to allow their control to be treated separately. This decoupling assumption is true for the vast majority of practical flight conditions for MAVs. The lateral-directional autopilot involved control loops for roll rate, roll angle, and heading. The longitudinal autopilot utilized control loops for pitch rate, pitch angle, altitude, and airspeed. While relatively straightforward in concept, the successive loop closure approach allows us to control a relatively complicated system in a way that will enable MAV guidance and navigation.

6.6.1 Summary of Design Process for Lateral Autopilot

RWB: Need to revise to include sideslip hold

Input: The transfer function coefficients a_{ϕ_1} and a_{ϕ_2}, the nominal airspeed V_a, and the aileron limit $\delta_a^{\max}$.

Tuning Parameters: The roll angle limit $\phi^{\max}$, the damping coefficients ζ_ϕ and ζ_ψ, the roll integrator gain k_{i_ϕ}, and the bandwidth separation σ.

Compute Natural Frequencies: Compute the natural frequency of the inner loop ω_{n_ϕ} using Equation (6.7), and the natural frequency of the outer loop using $\omega_{n_\psi} = \omega_{n_\phi}/\sigma$.

Compute Gains: Compute the gains k_{p_ϕ}, k_{d_ϕ}, k_{p_ψ}, k_{i_ψ} using Equations (6.8), (6.9), (6.13), and (6.14).

6.6.2 Summary of Design Process for Longitudinal Autopilot

Input: The transfer function coefficients a_{θ_1}, a_{θ_2}, a_{θ_2}, a_{V_1}, a_{V_2}, the nominal airspeed V_a, and the elevator limit $\delta_e^{\max}$.

Tuning Parameters: The pitch angle limit $\theta^{\max}$, the damping coefficients ζ_θ, ζ_h, ζ_V, ζ_{V_2}, the natural frequency ω_{n_V}, and the bandwidth separation for the altitude loop σ_h, and the airspeed using pitch loop σ_V.

Compute Natural Frequencies: RWB: Need to revise this. Compute the natural frequency of the inner loop ω_{n_θ} using Equation (6.7), and the natural frequency of the outer loop using $\omega_{n_\psi} = \omega_{n_\phi}/\sigma$.

Compute Gains: RWB: Need to revise this. Compute the gains k_{p_ϕ}, k_{d_ϕ}, k_{p_ψ}, k_{i_ψ} using Equations (6.8), (6.9), (6.13), and (6.14).

Notes and References

RWB: Add stuff here.

6.7 Design Project

In this assignment you will use simplified design models to turn the gains of the PID loops for the lateral and longitudinal autopilot. In order to do that, you will need to create some auxiliary Simulink models that implement the design models. The final step will be to implement the control loops on the full simulation model. Simulink models that will help you in this process are included on the book website.

6.1 Create a Matlab script that computes the gains for the roll attitude hold loop. Assume that the maximum aileron deflection is $\delta_a^{\max} = 45$ degrees, and that the saturation limit is achieved for a step size of $\phi^{\max} = 15$ degrees, and that the nominal airspeed is $V_a = 10$ m/s. Use the Simulink file `roll_loop.mdl` on the website to tune the values of ζ_ϕ and k_{i_ϕ} to get acceptable performance.

6.2 Augment your Matlab script to compute the gains for the heading hold loop. The Simulink file `heading_loop.mdl` implements the heading hold loop with the roll hold as an inner loop. Tune the bandwidth separation and the damping ratio ζ_ψ to get acceptable performance for step inputs on heading of 25 degrees.

6.3 Augment your Matlab script to compute the gains for sideslip hold. Use the Simulink model `sideslip_loop.mdl` on the website to tune the value of ζ_β.

6.4 Augment your Matlab script to compute the gains for the pitch attitude hold loop. Assume that the maximum elevator deflection is $\delta_e^{\max} = 45$ degrees, and that the saturation limit is achieved for a step size of $\theta^{\max} = 10$ degrees. Use the Simulink file `pitch_loop.mdl` on the website to tune the value of ζ_θ.

6.5 Augment your Matlab script to compute the gains for altitude hold using pitch as an input. Use the Simulink model `altitude_from_pitch_loop.mdl` on the website to tune the value of ζ_h and the bandwidth separation.

6.6 Augment your Matlab script to compute the gains for airspeed hold using pitch as an input. Use the Simulink model `airspeed_from_pitch_loop.mdl` on the website to tune the value of ζ and the bandwidth separation.

6.7 Augment your Matlab script to compute the gains for airspeed hold using throttle as an input. Use the Simulink model `airspeed_from_throttle_loop.mdl` on the website to tune the value of ζ and ω_n.

6.8 The final step of the design is to implement the lateral and longitudinal autopilot on the simulation model. Modify your simulation model so that the aircraft is in its own subsystem. To ensure that the autopilot code can be easily transferred to embedded code written in, for example, C/C++, write the autopilot function using a Matlab script. An example of how to organize your simulation is given in the Simulink model `mavsim_chap6.mdl` on the website. A gutted version of the autopilot code is also given on the web site. To implement the longitudinal autopilot, you will need to implement a state machine using, for example, the **`switch`** statement.

Chapter 7

Nonlinear Design Models

In Chapter 5 we developed linear design models that were used to design the low-level autopilot in Chapter 6. In Chapters 10–13 we will develop higher level guidance strategies for the MAV. The nonlinear equations of motion given in Equations (5.1)–(5.12) do not facilitate the design of these higher level guidance strategies for two reasons: first, they are too complex, and second they do not include the feedback loops of the low-level autopilot. The objective of this chapter is to develop 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.

7.1 Autopilot Model

The guidance models developed in this chapter use a high level representation of the autopilots developed in the previous chapter. The airspeed and roll are represented by the first order models

$$\dot{V}_a = b_{V_a}(V_a^c - V_a) \tag{7.1}$$

$$\dot{\phi} = b_\phi(\phi^c - \phi), \tag{7.2}$$

where b_{V_a} and b_ϕ are positive constants that depend on the implementation of the autopilot. The altitude and heading loops are represented by the second order models

$$\dot{h} = -b_{\dot{h}}\dot{h} + b_h(h^c - h) \tag{7.3}$$

$$\dot{\psi} = -b_{\dot{\psi}}\dot{\psi} + b_\psi(\psi^c - \psi), \tag{7.4}$$

where $b_{\dot{h}}$, b_h, $b_{\dot{\psi}}$, and b_ψ are also positive constants that depend on the implementation of the autopilot. As explained in subsequent sections, some of the guidance models also assume autopilot

hold loops for the flight path angle γ and the load factor n, where load factor is defined as lift divided by gravity. The first order autopilot loops for flight path angle and load factor are given by

$$\dot{\gamma} = b_\gamma(\gamma^c - \gamma) \tag{7.5}$$

$$\dot{n} = b_n(n^c - n), \tag{7.6}$$

where b_γ and b_n are positive constants that depend on the implementation of the autopilot.

7.2 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 $\dot{u}$, $\dot{v}$, $\dot{w}$, $\dot{p}$, $\dot{q}$, $\dot{r}$) from consideration. This eliminates 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.

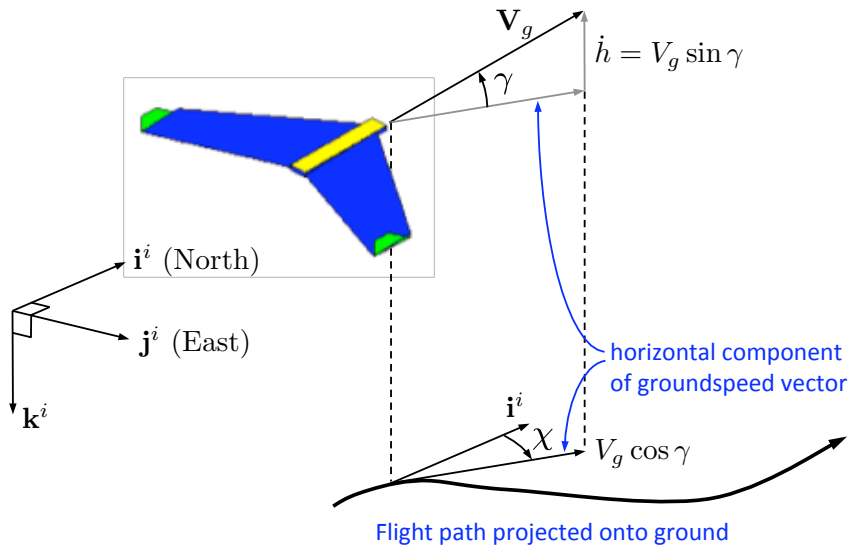


Figure 7.1: The flight path angles γ and χ.

The direction of the velocity vector of the aircraft $\mathbf{V}_g$ can be related to the inertial coordinate frame by two angles: the flight path angle γ and the course angle χ, as shown in Figure 7.1. The flight path angle is defined as the angle between the horizontal plane and the velocity vector, while the course is the angle between the projection of the velocity vector onto the horizontal plane and true North. Recalling that the magnitude of the aircraft velocity vector is the groundspeed,

$|\mathbf{V}_g| = V_g$, we can express the velocity in the inertial frame as

$$\mathbf{v}^i = \begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ -\dot{h} \end{pmatrix} = \begin{pmatrix} \cos\chi & -\sin\chi & 0 \\ \sin\chi & \cos\chi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\gamma & 0 & \sin\gamma \\ 0 & 1 & 0 \\ -\sin\gamma & 0 & \cos\gamma \end{pmatrix} \begin{pmatrix} V_g \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \cos\chi\cos\gamma \\ \sin\chi\cos\gamma \\ -\sin\gamma \end{pmatrix} V_g$$

or

$$\begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{h} \end{pmatrix} = \begin{pmatrix} \cos\chi\cos\gamma \\ \sin\chi\cos\gamma \\ \sin\gamma \end{pmatrix} V_g. \tag{7.7}$$

Because it is common to control the heading and airspeed of an aircraft, it is useful to express (7.7) in terms of ψ and V_a. This can be conceptualized by referring to the two-dimensional case ($\gamma = 0$) shown in Figure 2.10. In level flight,

$$\begin{aligned} V_g\cos\chi &= V_a\cos\psi + w_n \\ V_g\sin\chi &= V_a\sin\psi + w_e \end{aligned} \tag{7.8}$$

where w_n and w_e are the north and east components of the wind. By drawing on the relationships between χ, ψ, V_a, V_g, and the wind vector, and incorporating the effect on a non-zero flight path angle, we can express the derivatives of the inertial positions as

$$\begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{h} \end{pmatrix} = \begin{pmatrix} \cos\psi\cos\gamma \\ \sin\psi\cos\gamma \\ \sin\gamma \end{pmatrix} V_a + \begin{pmatrix} w_n \\ w_e \\ w_h \end{pmatrix}, \tag{7.9}$$

where w_h is the vertical component of the wind (positive up).

7.2.1 Coordinated Turn

The coordinated turn is a flight condition that is often sought for 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 relating heading rate and bank angle as shown by Phillips [23]. During a coordinated turn, the bank angle ϕ is set so that there is no net side force acting on the MAV. As shown in the free-body diagram of Figure 7.2, the centrifugal force acting on the MAV is equal and opposite to the

horizontal component of the lift force

$$\begin{aligned} L\sin\phi &= \mathsf{m}\frac{v^2}{R} \\ &= \mathsf{m}v\omega \\ &= \mathsf{m}\left(V_a\cos\gamma\right)\dot{\psi}. \end{aligned} \tag{7.10}$$

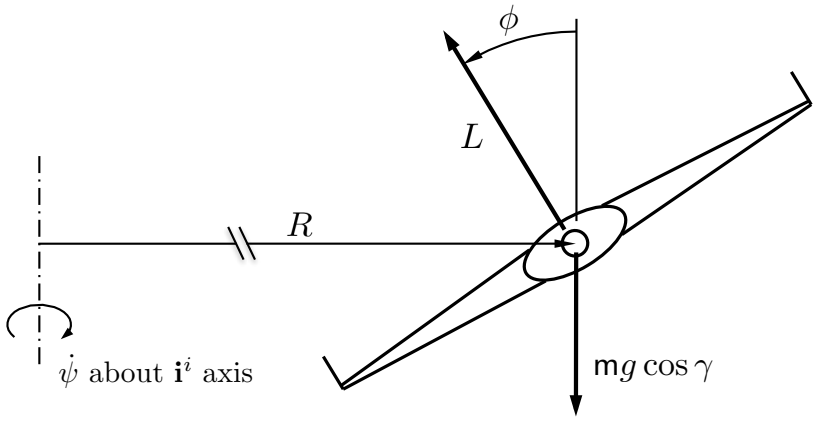


Figure 7.2: Free body diagram indicating forces on a MAV in a climbing coordinated turn. The MAV is flying at a flight path angle of γ with the nose of the MAV directed out of the page. The forces shown are in the $\mathbf{j}^b$-$\mathbf{k}^b$ plane.

The centrifugal force is calculated using the angular rate $\dot{\psi}$ about the vertical axis and the horizontal component of the airspeed, $V_a\cos\gamma$. Similarly, the vertical component of the lift force is equal and opposite to the projection of the gravitational force onto the $\mathbf{j}^b$-$\mathbf{k}^b$ plane

$$L\cos\phi = \mathsf{m}g\cos\gamma, \tag{7.11}$$

as shown in Figure 7.2. Dividing (7.10) by (7.11) and solving for $\dot{\psi}$ gives

$$\dot{\psi} = \frac{g}{V_a}\tan\phi, \tag{7.12}$$

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

$$R = \frac{V_a^2\cos\gamma}{g\tan\phi}. \tag{7.13}$$

7.2.2 Load Factor

Load Factor is defined as the ratio of the lift acting on the aircraft to the weight of the aircraft: $n \triangleq L/mg$. In wings-level, horizontal flight, $\phi = \gamma = 0$ and $n = 1$. 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 it is a dimensionless number, it is often referred by the number of "g's" that an aircraft experiences during a maneuver. By controlling the load factor as a state, we can ensure that the aircraft is always given commands that are within its structural capability. Drawing on the load factor definition and (7.11) we find that for a steady, climbing coordinated turn the load factor is

$$n = \frac{\cos\gamma}{\cos\phi}.$$

Considering load factor as a state variable that will be controlled by the autopilot of the system, Eq. (**??**) can be rewritten as

$$\dot{\psi} = \frac{g}{V_a}\frac{\sin\phi}{\cos\gamma}n. \tag{7.14}$$

7.2.3 Accelerating Climb

To derive the dynamics for flight path angle, we will consider a pull-up maneuver in which the aircraft climbs along an arc. Our derivation draws on the discussion in [2, p. 227–228]. The free-body diagram of the MAV in the $\mathbf{i}^b$-$\mathbf{k}^i$ plane is shown in Figure 7.3. Since the MAV has a roll angle of ϕ, the projection of the lift vector onto the $\mathbf{i}^b$-$\mathbf{k}^i$ plane is $L\cos\phi$. The centripetal force due to to the pull-up maneuver is $\mathsf{m}V_a\dot{\gamma}$. Therefore, summing the forces in the $\mathbf{i}^b$-$\mathbf{k}^i$ plane gives

$$L\cos\phi = \mathsf{m}V_a\dot{\gamma} + \mathsf{m}g\cos\gamma. \tag{7.15}$$

Taking into account the definition of load factor and solving for $\dot{\gamma}$ results in

$$\dot{\gamma} = \frac{g}{V_a}\left(n\cos\phi - \cos\gamma\right). \tag{7.16}$$

7.3 Kinematic Guidance Models

In this section we summarize several different kinematic guidance models MAVs. The first guidance model assumes that the low level autopilot controls controls airspeed, roll angle, and load

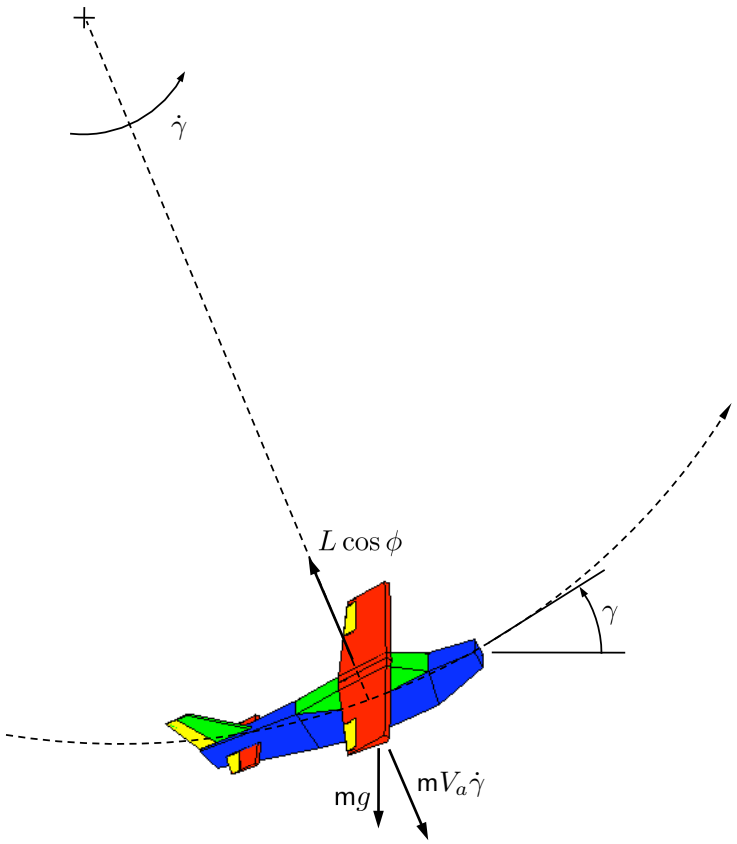


Figure 7.3: Free-body diagram for a pull up maneuver. The MAV is at a roll angle of ϕ.

factor. Therefore from Equations (7.9), (7.14), (7.16), we get

$$\begin{aligned}
\dot{p}_n &= V_a \cos\psi \cos\gamma + w_n \\
\dot{p}_e &= V_a \sin\psi \cos\gamma + w_e \\
\dot{h} &= V_a \sin\gamma + w_h \\
\dot{\psi} &= \frac{g}{V_a}\frac{\sin\phi}{\cos\gamma} n \\
\dot{\gamma} &= \frac{g}{V_a}\left(n\cos\phi - \cos\gamma\right) \\
\dot{V}_a &= b_{V_a}(V_a^c - V_a) \\
\dot{\phi} &= b_\phi(\phi^c - \phi) \\
\dot{n} &= b_n(n^c - n),
\end{aligned} \tag{7.17}$$

where the inputs to the model are V_a^c, ϕ^c, and n^c.

If instead of directly controlling the load factor, an autopilot loop controls the flight path angle,

then the model given by (7.17) reduces to

$$\begin{aligned}
\dot{p}_n &= V_a \cos\psi \cos\gamma + w_n \\
\dot{p}_e &= V_a \sin\psi \cos\gamma + w_e \\
\dot{h} &= V_a \sin\gamma + w_h \\
\dot{\psi} &= \frac{g}{V_a} \tan\phi \\
\dot{\gamma} &= b_\gamma(\gamma^c - \gamma) \\
\dot{V}_a &= b_{V_a}(V_a^c - V_a) \\
\dot{\phi} &= b_\phi(\phi^c - \phi),
\end{aligned} \tag{7.18}$$

where the inputs are now γ^c, V_a^c, and ϕ^c.

In the previous chapter, the altitude was controlled directly rather than through the flight path angle. To obtain a model that does not include the flight path angle γ, the influence of γ on $\dot{p}_n$ and $\dot{p}_e$ is dropped to obtain

$$\begin{aligned}
\dot{p}_n &= V_a \cos\psi + w_n \\
\dot{p}_e &= V_a \sin\psi + w_e \\
\dot{\psi} &= \frac{g}{V_a} \tan\phi \\
\ddot{h} &= -b_{\dot{h}}\dot{h} + b_h(h^c - h) \\
\dot{V}_a &= b_{V_a}(V_a^c - V_a) \\
\dot{\phi} &= b_\phi(\phi^c - \phi),
\end{aligned} \tag{7.19}$$

where the inputs are h^c, V_a^c, and ϕ^c.

If the heading is controlled directly rather than through the roll angle, then the guidance model becomes

$$\begin{aligned}
\dot{p}_n &= V_a \cos\psi + w_n \\
\dot{p}_e &= V_a \sin\psi + w_e \\
\ddot{\psi} &= -b_{\dot{\psi}}\dot{\psi} + b_\psi(\psi^c - \psi) \\
\ddot{h} &= -b_{\dot{h}}\dot{h} + b_h(h^c - h) \\
\dot{V}_a &= b_{V_a}(V_a^c - V_a),
\end{aligned} \tag{7.20}$$

where the inputs are h^c, V_a^c, and ψ^c.

7.4 Dynamic Guidance Model

The reduced-order guidance models derived above 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. Further, 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 utilizes relationships drawn from free-body diagrams. Lift, drag, and thrust forces are evident in these dynamic equations.

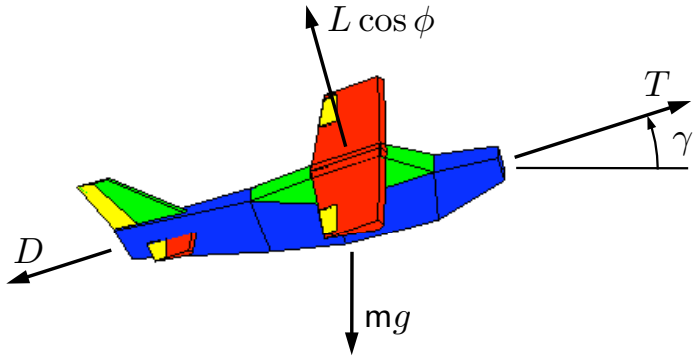


Figure 7.4: Free-body diagram indicating external forces on the UAV along the $\mathbf{i}_b$ axis. The UAV is assumed to be at a roll angle of ϕ.

Figure 7.4 shows a free-body diagram for a MAV climbing at flight-path angle γ and bank angle ϕ. Applying Newton's second law along the $\mathbf{i}_b$ axis and rearranging gives

$$\dot{V}_a = \frac{T}{\mathsf{m}} - \frac{D}{\mathsf{m}} - g\sin\gamma,$$

where T is the thrust and D is the drag. Two additional equations of motion can be formed by rearranging Equations (7.10) and (7.15).

$$\dot{\psi} = \frac{L}{\mathsf{m}V_a}\frac{\sin\phi}{\cos\gamma}$$
$$\dot{\gamma} = \frac{L}{\mathsf{m}V_a}\cos\phi - \frac{g}{V_a}\cos\gamma$$

These equations come from the free-body diagrams associated with a climbing turn (Figure 7.2) and an accelerating climb (Figure 7.3). Combining these dynamic equations with the kinematic

equations relating Cartesian position and velocity gives the following alternative equations of motion

$$\begin{aligned}
\dot{p}_n &= V_a \cos\psi \cos\gamma + w_n \\
\dot{p}_e &= V_a \sin\psi \cos\gamma + w_e \\
\dot{h} &= V_a \sin\gamma + w_h \\
\dot{V}_a &= \frac{T}{\mathsf{m}} - \frac{D}{\mathsf{m}} - g\sin\gamma \\
\dot{\psi} &= \frac{L}{\mathsf{m}V_a}\frac{\sin\phi}{\cos\gamma} \\
\dot{\gamma} &= \frac{L}{\mathsf{m}V_a}\cos\phi - \frac{g}{V_a}\cos\gamma.
\end{aligned} \tag{7.21}$$

The state variables are airspeed, heading, flight path angle, and inertial location (north, east, up) $[V_a, \psi, \gamma, p_n, p_e, h]^T$. Control variables are thrust, lift coefficient, and bank angle $[T, C_L, \phi]^T$. Lift and drag are given by

$$\begin{aligned}
L &= \frac{1}{2}\rho V_a^2 S C_L \\
D &= \frac{1}{2}\rho V_a^2 S C_D
\end{aligned}$$

with $C_D = C_{D_0} + KC_L^2$ [24]. The induced drag factor K can be determined from the aerodynamic efficiency

$$E_{\max} \triangleq \left(\frac{L}{D}\right)_{\max}$$

and the zero-lift drag coefficient, C_{D_0}

$$K = \frac{1}{4E_{\max}^2 C_{D_0}}.$$

The popularity of this point-mass model is likely due to the fact that it models aircraft behavior in response to inputs that a pilot commonly controls: engine thrust, lift from the lifting surfaces, and bank angle as observed using the attitude indicator.

7.5 Chapter Summary

The objective of this chapter was 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 (7.17), (7.18), (7.19), and (7.20). A dynamic design model often found in the literature is given in Equation (7.21).

Notes and References

RWB: This section needs to be completely revised.

The material in this chapter discussing the full twelve-state nonlinear dynamics can be found in most textbooks on flight mechanics including [13, 21, 1, 2, 5, 7, 23]. Material supporting the development of the reduced-order models can be found in [2, 23, 21, 24].

7.6 Design Project

The objective of the assignment in this chapter is to estimate the autopilot constants b_* and to develop a reduced order Simulink that can be used to test and debug the guidance algorithm discussed in later chapter, prior to implementation on the full simulation model. We will focus primarily on the models given in Equations (7.19) and (7.20).

7.1 Create a Simulink S-function that implements the model given in Equation (7.20) and insert it in your MAV simulator. For different inputs ψ^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, $b_{\dot{\psi}}$, and b_ψ to obtain similar behavior. You may need to re-tune the autopilot gains obtained from the previous chapter. You may want to use the Simulink file `mavsim_chap7_guidance1.mdl` on the website.

7.2 Modify your autopilot function so that it uses the commanded roll angle ϕ^c as an input instead of the commanded heading ψ^c. Create a Simulink S-function that implements the model given in Equation (7.19) and insert it in your MAV simulator. For different inputs ϕ^c, h^c, and V_a^c, compare the output of the two models, and tune the autopilot coefficients b_ϕ to obtain similar behavior. You may need to re-tune the autopilot gains obtained from the previous chapter. You may want to use the Simulink file `mavsim_chap7_guidance2.mdl` on the website. Using the simulation, find the achievable minimum turn radius $R_{\min}$ of the MAV when the commanded roll angle is $\phi^c = 30$ degrees.

Chapter 8

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), powerful microcontrollers, and more capable 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 on-board sensors typically used on MAVs and to quantify what they measure.

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.

8.1 Accelerometers

Acceleration transducers (accelerometers) typically employ a proof mass held in place by a compliant suspension as shown in Figure 8.1. When the case of the accelerometer experiences an acceleration, the proof mass moves relative to the case through a distance proportional to the acceleration. The acceleration experienced by the proof mass is converted to a displacement by the springs in the suspension. A simple force balance analysis of the proof-mass yields the relationship

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

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

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

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

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

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

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

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

The accelerometer in the figure 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, reluctive, strain based designs.

Accelerometers typical produce an analog voltage signal proportion to the acceleration being sensed. As with other analog devices, they are subject to signal bias and noise. The output of an accelerometer can be modeled as

$$\Upsilon_{accel} = k_{accel}A + \beta_{accel} + \eta_{accel},$$

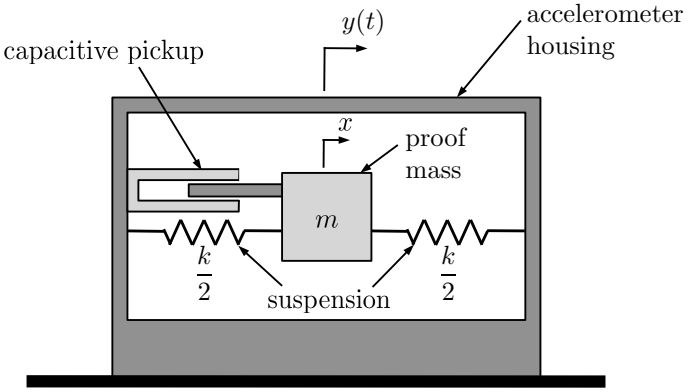


Figure 8.1: Conceptual depiction of MEMS accelerometer.

where Υ_{accel} is in Volts, k_{accel} is a gain, A is the acceleration in meters per second squared, β_{accel} is a bias term, and η_{acc} is zero mean Gaussian noise. The gain k_{accel} may be found on the data sheet of the sensor. However, due to variations in manufacturing it is imprecisely known. A one-time lab calibration is usually done to accurately determine the calibration constant, or gain, of the sensor. The bias term β_{accel} is dependent on temperature and should be calibrated prior to each flight.

In aircraft applications, three accelerometers are commonly used. The accelerometers are mounted near the center of mass, with the sensitive axis of one accelerometer aligned with each of the body axes. Accelerometers measure the specific force in the body frame of the vehicle. Another interpretation is that they measure the difference between the acceleration of the aircraft and the gravitational acceleration. A physically intuitive explanation is given in [7, p. 13-15]. Additional explanation is given in [14, p. 27]. Mathematically we have

$$\begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} = \frac{d\mathbf{v}}{dt_b} + \boldsymbol{\omega}_{b/i} \times \mathbf{v} - R_v^b \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix}$$

In component form we have

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

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

The output of an accelerometer is an analog voltage signal that is converted into a number corresponding to the voltage inside the autopilot microcontroller by an analog-to-digital converter

at a sample rate T_s. By dividing by the calibration constant, k_{acc}, this voltage can be converted to a numerical representation of the acceleration in meters per second squared. Assuming that the biases have been removed through the calibration process, the accelerometer signals inside the autopilot can be modeled as

$$\begin{aligned} y_{accel,x} &= \dot{u} + qw - rv + g\sin\theta + \frac{\eta_{accel,x}}{k_{accel}} \\ y_{accel,y} &= \dot{v} + ru - pw - g\cos\theta\sin\phi + \frac{\eta_{accel,y}}{k_{accel}} \\ y_{accel,z} &= \dot{w} + pv - qu - g\cos\theta\cos\phi + \frac{\eta_{accel,z}}{k_{accel}}, \end{aligned} \tag{8.1}$$

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

Depending on the organization of the simulation software, the terms $\dot{u}$, $\dot{v}$, and $\dot{w}$ (state derivatives), may be inconvenient to calculate for inclusion in Equation (8.1). As an alternative, we can substitute from Equations (5.4), (5.5), and (5.6) to obtain

$$\begin{aligned} y_{accel,x} &= \frac{\rho V_a^2 S}{2\mathsf{m}}\left[C_X(\alpha) + C_{X_q}(\alpha)\frac{\bar{c}q}{2V_a} + C_{X_{\delta_e}}(\alpha)\delta_e\right] + \frac{\rho S_{\text{prop}} C_{\text{prop}}}{2\mathsf{m}}\left[(k_{\text{motor}}\delta_t)^2 - V_a^2\right] + \frac{\eta_{accel,x}}{k_{accel}} \\ y_{accel,y} &= \frac{\rho V_a^2 S}{2\mathsf{m}}\left[C_{Y_0} + C_{Y_\beta}\beta + C_{Y_p}\frac{bp}{2V_a} + C_{Y_r}\frac{br}{2V_a} + C_{Y_{\delta_a}}\delta_a + C_{Y_{\delta_r}}\delta_r\right] + \frac{\eta_{accel,y}}{k_{accel}} \\ y_{accel,z} &= \frac{\rho V_a^2 S}{2\mathsf{m}}\left[C_Z(\alpha) + C_{Z_q}(\alpha)\frac{\bar{c}q}{2V_a} + C_{Z_{\delta_e}}(\alpha)\delta_e\right] + \frac{\eta_{accel,z}}{k_{accel}}. \end{aligned} \tag{8.2}$$

The result is that the accelerometer model calculations are now based only on system parameters, states, and inputs to the system. With the exception of the noise terms, the terms on the right hand sides of Equation (8.2) 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 (8.2) can be divided by g. The choice of units is up to the preference of the user, however, maintaining consistent units reduces the potential for mistakes in implementation.

8.2 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

$$\mathbf{a}_C = 2\boldsymbol{\Omega} \times \mathbf{v}, \tag{8.3}$$

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 8.2. In this figure, the cantilever and proof mass are actuated at their resonant frequency to cause oscillation in the vertical plane. The cantilever is actuated so that the velocity of the proof mass due to these oscillations is a constant amplitude sinusoid

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

where A is the amplitude of the oscillation and ω_n is the natural frequency of the oscillation. If the sensitive axis of the rate gyro is configured to be the longitudinal axis of the undeflected cantilever, then rotation about this axis will result in a Coriolis acceleration in the horizontal plane described by Equation 8.3 and shown in Figure 8.2. 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 capacitative coupling, through a piezoelectrically generated charge, or through a change in piezoresistance of the cantilever. Whatever the transduction method, a voltage proportional to the lateral Coriolis acceleration is produced.

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

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

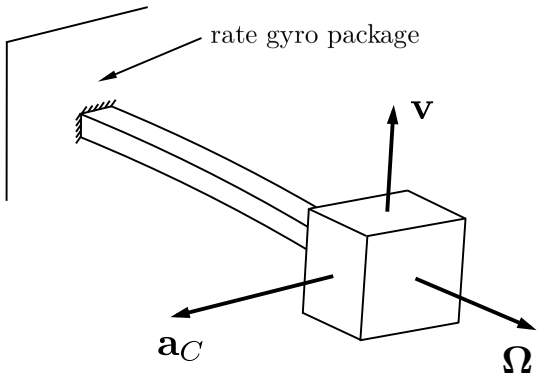


Figure 8.2: Conceptual depiction of proof mass rate gyro. $\boldsymbol{\omega}$ is the angular velocity of the sensor package to be measured. $\mathbf{v}$ is the actuated vibration velocity of the cantilever. $\mathbf{a}_C$ is the Coriolis acceleration that results as the sensor package undergoes an angular velocity.

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

The output of a rate gyro can be modeled as

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

where Υ_{gyro} corresponds the the measured rate of rotation in Volts, k_{gyro} is a gain converting the rate in radians per second to Volts, Ω is the angular rate in radians per second, β_{gyro} is a bias term, and η_{gyro} is zero mean Gaussian noise. An approximate value for the gain k_{gyro} should be given on the spec sheet of the sensor. To ensure accurate measurements, the value of this gain should be determined through experimental calibration. The bias term β_{gyro} is strongly dependent on temperature and should be calibrated prior to each flight.

If three rate gyros are aligned along the x, y, and z axes of the MAV, then the rate gyros measure the angular body rates p, q, and r as follows:

$$\begin{aligned}
\Upsilon_{gyro,x} &= k_{gyro,x}p + \beta_{gyro,x} + \eta_{gyro,x} \\
\Upsilon_{gyro,y} &= k_{gyro,y}q + \beta_{gyro,y} + \eta_{gyro,y} \\
\Upsilon_{gyro,z} &= k_{gyro,z}r + \beta_{gyro,z} + \eta_{gyro,z}.
\end{aligned}$$

For simulation purposes, we are interested in modeling the calibrated gyro signals inside the autopilot. The rate gyro signals are converted from analog voltages coming out of the sensor to numerical representations of angular rates (in units of rad/s) inside the autopilot. We assume that the gyros have been calibrated so that in the nominal case 1 rad/s or angular rate experienced by the sensor results in a numerical measurement inside the autopilot of 1 rad/s (i.e., the gain from the

physical rate to its numerical representation inside the autopilot is 1) are and that the biases have been estimated and subtracted from the measurements to produce

$$\begin{aligned} y_{gyro,x} &= p + \frac{\eta_{gyro,x}}{k_{gyro}} \\ y_{gyro,y} &= q + \frac{\eta_{gyro,y}}{k_{gyro}} \\ y_{gyro,z} &= r + \frac{\eta_{gyro,z}}{k_{gyro}}, \end{aligned} \tag{8.4}$$

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

8.3 Pressure Sensors

Pressure is a quantity commonly associated with fluids that 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.

8.3.1 Altitude Measurement

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

$$P_2 - P_1 = \rho g(z_2 - z_1), \tag{8.5}$$

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 (8.5) 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 Equa-

tion (8.5), the change in pressure due to a change in altitude is given by

$$\begin{aligned} P - P_{\text{ground}} &= -\rho g(h - h_{\text{ground}}) \\ &= -\rho g h_{\text{AGL}}, \end{aligned} \tag{8.6}$$

where h and h_{ground} are measured with respect to sea level and P is the corresponding absolute pressure measurement. The change in sign between Equations (8.5) and (8.6) 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 implementation, P_{ground} is the atmospheric pressure measured at ground level prior to take off and ρ is the air density at the flight location.

Equation (8.6) 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 [25]. This formula takes into account the change in density and pressure due to decreasing temperature with altitude and is given by

$$P = P_0 \left[\frac{T_0}{T_0 + L_0 h_{\text{ASL}}} \right]^{\frac{gM}{RL_0}}, \tag{8.7}$$

where P_0 = 101,325 N/m^2 is the standard pressure at sea level, T_0 = 288.15 K is the standard temperature at sea level, L_0 = -0.0065 K/m is the lapse rate or the rate of temperature decrease in the lower atmosphere, g = 9.80665 m/s^2 is the gravitational constant, R = 8.31432 N-m/(mol-K) is the universal gas constant for air, and M = 0.0289644 kg/mol is the standard molar mass of atmospheric air. The altitude h_{ASL} is referenced to sea level.

The relative significance of the constant-density assumption can be seen by comparing pressures calculated using Equations (8.6) and (8.7) as shown in Figure 8.3. It can be seen that over the full range of altitudes for which the barometric formula is valid (0 to 11,000 m), the pressure versus altitude relationship is not linear and that the linear approximation of Equation (8.6) is not valid. The plot on the right of Figure 8.6, however, shows that over narrower altitude ranges, such as those common to small unmanned aircraft, a linear approximation can be used with reasonable accuracy.

One key to accurately calculating altitude from pressure using Equation (8.6) is to have an accurate measure of air density at the flight location. This can be determined from the ideal gas

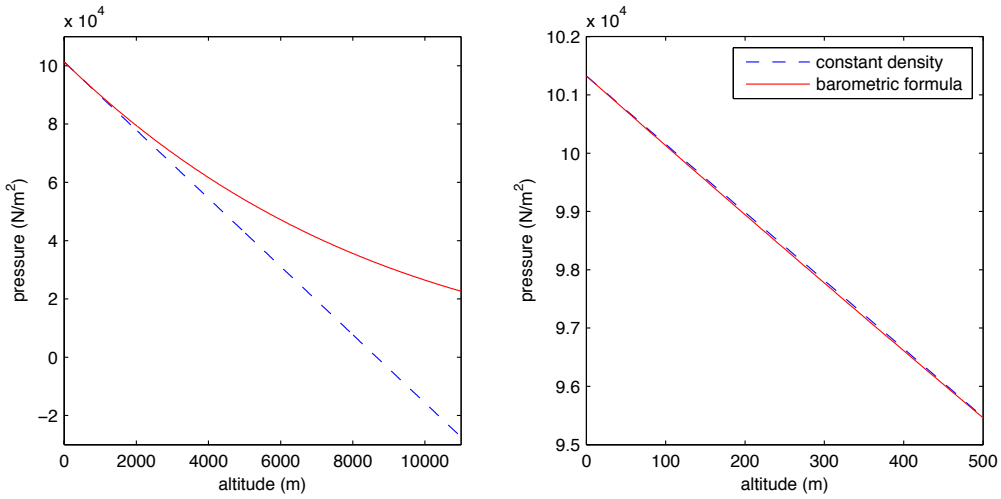


Figure 8.3: Comparison of atmospheric pressure calculations using constant-density and variable-density models.

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

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

The atmospheric pressure is expressed in N/m^2. Typical weather data reports pressure in inches of mercury (Hg). The conversion factor is

$$P\,[\text{N/m}^2] = 3385P\,[\text{in of Hg}].$$

where H is the pressure in inches of Mercury.

In practice, we will utilize the measurement of absolute pressure to give an indication of altitude above ground level of the aircraft. Figure 8.4 shows an example of an absolute pressure sensor in schematic form. The pressure sensor consist of two volumes separated by a diaphragm. The volume on the left is closed and at a constant reference pressure. The volume at the right 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.

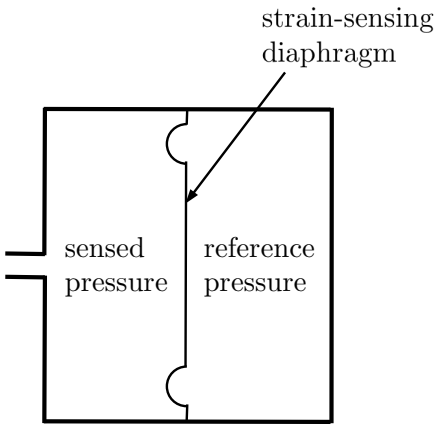


Figure 8.4: Schematic of an absolute pressure sensor.

Following Equation (8.6), the output of the absolute pressure sensor of interest is given by

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

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

8.3.2 Airspeed Sensor

Airspeed can be measured using a pitot-static probe in conjunction with a differential pressure transducer. Such a sensor is depicted schematically in Figure 8.5. 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 know 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.

Bernoulli's equation states that total pressure is the sum of the static pressure and dynamic

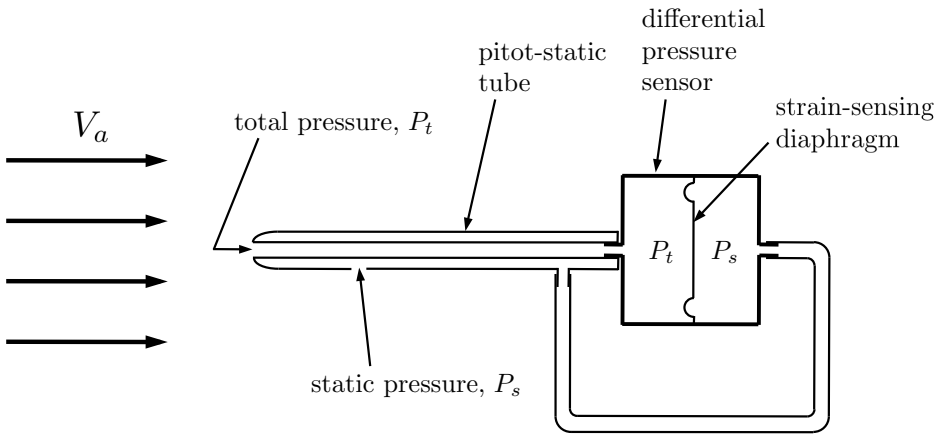


Figure 8.5: Schematic of pitot-static tube and differential pressure sensor. Not to scale.

pressure. In equation form, we can write this as

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

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

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

which is the quantity that the differential pressure sensor measures. With proper calibration to covert the sensor output from volts to a number inside the microcontroller representing pressure in units of N/m^2, we can model the output of the differential pressure sensor as

$$y_{\text{diff pres}} = \frac{\rho V_a^2}{2} + \eta_{\text{diff pres}}, \tag{8.8}$$

where $\eta_{\text{diff pres}}$ is zero mean Gaussian noise with variance $\sigma^2_{\text{diff pres}}$. The absolute and differential pressure sensors are analog devices that are sampled by the on-board processor. We will assume that the sample rate is given by T_s.

8.4 Magnetometers

The earth's magnetic field is three dimensional. There is a component that points to magnetic North, but there is also a vertical component. A three axis magnetometer measures the strength of the magnetic field along each of its axes. Let $\mathbf{m}_0$ be the magnetic field vector that is fixed in the inertial or vehicle frame. Given the orientation of the airframe, the strength of the magnetic field

along the body axes of the MAV are given by

$$\begin{aligned}\mathbf{m}(t) &= R_v^b \mathbf{m}_0 \\ &= \begin{pmatrix} c\theta c\psi & c\theta s\psi & -s\theta \\ s\phi s\theta c\psi - c\phi s\psi & s\phi s\theta s\psi + c\phi c\psi & s\phi c\theta \\ c\phi s\theta c\psi + s\phi s\psi & c\phi s\theta s\psi - s\phi c\psi & c\phi c\theta \end{pmatrix} \mathbf{m}_0.\end{aligned}$$

Unfortunately, when the electric motor is activated, an additional magnetic field is activated due to the rotation of the motor coils. Fortunately, the magnetic field is sinusoidal, with a frequency approximately equal to the frequency of the rotating rotor, and can be removed with a low-pass filter. We will model the effect due to the motor as

$$\mathbf{m}_{\text{motor}} = \sin(\omega_{\text{motor}} t)\mathbf{m}_{\text{motor}}$$

which is measured in the body frame.

Including bias and white noise terms, the output of the magnetometer is given by

$$\begin{aligned}\mathbf{y}_{\text{mag}} = &\begin{pmatrix} c\theta c\psi & c\theta s\psi & -s\theta \\ s\phi s\theta c\psi - c\phi s\psi & s\phi s\theta s\psi + c\phi c\psi & s\phi c\theta \\ c\phi s\theta c\psi + s\phi s\psi & c\phi s\theta s\psi - s\phi c\psi & c\phi c\theta \end{pmatrix} \mathbf{m}_0 \\ &+ \sin(\omega_{\text{motor}} t)\mathbf{m}_{\text{motor}} + \begin{pmatrix} \beta_{mag,x} \\ \beta_{mag,y} \\ \beta_{mag,z} \end{pmatrix} + \begin{pmatrix} \eta_{mag,x} \\ \eta_{mag,y} \\ \eta_{mag,z} \end{pmatrix}, \quad (8.9)\end{aligned}$$

where $\eta_{mag,x}$, $\eta_{mag,y}$, and $\eta_{mag,z}$ are zero mean Gaussian processes with variance $\sigma^2_{mag,x}$, $\sigma^2_{mag,y}$, $\sigma^2_{mag,z}$, respectively. The magnetometers are analog sensors that are sampled by the autopilot at sample rate T_s.

8.5 GPS

Typical GPS receivers provide information on the North position, East position, altitude, course angle, and ground speed of the MAV. However, each of these measurements contain error. There are several well known sources of error for GPS including atmospheric effects, satellite misalignment effects, satellite clock drift, multipath, and measurement noise. Atmospheric effects and clock drift introduce a constant bias into the measurement. Misalignment and multipath introduce a slowly time-varying bias, and measurement error can be modeled as a zero mean Gaussian process. Therefore, an abstract model for GPS measurements which is suitable for simulation purposes, is given

by

$$\begin{aligned} y_{GPS,n}(t) &= p_n + \nu_{n,\text{atmosphere}} + \nu_{\text{clock}} + \eta_{n,\text{measurement}}(t) \\ &\quad + A_{\text{n,geometry}} \sin(\omega_{\text{geometry}} t + \nu_{n,\text{geometry}}) + A_{\text{n,multipath}} \sin(\omega_{\text{multipath}} t + \nu_{n,\text{multipath}}) \end{aligned} \tag{8.10}$$

$$\begin{aligned} y_{GPS,e}(t) &= p_e + \nu_{e,\text{atmosphere}} + \nu_{e,\text{clock}} + \eta_{e,\text{measurement}}(t) \\ &\quad + A_{\text{e,geometry}} \sin(\omega_{\text{geometry}} t + \nu_{e,\text{geometry}}) + A_{\text{e,multipath}} \sin(\omega_{\text{multipath}} t + \nu_{e,\text{multipath}}) \end{aligned} \tag{8.11}$$

$$\begin{aligned} y_{GPS,h}(t) &= -p_d + \nu_{h,\text{atmosphere}} + \nu_{h,\text{clock}} + \eta_{h,\text{measurement}}(t), \\ &\quad + A_{\text{h,geometry}} \sin(\omega_{\text{geometry}} t + \nu_{h,\text{geometry}}) + A_{\text{h,multipath}} \sin(\omega_{\text{multipath}} t + \nu_{h,\text{multipath}}), \end{aligned} \tag{8.12}$$

$$y_{GPS,V_g} = \sqrt{\left(\frac{y_{GPS,n}(t + T_{s,GPS}) - y_{GPS,n}(t)}{T_{s,GPS}}\right)^2 + \left(\frac{y_{GPS,e}(t + T_{s,GPS}) - y_{GPS,e}(t)}{T_{s,GPS}}\right)^2} \tag{8.13}$$

$$y_{GPS,\chi} = \tan^{-1}\left(\frac{y_{GPS,e}(t + T_{s,GPS}) - y_{GPS,e}(t)}{y_{GPS,n}(t + T_{s,GPS}) - y_{GPS,n}(t)}\right), \tag{8.14}$$

where p_n, p_e, and h are the actual earth coordinates and altitude above sea level, $T_{s,GPS}$ is the sample rate of the GPS sensor, A_*, ω_*, and ν_* are random variables drawn from a uniform distribution at the beginning of the simulation, and η_* are zero mean Gaussian random processes with variance σ_*^2.

RWB: Work this in.

Sources of error:

The weather effects the speed of light in the atmosphere. However, this inaccuracy should be relatively constant for a given day. We will model the effect of the atmosphere by a constant random variable drawn from a uniform distribution.

The geometry of the Satellites viewed by the receiver is used to triangulate the location of the GPS receiver. Triangulation is much more effective in the horizontal plane than in the vertical direction. The satellite geometry is slowly changing in time. Therefore we will measure the effect of satellite geometry as a sinusoid with amplitude equal A_{geometry} and a frequency equal to $\omega_{\text{geometry}} = 10^{-6}$ and a phase η_{geometry}, each of which are modeled as constant random variables drawn from uniform distributions at the beginning of the simulation.

We will assume that the clock drift is relatively constant over time. Therefore, we will model the clock drift by a constant random variable drawn from a uniform distribution.

The multipath is a function of the position of the MAV. Therefore we will assume that the error is a sinusoidal signal with a magnitude of $A_{\text{multipath}}$, a frequency equal to $\omega_{\text{multipath}} = 10^{-3}$ and

phase equal to $\nu_{\text{multipath}}$, each of which are drawn from a uniform distribution at the beginning of the simulation.

8.6 Chapter Summary

Notes and References

8.7 Design Project

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

8.1 Download the files `chap8_sensors.zip` from the book website. Note that we have added a block for the sensors which contains to files: `sensors.m` and `gps.m`. The file `sensors.m` will model all of the sensors that update at rate T_s (gyros, accelerometers, pressure sensors), and `gps.m` will model the GPS sensor which is updated at rate $T_{s,GPS}$. Note also, that the order of the inputs for `autopilot.m` have changed so that the sensors are the first input, and the true states are appended to the input. The true states will be used to compare to the estimated states (subject of the next chapter). Therefore, you will need to modify the skeleton code `autopilot.m` to include the autopilot that you developed in Chapter 6.

8.2 Using the sensor parameters listed in F, modify `sensors.m` to simulate the output of the rate gyros (Eq. (8.4)), the accelerometers (Eq. (8.2)), and the pressure sensors (Eq (**??**) and (8.8)).

8.3 Using the sensor parameters listed in F, modify `gps.m` to simulate the output of the GPS sensor (Eq. (8.10)–(8.14)).

8.4 Using a Simulink scope, observe the output of each sensor and verify that its sign and magnitude are approximately correct, and that the shape of the waveform is approximately correct.

Chapter 9

State Estimation

The autopilots designed in Chapter 6 all assume 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 8. Since rate gyros directly measure roll rates in the body frame, the states p, q, and r can be recovered by low pass filtering the rate gyros. Therefore we begin the chapter by discussing digital implementation of low pass filter in Section 9.2. In Section 9.3 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 very well. Accordingly, in Section 9.4 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 9.5. The reader with limited exposure to stochastic processes is encouraged to refer to Appendix E which provides an overview of basic concepts from probability theory with a focus on Gaussian stochastic process. The last two sections of the chapter describe applications of the Kalman filter. In Section 9.6, the Kalman filter is used to estimate the attitude of the MAV, and in Section 9.7, the Kalman filter is used to estimate the position and heading the MAV, as well as the constant ambient wind.

9.1 Base 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 meters, and an airspeed of 10 meters/sec. The maneuver is defined by commanding

a constant airspeed of 10 meters/sec, by commanding altitude and heading as shown in Figure 9.1.

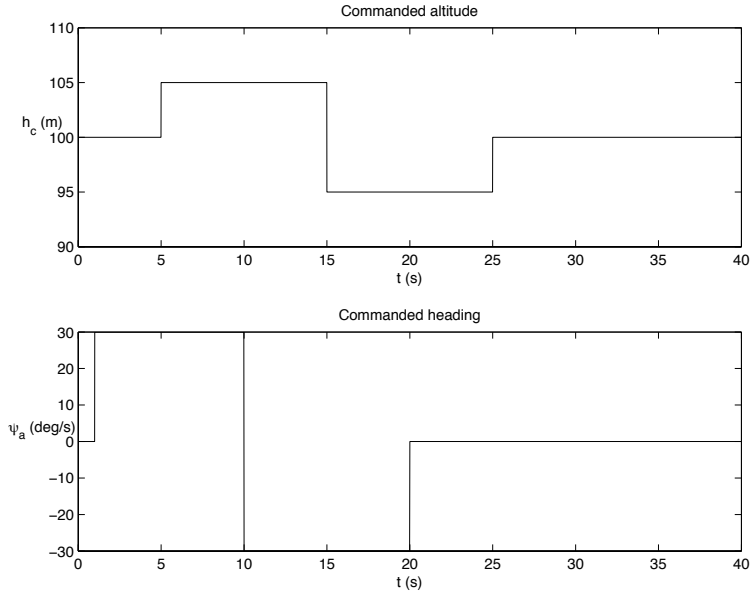


Figure 9.1: The altitude and heading commands that define the base maneuver that will be used to evaluate and tune the state estimation scheme.

9.2 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 the low pass filter. The Laplace transforms representation of a simple low-pass filter 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

$$\dot{y} = -ay + au. \tag{9.1}$$

Using a zeroth order approximation of the derivative we get

$$\frac{y(t+T_s) - y(t)}{T_s} = -ay(t) + au(t),$$

where T_s is the sample rate. Solving for $y(t+T_s)$ we get

$$y(t+T_s) = (1-aT)s)y(t) + aT_s u(t).$$

For the zeroth order approximation to be valid we need $aT_s \ll 1$. If we let $\alpha = aT_s$ then we get the simple form

$$y(t+T_s) = (1-\alpha)y(t) + \alpha u(t).$$

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

In the derivation of the discrete-time implementation of the low-pass filter, it is possible to be more precise. In particular, returning to Equation (9.1), from linear systems theory, it is well known that the solution is given by

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

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

$$\begin{aligned} y(t+T_s) &= e^{-aT_s} y(t) + a \int_0^{T_s} e^{-a(T_s-\tau)}\, d\tau u(t) y(t+T_s) \\ &= e^{-aT_s} y(t) + (1-e^{-aT_s}) u(t). \end{aligned} \tag{9.2}$$

Note that since $e^x = 1 + x + \frac{x^2}{2!} + \ldots$ we have that $e^{-aT_s} \approx 1 - aT_s$ and $1 - e^{-aT_s} \approx aT_s$.

Therefore, if the sample rate is not fixed (common for micro controllers), and it is desired to have a fixed cut-off frequency, then Equation (9.2) is the preferable way to implement a low-pass filter in digital hardware.

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.

9.3 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 8. While this method is effective for angular rates, altitude, and airspeed, it is not effective for estimating the Euler angles or the position and heading of the MAV.

9.3.1 Position and Heading

The position of the MAV can be estimated by low-pass filtering Equations (8.10) and (8.11). The biases due to multipath, clock, and satellite geometry will not be removed, but at present, there is not a known scheme for removing bias. The estimate of the position variables is therefore given by

$$\hat{p}_n = LPF(y_{\text{GPS,n}}) \tag{9.3}$$

$$\hat{p}_e = LPF(y_{\text{GPS,e}}). \tag{9.4}$$

Similarly, an estimate of the heading angle of the MAV can be obtained by low-pass filtering Equation (8.14), and assuming that there is no wind:

$$\hat{\psi} = LPF(y_{\text{GPS,heading}}). \tag{9.5}$$

The primary downside of low-pass filtering GPS signals is that since the sample rate is slow (usually on the order of 1 Hz), there is significant delay in the estimate. The estimation scheme described in Section 9.7 will resolve this problem.

For the base maneuver discussed in Section 9.1, the estimation error for the North and East position, and for the heading are shown in Figure 9.2. There is significant error due, in part, to the fact that the GPS sensor is only updated every one second. Clearly, simply low pass filtering the GPS data does not produce satisfactory results.

9.3.2 Angular Rates

Similarly, the angular rates p, q, and r can be estimated by low-pass filtering the rate gyro signals given by Equation (8.4) to obtain

$$\hat{p} = LPF(y_{\text{gyro,x}}) \tag{9.6}$$

$$\hat{q} = LPF(y_{\text{gyro,y}}) \tag{9.7}$$

$$\hat{r} = LPF(y_{\text{gyro,z}}). \tag{9.8}$$

For the base maneuver discussed in Section 9.1, the estimation error for p, q, and r are shown in Figure 9.3, which shows that low pass filtering the gyro measurements produces acceptable estimates.

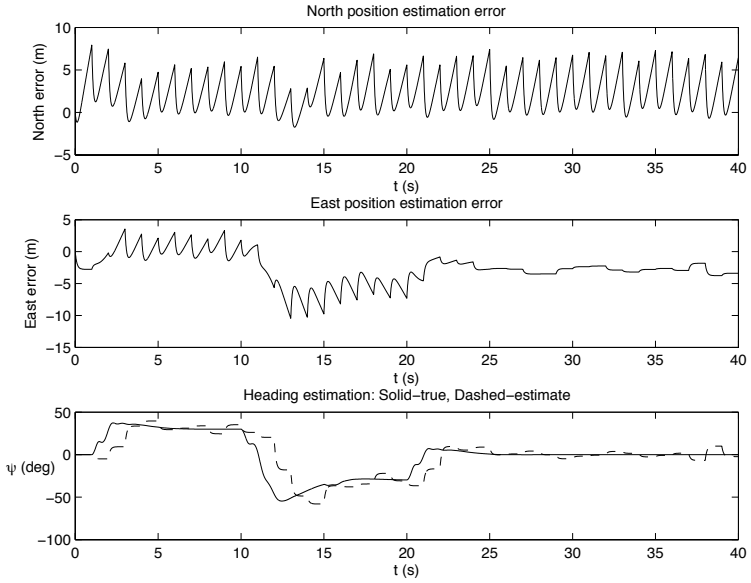


Figure 9.2: Estimation error on the North and East position and the heading obtained by low pass filtering the GPS sensors.

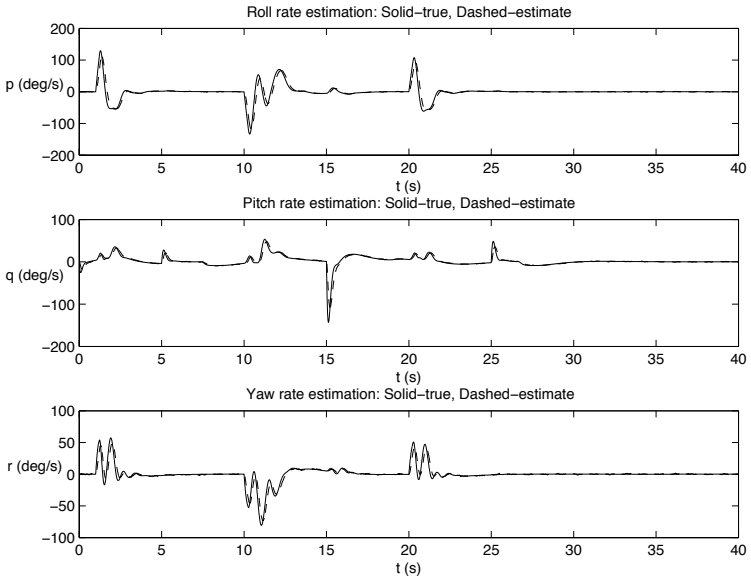


Figure 9.3: Estimation error on the angular rates obtained by low pass filtering the rate gyros.

9.3.3 Altitude

GPS is not accurate enough to estimate the altitude. Therefore, we will use the absolute pressure sensor. Applying a low pass filter to Equation (**??**) and dividing y ρg we get

$$\hat{h} = \frac{LPF(y_{\text{static pres}})}{\rho g}. \tag{9.9}$$

9.3.4 Airspeed

The airspeed can be estimated by applying a low pass filter to the differential pressure sensor represented by Equation (8.8), and inverting to obtain

$$\hat{V}_a = \sqrt{\frac{2}{\rho} LPF(y_{\text{diff pres}})}. \tag{9.10}$$

For the base maneuver discussed in Section 9.1, the estimation error for the altitude and airspeed are shown in Figure 9.4. As can be seen from the figure, inverting the sensor model produces a fairly accurate model of the altitude and airspeed.

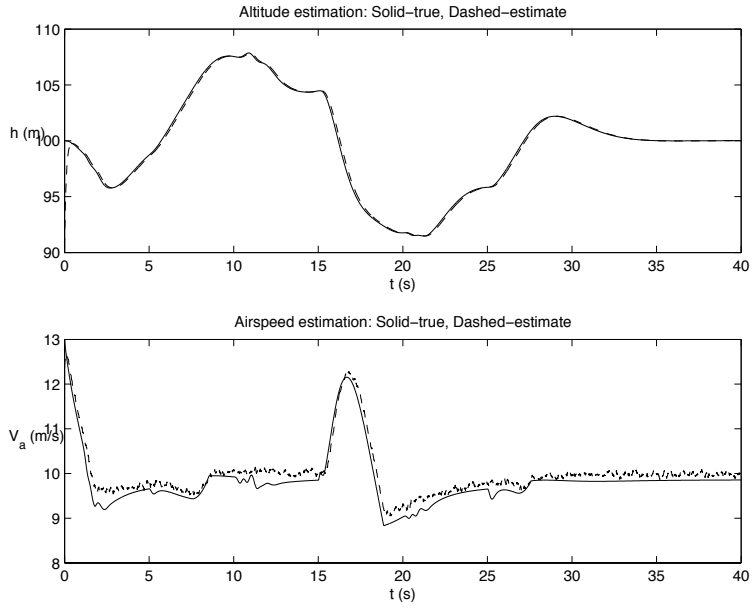


Figure 9.4: Estimation error of the altitude and airspeed obtained by low pass filtering the pressure sensors and inverting the sensor model. For altitude an airspeed, the accuracy of the simple scheme is adequate.

9.3.5 Roll and Pitch Angles

Roll and Pitch angles are the most difficult variables to estimate well on small and micro UAVs. An extremely simple scheme, that works in unaccelerated flight can be derived as follows. Recall from Equation (8.1) that

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

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

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

Solving for ϕ and θ we get

$$\hat{\phi}_{\text{accel}} = \tan^{-1}\left(\frac{LPF(y_{\text{accel,y}})}{LPF(y_{\text{accel,z}})}\right) \tag{9.11}$$

$$\hat{\theta}_{\text{accel}} = \sin^{-1}\left(\frac{LPF(y_{\text{accel,x}})}{g}\right). \tag{9.12}$$

The estimation error of the roll and pitch angles for the base maneuver discussed in Section 9.1 are shown in Figure 9.5, where it is clear that the estimation error during accelerated flight is unacceptable.

Another simple idea is to notice that we are not taking into account the rate gyro's, which give valuable information about the angles. Recalling from Equations (5.7) and (5.8) that

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

and assuming that $\phi \approx 0$ and $\theta \approx 0$ we get

$$\begin{aligned} \dot{\phi} &= p \\ \dot{\theta} &= q. \end{aligned}$$

Therefore we can integrate these equations to obtain an additional estimate of ϕ and θ:

$$\begin{aligned} \hat{\phi}_{\text{int}} &= \int_{-\infty}^{t} p(\tau)\, d\tau \\ \hat{\theta}_{\text{int}} &= \int_{-\infty}^{t} q(\tau)\, d\tau. \end{aligned}$$

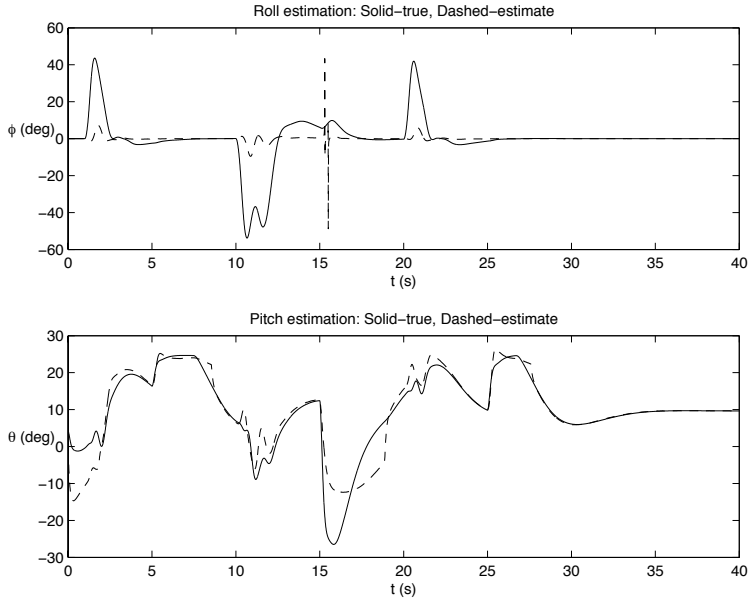


Figure 9.5: Estimation error on the roll and pitch angles obtained by low pass filtering the accelerometers and inverting the model. Since this scheme assumes unaccelerated flight, during maneuvers where acceleration exists, the estimation error can be unacceptably large.

Combining the estimate from the integrator and the accelerometers we obtain

$$\hat{\phi} = \kappa\hat{\phi}_{\text{int}} + (1-\kappa)\hat{\phi}_{\text{accel}} \tag{9.13}$$

$$\hat{\theta} = \kappa\hat{\theta}_{\text{int}} + (1-\kappa)\hat{\theta}_{\text{accel}}, \tag{9.14}$$

where $\kappa \in (0, 1)$.

The estimation error of the roll and pitch angles using this scheme is shown in Figure 9.6. The shape of the estimates are much closer to the actual values, but there is clearly a bias, which is a result of integrating the gyros. A variety of ad hoc schemes could be introduced to reset the integrators. However, a more satisfying approach that produces quality results is to dynamic observers as discussed in the remainder of this chapter.

9.4 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.

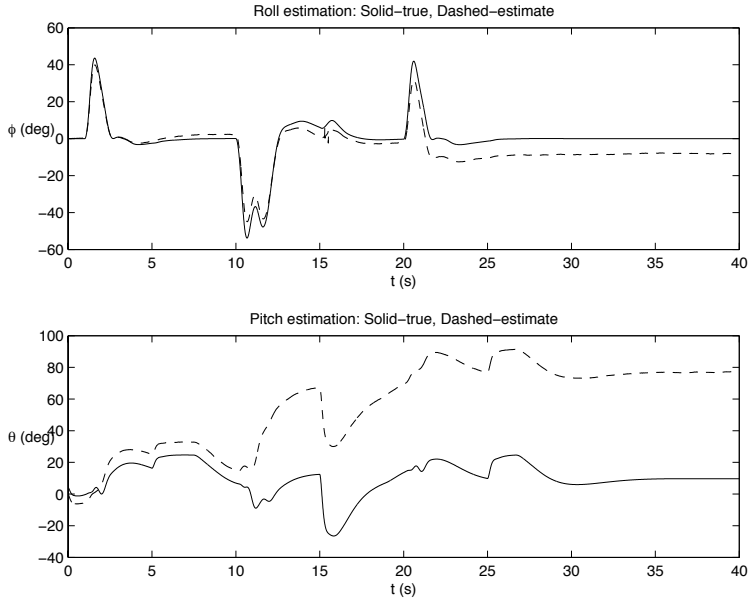


Figure 9.6: Estimation error on the roll and pitch angles obtained by using Equations (9.13) and (9.14). The bias is due to integrating the gyros.

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

$$\dot{\hat{x}} = \underbrace{A\hat{x} + Bu}_{\text{copy of the model}} + \underbrace{L\,(y - C\hat{x})}_{\text{correction due to sensor reading}}, \tag{9.15}$$

where $\hat{x}$ is the estimated value of x. Letting $\tilde{x} = x - \hat{x}$ we observe 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 some sample rate T_s. How do we modify the observer equation shown in Equation (9.15) to account for sampled sensor readings? The typical approach is to propagate the system model between samples using the equation

$$\dot{\hat{x}} = A\hat{x} + Bu \tag{9.16}$$

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

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

where t_k is the instant in time that the measurement is received and $\hat{x}^-$ is the state estimate produced by Equation (9.16) at time t_k. Equation (9.16) is then re-instantiated with initial conditions given by $\hat{x}^+$. The continuous-discrete observer is summarized in Table 9.4.

System model:
$\dot{x} = Ax + Bu$
$y(t_k) = Cx(t_k)$
Initial Condition $x(0)$.
Assumptions:
Knowledge of A, B, C, $u(t)$.
No measurement noise.
In between measurements $(t \in [t_{k-1}, t_k))$**:**
Propagate $\dot{\hat{x}} = A\hat{x} + Bu$.
Initial condition is $\hat{x}^+(t_{k-1})$.
Label the estimate at time t_k as $\hat{x}^-(t_k)$.
At sensor measurement $(t = t_k)$**:**
$\hat{x}^+(t_k) = \hat{x}^-(t_k) + L\left(y(t_k) - C\hat{x}^-(t_k)\right).$

Table 9.1: Continuous-Discrete Linear Observer.

The observation process is shown graphically in Figure 9.7. Note that it is not necessary to have a fixed sample rate. The continuous-discrete observer can be implemented using Algorithm 4.

9.5 Derivation of the Kalman Filter

In this section we derive the Kalman filter. Two forms of the Kalman filter are particularly useful for MAV applications. The first is the fixed gain Kalman filter which is derived (via the Continuous-Update, Continuous-Measurement Kalman Filter) in Section 9.5.1. The second is the continuous-update, discrete-measurement Kalman filter which is derived in Section 9.5.2.

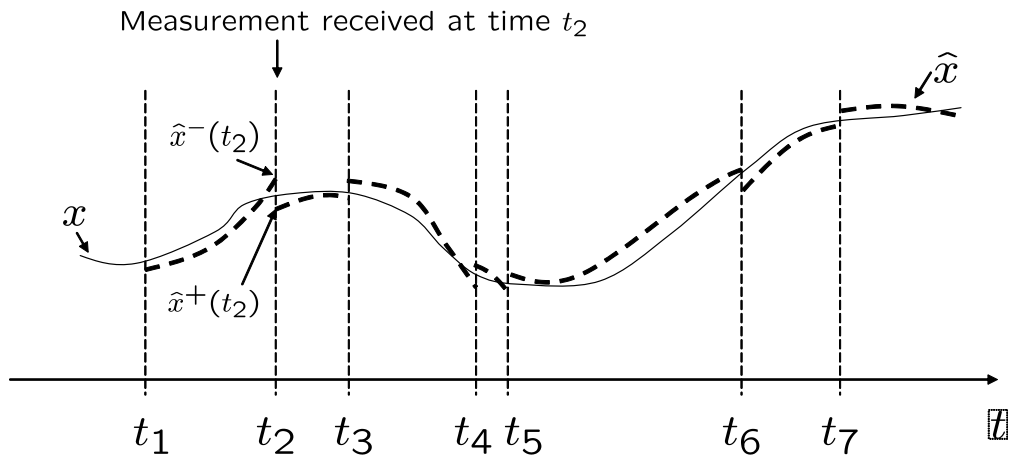


Figure 9.7: sdf

Algorithm 4 Continuous-Discrete Observer

Initialize: $\hat{x} = 0$.
Pick an output sample rate T_{out} which is much less than the sample rates of the sensors.
At each sample time T_{out}:
for $i = 1$ to N **do** {Propagate the state equation.}
    $\hat{x} = \hat{x} + \left(\frac{T_{out}}{N}\right)(A\hat{x} + Bu)$
end for
if A measurement has been received from sensor i **then** {Measurement Update}
    $\hat{x} = \hat{x} + L_i\left(y_i - C_i\hat{x}\right)$
end if

9.5.1 Fixed Gain Kalman Filter

If the state evolves according to a differential equation, and the measurements are obtained continuously in time, then an appropriate linear systems model is

$$\dot{x} = Ax + Bu + G\xi \tag{9.17}$$

$$y = Cx + \eta, \tag{9.18}$$

where $\xi \sim \mathcal{N}(0, Q)$ and $\eta \sim \mathcal{N}(0, R)$, i.e, $E\{\xi\} = 0$ and $E\{\xi(t)\xi(\tau)^T\} = Q\delta(t-\tau)$, and $E\{\eta\} = 0$ and $E\{\eta(t)\eta(\tau)^T\} = R\delta(t-\tau)$. The random variable ξ is called the process noise and represents modeling error and disturbances on the system. The random variable η is called the measurement noise and represents noise on the sensors. The covariance R can usually be estimated from sensor calibration, but the covariance Q is generally unknown and therefore becomes a system gain that can be tuned to improve the performance of the observer.

Similar to Equation (9.15), let the observer have the form

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - C\hat{x}). \tag{9.19}$$

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

$$P(t) = E\{\tilde{x}(t)\tilde{x}(t)^T\} \tag{9.20}$$

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 to minimize the eigenvalues of $P(t)$. Recall that

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

where $tr(P)$ is the trace of P and λ_i are the eigenvalues. Therefore, minimizing $tr(P)$ minimizes the estimation error covariance.

The Kalman filter is derived by choosing L in (9.19) so that the expected value of the estimation error is zero and the trace of the error covariance $P(t)$ is minimized. In the derivation that follows, we will need the following matrix relationships:

$$\frac{\partial}{\partial A} tr(A) = I \tag{9.21}$$

$$\frac{\partial}{\partial A} tr(BAD) = B^T D^T \tag{9.22}$$

$$\frac{\partial}{\partial A} tr(ABA^T) = 2AB \text{ if } B = B^T. \tag{9.23}$$

Note that

$$\begin{aligned}\dot{\tilde{x}} &= \dot{x} - \dot{\hat{x}} \\ &= Ax + Bu + G\xi - A\hat{x} - Bu - L(Cx + \eta - C\hat{x}) \\ &= (A - LC)\tilde{x} + G\xi - L\eta.\end{aligned}$$

Therefore, from linear systems theory we have

$$\tilde{x}(t) = e^{(A-LC)t}\tilde{x}(0) + \int_0^t e^{(A-LC)(t-\tau)}G\xi(\tau)\,d\tau - \int_0^t e^{(A-LC)(t-\tau)}L\eta(\tau)\,d\tau.$$

Differentiating Equation (9.20) we have

$$\begin{aligned}\dot{P}(t) &= \frac{d}{dt}E\{\tilde{x}(t)\tilde{x}(t)^T\} \\ &= E\{\dot{\tilde{x}}\tilde{x}^T + \tilde{x}\dot{\tilde{x}}\} \\ &= E\{(A - LC)\tilde{x}\tilde{x}^T + G\xi\tilde{x}^T - L\eta\tilde{x}^T + \tilde{x}\tilde{x}^T(A - LC)^T + \tilde{x}\xi^T G^T - \tilde{x}\eta^T L^T\} \\ &= (A - LC)P(t) + P(t)(A - LC)^T + GE\{\xi\tilde{x}^T\} + E\{\tilde{x}\xi^T\}G - LE\{\eta\tilde{x}^T\} - E\{\tilde{x}\eta^T\}L^T.\end{aligned}$$

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

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

where the $\frac{1}{2}$ is because we only use half of the area inside the delta function. Similar calculations give

$$\begin{aligned}E\{\xi\tilde{x}^T\} &= \frac{1}{2}QG^T \\ E\{\eta\tilde{x}^T\} &= \frac{1}{2}RL \\ E\{\tilde{x}\eta^T\} &= \frac{1}{2}L^TR.\end{aligned}$$

Therefore we have that the covariance of the estimation error satisfies the matrix differential equation given by

$$\dot{P} = (A - LC)P + P(A - LC)^T + GQG^T - LRL^T,$$

with the initial condition $P(0) = P_0$, where P_0 represents the estimation uncertainty at time $t = 0$.

If $(A-LC)$ is Hurwitz, then it can be shown that P approaches a constant value which implies that $\dot{P}$ approaches zero [26]. Therefore, in steady state we have

$$(A-LC)P + P(A-LC)^T + GQG^T - LRL^T = 0.$$

We now have the following constrained optimization problem:

$$\begin{aligned} &\text{minimize } \ \text{tr}(P) \\ &\text{subject to: } \ (A-LC)P + P(A-LC)^T + GQG^T - LRL^T = 0. \end{aligned}$$

Using a Lagrange multiplier [27] we can convert this problem to the following unconstrained optimization problem:

$$\text{minimize: } \ \mathcal{H} = \text{tr}(P) + \text{tr}\left((A-LC)P + P(A-LC)^T + GQG^T - LRL^T\right).$$

Necessary conditions for a minimum are

$$\begin{aligned} \frac{\partial \mathcal{H}}{\partial S} &= (A-LC)P + P(A-LC)^T + GQG^T - LRL^T = 0 \\ \frac{\partial \mathcal{H}}{\partial P} &= (A-LC)^T S + S(A-LC) + I = 0 \\ \frac{\partial \mathcal{H}}{\partial L} &= 2SPC^T - 2SLR = 0. \end{aligned}$$

From these necessary conditions we get that $L = PC^T R^{-1}$, where P satisfies the matrix Riccati Equation

$$AP + PA^T + GQG^T - PC^T R^{-1} CP = 0.$$

The fixed-gain Kalman observer is given in Algorithm 5. The application of Algorithm 5 to roll

Algorithm 5 Fixed Gain Kalman Filter

1: Initialize: $\hat{x} = 0$.
2: Solve the Riccati Equation $AP + PA^T + GQG^T - PC^T R^{-1} CP = 0$.
3: Set the observer gain $L = PC^T R^{-1}$.
4: Propagate the state estimate: $\hat{x} = \hat{x} + T_s \left(A\hat{x} + Bu + L(y - C\hat{x})\right.$.

and pitch angle estimation is described in Section 9.6.1.

9.5.2 Continuous-Discrete Kalman Filter

The previous section assumed that measurements are collected continuously. However, most sensors are sampled at a fixed sample rate. Therefore in this section we assume the state model

$$\begin{aligned} \dot{x} &= Ax + Bu + G\xi \\ y_k &= Cx_k + \eta_k, \end{aligned} \tag{9.24}$$

where $y_k = y(t_k)$ is the k^{th} sample of y, $x_k = x(t_k)$ is the k^{th} sample of x, η_k is the measurement noise at time t_k, ξ is a zero-mean Gaussian random process with covariance Q, and η_k is a zero-mean Gaussian random variable with covariance R. Note that the sample rate does not need to be be fixed.

The observer will therefore have the following form:

In between measurements: $(t \in [t_{k-1}, t_k])$

Propagate $\dot{\hat{x}} = A\hat{x} + Bu$.

Initial condition is $\hat{x}^+(t_{k-1})$.

Label the estimate at time t_k as $\hat{x}^-(t_k)$.

At sensor measurement $(t = t_k)$**:**

$\hat{x}^+(t_k) = \hat{x}^-(t_k) + L\left(y(t_k) - C\hat{x}^-(t_k)\right).$

Our objective is to pick L to minimize tr$(P(t))$.

Between Measurements.

Differentiating $\tilde{x}$ we get

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

Therefore we have that

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

We can therefore compute the evolution for P as

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

As in the previous section we get

$$\begin{aligned} E\{\xi\tilde{x}^T\} &= E\left\{\xi(t)\tilde{x}_0 e^{A^T t} + \int_0^t \xi(t)\xi^T(\tau)G^T e^{A^T(t-\tau)}\,d\tau\right\} \\ &= \frac{1}{2}QG^T, \end{aligned}$$

which implies that

$$\dot{P} = AP + PA^T + GQG^T.$$

At Measurements.

At a measurement we have that

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

Therefore

$$\begin{aligned} P^+ &= E\{\tilde{x}^+\tilde{x}^{+T}\} \\ &= E\left\{\left(\tilde{x}^- - LC\tilde{x}^- - L\eta\right)\left(\tilde{x}^- - LC\tilde{x}^- - L\eta\right)^T\right\} \\ &= E\left\{\tilde{x}^-\tilde{x}^{-T} - \tilde{x}^-\tilde{x}^{-T}C^TL^T - \tilde{x}^-\eta^TL^T\right. \\ &\qquad - LC\tilde{x}^-\tilde{x}^{-T} + LC\tilde{x}^-\tilde{x}^{-T}C^TL^T + LC\tilde{x}^-\eta^TL^T \\ &= \qquad \left. -L\eta\tilde{x}^{-T} + L\eta\tilde{x}^{-T}C^TL^T + L\eta\eta^TL^T\right\} \\ &= P^- - P^-C^TL^T - LCP^- + LCP^-C^TL^T + LRL^T. \end{aligned} \tag{9.25}$$

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

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

Plugging back into Equation (9.25) give

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

To this stage in the development, we have assumed that the system propagation models and measurement model are linear. However, for many applications, including the applications discussed later in this chapter, the system propagation model and the measurement model are nonlinear. In other words, Model (9.24) becomes

$$\dot{x} = f(x, u) + G\xi$$
$$y_k = h(x_k, u_k) + \eta_k.$$

For this case, the state propagation and update laws use the nonlinear model, but the propagation and update of the error covariance use the Jacobian of f for A, and the Jacobian of h for C. The resulting algorithm is called the Extended Kalman Filter (EKF), and is summarized in Algorithm 6. The application of Algorithm 6 to roll and pitch angle estimation is described in Section 9.6.2. The

Algorithm 6 Continuous-Discrete Extended Kalman Filter

Initialize: $\hat{x} = 0$.
Pick an output sample rate T_{out} which is much less than the sample rates of the sensors.
At each sample time T_{out}:
for $i = 1$ to N **do** {Prediction Step}
  $\hat{x} = \hat{x} + \left(\frac{T_{out}}{N}\right) f(\hat{x}, u)$
  $A = \frac{\partial f}{\partial x}(\hat{x}, u)$
  $P = P + \left(\frac{T_{out}}{N}\right) \left(AP + PA^T + GQG^T\right)$
end for
if Measurement has been received from sensor i **then** {Measurement Update}
  $C_i = \frac{\partial h_i}{\partial x}(\hat{x}, u_k)$
  $L_i = PC_i^T(R_i + C_iPC_i^T)^{-1}$
  $P = (I - L_iC_i)P$
  $\hat{x} = \hat{x} + L_i\left(y_i - h(\hat{x}, u_k)\right)$.
end if

application of Algorithm 6 to position, heading, and wind estimation is described in Section 9.7.

9.6 Attitude Estimation

9.6.1 Fixed Gain Kalman Filter

To derive the fixed gain Kalman filter, we lump the nonlinearities in Equations (5.7) and (5.8) into the noise variables as

$$\begin{aligned}\dot{\phi} &= p + q\sin\phi\tan\theta + r\cos\phi\tan\theta + \xi_\phi' \\ &= p + \xi_\phi \qquad (9.26)\\ \dot{\theta} &= q\cos\phi - r\sin\phi + \xi_\theta' \\ &= q + \left(-q + q\cos\phi - r\sin\phi + \xi_\theta'\right) \\ &= q + \xi_\theta, \qquad (9.27)\end{aligned}$$

where we will assume that $\xi_\phi \sim \mathcal{N}(0, Q_\phi)$ and $\xi_\theta \sim \mathcal{N}(0, Q_\theta)$.

We can use Equation (9.26) and (9.27) as the model with p and q considered as inputs. Also note that the equations are not coupled so we will derive two one-state Kalman filters where the system matrices in Equation (9.17) are given by $A_\phi = A_\theta = 0$, $B_\phi = B_\theta = 1$, and $G_\phi = G_\theta = 1$. We will use the accelerometers to produce the measurements. Therefore

$$\begin{aligned}y_\phi &= \hat{\phi}_{\text{accel}} = \phi + \eta_\phi \\ y_\theta &= \hat{\theta}_{\text{accel}} = \theta + \eta_\theta,\end{aligned}$$

where $\hat{\phi}_{\text{accel}}$ and $\hat{\theta}_{\text{accel}}$ are given by Equations (9.11) and (9.12), and where we will assume that $\eta_\phi \sim \mathcal{N}(0, R_\phi)$ and $\eta_\theta \sim \mathcal{N}(0, R_\theta)$. Therefore $C_\phi = C_\theta = 1$. From Algorithm 5, the steady-state Kalman filter equations are

$$\begin{aligned}\dot{\hat{x}} &= A\hat{x} + Bu + L(y - C\hat{x}) \\ L &= PC^T R^{-1} \\ 0 &= AP + PA^T + GQG^T - PC^T R^{-1} CP.\end{aligned}$$

The Riccati equation for the roll angle simplifies to

$$Q_\phi - \frac{P_\phi^2}{R_\phi} = 0.$$

Therefore we have

$$P_\phi = \sqrt{Q_\phi R_\phi},$$

and the Kalman gain becomes

$$\begin{aligned} L_\phi &= P_\phi C_\phi^T R_\phi^{-1} \\ &= \frac{\sqrt{Q_\phi R_\phi}}{R_\phi} \\ &= \sqrt{\frac{Q_\phi}{R_\phi}}. \end{aligned}$$

We can calculate L_θ using similar reasoning. Therefore, the steady state Kalman filters are given by

$$\begin{aligned} \dot{\hat{\phi}} &= p + \sqrt{\frac{Q_\phi}{R_\phi}} \left(\hat{\phi}_{\text{accel}} - \hat{\phi} \right) \\ \dot{\hat{\theta}} &= q + \sqrt{\frac{Q_\theta}{R_\theta}} \left(\hat{\theta}_{\text{accel}} - \hat{\theta} \right). \end{aligned}$$

For the base maneuver discussed in Section 9.1, the estimation error of the roll and pitch angles using a fixed-gain Kalman filter is shown in Figure 9.8. Comparing Figure 9.8 with Figures 9.5 and 9.6 shows that the fixed gain Kalman filter removes the bias seen in Figure 9.6, while doing a much better job during accelerated flight than in Figure 9.5. However, there is still significant estimation error, which we hope to correct by accounting for the nonlinear dynamics using the continuous-discrete Kalman filter.

9.6.2 Continuous-Discrete Kalman Filter

To apply the continous-discrete extended Kalman filter derived in Section 9.5.2 to roll and pitch estimation, we use the nonlinear propagation model

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

where we have added the noise terms ξ_ϕ and ξ_θ to model the noise on p, q, and r. Notice that the Gaussian noise assumption, i.e., $\xi_\phi \sim \mathcal{N}(0, Q_\phi)$ and $\xi_\theta \sim \mathcal{N}(0, Q_\theta)$, are much easier to justify in this case.

We would like to use the accelerometers as the output equations. From Equation (8.1) we have the accelerometer model

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

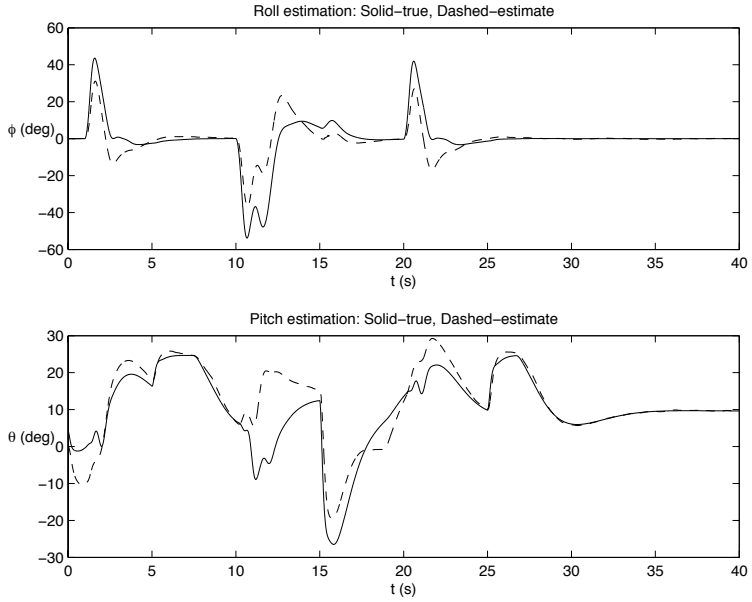


Figure 9.8: Estimation error on the roll and pitch angles using a fixed-gain Kalman filter.

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

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

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

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

Therefore, plugging into Equation (**??**) we get

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

Therefore letting $x = (\phi, \theta)^T$, $u = (p, q, r, V_a)^T$, $\xi = (\xi_\phi, \xi_\theta)^T$, and $\eta = (\eta_\phi, \eta_\theta)^T$, we get

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

where

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

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

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

The extended Kalman filter is implemented using Algorithm 6.

For the base maneuver discussed in Section 9.1, the estimation error of the roll and pitch angles using Algorithm 6 is shown in Figure 9.9. Comparing Figure 9.9 with Figures 9.5, 9.6, and 9.8 shows that the continous-discrete extended Kalman filter produces much better results than the other estimation schemes during accelerated flight.

9.7 GPS Smooting

In this section we will use GPS measurements to estimate the position and heading of the MAV, as well as the wind vector. If we assume that the flight path angle $\gamma = 0$, then from Equation (7.9) we have

$$\dot{p}_n = V_a\cos\psi + w_n$$
$$\dot{p}_e = V_a\sin\psi + w_e.$$

From Equation (5.9) the evolution of ψ is given by

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

Therefore, if we assume that the wind vector $(w_n, w_e)^T$ is constant, then the nonlinear propagation model position, heading, and wind is given by $\dot{x} = f(x,u)$, where $x = (p_n, p_e, \psi, w_n, w_e)^T$,

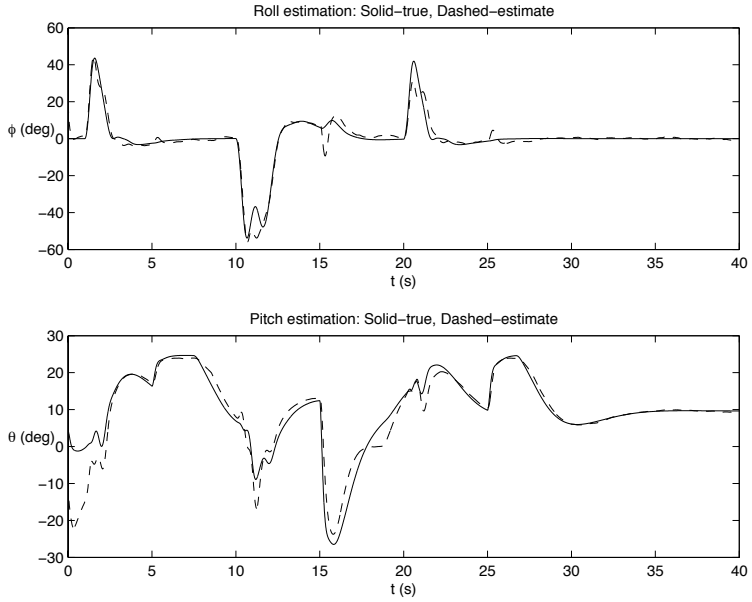


Figure 9.9: Estimation error on the roll and pitch angles using continuous-discrete extended Kalman filter.

$u = (V_a, q, r, \phi, \theta)^T$, and

$$f(x,u) \triangleq \begin{pmatrix} V_a \cos\psi + w_n \\ V_a \sin\psi + w_e \\ q\frac{\sin\phi}{\cos\theta} + r\frac{\cos\phi}{\cos\theta} \\ 0 \\ 0 \end{pmatrix},$$

and where the Jacobian of f is given by

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

We will use the GPS signals for North and East position, groundspeed, and course for measurement updates. From Equation (7.8) we can solve for the ground speed V_g and course angle χ

as

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

Therefore, the measurement model is given by

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

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

$$h(\hat{x},u) = \begin{pmatrix} \hat{p}_n \\ \hat{p}_e \\ \hat{V}_g \\ \tan^{-1}\left(\frac{\hat{V}_a \sin\hat{\psi}+\hat{w}_e}{\hat{V}_a \cos\hat{\psi}+\hat{w}_n}\right) \end{pmatrix},$$

where

$$\hat{V}_g = \sqrt{(\hat{V}_a \cos\hat{\psi} + \hat{w}_n)^2 + (\hat{V}_a \sin\hat{\psi} + \hat{w}_e)^2},$$

and where the Jacobian is given by

$$\frac{\partial h}{\partial x}(\hat{x},u) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{\hat{V}_a(-\hat{w}_n \sin\hat{\psi}+\hat{w}_e\cos\hat{\psi})}{\hat{V}_g} & \frac{\hat{V}_a\cos\hat{\psi}+\hat{w}_n}{\hat{V}_g} & \frac{\hat{V}_a\sin\hat{\psi}+\hat{w}_e}{\hat{V}_g} \\ 0 & 0 & \frac{\hat{V}_a^2+\hat{V}_a(\hat{w}_n\cos\hat{\psi}+\hat{w}_e\sin\hat{\psi})}{\hat{V}_g^2} & \frac{-(\hat{V}_a\sin\hat{\psi}+\hat{w}_e)}{\hat{V}_g^2} & \frac{(\hat{V}_a\cos\hat{\psi}+\hat{w}_n)}{\hat{V}_g^2} \end{pmatrix}.$$

The extended Kalman filter to estimate p_n, p_e, ψ, w_n, and w_e is implemented using Algorithm 6.

For the base maneuver discussed in Section 9.1, the estimation error for the position and heading using Algorithm 6 is shown in Figure 9.10. Comparing Figure 9.10 with Figures 9.2, shows that the continous-discrete extended Kalman filter produces much better results than a simple low pass filter.

9.8 Chapter Summary

RWB: To do.

Notes and References

RWB: To do.

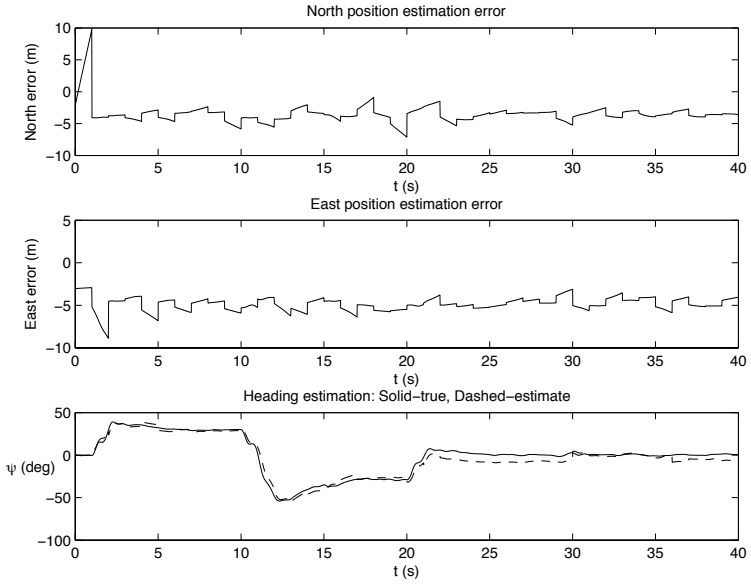


Figure 9.10: Estimation error on position and heading using continuous-discrete extended Kalman filter.

9.9 Design Project

9.1 Download the file `mavsim_chap9.zip` from web. Note that We have modified the `draw aircraft` block to plot the wind verses wind estimate. Also, the dimension of the output on the autopilot has changed to include estimates of the wind. Add your autopilot code to the skeleton function `autopilot.m` which has been modified to include a function call to `estimate_states.m`, which contains some sample code for a low pass filter, and a continuous-discrete Kalman filter.

9.2 Implement the simple schemes described in Section 9.3 using low pass filters and model inversion to estimate the states p_n, p_e, h, V_a, ϕ, θ, ψ, p, q, and r. Tune the bandwidth of the low pass filter to observe the effect.

9.3 Modify `estimate_states.m` to estimate roll and pitch angles using the fixed gain Kalman filter described in Section 9.6.1. Tune the filter until you are satisfied with the performance.

9.4 Modify `estimate_states.m` to implement the extended Kalman filter for roll and pitch angles described in Section 9.6.2. Tune the filter until you are satisfied with the performance.

9.5 Modify `estimate_states.m` to implement the extended Kalman filter for position, heading, and wind described in Section 9.7. Tune the filter until you are satisfied with the performance.

9.6 Modify `autopilot.m` so that all of the feedback loops are using estimated states instead actual states. 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 10

Straight-Line and Orbit Following

The objective of this chapter is to develop 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 the constant winds, which are almost always present. For small and micro air vehicles, 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 airspeed). The approach of tracking a moving point can result in significant problems for MAVs if disturbances, such as those due to wind, are not accounted for properly. If the MAV is flying into a strong wind (relative to its commanded groundspeed), the progression of the trajectory point must be slowed accordingly. Similarly, if the MAV is flying down wind, the speed of the tracking point must be increased to keep it from overrunning the desired position. Given that wind disturbances vary and are often not easily predicted, trajectory tracking can be challenging in anything other than calm conditions.

Rather than using a trajectory tracking approach, this chapter focuses on path following where the objective is to be *on the path* rather than at a certain point at a particular time. With path following, the time dependence of the problem is removed. For this chapter we will assume that

the controlled MAV is modeled by the guidance model given in Equation (7.20). 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 aerodynamically efficient. Therefore, to conserve fuel, it is desirable that the MAV maintain a constant airspeed. Accordingly, in this chapter we will assume 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

$$\mathcal{P}_{\text{line}}(\mathbf{r}, \vec{\mathbf{q}}) = \left\{ \mathbf{x} \in \mathbb{R}^3 : \mathbf{x} = \mathbf{r} + \alpha \vec{\mathbf{q}}, \alpha \in \mathbb{R} \right\},$$

where $\mathbf{r} \in \mathbb{R}^3$ is the origin of the path, and $\vec{\mathbf{q}} \in \mathbb{R}^3$ is a unit vector whose direction indicates the desired direction of travel. Figure 10.1 shows a top down or lateral view of $\mathcal{P}_{\text{line}}(\mathbf{r}, \vec{\mathbf{q}})$, and Figure 10.2 shows a side or longitudinal view. The course angle of $\mathcal{P}_{\text{line}}(\mathbf{r}, \vec{\mathbf{q}})$, as measured from North is given by

$$\chi_q \triangleq \tan^{-1} \frac{q_e}{q_n}. \tag{10.1}$$

The path following problem is most easily solved in a frame relative to the straight-line path. Let

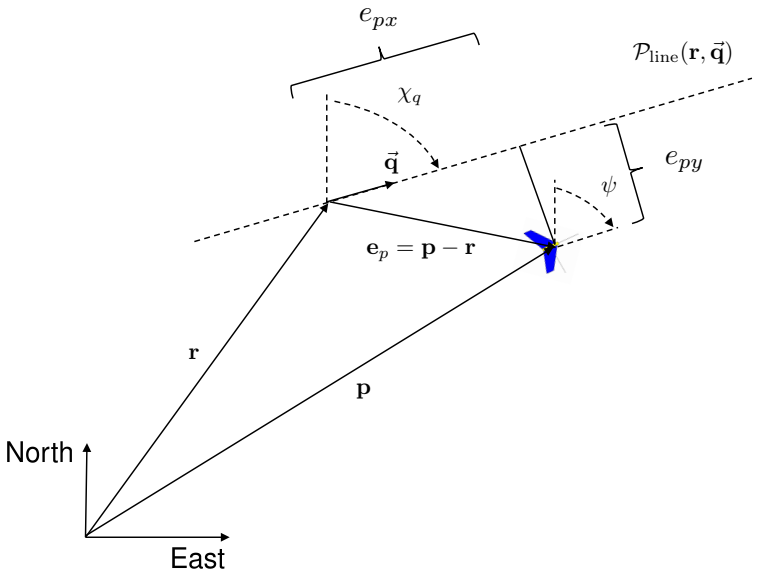


Figure 10.1: This figure shows the configuration of the MAV indicated by $(\mathbf{p}, \chi)$, and the configuration of the MAV relative to $\mathcal{P}_{\text{line}}$ indicated by $(\tilde{\mathbf{p}}, \tilde{\chi})$.

$\mathbf{r}$ be the center of the path-frame, with the x-axis aligned with the projection of $\vec{\mathbf{q}}$ onto the local

North-East plane, and the z-axis aligned with the inertial z-axis, and the y-axis selected to create a right-handed coordinate system. Let

$$R_i^{\mathcal{P}} \triangleq \begin{pmatrix} \cos\chi_{\vec{\mathbf{q}}} & \sin\chi_{\vec{\mathbf{q}}} & 0 \\ -\sin\chi_{\vec{\mathbf{q}}} & \cos\chi_{\vec{\mathbf{q}}} & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

be the transformation from the inertial frame to the path frame, and let

$$\mathbf{e}_p = \begin{pmatrix} e_{px} \\ e_{py} \\ e_{pz} \end{pmatrix} \triangleq R_i^{\mathbf{P}} \left(\mathbf{p}^i - \mathbf{r}^i \right)$$

be the relative path error expressed in the path frame, and let the wind vector in the North-East plane be expressed as

$$\begin{pmatrix} w_n \\ w_e \end{pmatrix} = V_w \begin{pmatrix} \cos\chi_w \\ \sin\chi_w \end{pmatrix}.$$

Then, the relative dynamics in the North-East inertial plane, expressed in the path frame, are given by

$$\begin{aligned} \begin{pmatrix} \dot{e}_{px} \\ \dot{e}_{py} \end{pmatrix} &= \begin{pmatrix} \cos\chi_q & \sin\chi_q \\ -\sin\chi_q & \cos\chi_q \end{pmatrix} \begin{pmatrix} V_a\cos\psi + V_w\cos\chi_w \\ V_a\sin\psi + V_w\sin\chi_q \end{pmatrix} \\ &= \begin{pmatrix} V_a\cos(\psi-\chi_q) + V_w\cos(\chi_w-\chi_q) \\ V_a\sin(\psi-\chi_q) + V_w\sin(\chi_w-\chi_q) \end{pmatrix}. \end{aligned} \tag{10.2}$$

For path following we desire to regulate the cross track error e_{py} to zero by commanding the heading angle. The relevant dynamics are therefore given by

$$\dot{e}_{py} = V_a\sin(\psi-\chi_q) + V_w\sin(\chi_w-\chi_q) \tag{10.3}$$

$$\ddot{\psi} = -b_{\dot{\psi}}\dot{\psi} + b_{\psi}(\psi^c - \psi). \tag{10.4}$$

The lateral straight-line path following problem is to select ψ^c so that $e_{py} \to 0$ when χ_q is known but V_w and χ_w are unknown constants.

The geometry in the longitudinal direction is shown in Figure 10.2. Referring to Figure 10.2 and using similar triangles, we have the relationship

$$\frac{-e_{pd}}{\sqrt{e_{pn}^2 + e_{pe}^2}} = \frac{-q_d}{\sqrt{q_n^2 + q_e^2}},$$

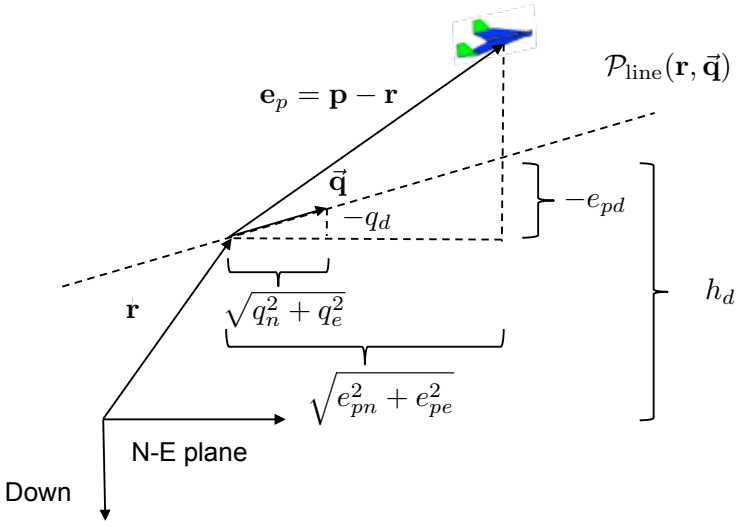


Figure 10.2: The flight path angle of the waypoint path $\gamma_{\vec{\mathbf{q}}}$.

where

$$\mathbf{e}_p^i = \begin{pmatrix} e_{pn} \\ e_{pe} \\ e_{pd} \end{pmatrix} \stackrel{\triangle}{=} \mathbf{p}^i - \mathbf{r}^i = \begin{pmatrix} p_n - r_n \\ p_e - r_e \\ p_d - r_d \end{pmatrix}.$$

Therefore the desired altitude is given by

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

Since the altitude dynamic are given by

$$\ddot{h} = -b_{\dot{h}} \dot{h} + b_h (h^c - h), \tag{10.6}$$

the lateral straight-line path following problem is to select h^c so that $h \to h^d(\mathbf{r}, \mathbf{p}, \vec{\mathbf{q}})$.

10.1.1 Longitudinal Guidance Strategy for Straight-Line Following

In this section we develop a longitudinal guidance law for tracking the altitude portion of the waypoint path. Defining the altitude error as

$$e_h \stackrel{\triangle}{=} h - h_d(\mathbf{r}, \mathbf{p}, \vec{\mathbf{q}}),$$

we get that

$$\begin{aligned}\dot{e}_h &= \dot{h} - \dot{h}_d \\ \ddot{e}_h &= \ddot{h} - \ddot{h}_d \\ &= -b_{\dot{h}}\dot{h} + b_h(h^c - h) - \ddot{h}_d,\end{aligned}$$

where $\dot{h}_d$ and $\ddot{h}_d$ depend on the wind and are not precisely known. If the commanded altitude were

$$h^c = h_d + \frac{b_{\dot{h}}}{b_h}\dot{h}_d + \frac{1}{b_h}\ddot{h}_d, \tag{10.7}$$

then the altitude error would be given by $\ddot{e}_h + b_{\dot{h}}\dot{e}_h + b_h e_h = 0$ which would ensure that $\dot{e}_h \to 0$. We could attempt to implement Equation (10.7) by numerically differentiating h_d. However, since numerical differentiation introduces errors, and since Equation (10.6) is only an approximation of the altitude dynamics, a more robust approach that works well in practice is to add an integrator to obtain

$$h^c(t) = h_d(t) + k_{hi}\int_{-\infty}^{t}(h_d(\tau) - h(\tau))d\tau. \tag{10.8}$$

Therefore, the error dynamics approximately satisfy

$$\dddot{e}_h + b_{\dot{h}}\ddot{e}_h + b_h\dot{e}_h + b_h k_{hi} e_h = 0.$$

Therefore, the integral gain needs to be selected so that the roots of the polynomial

$$s^3 + b_{\dot{h}}s^2 + b_h s + b_h k_{hi} = 0$$

are in the open left hand plane.

10.1.2 Lateral Guidance Strategy for Straight-Line Following

The objective in this section is to select the commanded heading angle ψ^c in Equation (10.4) so that e_{py} in Equation (10.3) is driven to zero asymptotically. The strategy in this section will be to construct a desired heading angle at every spatial point relative to the straight-line path that results in the MAV moving toward the path. The set of desired heading angles at every point will be called a vector field because the desired heading angle specifies a vector (relative to the straight line) with a magnitude of unity. Figure 10.3 depicts a potential vector field for straight-line path following. The objective is to 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 heading also approaches zero. Toward that end, define the desired heading of the MAV as

$$\psi_d(e_{py}) = -\chi^\infty \frac{2}{\pi}\tan^{-1}(k_{\text{path}} e_{py}), \tag{10.9}$$

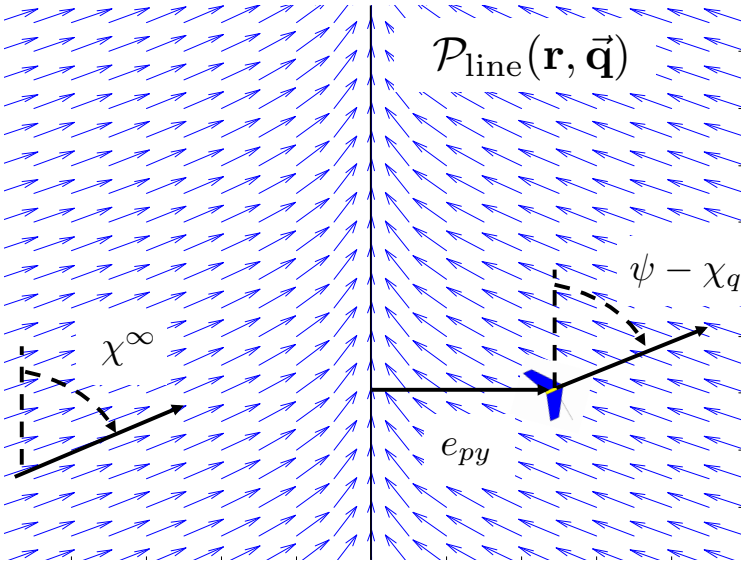


Figure 10.3: Vector field for straight-line path following. Far away from the waypoint path, the vector field is directed with an angle χ^∞ from the perpendicular to the path.

where k_{path} is a positive constant that influences the rate of the transition from χ^∞ to zero. Figure 10.4 shows how the choice of k_{path} affects the rate of transition. Large values of k_{path} result in short, abrupt transitions, while small values of k_{path} cause long, smooth transitions in the desired course.

If χ^∞ is restricted to be in the range $\chi^\infty \in (0, \frac{\pi}{2}]$ then clearly

$$-\frac{\pi}{2} < \chi^\infty \frac{2}{\pi} \tan^{-1}(k_{\text{path}} e_{py}) < \frac{\pi}{2}$$

for all values of e_{py}. Therefore since $\tan^{-1}(\cdot)$ is an odd function and $\sin(\cdot)$ is odd over $(-\frac{\pi}{2}, \frac{\pi}{2})$ we can use the Lyapunov function $W(e_{py}) = \frac{1}{2} e_{py}^2$ to argue that if $\psi = \chi_q + \psi^d(e_{py})$, and if the wind is zero, then $e_{py} \to 0$ asymptotically since

$$\dot{W} = -V_a e_{py} \sin\left(\chi^\infty \frac{2}{\pi} \tan^{-1}(k_{\text{path}} e_{py})\right)$$

is less than zero for $e_{py} \neq 0$.

When there is a constant wind, the MAV must crab into the wind to maintain the desired course angle along the straight-line path. Since wind is not precisely known, we use integral control to automatically seek for the desired crab angle. The command for lateral path following is therefore given by

$$\psi^c(t) = \chi_q - \chi^\infty \frac{2}{\pi} \tan^{-1}(k_{\text{path}} e_{py}(t)) - k_{\psi i} \int_{-\infty}^{t} e_{py}(\tau) d\tau \tag{10.10}$$

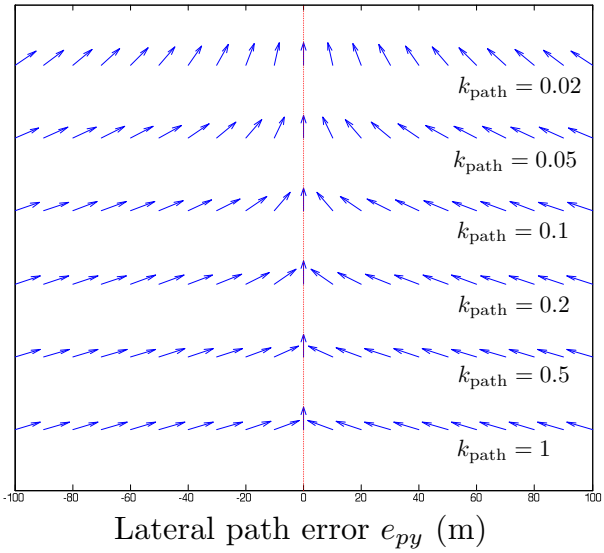


Figure 10.4: Vector fields for various values of k_{path}. Large values of k_{path} yield abrupt transitions from χ^∞ to zero, while small values of k_{path} give smooth transitions.

Before moving to orbit following, we note that using Equation (10.10) may result in undesirable behavior if χ_q is computed directly from Equation (10.1) where $\tan^{-1}$ returns an angle between $\pm\pi$. As an example, consider the scenario shown in Figure 10.5, where χ_q is a positive number slightly smaller than $+\pi$. Since the current heading is negative, Equation (10.10) will cause the MAV to turn right to align with the waypoint path. As an alternative, if χ_q is expressed as a negative angle slightly less than $-\pi$, then the MAV will turn left to align with the waypoint path. To alleviate this problem, χ_q should be computed as

$$\chi_q = \text{atan2}(q_e, q_n) + 2\pi m,$$

where $m \in \mathcal{N}$ is selected so that $-\pi \leq \chi_q - \psi \leq \pi$, and atan2 is a four-quadrant $\tan^{-1}$ function.

10.2 Orbit Following

An orbit path is described by a center $\mathbf{c} \in \mathbb{R}^3$, a radius $\rho \in \mathbb{R}$, and a direction $\lambda \in \{-1, 1\}$, as

$$\mathcal{P}_{\text{orbit}}(\mathbf{c}, \rho, \lambda) = \left\{ \mathbf{r} \in \mathbb{R}^3 : \mathbf{r} = \mathbf{c} + \lambda\rho \begin{pmatrix} \cos\varphi, & \sin\varphi & 0 \end{pmatrix}^T, \varphi \in [0, 2\pi) \right\},$$

where $\lambda = 1$ signifies a clockwise orbit and $\lambda = -1$ signifies a counterclockwise orbit. We assume that the center of the orbit is expressed in inertial coordinates so that $\mathbf{c} = (c_n, c_e, c_d)^T$,

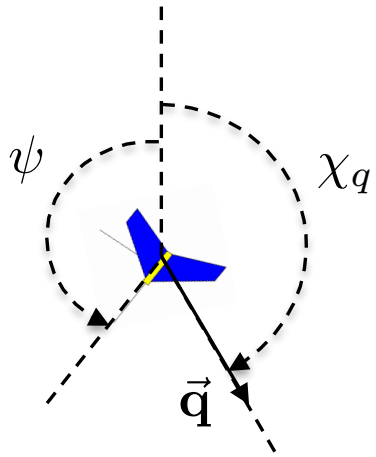


Figure 10.5: The calculation of χ_q needs to account for the current heading angle of the MAV. In this scenario, the MAV should turn left to align with the waypoint path, but if χ_q is computed with `atan2` the angle will be a positive number slightly smaller than $+\pi$, which will cause the MAV to turn right to align with the waypoint path.

where $-c_d$ represents the desired altitude of the orbit. In this chapter we assume that orbits are at constant altitude and use the longitudinal guidance scheme described in Section 10.1.1 to maintain the altitude in wind.

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

$$\varphi \triangleq \tan^{-1}\left(\frac{p_e - c_e}{p_n - c_n}\right) \tag{10.11}$$

be the phase angle of the relative position, as shown in Figure 10.6.

Differentiating d and using Equations (10.2) gives

$$\begin{aligned}\dot{d} &= \frac{(p_n - c_n)\dot{p}_n + (p_e - c_e)\dot{p}_e}{d} \\ &= \frac{(p_n - c_n)V_a\cos(\psi - \chi_q) + (p_e - c_e)V_a\sin(\psi - \chi_q)}{d} + \frac{(p_n - c_n)V_w\cos(\chi_w - \chi_q) + (p_e - c_e)V_a\sin(\chi_w - \chi}{d}\end{aligned}$$

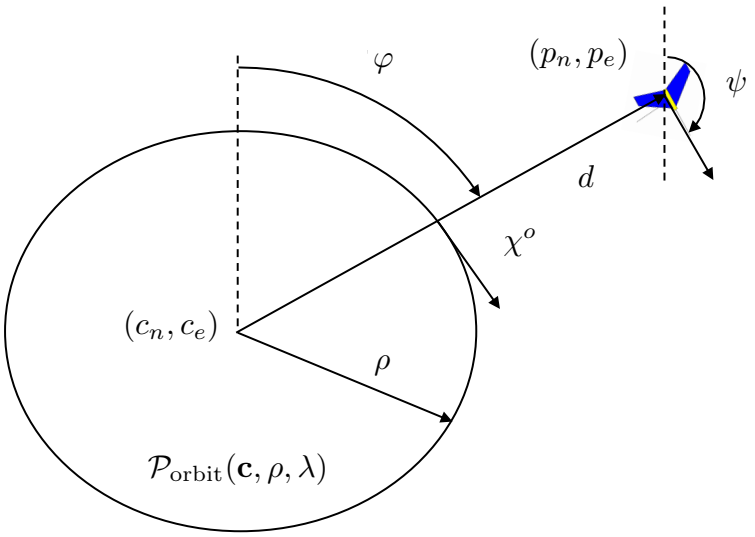


Figure 10.6: Orbital path with center (c_n, c_e), and radius ρ. The distance from the orbit center to the MAV is d, and the angular position of the MAV relative to the orbit is φ.

Using Equation (10.11) gives

$$\begin{aligned}
\dot{d} &= V_a \frac{(p_n - c_n)\cos(\psi - \chi_q) + (p_e - c_e)\sin(\psi - \chi_q)}{d} + V_w \frac{(p_n - c_n)\cos(\chi_w - \chi_q) + (p_e - c_e)\sin(\chi_w - \chi_q)}{d} \\
&= V_a \left(\frac{p_n - c_n}{d}\right)(\cos(\psi - \chi_q) + \sin(\psi - \chi_q)\tan\varphi) + V_w \left(\frac{p_n - c_n}{d}\right)(\cos(\chi_w - \chi_q) + \sin(\chi_w - \chi_q)\tan\varphi) \\
&= V_a \cos\varphi\,(\cos(\psi - \chi_q) + \sin(\psi - \chi_q)\tan\varphi) + V_w \cos\varphi\,(\cos(\chi_w - \chi_q) + \sin(\chi_w - \chi_q)\tan\varphi) \\
&= V_a\,(\cos(\psi - \chi_q)\cos\varphi + \sin(\psi - \chi_q)\sin\varphi) + V_w\,(\cos(\chi_w - \chi_q)\cos\varphi + \sin(\chi_w - \chi_q)\sin\varphi) \\
&= V_a \cos(\psi - \chi_q - \varphi) + V_w \cos(\chi_w - \chi_q - \varphi).
\end{aligned}$$

The orbital kinematics in polar coordinates are therefore given by

$$\dot{d} = V_a \cos(\psi - \chi_q - \varphi) + V_w \cos(\chi_w - \chi_q - \varphi) \tag{10.12}$$

$$\ddot{\psi} = -b_{\dot{\psi}}\dot{\psi} + b_{\psi}(\psi^c - \psi) \tag{10.13}$$

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 ρ and to drive the course angle $\chi(t)$ to χ^o in the presence of wind.

Our approach to orbit following is similar to the ideas developed in Section 10.1.2. The strategy is to construct a desired heading 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 $d >> \rho$ the desired heading is

$$\psi_d \approx \chi^o + \lambda \frac{\pi}{2},$$

and when $\tilde{d} = 0$ the desired heading is $\psi_d = \chi^o$. Therefore, a candidate heading field is given by

$$\psi_d(d - \rho, \lambda) = \chi^o + \lambda \tan^{-1}\left(k_{\text{orbit}}(d - \rho)\right), \tag{10.14}$$

where $k_{\text{orbit}} > 0$ is a constant that specifies the rate of transition from $\lambda\pi/2$ to zero. This expression for ψ_d is valid for all values of $d \geq 0$.

In zero wind conditions, we can again use the Lyapunov function $W = \frac{1}{2}(d - \rho)^2$ to argue that if $\psi = \psi_d$, then the tracking objective is satified. Differentiating W along the system trajectory gives

$$\dot{W} = -V_a(d - \rho) \sin\left(\tan^{-1}(k_{\text{orbit}}(d - \rho))\right),$$

which is negative definite since the argument of $\sin$ is in the set $(-\pi/2, \pi/2)$ for all $d > 0$, implying that $d \to \rho$ asymptotically. To account for wind and the fact that Equation (10.13) is only an approximation of the heading dynamics, we use integral control. The command for lateral orbit following is therefore given by

$$\psi^c(t) = \varphi + \lambda \left[\frac{\pi}{2} + \tan^{-1}\left(k_{\text{orbit}}(d - \rho)\right) + k_{\psi i} \int_{-\infty}^{t} (d(\tau) - \rho) d\tau\right]. \tag{10.15}$$

Similar to the computation of the path angle χ_q, if the angular position in the orbit φ is computed to be between $\pm\pi$, then there will be a sudden jump of 2π in the commanded heading as the MAV transitions from $\varphi = \pi$ to $\varphi = -\pi$. To alleviate this problem, φ should be computed as

$$\varphi = \text{atan2}(p_e - c_e, p_n - c_n) + 2\pi m,$$

where $m \in \mathcal{N}$ is selected so that $-\pi \leq \varphi - \psi \leq \pi$.

10.3 Chapter Summary

This chapter has 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 than trajectory tracking where the vehicle would be commanded to follow a time varying location.

The algorithms developed in this chapter are summarized in Algorithms 7 and 8.

Algorithm 7 Straight-Line Following: $[h^c, \psi^c] = \textsf{followStraightLine}(\mathbf{r}, \vec{\mathbf{q}}, \mathbf{p}, \psi)$

Input: Path definition $\mathbf{r} = (r_n, r_e, r_d)^T$, and $\vec{\mathbf{q}} = (q_n, q_e, q_d)^T$. MAV position $\mathbf{p} = (p_n, p_e, p_d)^T$, heading ψ, gains χ_∞, k_{path}, k_{hi}, $k_{\psi i}$, sample rate T_s.

Define and initialize persistent variables I_h and $I_{e_{py}}$ (integrals of altitude and path error)
$h^d \leftarrow -r_d - \sqrt{(p_n - r_n)^2 + (p_e - r_e)^2}\frac{q_d}{q_n^2+q_e^2}$
$I_h \leftarrow I_h + T_s(h^d + p_d)$
$h^c \leftarrow h^d + k_{hi} I_h$
$\chi_q \leftarrow \text{atan2}(q_e, q_n)$.
while $\chi_q - \psi < -\pi$ **do**
    $\chi_q \leftarrow \chi_q + 2\pi$
end while
while $\chi_q - \psi > \pi$ **do**
    $\chi_q \leftarrow \chi_q - 2\pi$
end while
$e_{py} \leftarrow -\sin\chi_q(p_n - r_n) + \cos\chi_q(p_e - r_e)$
$I_{e_py} \leftarrow I_{e_py} + T_s e_{py}$
$\psi^c \leftarrow \chi_q - \chi^\infty \frac{2}{\pi} tan^{-1}(k_{\text{path}} e_{py}) - k_{\psi i} I_{e_py}$
return h^c, ψ^c.

Algorithm 8 Circular Orbit Following: $[h^c, \psi^c] = \textsf{followOrbit}(\mathbf{c}, \rho, \lambda, \mathbf{p}, \psi)$

Input: Orbit center $\mathbf{c} = (c_n, c_e, c_d)^T$, radius ρ, and direction λ, MAV position $\mathbf{p} = (p_n, p_e, p_d)^T$, heading ψ, gains k_{orbit}, $k_{\psi i}$, k_{hi}, sample rate T_s

Define and initialize persistent variables I_h and I_d (integrals of altitude and orbit error)
$I_h \leftarrow I_h + T_s(-c_d + p_d)$
$h^c \leftarrow -c_d + k_{hi} I_h$
$d \leftarrow \sqrt{(p_n - c_n)^2 + (p_e - c_e)^2}$.
$\varphi \leftarrow \text{atan2}(p_e - c_e, p_n - c_n)$
while $\varphi - \psi < -\pi$ **do**
    $\varphi \leftarrow \varphi + 2\pi$
end while
while $\varphi - \psi > \pi$ **do**
    $\varphi \leftarrow \varphi - 2\pi$
end while
$\psi^c \leftarrow \varphi + \lambda\left(\frac{\pi}{2} + \tan^{-1}(k_{\text{orbit}}(d - \rho)) + k_{\psi i} I_d\right)$
return h^c, ψ^c.

Notes and References

RWB: Need to revise to reflect new material.

The method described in Sections **??** and **??** is a variation of the one described in [28, 29], and is based on the notion of a vector field, which calculates a desired heading based on the distance from the path. 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., [30]). It has also been suggested in [31] that potential fields can be used in UAV navigation for obstacle and collision avoidance applications. The method of [31] 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 [32]. 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 [33] 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.

The path following approach is studied in [34, 35], where performance limits for reference-tracking and path-following controllers are investigated and the difference between them is highlighted. It is shown that there is not a fundamental performance limitation for path following for systems with unstable zero dynamics as there is for reference tracking.

Building on the work presented in [36] on maneuver modified trajectory tracking, [37] develops an approach that combines the features of trajectory tracking and path following for marine vehicles. Similar to this work is that of Skjetne, et al. [38] which develops an output maneuvering method composed of two tasks: forcing the output to converge to the desired path and then satisfying a desired speed assignment along the path. The method is demonstrated using a marine vessel simulation. Ref. [39] presents a path following method for UAVs that provides a constant line of sight between the UAV and an observation target.

RWB: Need to reference Eric Frew's stuff on Lyapunov Vector Fields.

10.4 Design Project

The objective of this assignment is to implement Algorithms 7 and 8. Download the sample code for this chapter from the book website and note the addition of two blocks labeled `PathManager` and `PathFollower`. The output of the path manager is

$$y_{\text{manager}} = \begin{pmatrix} \texttt{flag} \\ V_a^d \\ \mathbf{r} \\ \vec{\mathbf{q}} \\ \mathbf{c} \\ \rho \\ \lambda \end{pmatrix},$$

where `flag`=1 indicates that $\mathcal{P}_{\text{line}}(\mathbf{r}, \vec{\mathbf{q}})$ should be followed, and `flag`=2 indicates that $\mathcal{P}_{\text{orbit}}(\mathbf{c}, \rho, \lambda)$ should be followed, and where V_a^d is the desired airspeed.

10.1 Modify `path_follow.m` to implement Algorithms 7 and 8. By modifying `path_manager.m` test both the straight-line and orbit following algorithms on the guidance model given in Equation (7.20). An example Simulink diagram is given in `mavsim_chap10_model.mdl`. Test your design with significant constant winds (e.g., $w_n = 3$, $w_e = -3$). Tune the gains to get acceptable performance.

10.2 Implement the path following algorithms on the full 6 DOF simulation of the MAV. An example Simulink diagram is given in `mavsim_chap10.mdl`. Test your design with significant constant winds (e.g., $w_n = 3$, $w_e = -3$). If necessary, tune the gains to get acceptable performance.

Chapter 11

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 Dubin's 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

In this section we define a waypoint path as an ordered sequence of waypoints

$$\mathcal{W} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N\}, \tag{11.1}$$

where $\mathbf{w}_i = (w_{n,i}, w_{e,i}, w_{d,i})^T \in \mathbb{R}^3$.

In this section we address the problem of switching from one waypoint segment to another. Consider the scenario shown in Figure 11.1 where a MAV is currently tracking the straight-line segment $\overline{\mathbf{w}_{i-1}\mathbf{w}_i}$. Intuitively, when the MAV reaches $\mathbf{w}_i$, we desire to switch the guidance algorithm so that it will track the straight-line segment $\overline{\mathbf{w}_i\mathbf{w}_{i+1}}$. What is the best method for determining whether the MAV has reached $\mathbf{w}_i$? One possible strategy is to switch when the MAV enters a ball around $\mathbf{w}_i$. In other words, the guidance algorithm would switch at the first time instant when

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

where b is the size of the ball and $\mathbf{p}(t)$ is the location of the MAV. However, if there are disturbances like wind, or if b is 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 $\mathbf{w}_i$.

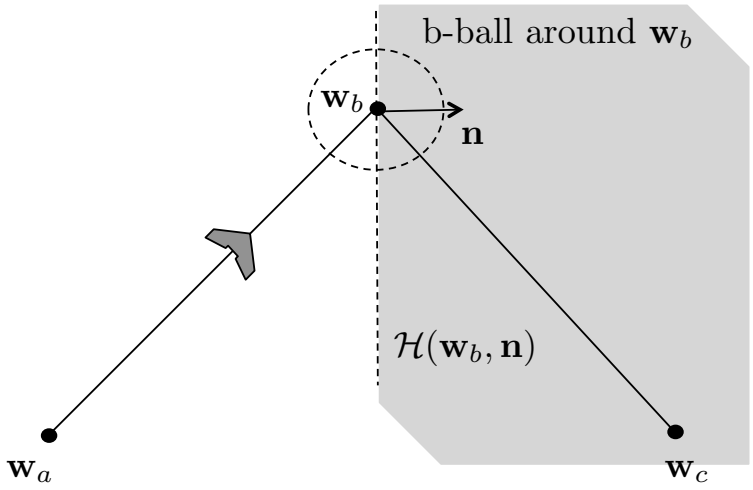


Figure 11.1: When transitioning from one straight-line segment to another, a criteria is needed to indicate when the MAV has completed the first straight-line segment. A possible option is to switch when the MAV enters a b-ball around the transition waypoint. A better option is to switch when the MAV enters the half-plane $\mathcal{H}(\mathbf{w}_i, \mathbf{n})$.

A better approach 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 $\mathbf{n} \in \mathbb{R}^3$, define the half plane

$$\mathcal{H}(\mathbf{r}, \mathbf{n}) \stackrel{\triangle}{=} \left\{ \mathbf{q} \in \mathbb{R}^3 : (\mathbf{q} - \mathbf{r})^T \mathbf{n} \geq 0 \right\}.$$

In reference to Figure 11.1, the normal to the 3D half plane that separates the straight-line $\overline{\mathbf{w}_{i-1}\mathbf{w}_i}$ from the straight-line $\overline{\mathbf{w}_i\mathbf{w}_{i+1}}$ is given by

$$\mathbf{n}_i \stackrel{\triangle}{=} \frac{\mathbf{w}_{i+1} - \mathbf{w}_{i-1}}{\|\mathbf{w}_{i+1} - \mathbf{w}_{i-1}\|}.$$

The MAV tracks the straight-line path from $\mathbf{w}_{i-1}$ to $\mathbf{w}_i$ until it enters $\mathcal{H}(\mathbf{w}_i, \mathbf{n}_i)$, at which point it will track the straight-line path from $\mathbf{w}_i$ to $\mathbf{w}_{i+1}$.

A simple algorithm for following the sequence of waypoints (11.1) is given in Algorithm 9. The first time that the algorithm is executed, the waypoint pointer is initialized to the first waypoint in Line 1. Line 2 labels the current waypoint, and the next two waypoints as 'a', 'b', and 'c'. Lines 3 and 4 define $\mathbf{r}$ and $\mathbf{q}$ for the current waypoint segment. Line 5 defines the unit vector along the next waypoint path, and Line 6 is a vector that is perpendicular to the half plane that separates $\overline{\mathbf{w}_a\mathbf{w}_b}$

from $\overline{\mathbf{w}_b\mathbf{w}_c}$. Line 7 checks to see if the half plane defining the next waypoint segment has been reached by the MAV. If it has, then Lines 8-9 will cycle to the next waypoint segment.

Algorithm 9 Follow Waypoints: $(\mathbf{r}, \mathbf{q}) = \textsf{followWpp}(\mathcal{W}, \mathbf{p})$

Input: Waypoint path $\mathcal{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$, MAV position $\mathbf{p} = (p_n, p_e, p_d)^T$.

Require: $N \geq 3$.

1: Initialize waypoint pointer $i \leftarrow 1$.
2: $a \leftarrow i, \quad b \leftarrow (i+1) \bmod N, \quad c \leftarrow (i+2) \bmod N.$
3: $\mathbf{r} \leftarrow \mathbf{w}_a$
4: $\mathbf{q} \leftarrow \frac{\mathbf{w}_b - \mathbf{w}_a}{\|\mathbf{w}_b - \mathbf{w}_a\|}$.
5: $\mathbf{q}_{bc} \leftarrow \frac{\mathbf{w}_c - \mathbf{w}_b}{\|\mathbf{w}_c - \mathbf{w}_b\|}$.
6: $\mathbf{n} \leftarrow \frac{\mathbf{q} + \mathbf{q}_{bc}}{2}$
7: **if** $\mathbf{p} \in \mathcal{H}(\mathbf{w}_b, \mathbf{n})$ **then**
8: $\quad i \leftarrow (i+1) \bmod N.$
9: $\quad a \leftarrow i, \quad b \leftarrow (i+1) \bmod N, \quad c \leftarrow (i+2) \bmod N.$
10: **end if**
11: **return** $\mathbf{r}$, $\mathbf{q}$ at each time step.

Algorithm 9 will produce paths like that shown in Figure 11.2. The advantage of Algorithm 9 is 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 do not provide a smooth or 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 around the transition is shown in Figure 11.4. Let

$$\mathbf{q}_{ab} \stackrel{\triangle}{=} \frac{\mathbf{w}_b - \mathbf{w}_a}{\|\mathbf{w}_b - \mathbf{w}_a\|}$$

be the unit vector aligned with the straight-line between waypoints $\mathbf{w}_a$ and $\mathbf{w}_b$, and let $\mathbf{q}_{bc}$ be the unit vector between $\mathbf{w}_b$ and $\mathbf{w}_c$, then the angle between $\overline{\mathbf{w}_a\mathbf{w}_b}$ and $\overline{\mathbf{w}_b\mathbf{w}_c}$ is given by

$$\beta \stackrel{\triangle}{=} \cos^{-1}\left(-\mathbf{q}_{ab}^T\mathbf{q}_{bc}\right). \tag{11.2}$$

If the radius of the fillet is R, as shown in Figure 11.4, then the distance between the waypoint $\mathbf{w}_b$ and the location where the the fillet intersects the line $\overline{\mathbf{w}_b\mathbf{w}_c}$ is $R/\tan\frac{\beta}{2}$, and the distance between

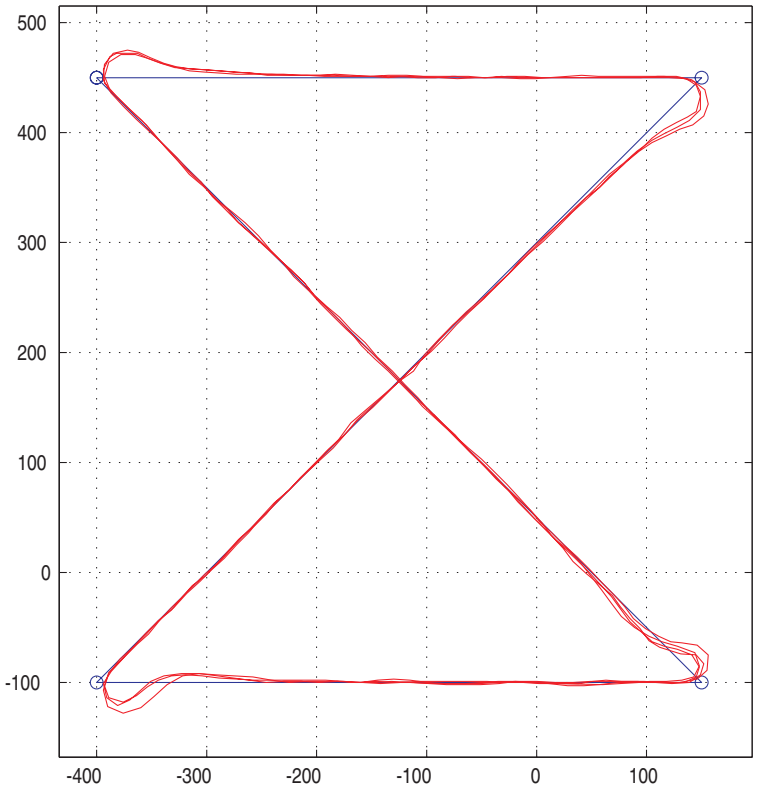


Figure 11.2: Path generated using the path following algorithm given Algorithm 9. The MAV follows the straight-line path until reaching the waypoint, and then maneuvers onto the next straight-line section.

$\mathbf{w}_b$ and the center of the fillet circle is $R/\sin\frac{\beta}{2}$. Therefore the distance between $\mathbf{w}_b$ and the edge of the fillet circle along the bisector of β is given by $R/\sin\frac{\beta}{2} - R$.

To implement the fillet maneuver using the path following algorithms described in Chapter 10, we will follow the straight-line segment $\overline{\mathbf{w}_a\mathbf{w}_b}$ 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 halfplane $\mathcal{H}_2$ shown in Figure 11.5 at which point the straight-line segment $\overline{\mathbf{w}_i\mathbf{w}_{i+1}}$ is followed.

As shown in Figure 11.5, the center of the fillet is given by

$$\mathbf{c} = \mathbf{w}_b - \left(\frac{R}{\sin\frac{\beta}{2}}\right)\frac{\mathbf{q}_{ab} - \mathbf{q}_{bc}}{\|\mathbf{q}_{ab} - \mathbf{q}_{bc}\|}.$$

Similarly, the halfplane $\mathcal{H}_1$ is defined by the location

$$\mathbf{r}_1 = \mathbf{w}_b - \left(\frac{R}{\tan\frac{\beta}{2}}\right)\mathbf{q}_{ab},$$

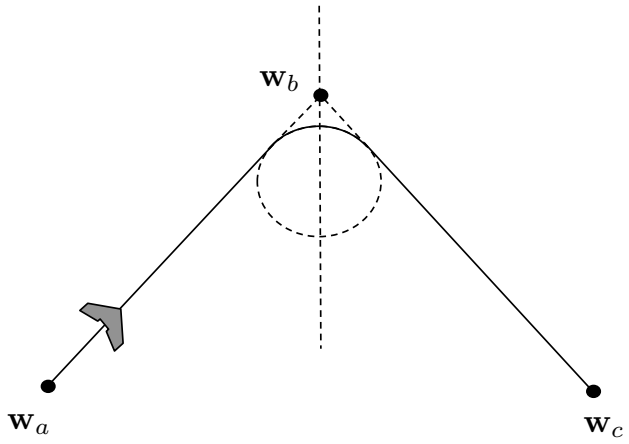


Figure 11.3: The transition from straight-line path $\overline{\mathbf{w}_{i-1}\mathbf{w}_i}$ to $\overline{\mathbf{w}_i\mathbf{w}_{i+1}}$ can be smoothed by inserting a filet.

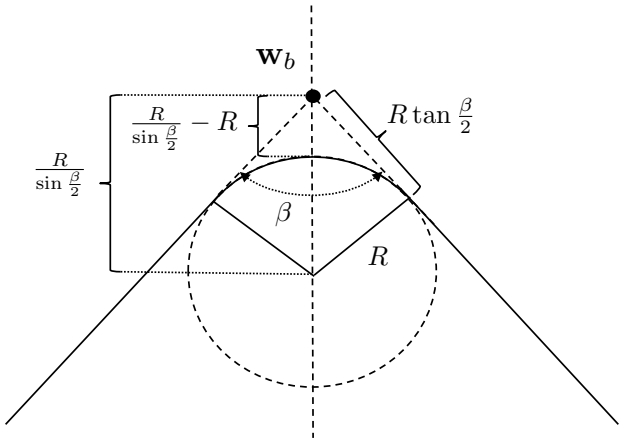


Figure 11.4: The geometry associated with inserting a fillet between waypoint segments.

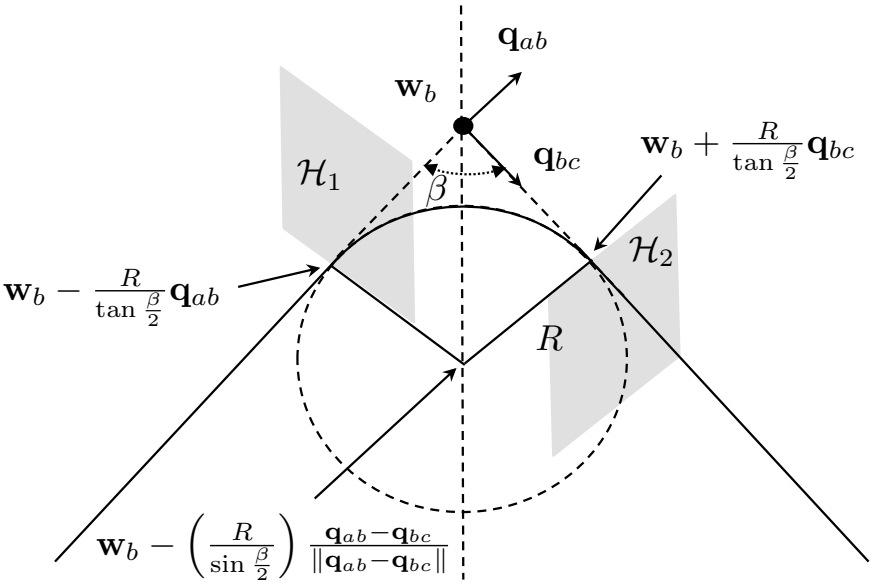


Figure 11.5: Definitions of the half-planes associated with following a fillet inserted between waypoint segments.

and the normal vector $\mathbf{q}_{ab}$. The halfplane $\mathcal{H}_2$ is defined by the location

$$\mathbf{r}_2 = \mathbf{w}_b + \left(\frac{R}{\tan\frac{\beta}{2}}\right)\mathbf{q}_{bc},$$

and the normal vector $\mathbf{q}_{bc}$.

The algorithm for maneuvering along the waypoint path $\mathcal{W}$ using fillets to smooth between the straight-line segments, is given by Algorithm 10. 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 is Line 2. The waypoints are labeled "a", "b", and "c" in line 4 and the unit vectors $\mathbf{q}_{ab}$ and $\mathbf{q}_{bc}$ and the angle β are computed in lines 5–7.

When the state machine is in state `state=1` the MAV is commanded to follow the straight-line path along $\mathbf{w}_a\mathbf{w}_b$, which is parameterized by $\mathbf{r} = \mathbf{w}_a$, and $\mathbf{q} = \mathbf{q}_{ab}$, which are assigned in Lines 9–11. Lines12–15 test to see if the MAV has transitioned into the half-plane shown as $\mathcal{H}_1$ in Figure 11.5. If the MAV has transitioned into $\mathcal{H}_2$, then the state machine is updated to `state=2`.

When the state machine is in `state=2` the MAV is commanded to follow the orbit that defines the fillet. The center, radius, and direction of the orbit are assigned in Lines 18–20. In Line 20, $q_{ab,n}$ and $q_{ab,e}$ denote the North and East components of $\mathbf{q}_{ab}$. Lines20–25 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 $\overline{\mathbf{w}_b\mathbf{w}_c}$. Algorithm 10 produces paths like that shown in Figure 11.6.

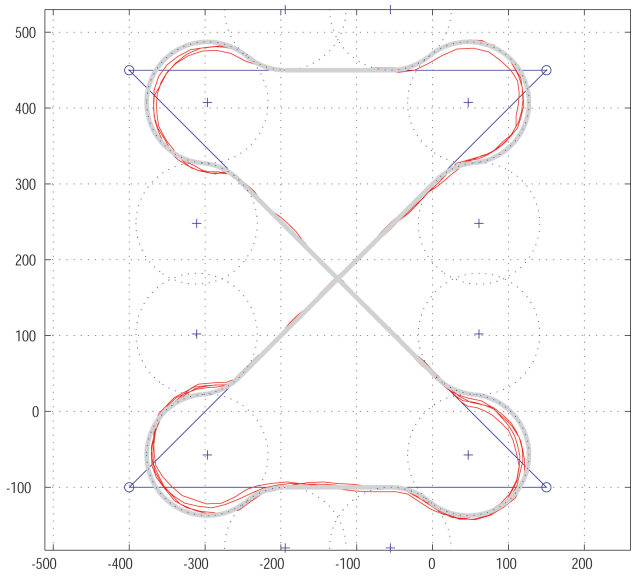


Figure 11.6: An example of the types of flight paths produced by Algorithm 10.RWB: Replace this figure with regenerated (and correct) flight paths.

One of the disadvantages of the fillet method as given in Algorithm 10 is that the path length is changed when fillets are inserted. For certain applications like the cooperative timing problems discussed in [40], 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 estimate of the path length of $\mathcal{W}$ after fillets have been inserted.

To be precise, let

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

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 10. From Figure 11.4 we see that the length of the fillet traversed by the corrected path is $R(\pi - \beta_i)$. In addition, it is clear that the length of the straight-line segment removed from $|\mathcal{W}|$ by traversing the fillet is $2R\tan\frac{\beta_i}{2}$. Therefore

$$|\mathcal{W}|_F = |\mathcal{W}| + \sum_{i=2}^{N} \left(R(\pi - \beta_i) - \frac{2R}{\tan\frac{\beta_i}{2}} \right), \tag{11.3}$$

where β_i is given in Eq. (11.2).

Algorithm 10 Follow Waypoints with Fillets: $(\text{flag}, \mathbf{r}, \mathbf{q}, \mathbf{c}, \rho, \lambda) = \text{followWppFillet}(\mathcal{W}, \mathbf{p}, R)$

Input: Waypoint path $\mathcal{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$, MAV position $\mathbf{p} = (p_n, p_e, p_d)^T$, fillet radius R.

Require: $N \geq 3$.

if New waypoint path $\mathcal{W}$ is received **then**
  Initialize waypoint pointer: $i \leftarrow 1$, and state machine: state $\leftarrow 1$.
end if
$a \leftarrow i, \quad b \leftarrow (i+1) \bmod N, \quad c \leftarrow (i+2) \bmod N.$
$\mathbf{q}_{ab} \leftarrow \frac{\mathbf{w}_b - \mathbf{w}_a}{\|\mathbf{w}_b - \mathbf{w}_a\|}$.
$\mathbf{q}_{bc} \leftarrow \frac{\mathbf{w}_c - \mathbf{w}_b}{\|\mathbf{w}_c - \mathbf{w}_b\|}$.
$\beta \leftarrow \cos^{-1}\left(-\mathbf{q}_{ab}^T \mathbf{q}_{bc}\right)$.
if state $= 1$ **then**
  flag $\leftarrow 1$
  $\mathbf{r} \leftarrow \mathbf{w}_a$
  $\mathbf{q} \leftarrow \mathbf{q}_{ab}$
  $\mathbf{z} \leftarrow \mathbf{w}_b - \left(\frac{R}{\tan(\beta/2)} \mathbf{q}_{ab}\right)$
  if $\mathbf{p} \in \mathcal{H}(\mathbf{z}, \mathbf{q}_{ab})$ **then**
    state $\leftarrow 2$
  end if
else if state $= 2$ **then**
  flag $\leftarrow 2$
  $\mathbf{c} \leftarrow \mathbf{w}_b - \left(\frac{R}{\sin(\beta/2)}\right) \frac{\mathbf{q}_{bc} - \mathbf{q}_{ab}}{\|\mathbf{q}_{bc} - \mathbf{q}_{ab}\|}$
  $\rho \leftarrow R$
  $\lambda \leftarrow \text{sign}(q_{ab,n} q_{bc,e} - q_{ab,e} q_{bc,n})$.
  $\mathbf{z} \leftarrow \mathbf{w}_b + \left(\frac{R}{\tan(\beta/2)} \mathbf{q}_{bc}\right)$
  if $\mathbf{p} \in \mathcal{H}(\mathbf{z}, \mathbf{q}_{bc})$ **then**
    $i \leftarrow (i+1) \bmod N$.
    state $\leftarrow 1$
  end if
end if
return flag, $\mathbf{r}$, $\mathbf{q}$, $\mathbf{c}$, ρ, λ.

11.2 Dubins Paths

RWB: This section needs to be updated to conform to the format in the previous section and so that the output of the path manager is (flag, r, q, c, rho, lambda).

11.2.1 Definition of Dubin's Path

This section focuses on so-called Dubins paths where, rather than following a waypoint path, the objective is to transition from one configuration to another. It was shown in [41] that for a vehicle with kinematics given by

$$\begin{aligned}\dot{x} &= V\cos\chi\\ \dot{y} &= V\sin\chi\\ \dot{\chi} &= u,\end{aligned}$$

where V is constant and $u \in [-\bar{u}, \bar{u}]$, that the time optimal path between two different configurations consists of a circular arc, followed by a straight-line, concluding with another circular arc to the final configuration, and where the radius of the circular arcs is $V/\bar{u}$. Therefore, in the context of UAVs, Dubins paths are defined for constant-altitute, 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, the configuration of the UAV at time t is defined as the position $\mathbf{p}(t)$ and the course angle $\chi(t)$.

Given a start configuration denoted as $(\mathbf{p}_s, \chi_s)$ and an end configuration $(\mathbf{p}_e, \chi_e)$, a Dubin's 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 is a right-handed arc followed by a straight-line followed by another right-handed arc. Case II is a right-handed arc followed by a straight-line followed by a left-handed arc. Case III is a left-handed arc followed by a straight-line followed by a right handed arc. Case IV is a left-handed arc followed by a straight-line followed by another left-handed arc. The Dubin's path is defined as the case with the shortest path length.

11.2.2 Path Length Computation

In order to determine the Dubin's 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

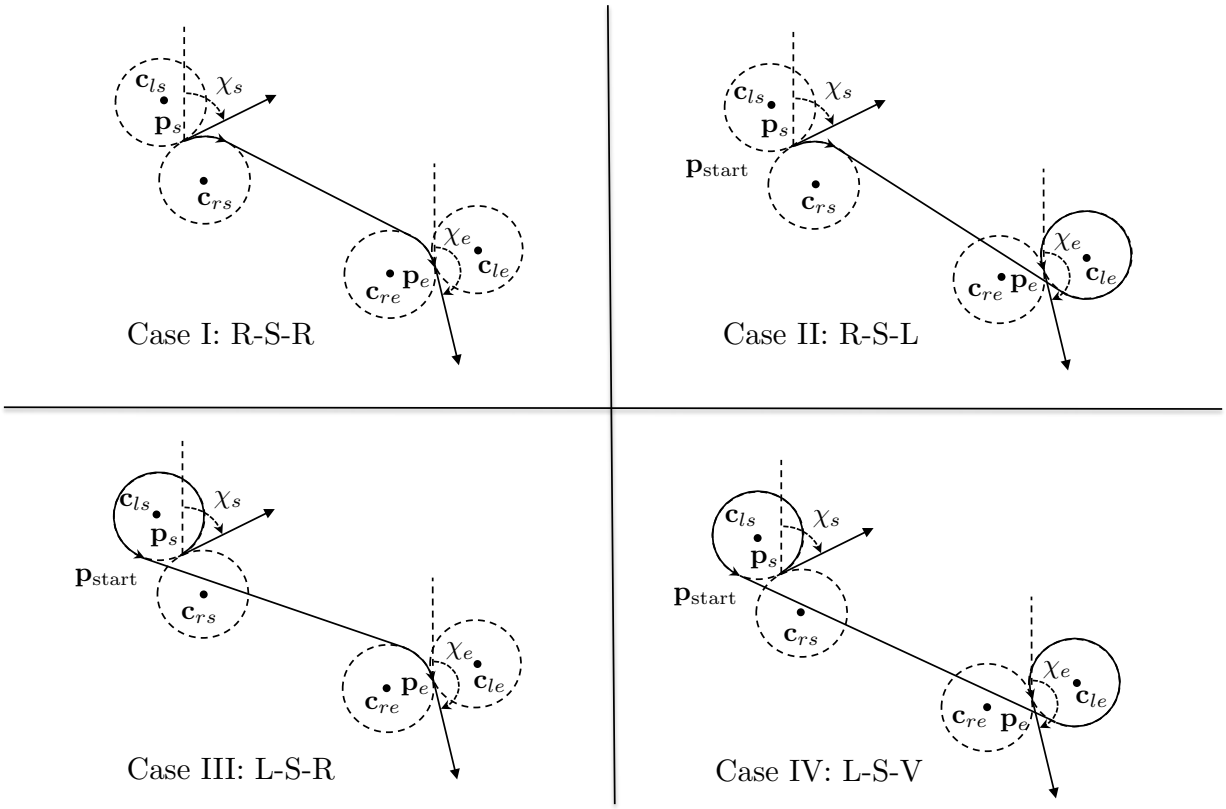


Figure 11.7: Given a start configuration $(\mathbf{p}_s, \chi_s)$, an end configuration $(\mathbf{p}_e, \chi_e)$, and a radius R, there are four possible paths consisting of an arc, a straight-line, and an arc. The Dubin's path is defined as the case that results in the shortest path length, which for this scenario is Case I.

case. Given the position $\mathbf{p}$, the orientation χ, and the radius R, the centers of the right and left turning circles are given by

$$\mathbf{c}_r = \mathbf{p} + R \begin{pmatrix} \cos(\chi + \frac{\pi}{2}), & \sin(\chi + \frac{\pi}{2}), & 0 \end{pmatrix}^T \tag{11.4}$$

$$\mathbf{c}_l = \mathbf{p} + R \begin{pmatrix} \cos(\chi - \frac{\pi}{2}), & \sin(\chi - \frac{\pi}{2}), & 0 \end{pmatrix}^T \tag{11.5}$$

.

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 assume that both θ_1 and θ_2 are between 0 and 2π. For clockwise

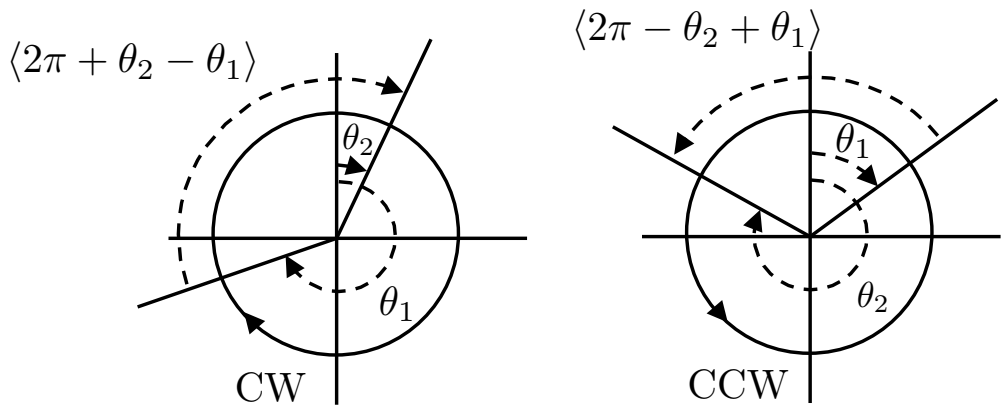


Figure 11.8: The angular distance between angles θ_1 and θ_2 for clockwise (CW) and counter clockwise (CCW) circles.

circles, the angular distance between θ_1 and θ_2 is given by

$$|\theta_2 - \theta_1|_{CW} \stackrel{\triangle}{=} \langle 2\pi + \theta_2 - \theta_1 \rangle, \tag{11.6}$$

where

$$\langle \varphi \rangle \stackrel{\triangle}{=} \varphi \mod 2\pi.$$

Similarly, for counter clockwise circles we get

$$|\theta_2 - \theta_1|_{CCW} \stackrel{\triangle}{=} \langle 2\pi - \theta_2 + \theta_1 \rangle. \tag{11.7}$$

Case I: R-S-R

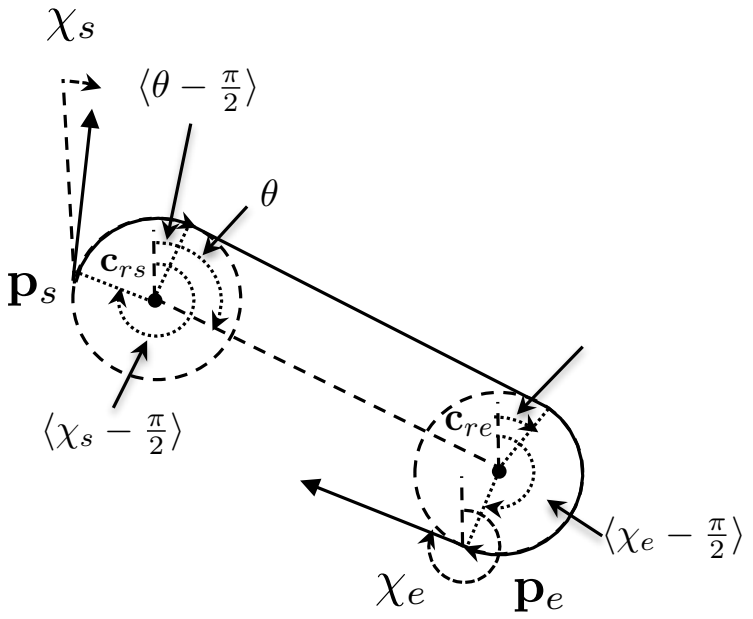


Figure 11.9: Dubins path, Case I.

The geometry for Case I is shown in Figure 11.9, where θ is the angle formed by the line between $\mathbf{c}_{rs}$ and $\mathbf{c}_{re}$. Using Eq. (11.6), the angular distance traveled along $\mathbf{c}_{rs}$ is given by

$$R\langle 2\pi + \langle\theta - \frac{\pi}{2}\rangle - \langle\chi_s - \frac{\pi}{2}\rangle\rangle.$$

Similarly, using Eq. (11.6), the angular distance traveled along $\mathbf{c}_{re}$ is given by

$$R\langle 2\pi + \langle\chi_e - \frac{\pi}{2}\rangle - \langle\theta - \frac{\pi}{2}\rangle\rangle.$$

The total path length for Case I is therefore given by

$$L_1 = \|\mathbf{c}_{rs} - \mathbf{c}_{re}\| + R\langle 2\pi + \langle\theta - \frac{\pi}{2}\rangle - \langle\chi_s - \frac{\pi}{2}\rangle\rangle + R\langle 2\pi + \langle\chi_e - \frac{\pi}{2}\rangle - \langle\theta - \frac{\pi}{2}\rangle\rangle. \quad (11.8)$$

Case II: R-S-L

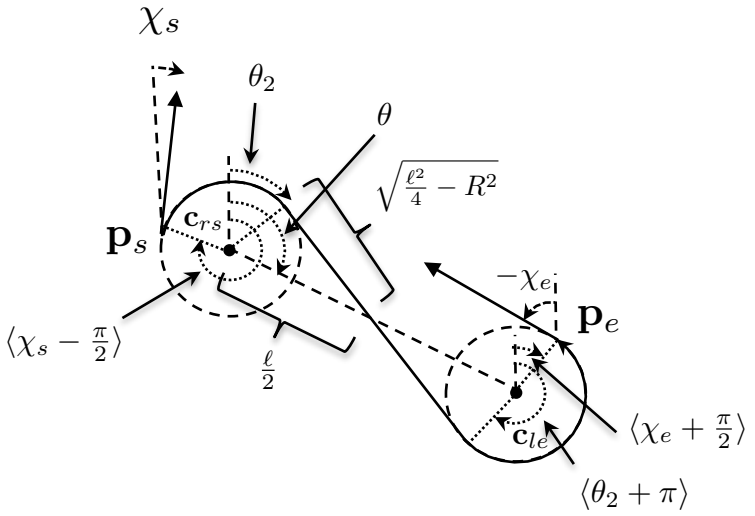


Figure 11.10: Dubins path, Case II.

The geometry for Case II is shown in Figure 11.10, where θ is the angle formed by the line between $\mathbf{c}_{rs}$ and $\mathbf{c}_{le}$, $\ell = \|\mathbf{c}_{le} - \mathbf{c}_{rs}\|$, and

$$\theta_2 = \theta - \frac{\pi}{2} + \sin^{-1}\left(\frac{2R}{\ell}\right).$$

Using Eq. (11.6), the angular distance traveled along $\mathbf{c}_{rs}$ is given by

$$R\langle 2\pi + \langle \theta - \theta_2 \rangle - \langle \chi_s - \frac{\pi}{2} \rangle \rangle.$$

Similarly, using Eq. (11.7), the angular distance traveled along $\mathbf{c}_{le}$ is given by

$$R\langle 2\pi + \langle \theta_2 + \pi \rangle \rangle - \langle \chi_e + \frac{\pi}{2} \rangle.$$

The total path length for Case II is therefore given by

$$L_2 = \sqrt{\ell^2 - 4R^2} + R\langle 2\pi + \langle \theta - \theta_2 \rangle - \langle \chi_s - \frac{\pi}{2} \rangle \rangle, + R\langle 2\pi + \langle \theta_2 + \pi \rangle \rangle - \langle \chi_e + \frac{\pi}{2} \rangle. \quad (11.9)$$

Case III: L-S-R

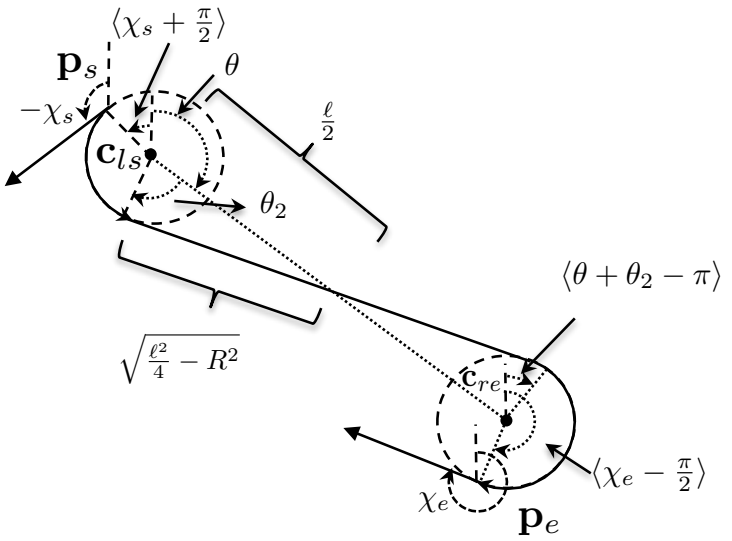


Figure 11.11: Dubins path, Case III.

The geometry for Case III is shown in Figure 11.11, where θ is the angle formed by the line between $\mathbf{c}_{ls}$ and $\mathbf{c}_{re}$, $\ell = \|\mathbf{c}_{re} - \mathbf{c}_{ls}\|$, and

$$\theta_2 = \cos^{-1} \frac{2R}{\ell}.$$

Using Eq. (11.7), the angular distance traveled along $\mathbf{c}_{ls}$ is given by

$$R\langle 2\pi + \langle \chi_s + \frac{\pi}{2} \rangle \rangle - \langle \theta + \theta_2 \rangle.$$

Similarly, using Eq. (11.6), the angular distance traveled along $\mathbf{c}_{re}$ is given by

$$R\langle 2\pi + \langle \chi_e - \frac{\pi}{2} \rangle - \langle \theta + \theta_2 - \pi \rangle \rangle.$$

The total path length for Case III is therefore given by

$$L_3 = \sqrt{\ell^2 - 4R^2} + R\langle 2\pi + \langle \chi_s + \frac{\pi}{2} \rangle \rangle - \langle \theta + \theta_2 \rangle + R\langle 2\pi + \langle \chi_e - \frac{\pi}{2} \rangle - \langle \theta + \theta_2 - \pi \rangle \rangle. \quad (11.10)$$

Case IV: L-S-L

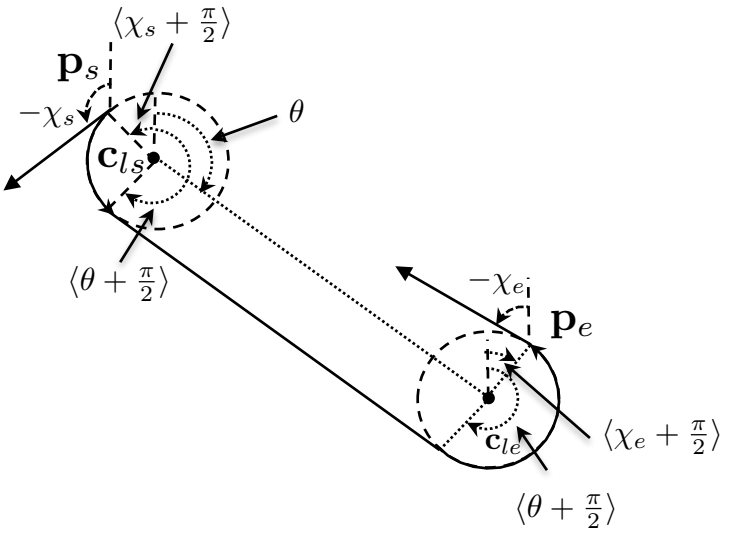


Figure 11.12: Dubins path, Case IV.

The geometry for Case IV is shown in Figure 11.12, where θ is the angle formed by the line between $\mathbf{c}_{ls}$ and $\mathbf{c}_{le}$. Using Eq. (11.7), the angular distance traveled along $\mathbf{c}_{ls}$ is given by

$$R\langle 2\pi + \langle \chi_s + \frac{\pi}{2} \rangle \rangle - \langle \theta + \frac{\pi}{2} \rangle.$$

Similarly, using Eq. (11.7), the angular distance traveled along $\mathbf{c}_{le}$ is given by

$$R\langle 2\pi + \langle \theta + \frac{\pi}{2} \rangle \rangle - \langle \chi_e + \frac{\pi}{2} \rangle.$$

The total path length for Case IV is therefore given by

$$L_4 = \|\mathbf{c}_{ls} - \mathbf{c}_{le}\| + R\langle 2\pi + \langle \chi_s + \frac{\pi}{2} \rangle \rangle - \langle \theta + \frac{\pi}{2} \rangle + R\langle 2\pi + \langle \theta + \frac{\pi}{2} \rangle \rangle - \langle \chi_e + \frac{\pi}{2} \rangle. \quad (11.11)$$

11.2.3 Algorithm for tracking Dubin's paths

The guidance algorithm for tracking a Dubin's 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 Dubin's path.

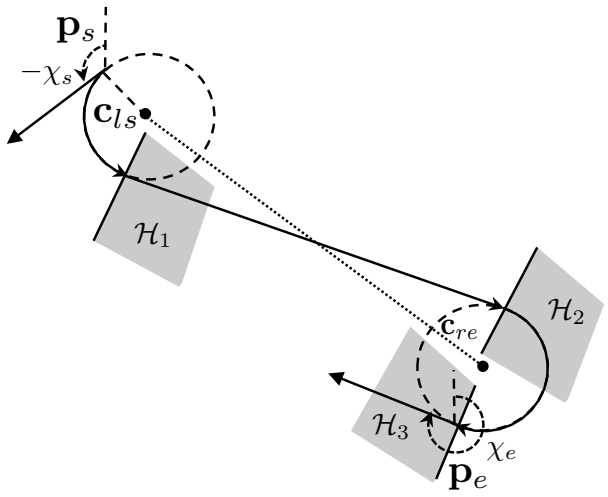


Figure 11.13: Definition of switching half-planes for Dubin's paths. The algorithm begins in a circular orbit, switching to straight-line tracking when $\mathcal{H}_1$ is entered. Orbit tracking is again initialized upon entering $\mathcal{H}_2$. The half-plane $\mathcal{H}_3$ defines the end of the Dubin's path.

The Dubin's path following guidance scheme is shown in Algorithm 11. The inputs to the algorithm include the start configuration $(\mathbf{p}_s, \chi_s)$, the end configuration $(\mathbf{p}_e, \chi_e)$, the orbit radius R which is assumed to be larger than the minimum turn radius of the MAV, and the MAV configuration $(\mathbf{p}, \chi, V_g)$. Line 1 calls the subroutine listed as Algorithm 12 which will be described below, and which returns the Dubin's path parameters. In Line 2 the initial state is set to 1. In State 1, which is defined in Lines 3–7, an orbit around the start center $\mathbf{c}_s$ with direction λ_s is commanded. The start orbit is followed until the MAV enters the half-plane $\mathcal{H}(\mathbf{w}_1, \vec{\mathbf{q}}_1)$ which is depicted as $\mathcal{H}_1$ in Figure 11.13, and the state is assigned to 2 in Line 6. In State 2, which is implemented in Lines 8–12, the vehicle is commanded to follow the straight-line segment defined by $\mathbf{w}_1$ and $\vec{\mathbf{q}}_1$ until entering the half-plane $\mathcal{H}(\mathbf{w}_2, \vec{\mathbf{q}}_1)$, which is depicted as $\mathcal{H}_2$ in Figure 11.13. In State 3, which is implemented in Lines 13–17, the end orbit is followed until entering the half-plane $\mathcal{H}(\mathbf{w}_3, \vec{\mathbf{q}}_3)$ which is depicted as $\mathcal{H}_3$ in Figure 11.13. When State 4 has been reached, the Dubin's path is complete.

The parameters of the Dubin's path are computed in Algorithm 12, where the inputs are the initial configuration $(\mathbf{p}_s, \chi_s)$, the end configuration $(\mathbf{p}_e, \chi_e)$, and the radius R, and the outputs are the length of the Dubin's path L, the center of the start orbit $\mathbf{c}_s$, the direction of the start orbit λ_s, the center of the end orbit $\mathbf{c}_e$, the direction of the end orbit λ_e, the position $\mathbf{w}_1$ and direction $\vec{\mathbf{q}}_1$ of the half-plane $\mathbf{H}_1$, the position $\mathbf{w}_2$ and direction $\vec{\mathbf{q}}_1$ of the half-plane $\mathbf{H}_2$, and the position $\mathbf{w}_3$ and direction $\vec{\mathbf{q}}_3$ of the half-plane $\mathbf{H}_3$. Lines 1–4 compute the potential positions of the centers of the start and end orbits as shown in Figure 11.7 using Equations (11.4) and (11.5). In Line 6 the

Algorithm 11 Follow Dubin's Path: $(\chi^c, \text{flag}) = \textsf{followDubinsPath}(\mathbf{p}_s, \chi_s, \mathbf{p}_e, \chi_e, R, \mathbf{p}, \chi, V_g)$

Input: Start configuration $(\mathbf{p}_s, \chi_s)$, end configuration $(\mathbf{p}_e, \chi_e)$, radius R, MAV position $\mathbf{p} = (p_n, p_e, p_d)^T$, course χ, ground speed V_g.

Require: $\|\mathbf{p}_s - \mathbf{p}_e\| \geq 3R$.

Require: R is larger than minimum turn radius of MAV.

$(L, \mathbf{c}_s, \lambda_s, \mathbf{c}_e, \lambda_e, \mathbf{w}_1, \vec{\mathbf{q}}_1, \mathbf{w}_2, \mathbf{w}_3, \vec{\mathbf{q}}_3) \leftarrow \textsf{findDubinsParameters}(\mathbf{p}_s, \chi_s, \mathbf{p}_e, \chi_e, R)$
state $\leftarrow 1$, flag $\leftarrow 0$.
if state $= 1$ **then**
  $\chi^c \leftarrow \textsf{orbit}\,(\mathbf{c}_s, R, \lambda_s, \mathbf{p}, \chi, V_g)$.
  if $\mathbf{p} \in \mathcal{H}(\mathbf{w}_1, \vec{\mathbf{q}}_1)$ **then**
    state $\leftarrow 2$
  end if
else if state $= 2$ **then**
  $\chi^c \leftarrow \textsf{wppLateral}\,(\mathbf{w}_1, \vec{\mathbf{q}}_1, \mathbf{p}, \chi, V_g)$.
  if $\mathbf{p} \in \mathcal{H}(\mathbf{w}_2, \vec{\mathbf{q}}_1)$ **then**
    state $\leftarrow 3$
  end if
else if state $= 3$ **then**
  $\chi^c \leftarrow \textsf{orbit}\,(\mathbf{c}_e, R, \lambda_e, \mathbf{p}, \chi, V_g)$.
  if $\mathbf{p} \in \mathcal{H}(\mathbf{w}_3, \vec{\mathbf{q}}_3)$ **then**
    state $\leftarrow 4$
  end if
else if state $= 4$ **then**
  Dubin's path is complete: flag $\leftarrow 1$, $\chi_c \leftarrow 0$.
end if
return χ^c at each time step.

lengths of each of the four cases shown in Figure 11.7 are computed, and the length of the Dubin's path is assigned to the smallest length. The remaining algorithm assigns the Dubin's parameters for each case as derived in Sections 11.2.2–11.2.2. The notation $\mathcal{R}_z(\theta)$ denotes the rotation matrix for a right-handed rotation of θ about the z-axis, and $\vec{\mathbf{e}}_1 = (1, 0, 0)^T$.

Algorithm 12 Find Dubin's Parameters:
$(L, \mathbf{c}_s, \lambda_s, \mathbf{c}_e, \lambda_e, \mathbf{w}_1, \vec{\mathbf{q}}_1, \mathbf{w}_2, \mathbf{w}_3, \vec{\mathbf{q}}_3) = \mathsf{findDubinsParameters}(\mathbf{p}_s, \chi_s, \mathbf{p}_e, \chi_e, R)$

Input: Start configuration $(\mathbf{p}_s, \chi_s)$, End configuration $(\mathbf{p}_e, \chi_e)$, Radius R,

Require: $\|\mathbf{p}_s - \mathbf{p}_e\| \geq 3R$.

Require: R is larger than minimum turn radius of MAV.

$\mathbf{c}_{rs} \leftarrow \mathbf{p}_s + R\mathcal{R}_z\left(\frac{\pi}{2}\right)(\cos(\chi_s), \sin(\chi_s), 0)^T$
$\mathbf{c}_{ls} \leftarrow \mathbf{p}_s + R\mathcal{R}_z\left(\frac{-\pi}{2}\right)(\cos(\chi_s), \sin(\chi_s), 0)^T$
$\mathbf{c}_{re} \leftarrow \mathbf{p}_e + R\mathcal{R}_z\left(\frac{\pi}{2}\right)(\cos(\chi_e), \sin(\chi_e), 0)^T$
$\mathbf{c}_{le} \leftarrow \mathbf{p}_e + R\mathcal{R}_z\left(-\frac{\pi}{2}\right)(\cos(\chi_e), \sin(\chi_e), 0)^T$
Compute L_1, L_2, L_3, and L_3 using Eq. (11.8), (11.9), (11.10), and (11.11).
$L \leftarrow \min\{L_1, L_2, L_3, L_4\}$.
if $\arg\min\{L_1, L_2, L_3, L_4\} = 1$ **then**
  $\mathbf{c}_s \leftarrow \mathbf{c}_{rs}$, $\lambda_s \leftarrow +1$, $\mathbf{c}_e \leftarrow \mathbf{c}_{re}$, $\lambda_e \leftarrow +1$
  $\vec{\mathbf{q}}_1 \leftarrow \frac{\mathbf{c}_e - \mathbf{c}_s}{\|\mathbf{c}_e - \mathbf{c}_s\|}$.
  $\mathbf{w}_1 \leftarrow \mathbf{c}_s + R\mathcal{R}_z\left(-\frac{\pi}{2}\right)\vec{\mathbf{q}}_1$
  $\mathbf{w}_2 \leftarrow \mathbf{c}_e + R\mathcal{R}_z\left(-\frac{\pi}{2}\right)\vec{\mathbf{q}}_2$
else if $\arg\min\{L_1, L_2, L_3, L_4\} = 2$ **then**
  $\ell \leftarrow \|\mathbf{c}_e - \mathbf{c}_s\|$
  $\theta \leftarrow \text{angle}(\mathbf{c}_e - \mathbf{c}_s)$
  $\theta_2 \leftarrow \theta - \frac{\pi}{2} + \sin^{-1}\frac{2R}{\ell}$
  $\mathbf{c}_s \leftarrow \mathbf{c}_{rs}$, $\lambda_s \leftarrow +1$, $\mathbf{c}_e \leftarrow \mathbf{c}_{le}$, $\lambda_e \leftarrow -1$
  $\vec{\mathbf{q}}_1 \leftarrow \mathcal{R}_z\left(\theta_2 + \frac{\pi}{2}\right)\mathbf{e}_1$.
  $\mathbf{w}_1 \leftarrow \mathbf{c}_s + R\mathcal{R}_z(\theta_2)\mathbf{e}_1$
  $\mathbf{w}_2 \leftarrow \mathbf{c}_e + R\mathcal{R}_z(\theta_2 + \pi)\mathbf{e}_1$
else if $\arg\min\{L_1, L_2, L_3, L_4\} = 3$ **then**
  $\mathbf{c}_s \leftarrow \mathbf{c}_{ls}$, $\lambda_s \leftarrow -1$, $\mathbf{c}_e \leftarrow \mathbf{c}_{re}$, $\lambda_e \leftarrow +1$
  $\ell \leftarrow \|\mathbf{c}_e - \mathbf{c}_s\|$
  $\theta \leftarrow \text{angle}(\mathbf{c}_e - \mathbf{c}_s)$
  $\theta_2 \leftarrow \cos^{-1}\frac{2R}{\ell}$
  $\vec{\mathbf{q}}_1 \leftarrow \mathcal{R}_z\left(\theta + \theta_2 - \frac{\pi}{2}\right)\mathbf{e}_1$.
  $\mathbf{w}_1 \leftarrow \mathbf{c}_s + R\mathcal{R}_z(\theta + \theta_2)\mathbf{e}_1$
  $\mathbf{w}_2 \leftarrow \mathbf{c}_e + R\mathcal{R}_z(\theta + \theta_2 - \pi)\mathbf{e}_1$
else if $\arg\min\{L_1, L_2, L_3, L_4\} = 4$ **then**
  $\mathbf{c}_s \leftarrow \mathbf{c}_{ls}$, $\lambda_s \leftarrow -1$, $\mathbf{c}_e \leftarrow \mathbf{c}_{le}$, $\lambda_e \leftarrow -1$
  $\vec{\mathbf{q}}_1 \leftarrow \frac{\mathbf{c}_e - \mathbf{c}_s}{\|\mathbf{c}_e - \mathbf{c}_s\|}$.
  $\mathbf{w}_1 \leftarrow \mathbf{c}_s + R\mathcal{R}_z\left(\frac{\pi}{2}\right)\vec{\mathbf{q}}_1$
  $\mathbf{w}_2 \leftarrow \mathbf{c}_e + R\mathcal{R}_z\left(\frac{\pi}{2}\right)\vec{\mathbf{q}}_2$
end if
$\mathbf{w}_3 \leftarrow \mathbf{p}_e$
$\vec{\mathbf{q}}_3 \leftarrow \mathcal{R}_z(\chi_e)\vec{\mathbf{e}}_1$

11.2.4 Sequence of Configurations

In the final section of this chapter we consider the problem of transitioning between a sequence of configurations. Define a *configuration path* as a sequence of configuration

$$\mathcal{P} = \{(\mathbf{p}_1, \chi_1), (\mathbf{p}_2, \chi_2), \cdots, (\mathbf{p}_N, \chi_N)\}. \tag{11.12}$$

A path management algorithm that transitions between the N configurations is given in Algorithm 13.

Algorithm 13 Follow configuration path: $\chi^c = \text{followConfigPath}(\mathcal{P}, \mathbf{p}, \chi, V_g)$

Input: Configuration path $\mathcal{W} = \{(\mathbf{p}_1, \chi_1), \ldots, (\mathbf{p}_N, \chi_N)\}$, MAV position $\mathbf{p} = (p_n, p_e, p_d)^T$, course χ, ground speed V_g, fillet radius R.

Require: $N \geq 3$.

$i \leftarrow 1$.
$a \leftarrow i$
$b \leftarrow (i+1) \bmod N$.
while 1 **do**
  $(\chi^c, \text{flag}) = \text{followDubinsPath}(\mathbf{p}_a, \chi_a, \mathbf{p}_b, \chi_b, R, \mathbf{p}, \chi, V_g)$.
  if flag $= 1$ **then**
    $i \leftarrow (i+1) \bmod N$
    $a \leftarrow i$
    $b \leftarrow (i+1) \bmod N$.
  end if
end while
return χ^c at each time step.

To do: Add figures showing paths that result from Algorithm 13.

11.3 Chapter Summary

Notes and References

Section 11.1 is based largely on [42]. Dubin's paths were introduced in [41]. In certain degenerate cases, the Dubin's 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 [43] 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 [44], Dubins circles are superimposed as fillets at the junction of straight-line waypoint paths produced from a Voronoi diagram. In some application, like the cooperative timing problem described in [40] 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 [42].

11.4 Design Project

The objective of this assignment is to implement Algorithms 9 and 10 for following a set of waypoints denoted as $\mathcal{W}$, and Algorithm 13 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

$$y_{\text{manager}} = \begin{pmatrix} \texttt{flag} \\ V_a^d \\ \mathbf{r} \\ \vec{\mathbf{q}} \\ \mathbf{c} \\ \rho \\ \lambda \end{pmatrix}.$$

Skeleton code for this chapter is given on the website as `chap11_manager.zip`.

RWB: Winter 2010 Semester: Only do problems 1 and 2.

11.1 Modify `path_manager.m` to implement Algorithm 9 to follow the waypoint path defined in `path_planner.m`. Test and debug the algorithm on the guidance model given in Equation (7.20). When the algorithm is working well on the guidance model, verify that it performs adequately for the full 6-DOF model.

11.2 Rename `path_manager.m` as `path_manager_fillet.m` and implement Algorithm 10 to follow the waypoint path defined in `path_planner.m`. Test and debug the algorithm on the guidance model given in Equation (7.20). When the algorithm is working well on the guidance model, verify that it performs adequately for the full 6-DOF model.

11.3 Rename `path_manager.m` as `path_manager_dubins.m` and implement Algorithm 13 to follow the path configuration defined in `path_planner_dubins.m`. Test and debug

the algorithm on the guidance model given in Equation (7.20). When the algorithm is working well on the guidance model, verify that it performs adequately for the full 6-DOF model.

Chapter 12

Path Planning

In the robotics literature, there are roughly two different approaches to motion planning: *deliberative* motion planning where explicit paths and trajectories are computed based on global world knowledge [45, 46, 47], and *reactive* motion planning which uses behavioral methods to react to local sensor information [48, 49]. 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 lacks the ability to specify and direct motion plans.

This chapter focuses on deliberative path planning techniques that have proved effective and efficient for micro air vehicles. In deliberative approaches, the MAVs 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 to describe the motion of the vehicle. Unfortunately, precise modeling of the atmosphere and the vehicle dynamics is not possible. Therefore, to compensate for the inherent uncertainty, the path planning algorithms will need to be executed on a regular basis in an outer feedback loop. Therefore, it is essential that the path planning algorithms are 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 micro air vehicles. In reference to the architecture shown in Figure 1.1, this chapter describes the design of the Waypoint Path Planner. We will describe path planning algorithms for two types of problems. In Section 12.1 we will first address point-to-point problems where the objective is to plan a waypoint path from one point to another through an obstacle field. In Section 12.2 we will

then 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 either be a sequence of waypoints, or a sequence of 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 where the MAV is tasked to maneuver through a congested airspace with obstacles that are small relative to the turning radius of the MAV. The relative size allows the obstacles to be modeled as points with zero area. The Voronoi method is essentially restricted to $2\ 1/2$-D path planning, where the altitude at each node is fixed in the map.

Given a finite set $\mathcal{P}$ of points in $\mathbb{R}^2$, the Voronoi graphs divides $\mathbb{R}^2$ into P convex cells, each containing exactly one point in $\mathcal{P}$. 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 $\mathcal{P}$. An example of a Voronoi graph is shown in Figure 12.1.

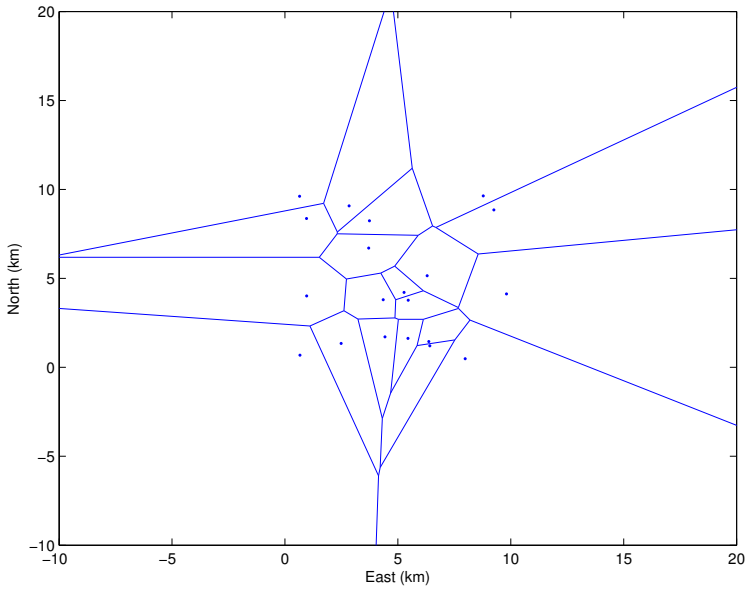


Figure 12.1: An example of a Voronoi graph with $P = 20$ point obstacles.

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 $\mathcal{P}$. Therefore, following the edges of the Voronoi graph potentially produces paths that avoid the points in $\mathcal{P}$. However, Figure 12.1 illustrates several potential pitfalls in 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 will lead to unnecessarily long excursions. Finally, note that for the two points in the lower right hand corner of Figure 12.1, the Voronoi graph produces an edge between the two points, however, since the edge is so close to the points, the corresponding waypoint path may not be desirable.

There are well established and widely available algorithms for generating Voronoi graphs. For example, Matlab has a built in Voronoi function, and C++ implementations are publicly available on the internet. Given the availability of Voronoi code, we will not discuss implementation of the algorithm. For additional discussions see [50, 51, 52].

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 $\mathcal{P}$. 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.

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 $\mathcal{P}$. 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 the line segment can be written as

$$\mathbf{w}(\sigma) = (1 - \sigma)\mathbf{v}_1 + \sigma\mathbf{v}_2,$$

where $\sigma \in [0, 1]$. The minimum distance between $\mathbf{p}$ and the graph edge can be expressed as

$$\begin{aligned}
D(\mathbf{v}_1, \mathbf{v}_2, \mathbf{p}) &\stackrel{\triangle}{=} \min_{\sigma\in[0,1]} \|\mathbf{p} - \mathbf{w}(\sigma)\| \\
&= \min_{\sigma\in[0,1]} \sqrt{(\mathbf{p} - \mathbf{w}(\sigma))^T(\mathbf{p} - \mathbf{w}(\sigma))} \\
&= \min_{\sigma\in[0,1]} \sqrt{\mathbf{p}^T\mathbf{p} - 2(1-\sigma)\sigma\mathbf{p}^T\mathbf{v}_1 - \sigma\mathbf{p}^T\mathbf{v}_2 + (1-\sigma)^2\mathbf{v}_1^T\mathbf{v}_1 + 2(1-\sigma)\sigma\mathbf{v}_1^T\mathbf{v}_2 + \sigma^2\mathbf{v}_2^T\mathbf{v}_2} \\
&= \min_{\sigma\in[0,1]} \sqrt{\|\mathbf{p} - \mathbf{v}_1\|^2 + 2\sigma(\mathbf{p} - \mathbf{v}_1)^T(\mathbf{v}_1 - \mathbf{v}_2) + \sigma^2\|\mathbf{v}_1 - \mathbf{v}_2\|^2}.
\end{aligned}$$

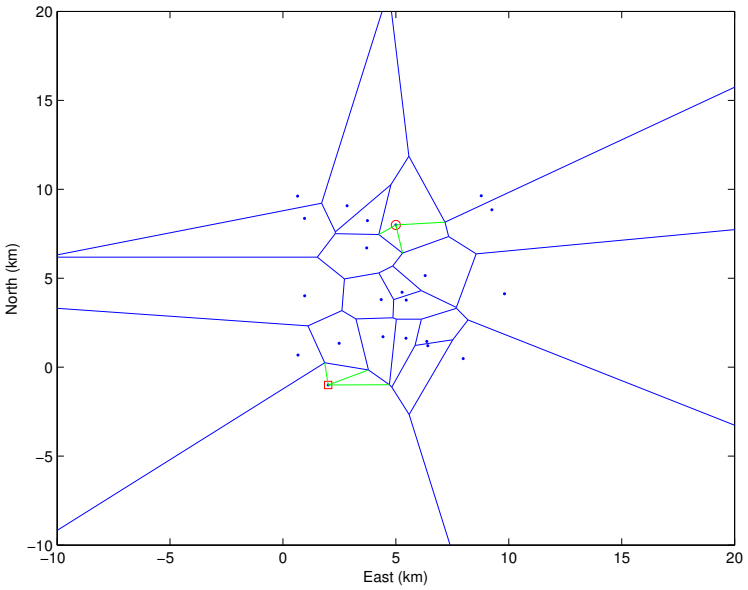


Figure 12.2: The Voronoi graph of $\mathcal{P}$ is augmented with start and end nodes, and with edges that connect the start and end notes to $\mathcal{P}$.

If σ is unconstrained, then its optimizing value is

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

and

$$\mathbf{w}(\sigma^*) = \sqrt{\|\mathbf{p} - \mathbf{v}_1\|^2 - \frac{((\mathbf{v}_1 - \mathbf{p})^T(\mathbf{v}_1 - \mathbf{v}_2))^2}{\|\mathbf{v}_1 - \mathbf{v}_2\|^2}}.$$

If we define

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

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

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

The cost for the edge defined by $(\mathbf{v}_1, \mathbf{v}_2)$ is assigned as

$$J(\mathbf{v}_1, \mathbf{v}_2) = \gamma_1 \|\mathbf{v}_1 - \mathbf{v}_2\| + \frac{\gamma_2}{D(\mathbf{v}_1, \mathbf{v}_2, \mathcal{P})}, \tag{12.1}$$

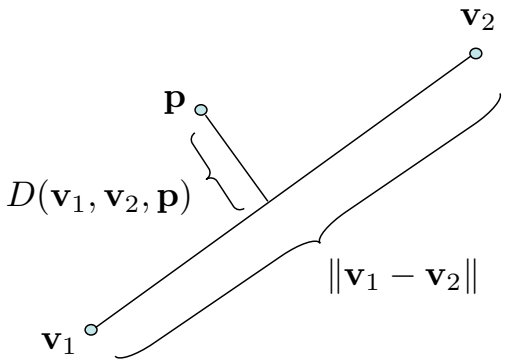


Figure 12.3: The cost penalty assigned to each edge of the Voronoi graph is proportional to the path length $\|\mathbf{v}_1 - \mathbf{v}_2\|$ and the reciprocal of the minimum distance from the path to a point in $\mathcal{P}$.

where γ_1 and γ_2 are positive weights. The first term in Eq. (12.1) is the length of the edge, and the second term is the reciprocal of the distance from the edge to the closest point in $\mathcal{P}$.

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 [52]. A well known algorithm with readily available code is the Dijkstra's algorithm[27] which has a computational complexity equal to $\bigcirc(|V|)$. An example of a path found by Dijkstra's algorithm with $\gamma_1 = 0.1$ and $\gamma_2 = 0.9$ is shown in Figure 12.4.

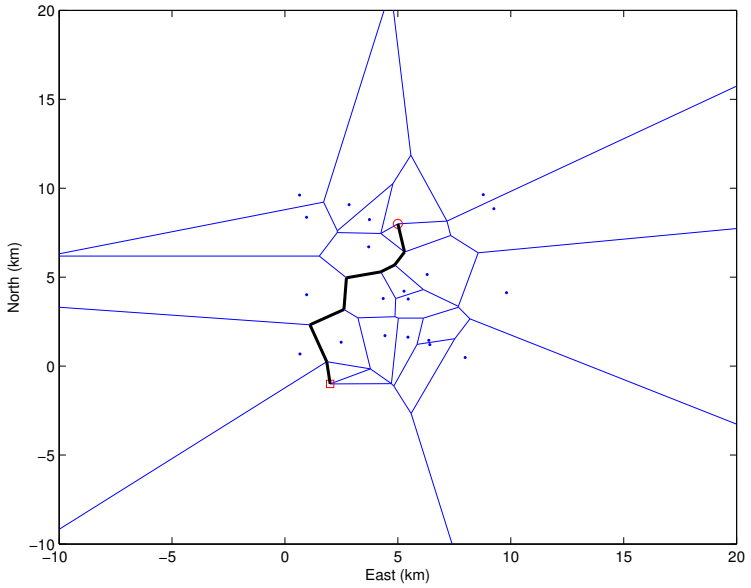


Figure 12.4: Optimal path through the Voronoi graph.

Pseudo-code for the Voronoi path planning method is listed in Algorithm 14. If there are not

a sufficient number of obstacle points in $\mathcal{P}$, the resulting Voronoi graph will be sparse and could potentially have many edges extending to infinity. To avoid that situation, Algorithm 14 requires that $\mathcal{P}$ 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 $\mathcal{P}$ 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 14 Plan Voronoi Path: $\mathcal{W} = \mathsf{planVoronoi}(\mathcal{P}, \mathbf{p}_s, \mathbf{p}_e)$

Input: Obstacle points $\mathcal{P}$, start position $\mathbf{p}_s$, end position $\mathbf{p}_e$.

Require: $|\mathcal{P}| \geq 10$. Randomly add points if necessary.

1: $(V, E) = \mathsf{constructVoronoiGraph}(\mathcal{P})$.
2: $V^+ = V \bigcup \{\mathbf{p}_s\} \bigcup \{\mathbf{p}_e\}$.
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: $E^+ = E \bigcup_{i=1,2,3} (\mathbf{v}_{is}, \mathbf{p}_s) \bigcup_{i=1,2,3} (\mathbf{v}_{ie}, \mathbf{p}_e)$.
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 Eq. (12.1).
7: **end for**
8: $\mathcal{W} = \mathsf{DijkstraSearch}(V^+, E^+, \mathbf{J})$.
9: **return** $\mathcal{W}$.

One of the disadvantages of the Voronoi method described in Algorithm 14 is that it is limited to point obstacles. However, there are straight-forward 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 either contained inside an obstacle, or that 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 planned path. Additional examples of Voronoi graphs, and the associated shortest paths, are shown in Figure 12.6.

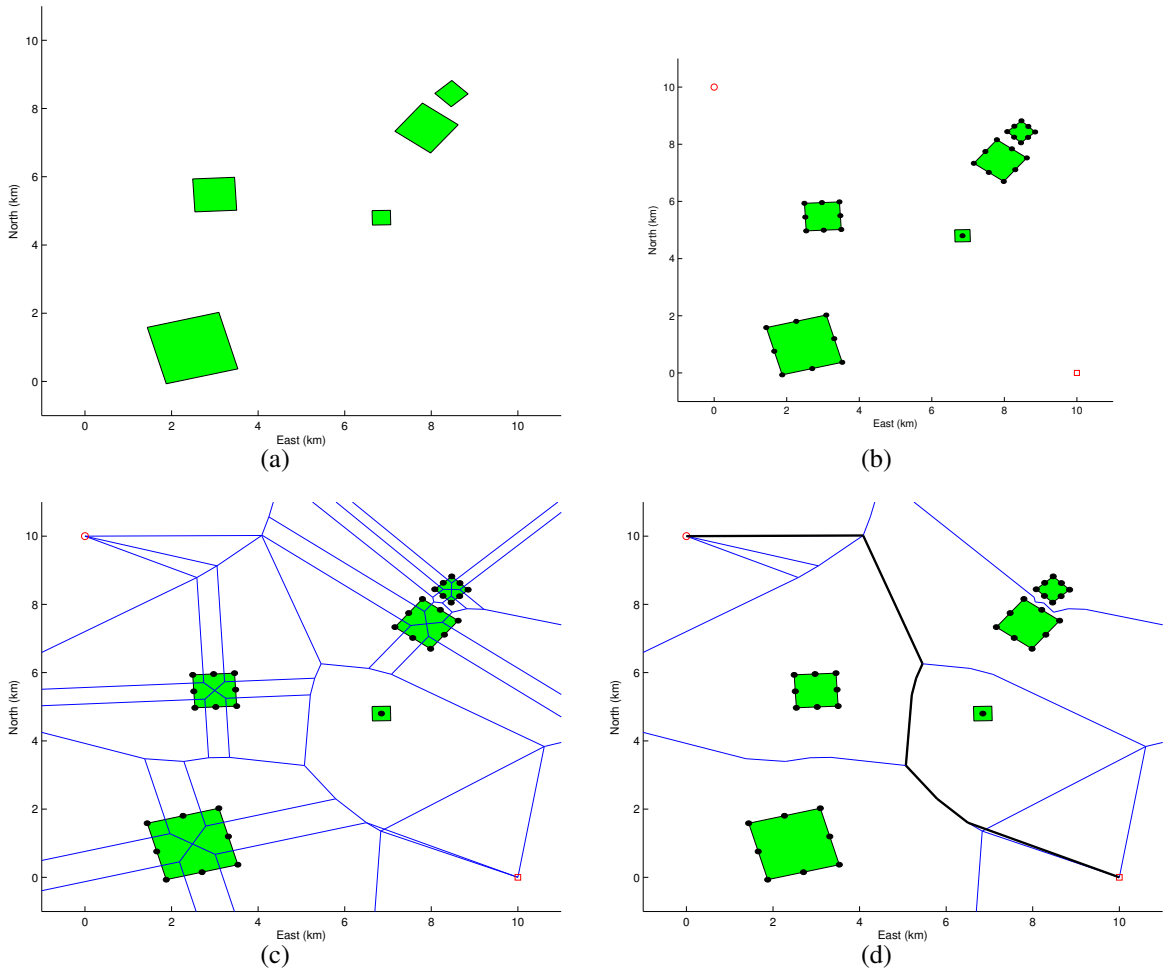


Figure 12.5: (a) An obstacle field with non-point obstacles. (b) The first step in using the Voronoi method to construct a path through the obstacle field is to insert points around the perimeter of the obstacles. (c) The resulting Voronoi graph includes many infeasible links that are either contained inside the obstacles, or terminate on the boundary of the obstacle. (d) When infeasible links are removed, the resulting graph can be used to plan paths through the obstacle field.

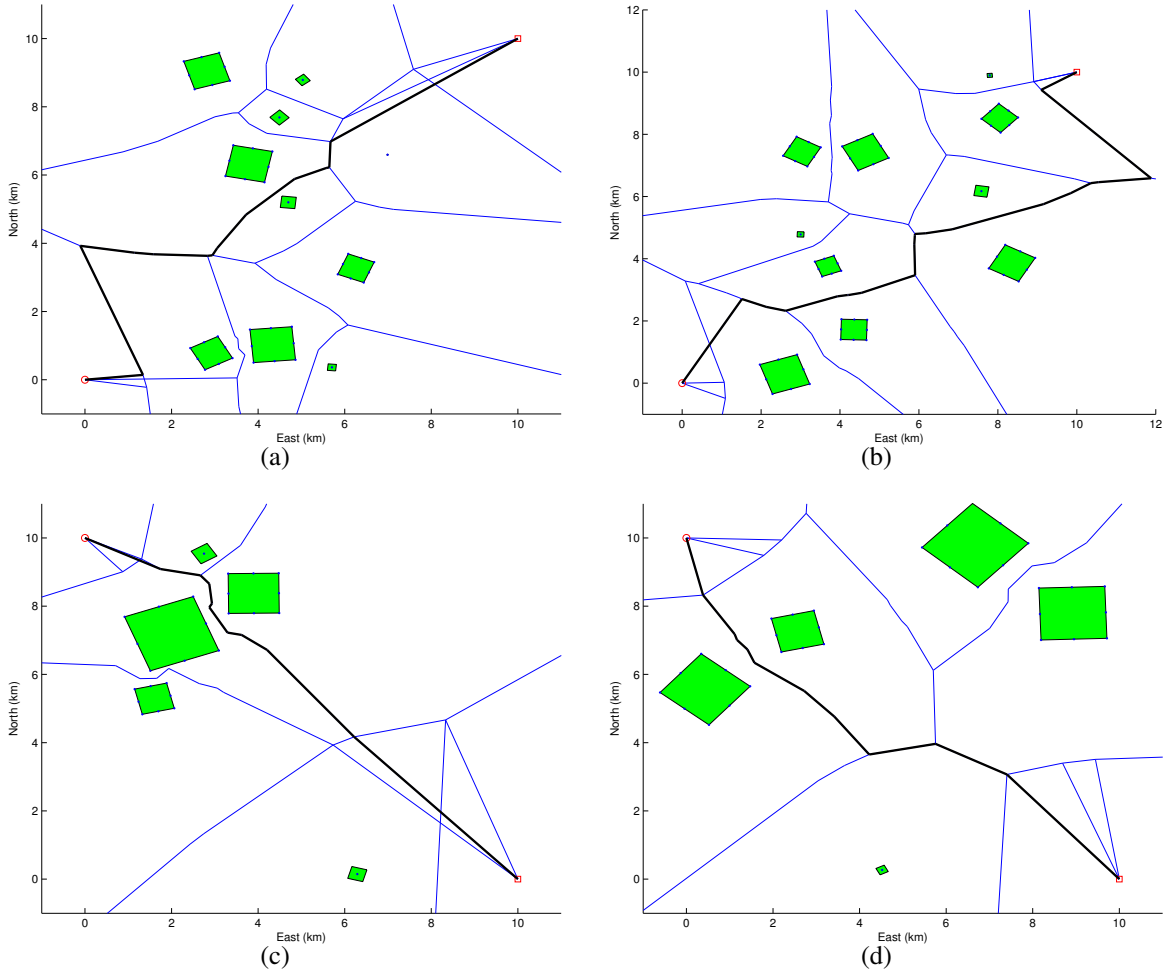


Figure 12.6: Additional results for path planning through non-point obstacle fields using the Voronoi path planning method.

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 12.7 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.

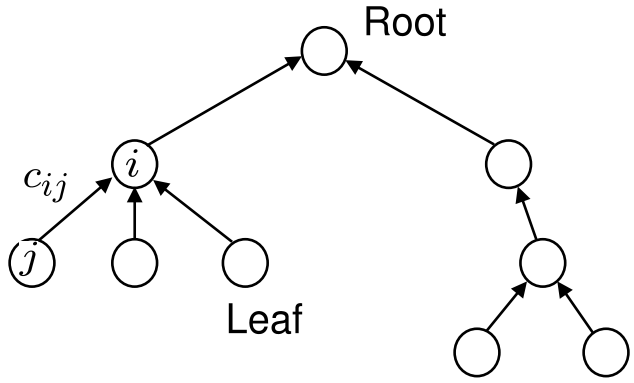


Figure 12.7: A tree is a special graph where every node, except the root, has exactly one parent.

The basic idea of the Rapidly Exploring Random Tree (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 12.8 depicts the basic RRT algorithm. As shown in subfigure 12.8(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 point $\mathbf{p}$ in the workspace. As shown in subfigure 12.8(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 subfigure 12.8(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 subfigure 12.8(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, and as shown in subfigure 12.8(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 15 gives the basic RRT algorithm. In Line 1 the RRT graph G is initialized to contain only the start node. The while loop in Lines 2–14 adds nodes to the RRT graph until the end nodes 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 7, then $\mathbf{v}^+$ is added to G in Line 7 and the cost matrix is updated in Line 8. The *if* statement in Line 10 checks to see if the new node $\mathbf{v}^+$ can be connected directly to the end node $\mathbf{p}_e$. If so, $\mathbf{p}_e$ is added to G in Line 11–12, and the algorithm ends in Line 15 by returning the shortest waypoint path in G.

The result of implementing Algorithm 15 for four different randomly generated obstacle fields and randomly generated start and end nodes, is shown in red in Figure 12.9. Note that the paths generated by Algorithm 15 sometimes vary needlessly, and that eliminating some nodes may result in more efficient path. Algorithm 16 gives a simple scheme for smoothing the paths generated by Algorithm 15. The basic idea is to remove intermediate nodes if a feasible path still exists. The result of applying Algorithm 16 is shown in black in Figure 12.9.

There are numerous extensions to the basic RRT algorithm. A common extension, which is discussed in [53], 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, namely, waypoint planning over 3D terrain, and using Dubin's paths to plan kinematically feasible paths in complex 2D terrain.

Algorithm 15 Plan RRT Path: $\mathcal{W} = \mathsf{planRRT}(\mathcal{T}, \mathbf{p}_s, \mathbf{p}_e)$

Input: Terrain map $\mathcal{T}$, start configuration $\mathbf{p}_s$, end configuration $\mathbf{p}_e$.

Initialize RRT graph $G = (V, E)$ as $V = \mathbf{p}_s$, $E = \emptyset$.
while The end node $\mathbf{p}_e$ is not connected to G, i.e., $\mathbf{p}_e \notin V$ **do**
  $\mathbf{p} \leftarrow \mathsf{generateRandomConfiguration}(\mathcal{T})$
  $\mathbf{v}^* \leftarrow \mathsf{findClosestConfiguration}(\mathbf{p}, V)$
  $\mathbf{v}^+ \leftarrow \mathsf{planPath}(\mathbf{v}^*, \mathbf{p}, D)$
  if $\mathsf{existFeasiblePath}(\mathcal{T}, \mathbf{v}^*, \mathbf{v}^+)$ **then**
    Update graph $G = (V, E)$ as $V \leftarrow V \bigcup \{\mathbf{v}^+\}$, $E \leftarrow E \bigcup \{(\mathbf{v}^*, \mathbf{v}^+)\}$
    Update edge costs as $C[(\mathbf{v}^*, \mathbf{v}^+)] \leftarrow \mathsf{pathLength}(\mathbf{v}^*, \mathbf{v}^+)$
  end if
  if $\mathsf{existFeasiblePath}(\mathcal{T}, \mathbf{v}^+, \mathbf{p}_e)$ **then**
    Update graph $G = (V, E)$ as $V \leftarrow V \bigcup \{\mathbf{p}_e\}$ $E \leftarrow E \bigcup \{(\mathbf{v}^*, \mathbf{p}_e)\}$
    Update edge costs as $C[(\mathbf{v}^*, \mathbf{p}_e)] \leftarrow \mathsf{pathLength}(\mathbf{v}^*, \mathbf{p}_e)$
  end if
end while
$\mathcal{W} = \mathsf{findShortestPath}(G, C)$.
return $\mathcal{W}$.

Algorithm 16 Smooth RRT Path: $(\mathcal{W}_s, C_s) = \mathsf{smoothRRT}(\mathcal{T}, \mathcal{W}, C)$

Input: Terrain map $\mathcal{T}$, waypoint path $\mathcal{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$, cost matrix C.

Initialized smoothed path $\mathcal{W}_s \leftarrow \{\mathbf{w}_1\}$
Initialize pointer to current node in $\mathcal{W}_s$: $i \leftarrow 1$.
Initialize pointer to next node in $\mathcal{W}$: $j \leftarrow 2$
while $j < N$ **do**
  $\mathbf{w}_s \leftarrow \mathsf{getNode}(\mathcal{W}_s, i)$
  $\mathbf{w}^+ \leftarrow \mathsf{getNode}(\mathcal{W}, j+1)$
  if $\mathsf{existFeasiblePath}(\mathcal{T}, \mathbf{w}_s, \mathbf{w}^+) =$ FALSE **then**
    Get last node: $\mathbf{w} \leftarrow \mathsf{getNode}(\mathcal{W}, j)$
    Add deconflicted node to smoothed path: $\mathcal{W}_s \leftarrow \mathcal{W}_s \bigcup \{\mathbf{w}\}$
    Update smoothed cost: $C_s[(\mathbf{w}_s, \mathbf{w})] \leftarrow \mathsf{pathLength}(\mathbf{w}_s, \mathbf{w})$
    $i \leftarrow i + 1$
  end if
  $j \leftarrow j + 1$
end while
Add last node from $\mathcal{W}$: $\mathcal{W}_s \leftarrow \mathcal{W}_s \bigcup \{\mathbf{w}_N\}$
Update smoothed cost: $C_s[(\mathbf{w}_i, \mathbf{w}_N)] \leftarrow \mathsf{pathLength}(\mathbf{w}_i, \mathbf{w}_N)$
return $\mathcal{W}_s$.

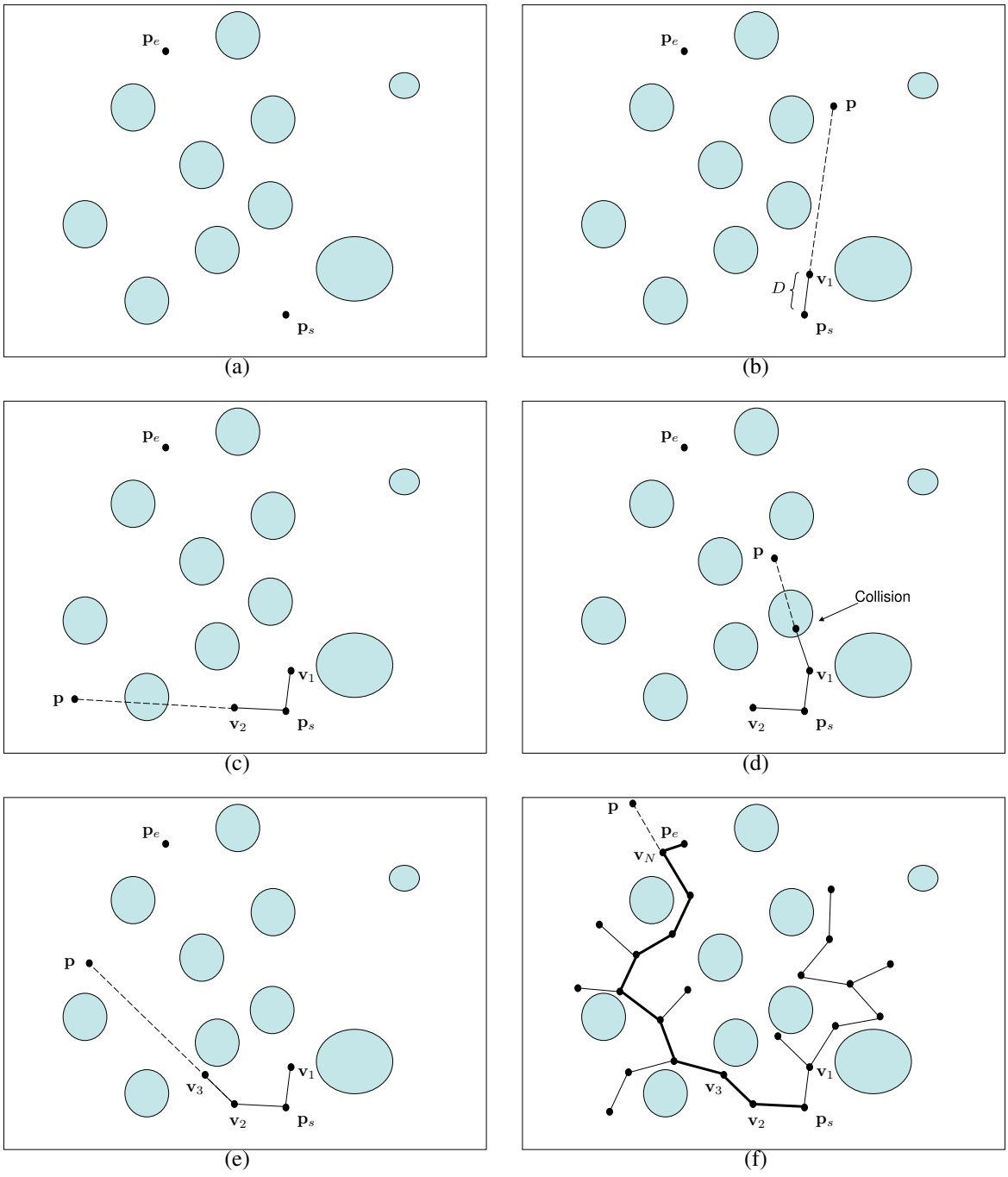


Figure 12.8: (a) The RRT algorithm is initialized with a terrain map and a start-node and an end-node. (b) and (c) The RRT graph is extended by randomly generating a point $\mathbf{p}$ in the terrain, and planning a path of length D in the direction of $\mathbf{p}$. (d) If the resulting configuration is not feasible, then it is not added to the RRT graph, and the process continues as shown in (e). (f) The RRT algorithm completes when the end-node is added to the the RRT graph.

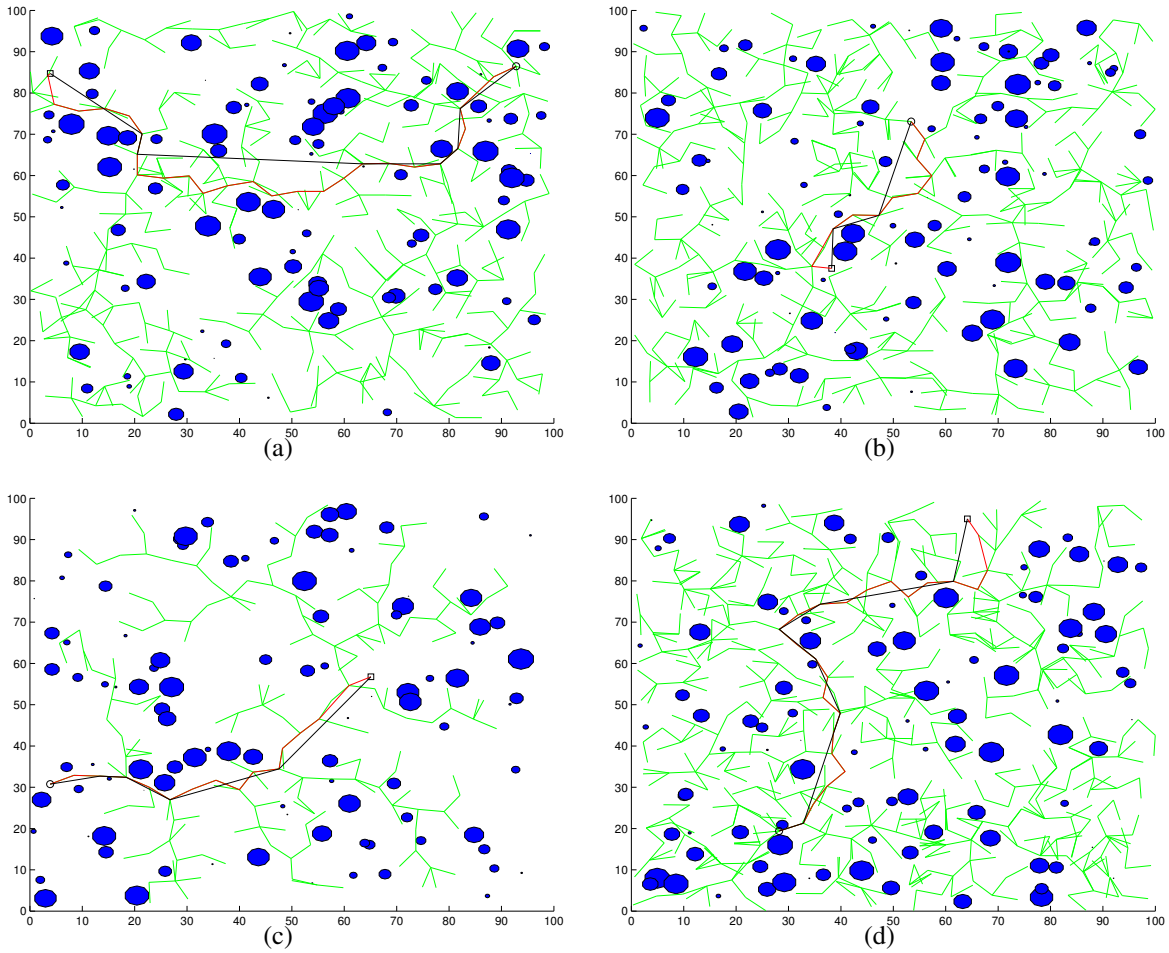


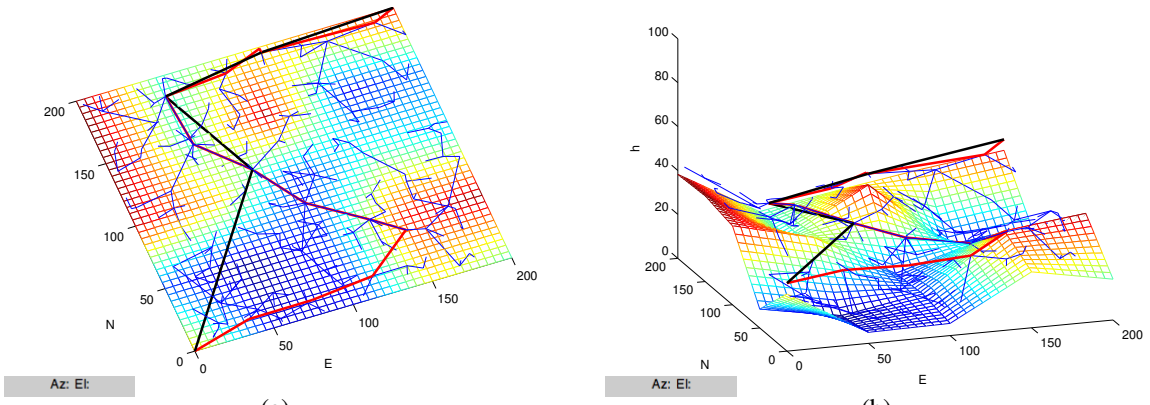
Figure 12.9: The result of Algorithm 15 for four randomly generated obstacle fields and randomly generated start and end nodes is indicated by the red line. The smoothed paths generated by Algorithm 16 are indicated by the black lines.

RRT Waypoint Planning over 3D Terrain

In this section we will consider the extension of the basic RRT algorithm to planning waypoint paths over 3D 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 3D 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 to randomly select one of these levels.

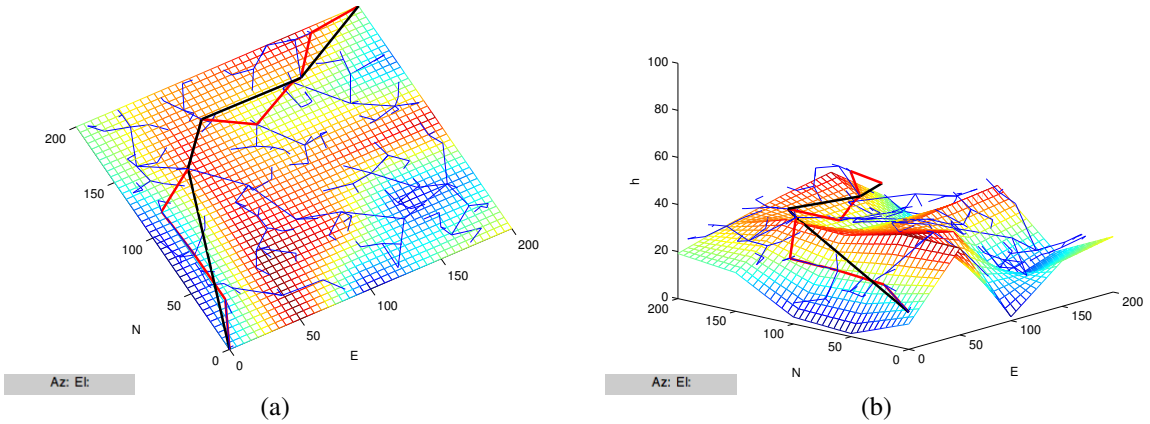
In this section, we will select the altitude as a fixed height-above-ground h_{AGL}. Therefore, the RRT graph will, in essence, be a 2D graph that follows the contour of the terrain. The output of Algorithm 15 will be a path that follows the terrain at a fixed altitude h_{AGL}. However, the smoothing step represented by Algorithm 16 will result in paths with much less altitude variation. For 3D terrain, the climb rate and descent rates of the MAV are usually constrained to be within certain limits. The function existFeasiblePath in Algorithms 15 and 16 can be modified to ensure that the climb and descent rates are satisfied.

The results of the 3D RRT algorithm are shown in Figures 12.10 and 12.11 where the blue lines represent the RRT tree, the red line is the RRT path generated by Algorithm 15 and the solid black line is the smoothed path generated by Algorithm 16.



(a) (b)

Figure 12.10: (a) Overhead view of the results of the 3D RRT waypoint path planning algorithm. (b) Side view of the results of the 3D RRT waypoint path planning algorithm. The blue lines are the RRT graph, the red line is the RRT path returned by Algorithm 15, and the thick black line is the smoothed path. The RRT graph is generated at a fixed height above the terrain.



(a) (b)

Figure 12.11: (a) Overhead view of the results of the 3D RRT waypoint path planning algorithm. (b) Side view of the results of the 3D RRT waypoint path planning algorithm. The blue lines are the RRT graph, the red line is the RRT path returned by Algorithm 15, and the thick black line is the smoothed path. The RRT graph is generated at a fixed height above the terrain.

RRT Dubin's Path Planning in a 2D 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 Dubin's paths, as discussed in Section 11.2. Dubin's 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.

In this section we will generate a random configuration as follows:

1. Generate a random North-East position in the environment.

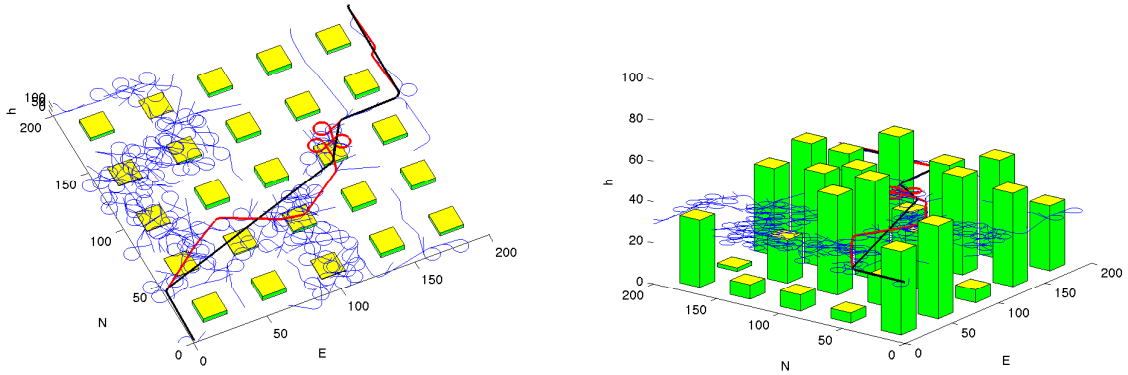
2. Find the closest node in the RRT to the new point.

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.

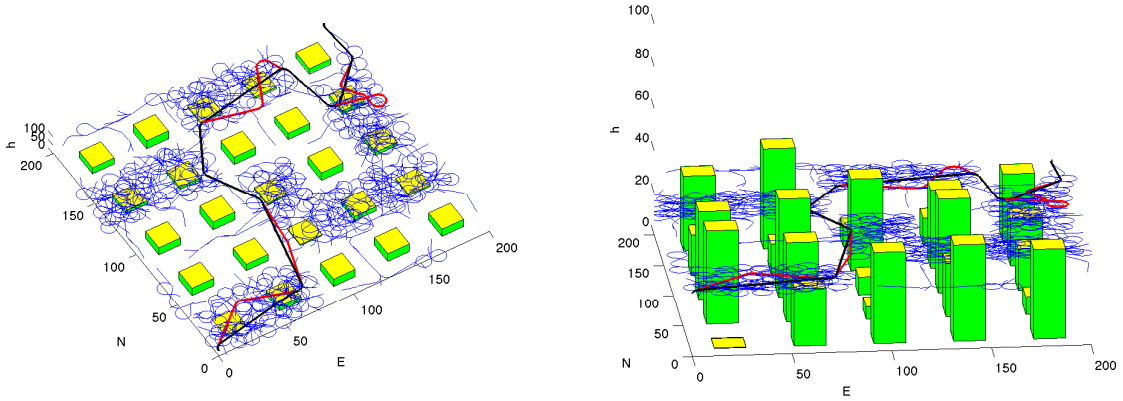
The RRT algorithm is then implemented as described in Algorithm 15 where the function pathLength returns the length of the Dubin's path between configuration.

The results of the RRT Dubins algorithm are shown in Figures 12.12 and 12.13 where the blue lines represent the RRT tree, the red line is the RRT path generated by Algorithm 15 and the solid black line is the smoothed path generated by Algorithm 16.



(a) (b)

Figure 12.12: (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 blue lines are the RRT graph, the red line is the RRT path returned by Algorithm 15, and the thick black line is the smoothed path. The RRT graph is generated at a fixed height above the terrain.



(a) (b)

Figure 12.13: (a) Overhead view of the results of the 3D RRT waypoint path planning algorithm. (b) Side view of the results of the 3D RRT waypoint path planning algorithm. The blue lines are the RRT graph, the red line is the RRT path returned by Algorithm 15, and the thick black line is the smoothed path. The RRT graph is generated at a fixed height above the terrain.

12.2 Coverage Algorithms

In this section we will briefly discuss coverage algorithms where the objective is not to transition from a start configuration to an end configuration, but rather to cover as much area as possible. Coverage algorithms are used, for example, in search problems where the air vehicle is searching for objects of interest within a given region. Since the location of the object may be unknown, the region should be searched as uniformly as possible. The particular algorithm that we present in this chapter allows for prior information about possible locations of objects to be included.

The basic idea is to maintain two maps in memory: the terrain map, and the return map. The terrain map is used to detect possible collisions with the environment. We will model the benefit of being at a particular location in the terrain by a return value R_i, where i indexes the location in the terrain. In order to ensure uniform coverage of an area, the return map is initialized so that all locations in the terrain have the same initial return value. As locations are visited, the return value is decremented by a fixed amount according to

$$R_i[k] = R_i[k] - c, \tag{12.2}$$

where c is a positive constant. The path planner searches for paths that provide the largest possible return over a finite look-ahead window.

The basic coverage algorithm is listed in Algorithm 17. At each step of the algorithm, a look-ahead tree is generated from the current MAV configuration, and is used to search for regions of the terrain with a large return value. In Line 1 the look-ahead tree is initialized to the start configuration $\mathbf{p}_s$, and in Line 8 the look-ahead tree is reset to the current configuration. More advanced algorithms could be designed to retain portions of the look-ahead tree that have already been explored. The return map is initialized in Line 2. We initialize the return map to be a large constant number plus additive noise. The additive noise facilitates choices in the initial stages of the algorithm where all regions of the terrain need to be search and therefore produce equal return values. After the MAV has moved into a new region of the terrain, the return map $\mathcal{R}$ is updated in Line 9 according to Equation 12.2. In Line 5 the look-ahead tree is generated, starting from the current configuration. The look-ahead tree is generated in a way that avoids obstacles in the terrain. In Line 6 the look-ahead tree G is searched for the the path that produces the largest return value.

There are several techniques that can be used to generate the look-ahead tree in Line 5 of Algorithm 17, and we will briefly describe two possible methods. In the first method, the path is given by waypoints, and the look-ahead tree is generated by taking finite steps of length L and, at the end of each step, allowing the MAV to move straight or change heading by $\pm\lambda$ at each configuration. A three step look-ahead tree where $\lambda = \pi/6$ is shown in Figure 12.14.

Algorithm 17 Plan Cover Path: planCover$(\mathcal{T}, \mathcal{R}, \mathbf{p})$

Input: Terrain map $\mathcal{T}$, return map $\mathcal{R}$, initial configuration $\mathbf{p}_s$.

Initialize look-ahead tree $G = (V, E)$ as $V = \{\mathbf{p}_s\}$, $E = \emptyset$
Initialize return map $\mathbf{R} = \{R_i : i \text{ indexes the terrain}\}$
$\mathbf{p} = \mathbf{p}_s$
for Each planning cycle **do**
    $G =$ generateTree$(\mathbf{p}, \mathcal{T}, \mathcal{R})$
    $\mathcal{W} =$ highestReturnPath(G)
    Update $\mathbf{p}$ by moving along the first segment of $\mathcal{W}$
    Reset $G = (V, E)$ as $V = \{\mathbf{p}\}$, $E = \emptyset$
    $\mathcal{R} =$ updateReturnMap$(\mathcal{R}, \mathbf{p})$
end for

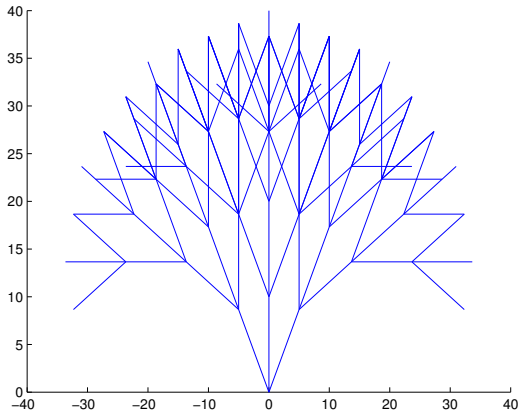


Figure 12.14: A look-ahead tree of depth three is generated from the position $(0, 0)$, where $L = 10$, and $\lambda = \pi/6$.

The results of using a waypoint look-ahead tree in Algorithm 17 are shown in Figure 12.15. Figure 12.15 (a) shows paths through an obstacle field where the look-ahead length is $L = 5$, the allowed heading change at each step is $\lambda = \pi/6$, and the depth of the look-ahead tree is three. The associated return map after 200 iterations of the algorithm is shown in Figure 12.15 (b). Figure 12.15 (c) shows paths through an obstacle field where the look-ahead length is $L = 5$, the allowed heading change at each step is $\lambda = \pi/3$, and the depth of the look-ahead tree is three. The associated return map after 200 iterations of the algorithm is shown in Figure 12.15 (d). Note that the area is approximately uniformly covered, but that paths through regions are often repeated, especially through tight regions.

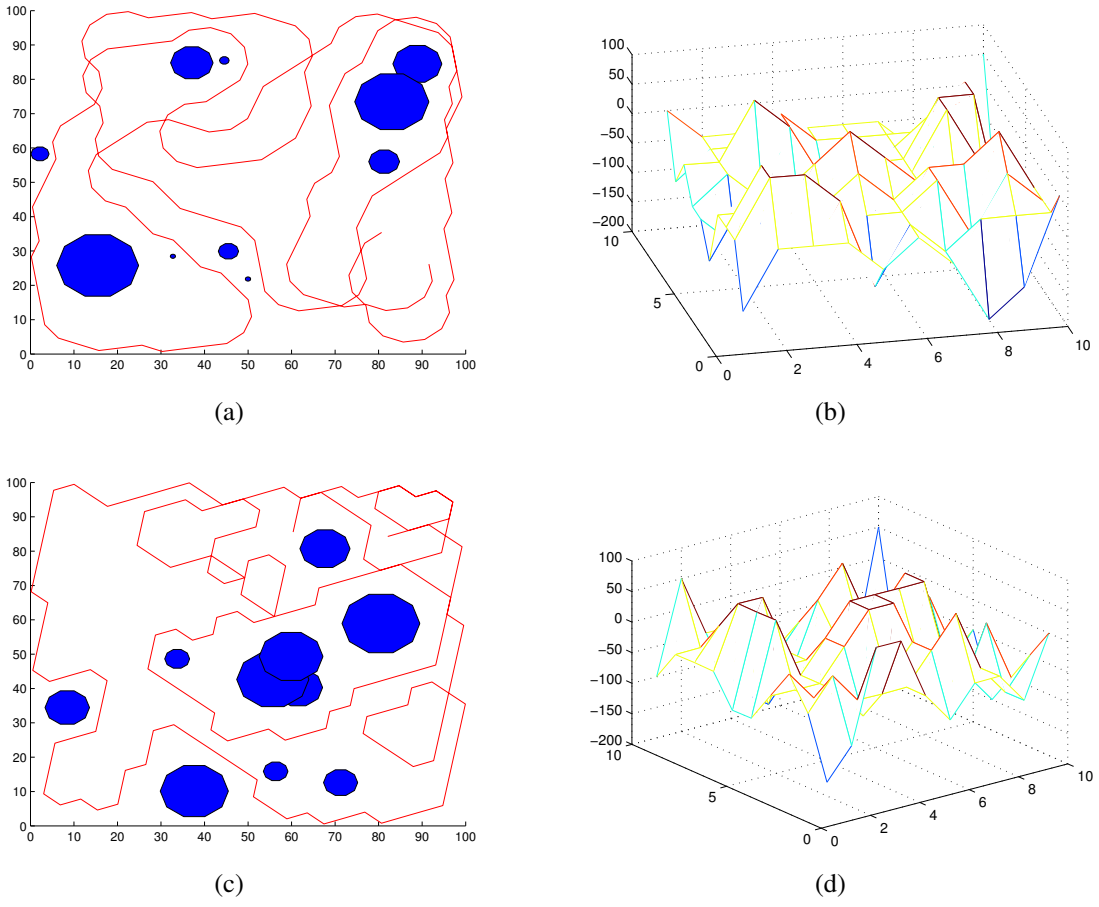


Figure 12.15: (a) and (c) Overhead view of the results of the coverage algorithm using a three angle expansion tree. The associated return maps after 200 planning cycles is shown in (b) and (d). In (a) and (b) the allowed heading change is $\lambda = \pi/6$, and in (c) and (d) the allowed heading change is $\lambda = \pi/3$.

Another method that can be used to generate the look-ahead tree in Line 5 of Algorithm 17 is an RRT algorithm using Dubin's paths. Given the current configuration, N steps of the RRT tree expansion algorithm are used to generate the look-ahead tree. Results using this algorithm in a 3D urban environment are shown in Figure 12.16. Figures 12.16(a) and 12.16(d) show overhead views of two different instances of the algorithm. The altitude of the MAV is fixed at a certain height and so buildings prevent certain regions from being searched. A side view of the results are shown in Figures 12.16(b) and 12.16(e) to provide the 3D perspective. The associated return maps are shown in Figures 12.16(c) and 12.16(f). The results again show that the area is covered fairly uniformly, but it also highlights that the coverage algorithm is not particularly efficient.

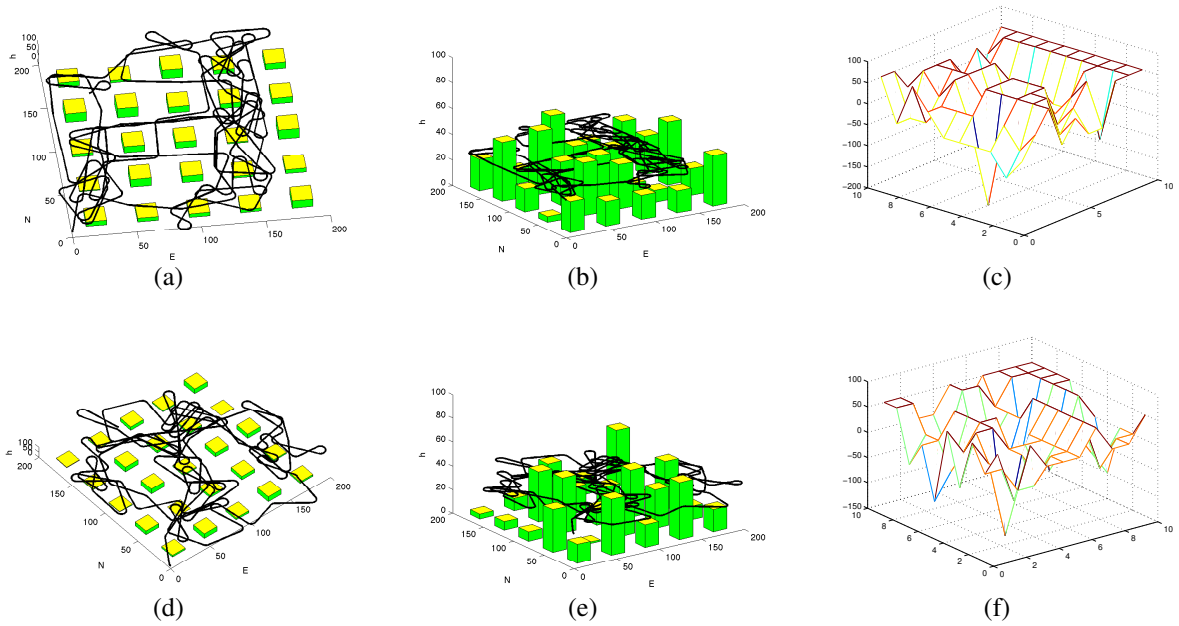


Figure 12.16: (a), (d) Overhead view of the results of the coverage algorithm using the RRT planner to find new regions using Dubin's paths between configurations. (b), (e) Side view of the results. (c), (f) The return map after 100 iterations of the search algorithm.

12.3 Chapter Summary

Notes and References

There is an extensive literature on path planning methods including the textbooks [54, 55, 56], which contain a thorough review of the literature.

An introduction to the Voronoi graph is contained in [51] and early applications of Voronoi techniques to UAV path planning is described in [44, 57, 58, 59]. Incremental construction of Voronoi graphs based on sensor measurements are described in [60, 61]. An effective search technique for Voronoi diagrams is the Eppstein's k-shortest paths algorithm [62].

The RRT algorithm was first introduced in [63] and applied to nonholonomic robotic vehicles in[64, 53]. There are numerous applications of RRTs reported in the literature, as well as extensions to the basic algorithms [65]. The RRT algorithm is closely related to the probabilistic roadmap technique described in [66], and which is applied to UAVs in [67].

There are several coverage algorithms discussed in the literature. In [68] a coverage algorithm that plan paths such that the robot passes over all points in its free space is described, as well as a nice survey of other coverage algorithms reported in the literature. A coverage algorithm in the presence of moving obstacles is described in [69]. Multiple vehicle coverage algorithms are discussed in [70], and coverage algorithms in the context of mobile sensor networks are described in [71, 72].

12.4 Design Project

12.1 Homework problem 1.

Chapter 13

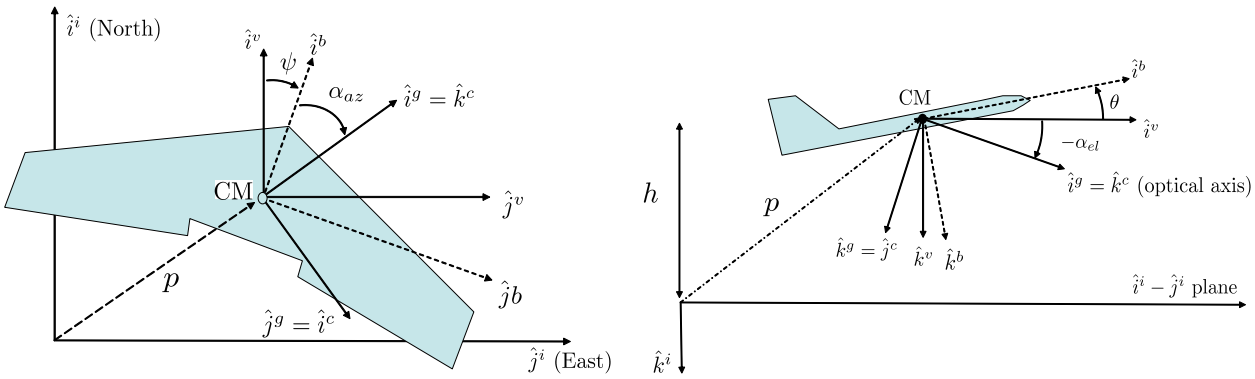
Vision Guided Navigation

One of the primary reasons for the current interest in micro air vehicles 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 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 [73].

The objective of this chapter is to briefly introduce some of the issues that arise in vision based guidance and control of micro air vehicles. 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 3D objects to their 2D 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 geo-location 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 target motion and by translational and rotational motion of the air platform. 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, in Section 13.6 we describe an algorithm that uses vision to precisely land on a ground based target.

13.1 Gimbal and Camera Frames and Projective Geometry

In this section we will assume that the origins of the camera and gimbal frames are located at the center of mass of the vehicle. For more general geometry see [74]. Figure 13.1 show the relationship between the vehicle and body frames of the MAV and the gimbal and camera frames. There are three frames of interest: the gimbal-1 frame denoted by $\mathcal{F}^{g1} = (\hat{i}^{g1}, \hat{j}^{g1}, \hat{k}^{g1})$, the gimbal



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

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

frame denoted by $\mathcal{F}^g = (\hat{i}^g, \hat{j}^g, \hat{k}^g)$, and the camera frame denoted by $\mathcal{F}^c = (\hat{i}^c, \hat{j}^c, \hat{k}^c)$. The gimbal-1 frame is obtained by rotating the body frame about the $\hat{k}^b$-axis by an angle of α_{az}, which is called the gimbal azimuth angle. The rotation from the body to the gimbal-1 frame is given by

$$R_b^{g1}(\alpha_{az}) \stackrel{\triangle}{=} \begin{pmatrix} \cos\alpha_{az} & \sin\alpha_{az} & 0 \\ -\sin\alpha_{az} & \cos\alpha_{az} & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{13.1}$$

The gimbal frame is obtained by rotating the gimbal-1 frame about the $\hat{j}^{g1}$ axis by an angle of α_{el}, which is called the gimbal elevation angle. Note that a negative elevation angle points the camera toward the ground. The rotation from the gimbal-1 frame to the gimbal frame is given by

$$R_{g1}^{g}(\alpha_{el}) \stackrel{\triangle}{=} \begin{pmatrix} \cos\alpha_{el} & 0 & -\sin\alpha_{el} \\ 0 & 1 & 0 \\ \sin\alpha_{el} & 0 & \cos\alpha_{el} \end{pmatrix}. \tag{13.2}$$

Therefore, the rotation from the body to the gimbal frame is given by

$$R_b^g = R_{g1}^g R_b^{g1} = \begin{pmatrix} \cos\alpha_{el}\cos\alpha_{az} & \cos\alpha_{el}\sin\alpha_{az} & -\sin\alpha_{el} \\ -\sin\alpha_{az} & \cos\alpha_{az} & 0 \\ \sin\alpha_{el}\cos\alpha_{az} & \sin\alpha_{el}\sin\alpha_{az} & \cos\alpha_{el} \end{pmatrix}. \tag{13.3}$$

The literature in computer vision and image processing traditionally aligns the coordinate axis of the camera such that $\hat{i}^c$ points to the right in the image, $\hat{j}^c$ points down in the image, and $\hat{k}^c$ points along the optical axis. Therefore, the transformation from the gimbal frame to the camera frame is given by

$$R_g^c = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}. \tag{13.4}$$

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 is M and the field-of-view of the camera η is known, then the focal length f can be expressed as

$$f = \frac{M}{2\tan\left(\frac{\eta}{2}\right)}. \tag{13.5}$$

The location of the object in the camera frame is given by $(P\epsilon_x, P\epsilon_y, Pf)$, where ϵ_x and ϵ_y are the pixel location (in units of pixels) of the object. The distance from the origin of the camera frame to the pixel location (ϵ_x, ϵ_y), as shown in Figure 13.2, is PF where

$$F = \sqrt{f^2 + \epsilon_x^2 + \epsilon_y^2}. \tag{13.6}$$

Using similar triangles in Figure 13.2 we get

$$\frac{\ell_x^c}{L} = \frac{P\epsilon_x}{PF} = \frac{\epsilon_x}{F}. \tag{13.7}$$

Similarly we get that $\ell_y^c/L = \epsilon_y/F$ and $\ell_z^c/L = f/F$. Therefore

$$\ell^c = \frac{L}{F}\begin{pmatrix} \epsilon_x \\ \epsilon_y \\ f \end{pmatrix}. \tag{13.8}$$

Note that ℓ^c cannot be determined strictly from camera data since $L = \|\ell^c\|$ is unknown. However, we can determine the unit direction vector to the target as

$$\hat{\ell}^c \triangleq \begin{pmatrix} \hat{\ell}_x^c \\ \hat{\ell}_y^c \\ \hat{\ell}_z^c \end{pmatrix} \triangleq \frac{\ell^c}{L} = \frac{1}{F}\begin{pmatrix} \epsilon_x \\ \epsilon_y \\ f \end{pmatrix}. \tag{13.9}$$

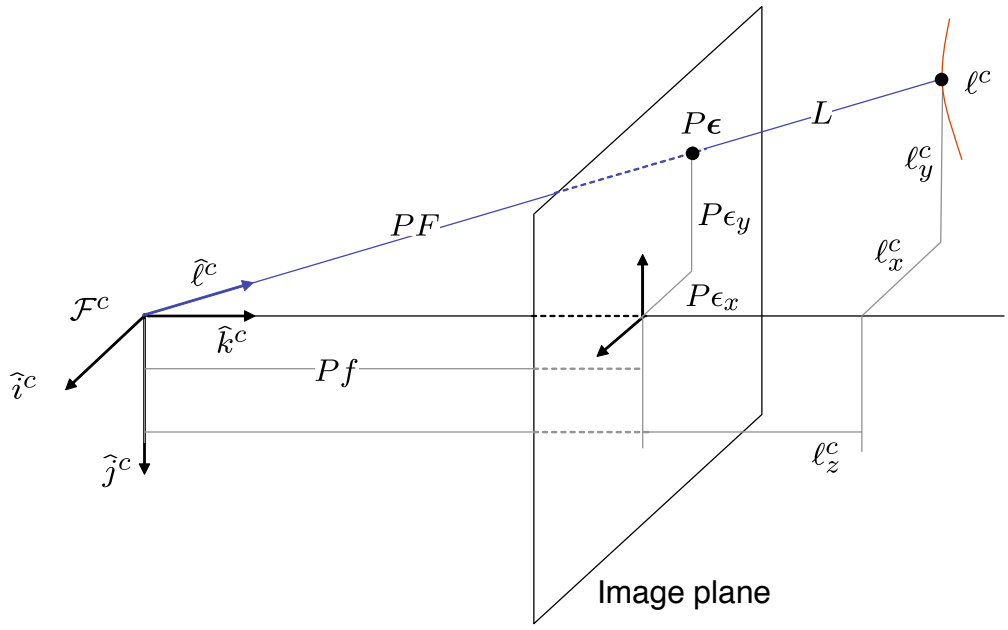


Figure 13.2: The camera frame. The target in the camera frame is represented by ℓ^c. The projection of the target onto the image plane is represented by $\boldsymbol{\epsilon}$. The zero pixel location corresponds to the center of the image which is assumed to be aligned with the optical axis. The distance to the target is given by L. $\boldsymbol{\epsilon}$ and f are in units of pixels. ℓ is in units of meters.

13.2 Gimbal Pointing

Small and micro 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 are given by

$$\dot{\alpha}_{az} = u_{az}$$
$$\dot{\alpha}_{el} = u_{el},$$

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 p^i_{target} be the known location of the target in the inertial frame. The objective is to align the optical axis of the camera with the relative position vector

$$\left(\ell^i\right)^d \triangleq p^i_{target} - p^i_{MAV},$$

where $p^i_{MAV} = (p_n, p_e, p_d)^T$ is the inertial position of the MAV, and where the superscript d indicates a desired quantity. The body frame unit vector that points in the desired direction of the target is given by

$$\left(\hat{\ell}^b\right)^d = \frac{1}{\left\|(\ell^i)^d\right\|} R^b_i \left(\ell^i\right)^d.$$

For the second scenario, suppose that we desire to maneuver the gimbal so that the pixel location $\boldsymbol{\epsilon}$ is pushed to the center of the image. Using (13.9), the desired direction of the optical axis in the camera frame, is given by

$$\left(\hat{\ell}^c\right)^d = \frac{1}{\sqrt{f^2 + \epsilon_x^2 + \epsilon_y^2}} \begin{pmatrix} \epsilon_x \\ \epsilon_y \\ f \end{pmatrix}.$$

In the body frame, the desired direction of the optical axis is

$$\left(\hat{\ell}^b\right)^d = R^b_g R^g_c \left(\hat{\ell}^c\right)^d.$$

The next step is to determine the desired azimuth and elevation angles that will align the optical axis with $\left(\hat{\ell}^b\right)^d$. In the camera frame, the optical axis is given by $(0,0,1)^c$. Therefore, the objective is to select the desired gimbal angles α^d_{az} and α^d_{el} so that

$$\left(\hat{\ell}^b\right)^d \triangleq \begin{pmatrix} \left(\hat{\ell}^b_x\right)^d \\ \left(\hat{\ell}^b_y\right)^d \\ \left(\hat{\ell}^b_z\right)^d \end{pmatrix} = R^b_g(\alpha^d_{az}, \alpha^d_{el}) R^g_c \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{13.10}$$

$$= \begin{pmatrix} \cos\alpha^d_{el}\cos\alpha^d_{az} & -\sin\alpha^d_{el} & -\sin\alpha^d_{el}\cos\alpha^d_{az} \\ \cos\alpha^d_{el}\sin\alpha^d_{az} & \cos\alpha^d_{az} & -\sin\alpha^d_{el}\sin\alpha^d_{az} \\ \sin\alpha^d_{el} & 0 & \cos\alpha^d_{el} \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{13.11}$$

$$= \begin{pmatrix} \cos\alpha^d_{el}\cos\alpha^d_{az} \\ \cos\alpha^d_{el}\sin\alpha^d_{az} \\ \sin\alpha^d_{el} \end{pmatrix}. \tag{13.12}$$

Therefore the desired azimuth and elevation angles are given by

$$\alpha_{az}^d = \tan^{-1}\left(\frac{\left(\hat{\ell}_y^b\right)^d}{\left(\hat{\ell}_x^b\right)^d}\right) \tag{13.13}$$

$$\alpha_{el}^d = \sin^{-1}\left(\left(\hat{\ell}_z^b\right)^d\right). \tag{13.14}$$

The gimbal servo commands can be selected as

$$\begin{aligned} u_{az} &= k_{az}(\alpha_{az}^d - \alpha_{az}) \\ u_{el} &= k_{el}(\alpha_{el}^d - \alpha_{el}), \end{aligned} \tag{13.15}$$

where k_{az} and k_{el} are positive control gains.

13.3 Geolocation

This section presents a method for determining the location of objects in world/inertial coordinates using a gimbaled EO/IR camera on-board a fixed-wing MAV. We assume that the MAV can measure its own world coordinates using, for example, a GPS receiver, and that other MAV state variables are also available.

Following Section 13.1, let $\ell = p_{obj} - p_{MAV}$ be the relative position vector between the target of interest and the MAV, and define $L = \|\ell\|$ and $\hat{\ell} = \ell/L$. Then from geometry we have the relationship

$$\begin{aligned} p_{obj}^i &= p_{MAV}^i + R_b^i R_g^b R_c^g \ell^c \\ &= p_{MAV}^i + L\left(R_b^i R_g^b R_c^g \hat{\ell}^c\right), \end{aligned} \tag{13.16}$$

where $p_{MAV}^i = (p_n, p_e, p_d)^T$, $R_b^i = R_b^i(\phi, \theta, \psi)$ and $R_g^b = R_g^b(\alpha_{az}, \alpha_{el})$. The only element on the right hand side of Eq. (13.16) that is unknown is L. Therefore, solving the geolocation problem reduces to the problem of estimating the range to the target 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 L is to assume a flat earth model [74]. Figure 13.3 shows the geometry of the situation, where $h = -p_d$ is the height-above-ground, and φ is the angle between ℓ and the $\hat{k}^i$ axis. It is clear from Figure 13.3

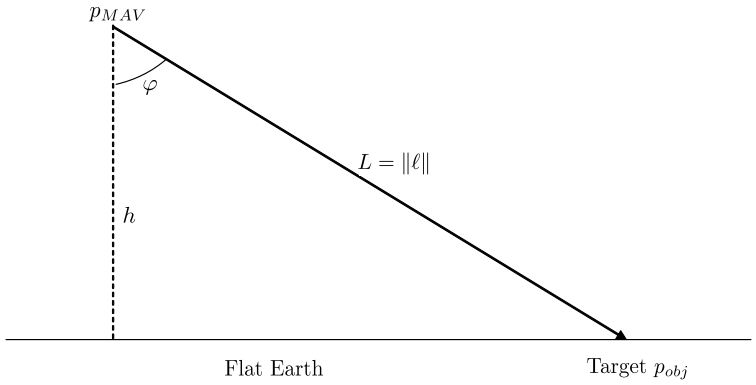


Figure 13.3: Range estimation using the flat earth assumption.

that

$$L = \frac{h}{\cos\varphi},$$

where

$$\begin{aligned}\cos\varphi &= \hat{k}^i \cdot \hat{\ell}^i \\ &= \hat{k}^i \cdot R_b^i R_g^b R_c^g \hat{\ell}^c.\end{aligned}$$

Therefore, the range estimate using the flat earth model is given by

$$L = \frac{h}{\hat{k}^i \cdot R_b^i R_g^b R_c^g \hat{\ell}^c}. \tag{13.17}$$

The geolocation estimate is given by combining Eqs. (13.16) and (13.17) to produce

$$p_{obj}^i = \begin{pmatrix} p_n \\ p_e \\ p_d \end{pmatrix} + h \frac{R_b^i R_g^b R_c^g \hat{\ell}^c}{\hat{k}^i \cdot R_b^i R_g^b R_c^g \hat{\ell}^c}. \tag{13.18}$$

13.3.2 Geo-location using an extended Kalman filter

The geo-location estimate in Eq. (13.18) provides a one-shot estimate of the target location. Unfortunately, this equation is highly sensitive to measurement errors, especially attitude errors. In this section we will describe the use of the extended Kalman filter (EKF) to solve the geo-location problem.

Rearranging to Equation (13.16) we get

$$p_{MAV}^i = p_{obj}^i - L\left(R_b^i R_g^b R_c^g \hat{\ell}^c\right), \tag{13.19}$$

which, since p^i_{MAV} 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 geo-location error will also contain a bias.

If we assume that the object is stationary, we have

$$\dot{p}^i_{obj} = 0.$$

Since $L = \left\| p^i_{obj} - p^i_{MAV} \right\|$, we have

$$\begin{aligned}
\dot{L} &= \frac{d}{dt}\sqrt{(p^i_{obj} - p^i_{MAV})^T (p^i_{obj} - p^i_{MAV})} \\
&= \frac{(p^i_{obj} - p^i_{MAV})^T (\dot{p}^i_{obj} - \dot{p}^i_{MAV})}{L} \\
&= -\frac{(p^i_{obj} - p^i_{MAV})^T \dot{p}^i_{MAV}}{L},
\end{aligned}$$

where $\dot{p}_{MAV}$ is provided by the GPS sensor.

A block diagram of the geo-location algorithm is shown in Figure 13.4. The input to the geo-location algorithm is the position and the velocity of the MAV in the inertial frame as estimated by the GPS smoother described in Section 9.7, the estimate of the normalized line of sight vector as given in Equation (13.9), and the estimated attitude as estimated by the scheme described in Section **??**.

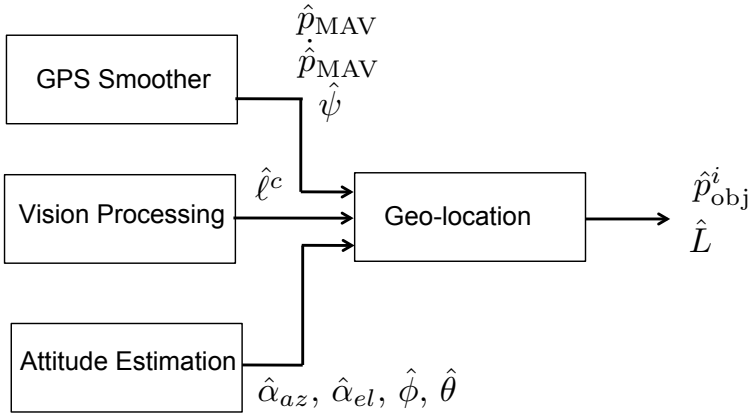


Figure 13.4: The geo-location algorithm uses the output of the GPS smoother, the normalized line-of-sight vector from the vision algorithm, and the attitude, to estimate the position of the object in the inertial frame and the distance to the object.

The geo-location algorithm is an extended Kalman filter with state $\hat{x} = (\hat{p}^i_{\rm obj}, \hat{L})$ and prediction

equations given by

$$\begin{pmatrix} \hat{p}^i_{\text{obj}} \\ \hat{L} \end{pmatrix} = \begin{pmatrix} 0 \\ -\frac{(\hat{p}^i_{obj} - \hat{p}^i_{MAV})^T \dot{\hat{p}}^i_{MAV}}{\hat{L}} \end{pmatrix}.$$

The Jacobian is therefore given by

$$\frac{\partial f}{\partial x} = \begin{pmatrix} 0 & 0 \\ \text{diag}\left(\frac{\dot{\hat{p}}^i_{MAV}}{\hat{L}}\right) & \frac{(\hat{p}^i_{obj} - \hat{p}^i_{MAV})^T \dot{\hat{p}}_{MAV}}{\hat{L}^2} \end{pmatrix}.$$

The output equation is given by Equation (13.19), where the Jacobian of the output equation is

$$\frac{\partial h}{\partial x} = \begin{pmatrix} I & R^i_b R^b_g R^g_c \hat{\ell}^c \end{pmatrix}.$$

RWB: Add simulation results.

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 often noisy. The guidance algorithm described in the next section needs both the pixel location of the target, as well as 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.

In the remainder of this chapter, we will need to be careful about which frame we are taking derivatives. We will use an overhead dot to denote differentiation with respect to the inertial frame. We will use an overhead prime to denote differentiation with respect to the camera frame.

13.4.1 Digital low pass filter and differentiation

Let $\hat{\boldsymbol{\epsilon}} = (\hat{\epsilon}_x, \hat{\epsilon}_y)^T$ denote the raw pixel measurements, and $\boldsymbol{\epsilon} = (\epsilon_x, \epsilon_y)^T$ denote the filtered pixel location, and $\boldsymbol{\epsilon}' = (\epsilon'_x, \epsilon'_y)^T$ denote the filtered pixel velocity as observed in the camera frame. The basic idea is to low pass filter the raw pixel measurements as

$$\boldsymbol{\epsilon}(s) = \frac{1}{\tau s + 1} \hat{\boldsymbol{\epsilon}}. \tag{13.20}$$

The pixel rates are obtained via a bandwidth limited differentiator as

$$\acute{\boldsymbol{\epsilon}}(s) = \frac{s}{\tau s + 1}\hat{\boldsymbol{\epsilon}}. \tag{13.21}$$

In the time domain, Equation (13.20) can be written as

$$\boldsymbol{\epsilon}' = -\frac{1}{\tau}\boldsymbol{\epsilon} + \frac{1}{\tau}\hat{\boldsymbol{\epsilon}}. \tag{13.22}$$

Solving this differential equation gives

$$\boldsymbol{\epsilon}(t) = e^{-(t-t_0)/\tau}\boldsymbol{\epsilon}(t_0) + \frac{1}{\tau}\int_{t_0}^{t} e^{-(t-\sigma)/\tau}\hat{\boldsymbol{\epsilon}}(\sigma)\, d\sigma.$$

Therefore, if $\boldsymbol{\epsilon}[k]$ denotes the filtered pixel location at the k^{th} sample, and T is the sample period, then

$$\boldsymbol{\epsilon}[k] = e^{-T/\tau}\boldsymbol{\epsilon}[k-1] + (1 - e^{-T/\tau})\hat{\boldsymbol{\epsilon}}[k]. \tag{13.23}$$

In the time domain, Equation (13.21) can be written as

$$\boldsymbol{\epsilon}'' = \frac{1}{\tau}(\hat{\boldsymbol{\epsilon}}' - \boldsymbol{\epsilon}').$$

Integrating once gives

$$\boldsymbol{\epsilon}' = \frac{1}{\tau}(\hat{\boldsymbol{\epsilon}} - \int \boldsymbol{\epsilon}'). \tag{13.24}$$

Defining $\mathbf{x} \triangleq \int \boldsymbol{\epsilon}'$ and differentiating with respect to time gives

$$\begin{aligned}\mathbf{x}' &= \boldsymbol{\epsilon}' \\ &= \frac{1}{\tau}(\hat{\boldsymbol{\epsilon}} - \mathbf{x}) \\ &= -\frac{1}{\tau}\mathbf{x} + \frac{1}{\tau}\hat{\boldsymbol{\epsilon}},\end{aligned}$$

which is identical to Equation (13.22). Therefore, the solution $\mathbf{x}[k]$ is given by Equation (13.23). Combining Equation (13.23) and (13.24) we get

$$\boldsymbol{\epsilon}[k] = e^{-T/\tau}\boldsymbol{\epsilon}[k-1] + (1 - e^{-T/\tau})\hat{\boldsymbol{\epsilon}}[k] \tag{13.25}$$

$$\boldsymbol{\epsilon}'[k] = \frac{1}{\tau}(\hat{\boldsymbol{\epsilon}} - \boldsymbol{\epsilon}[k]), \tag{13.26}$$

where $\boldsymbol{\epsilon}[0] = \hat{\boldsymbol{\epsilon}}[0]$ and $\boldsymbol{\epsilon}'[0] = 0$.

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 $\hat{\ell} = \ell/L = (p_{obj} - p_{MAV})/\|p_{obj} - p_{MAV}\|$ be the normalized relative position vector between the target and the MAV. Using the Coriolis formula (2.13) we get

$$\frac{d\hat{\ell}}{dt_i} = \frac{d\hat{\ell}}{dt_c} + \omega_{c/i} \times \hat{\ell}. \tag{13.27}$$

The expression on the left-hand-side of (13.27) is the true relative translation motion between the target and the MAV. The first expression on the right-hand-side of (13.27) 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 (13.27) is the apparent motion due to the rotation of the MAV and gimbal platform. Equation (13.27) can be expressed in the camera frame as

$$\frac{d\hat{\ell}^c}{dt_i} = \frac{d\hat{\ell}^c}{dt_c} + \omega^c_{c/i} \times \hat{\ell}^c. \tag{13.28}$$

The first expression on the right-hand-side of (13.28) can be computed as

$$\begin{aligned}
\frac{d\hat{\ell}^c}{dt_c} &= \frac{d}{dt_c}\frac{\begin{pmatrix}\epsilon_x\\ \epsilon_y\\ f\end{pmatrix}}{F} = \frac{F\begin{pmatrix}\epsilon'_x\\ \epsilon'_y\\ 0\end{pmatrix} - F'\begin{pmatrix}\epsilon_x\\ \epsilon_y\\ f\end{pmatrix}}{F^2} = \frac{F\begin{pmatrix}\epsilon'_x\\ \epsilon'_y\\ 0\end{pmatrix} - \frac{\epsilon_x\epsilon'_x+\epsilon_y\epsilon'_y}{F}\begin{pmatrix}\epsilon_x\\ \epsilon_y\\ f\end{pmatrix}}{F^2} \\
&= \frac{1}{F^3}\begin{pmatrix}F^2-\epsilon_x^2 & -\epsilon_x\epsilon_y\\ -\epsilon_x\epsilon_y & F^2-\epsilon_y^2\\ -\epsilon_x f & -\epsilon_y f\end{pmatrix}\begin{pmatrix}\epsilon'_x\\ \epsilon'_y\end{pmatrix} = \frac{1}{F^3}\begin{pmatrix}\epsilon_y^2+f^2 & -\epsilon_x\epsilon_y\\ -\epsilon_x\epsilon_y & \epsilon_x^2+f^2\\ -\epsilon_x f & -\epsilon_y f\end{pmatrix}\boldsymbol{\epsilon}' \\
&\triangleq Z(\boldsymbol{\epsilon})\boldsymbol{\epsilon}'.
\end{aligned} \tag{13.29}$$

To compute the second term on the right-hand-side of (13.28) we need an expression for $\omega^c_{c/i}$ which can be decomposed as

$$\omega^c_{c/i} = \omega^c_{c/g} + \omega^c_{g/b} + \omega^c_{b/i}. \tag{13.30}$$

Since the camera is fixed in the gimbal frame, we have that $\omega^c_{c/g} = 0$. Letting p, q, and r denote the angular body rates of the platform as measured by on-board rate gyros that are aligned with the

body frame axes, gives $\omega^b_{b/i} = (p, q, r)^T$. Therefore

$$\omega^c_{b/i} = R^c_g R^g_b \omega^b_{b/i} = R^c_g R^g_b \begin{pmatrix} p \\ q \\ r \end{pmatrix}. \tag{13.31}$$

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 α_{az} is defined with respect to the body frame, and that the elevation angle α_{el} is defined with respect to the gimbal-1 frame. The gimbal frame is obtained by rotating the gimbal-1 frame about its y-axis by α_{el}. Therefore $\dot{\alpha}_{az}$ is defined with respect to the gimbal-1 frame, and $\dot{\alpha}_{el}$ is defined with respect to the gimbal frame. This implies that

$$\omega^b_{g/b} = R^b_{g1}(\alpha_{az}) R^{g1}_g(\alpha_{el}) \begin{pmatrix} 0 \\ \dot{\alpha}_{el} \\ 0 \end{pmatrix}^g + R^b_{g1}(\alpha_{az}) \begin{pmatrix} 0 \\ 0 \\ \dot{\alpha}_{az} \end{pmatrix}^{g1}.$$

Noting that R^{g1}_g is a y-axis rotation we get

$$\omega^b_{g/b} = R^b_{g1}(\alpha_{az}) \begin{pmatrix} 0 \\ \dot{\alpha}_{el} \\ \dot{\alpha}_{az} \end{pmatrix} = \begin{pmatrix} \cos\alpha_{az} & -\sin\alpha_{az} & 0 \\ \sin\alpha_{az} & \cos\alpha_{az} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ \dot{\alpha}_{el} \\ \dot{\alpha}_{az} \end{pmatrix} = \begin{pmatrix} -\sin(\alpha_{az})\dot{\alpha}_{el} \\ \cos(\alpha_{az})\dot{\alpha}_{el} \\ \dot{\alpha}_{az} \end{pmatrix}$$

Therefore,

$$\omega^c_{g/b} = R^c_g R^g_b \omega^b_{g/b} = R^c_g R^g_b \begin{pmatrix} -\sin(\alpha_{az})\dot{\alpha}_{el} \\ \cos(\alpha_{az})\dot{\alpha}_{el} \\ \dot{\alpha}_{az} \end{pmatrix}. \tag{13.32}$$

Equation (13.28) can therefore be expressed as

$$\frac{d\hat{\ell}^c}{dt_i} = Z(\boldsymbol{\epsilon})\dot{\boldsymbol{\epsilon}} + \dot{\boldsymbol{\epsilon}}^{app}, \tag{13.33}$$

where

$$\dot{\boldsymbol{\epsilon}}^{app} \triangleq \frac{1}{F} \left[R^c_g R^g_b \begin{pmatrix} p - \sin(\alpha_{az})\dot{\alpha}_{el} \\ q + \cos(\alpha_{az})\dot{\alpha}_{el} \\ r + \dot{\alpha}_{az} \end{pmatrix} \right] \times \begin{pmatrix} \epsilon_x \\ \epsilon_y \\ f \end{pmatrix} \tag{13.34}$$

is the apparent motion on the image plane due to rotation of the gimbal and body.

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 L is the length of the line-of-sight vector between the MAV and an object, then the time-to-collision is given by

$$t_c \triangleq \frac{L}{\dot{L}}.$$

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 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 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

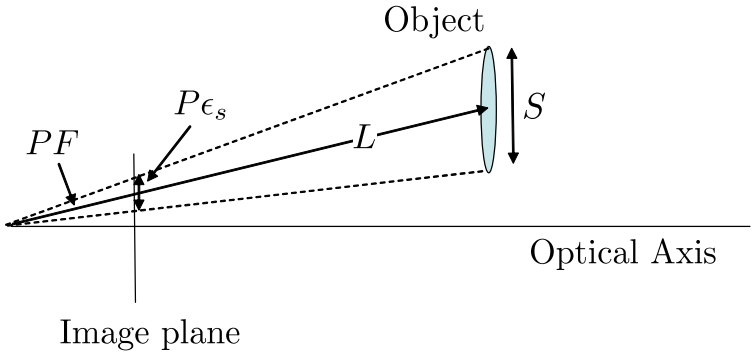


Figure 13.5: The size and growth of the target in the image frame can be used to estimate the time-to-collision.

the relationship

$$\frac{S}{L} = \frac{P\epsilon_s}{PF} = \frac{\epsilon_s}{F}, \tag{13.35}$$

where the size of the target in meters is S, and the size of the target in pixels is given by ϵ_s. We assume that the size of the target S is not changing in time. Differentiating Eq. (13.35) and solving

for $\dot{L}/L$ we obtain

$$\begin{aligned}\frac{\dot{L}}{L} &= \frac{L}{S}\left[\frac{\epsilon_s}{F}\frac{\dot{F}}{F} - \frac{\dot{\epsilon}_s}{F}\right] \\ &= \frac{F}{\epsilon_s}\left[\frac{\epsilon_s}{F}\frac{\dot{F}}{F} - \frac{\dot{\epsilon}_s}{F}\right] \\ &= \frac{\dot{F}}{F} - \frac{\dot{\epsilon}_s}{\epsilon_s}.\end{aligned} \tag{13.36}$$

Since ϵ_s and F are scalars, differentiation in the inertial frame is equal to differentiation in the camera frame. Therefore,

$$\begin{aligned}\frac{\dot{L}}{L} &= \frac{\acute{F}}{F} - \frac{\acute{\epsilon}_s}{\epsilon_s} \\ &= \frac{\epsilon_x\acute{\epsilon}_x + \epsilon_y\acute{\epsilon}_y}{F} - \frac{\acute{\epsilon}_s}{\epsilon_s}.\end{aligned} \tag{13.37}$$

13.5.2 Computing time-to-collision from flat earth model

A popular computer vision algorithm for target tracking is to track features on the target [75]. 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

$$L = \frac{h}{\cos\varphi},$$

where $h = -p_d$ is the altitude. Differentiating in the inertial frame gives

$$\begin{aligned}\frac{\dot{L}}{L} &= \frac{1}{L}\left(\frac{\cos\varphi\dot{h} + h\dot{\varphi}\sin\varphi}{\cos^2\varphi}\right) \\ &= \frac{\cos\varphi}{h}\left(\frac{\cos\varphi\dot{h} + h\dot{\varphi}\sin\varphi}{\cos^2\varphi}\right) \\ &= \frac{\dot{h}}{h} + \dot{\varphi}\tan\varphi.\end{aligned}$$

In the inertial frame we have that

$$\cos\varphi = \hat{\ell}^i \cdot \hat{k}, \tag{13.38}$$

where $\hat{k} = (0, 0, 1)^T$. Therefore

$$\cos\varphi = \hat{\ell}^i_z. \tag{13.39}$$

Differentiating Equation (13.39) and solving for $\dot{\varphi}$ gives

$$\dot{\varphi} = -\frac{1}{\sin\varphi}\frac{d}{dt_i}\hat{\ell}^i_z. \tag{13.40}$$

Therefore

$$\dot{\varphi}\tan\varphi = -\frac{1}{\cos\varphi}\frac{d}{dt_i}\hat{\ell}^i_z = -\frac{1}{\hat{\ell}^i_z}\frac{d}{dt_i}\hat{\ell}^i_z, \tag{13.41}$$

where $d\hat{\ell}^i_z/dt_i$ can be determined by rotating Eq. (13.33) into the inertial frame.

RWB: Add some simulation results using noising camera information. Compare the two methods with truth data.

13.6 Precision Landing

Our objective in this section is to use the camera to guide the MAV to precisely land 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 since its discovery [76]. In this section, we present a method for implementing a 3D 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-site (LOS) rates multiplied by the closing velocity. PN is often implemented as two 2D 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 3D 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

$$\dot{p}_n = V\cos\chi\cos\gamma \tag{13.42}$$

$$\dot{p}_e = V\sin\chi\cos\gamma \tag{13.43}$$

$$\dot{p}_d = -V\sin\gamma \tag{13.44}$$

$$\dot{\chi} = \frac{g}{V}\tan\phi \tag{13.45}$$

$$\dot{\phi} = u_1 \tag{13.46}$$

$$\dot{\gamma} = u_2, \tag{13.47}$$

where (p_n, p_e, p_d) are the North-East-Down position of the MAV, V is the airspeed (assumed constant), χ is the course angle, γ is the flight path angle, ϕ is the roll angle, g is the gravitational constant, and u_1, and u_2 are control variables, and where we have assumed zero wind conditions.

The objective of this section is to design a vision based guidance law that causes a MAV to intercept a potentially moving, ground based target. The position and velocity vector of the target are given by r_T and v_T respectively. Similarly, let r_I and v_I denote the position and velocity of the MAV interceptor.

In the inertial frame, the position and velocity of the MAV can be expressed as

$$r_I^i = \begin{pmatrix} p_n, & p_e, & p_d \end{pmatrix}^T,$$
$$v_I^i = \begin{pmatrix} V\cos\chi\cos\gamma, & V\sin\chi\cos\gamma, & -V\sin\gamma \end{pmatrix}^T.$$

However, in the vehicle-2 frame, the velocity vector of the MAV is given by

$$v_I^{v_2} = \begin{pmatrix} V, & 0, & 0 \end{pmatrix}.$$

Define $\ell = r_T - r_I$ and $\dot{\ell} = \dot{r}_T - \dot{r}_I$, and let $L = \|\ell\|$, then the 3D equations for pure proportional navigation (PPNG) are given in [77] as

$$a_I = N\Omega_\perp \times \mu v_I, \tag{13.48}$$

where

$$\Omega_\perp = \frac{\ell}{L} \times \frac{\dot{\ell}}{L},$$
$$\mu = \frac{\left\|\dot{\ell}\right\|}{\|v_I\|},$$

and where a_I is the commanded acceleration vector for the MAV. Furthermore, following the previous sections by defining

$$\hat{\ell} \triangleq \frac{\ell}{L} \tag{13.49}$$
$$\dot{\hat{\ell}} \triangleq \frac{\dot{\ell}}{L}, \tag{13.50}$$

then $\Omega_\perp$ can be expressed as

$$\Omega_\perp = \hat{\ell} \times \dot{\hat{\ell}}. \tag{13.51}$$

The term $\mu = \left\|\dot{\ell}\right\| / \|v_I\|$ is the normalized closing velocity and, unfortunately, cannot be determined directly by a passive sensor. We assume that the velocity of the target is slow compared to the velocity of the interceptor; therefore $\mu \approx 1$.

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 a_I must be resolved in the vehicle-2 frame as

$$\begin{aligned} a_I^{v2} &= \mu N \Omega_\perp^{v2} \times V_I^{v2} \\ &= \mu N \begin{pmatrix} \Omega_{\perp,x}^{v2} \\ \Omega_{\perp,y}^{v2} \\ \Omega_{\perp,z}^{v2} \end{pmatrix} \times \begin{pmatrix} V \\ 0 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ \mu N V \Omega_{\perp,z}^{v2} \\ -\mu N V \Omega_{\perp,y}^{v2} \end{pmatrix}. \end{aligned} \tag{13.52}$$

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 $\Omega_\perp = \hat{\ell} \times \dot{\hat{\ell}}$ 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

$$\Omega_\perp^{v2} = R_b^{v2} R_g^b R_c^g \Omega_\perp^c. \tag{13.53}$$

The normalized line-of-sight vector $\hat{\ell}^c$ can be estimated directly from camera data using Equation (13.9). Differentiating $\hat{\ell}^c = \ell^c / L$ gives

$$\frac{d\hat{\ell}^c}{dt_i} = \frac{d}{dt_i}\frac{\ell^c}{L} = \frac{L\dot{\ell}^c - \dot{L}\ell^c}{L^2} = \frac{\dot{\ell}^c}{L} - \frac{\dot{L}}{L}\hat{\ell}^c, \tag{13.54}$$

which, when combined with Equation (13.33), results in the expression

$$\frac{\dot{\ell}^c}{L} = \frac{\dot{L}}{L}\hat{\ell}^c + Z(\boldsymbol{\epsilon})\dot{\boldsymbol{\epsilon}} + \frac{1}{F}\left[R_g^c R_b^g \begin{pmatrix} p - \sin(\alpha_{az})\dot{\alpha}_{el} \\ q + \cos(\alpha_{az})\dot{\alpha}_{el} \\ r + \dot{\alpha}_{az} \end{pmatrix} \right] \times \begin{pmatrix} \epsilon_x \\ \epsilon_y \\ f \end{pmatrix}, \tag{13.55}$$

where the inverse of time-to-collision $\dot{L}/L$ can be estimated using one of the techniques discussed in Section 13.5.

Equation (13.52) gives an acceleration command in the vehicle-2 frame. In this section, we will describe how the acceleration command is converted into a roll command and a pitch rate command. The standard approach is to use polar control logic [78], which is shown in Figure 13.6. From Figure 13.6 it is clear that for $a_z^{v2} < 0$, we have

$$\begin{aligned} {}^c\phi &= \tan^{-1}\left(\frac{a_y^{v2}}{-a_z^{v2}}\right) \\ V\,{}^c\dot{\gamma} &= \sqrt{(a_y^{v2})^2 + (a_z^{v2})^2}. \end{aligned}$$

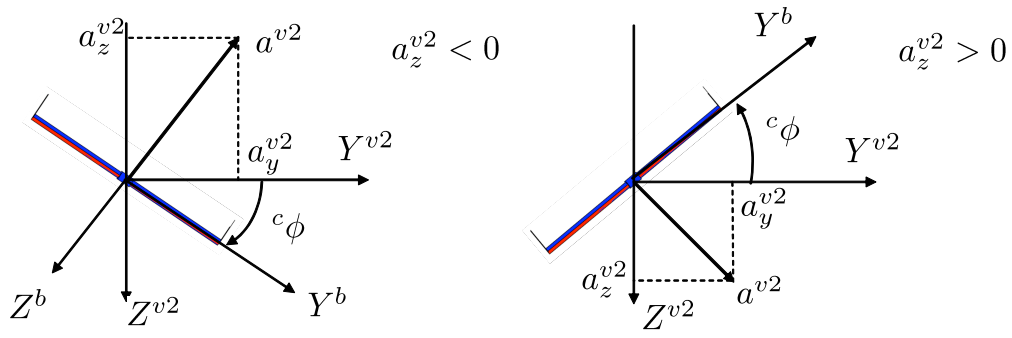


Figure 13.6: Polar converting logic that transforms an acceleration command a^{v2} to a commanded roll angle $^c\phi$ and a commanded normal acceleration $V\,^c\dot{\gamma}$.

Similarly, when $a_z^{v2} > 0$ we have

$$
\begin{aligned}
^c\phi &= \tan^{-1}\left(\frac{a_y^{v2}}{a_z^{v2}}\right) \\
V\,^c\dot{\gamma} &= -\sqrt{(a_y^{v2})^2 + (a_z^{v2})^2}.
\end{aligned}
$$

Therefore, the general rule is

$$
^c\phi = \tan^{-1}\left(\frac{a_y^{v2}}{|a_z^{v2}|}\right) \tag{13.56}
$$

$$
^c\dot{\gamma} = -\text{sign}(a_z^{v2})\frac{1}{V}\sqrt{(a_y^{v2})^2 + (a_z^{v2})^2}. \tag{13.57}
$$

Unfortunately, Equation (13.56) has a discontinuity at $(a_y^{v2}, a_z^{v2}) = (0, 0)$. For example, when $a_z^{v2} = 0$ the commanded roll angle is $^c\phi = \pi/2$ when $a_y^{v2} > 0$ and $^c\phi = -\pi/2$ when $a_y^{v2} < 0$. The discontinuity can be removed by multiplying Equation (13.56) by the signed-sigmoidal function

$$
\sigma(a_y^{v2}) = \text{sign}(a_y^{v2})\frac{1 - e^{-ka_y^{v2}}}{1 + e^{-ka_y^{v2}}}, \tag{13.58}
$$

where k is a positive control gain. A plot of $^c\phi(a_y^{v2}, a_z^{v2} = 0)$ with and without the sigmoidal function, is shown in Figure 13.7. The gain k adjusts the rate of the transition. The proportional navigation algorithm for precision landing, is summarized in Algorithm 18.

RWB: Add simulation results.

Algorithm 18 Vision Based Proportional Navigation: planCover$(\mathcal{T}, \mathcal{R}, \mathbf{p})$

Input: Input from camera: ϵ_x, ϵ_y, ϵ_s, $\dot{\epsilon}_x$, $\dot{\epsilon}_y$, $\dot{\epsilon}_s$. Input from gimbal: α_{az}, α_{el}. Input from IMU: V, $\dot{\phi}$, $\dot{\gamma}$, $\dot{\chi}$, p, q, r.

1: Remove apparent motion:

$$\begin{pmatrix} \dot{\epsilon}_x \\ \dot{\epsilon}_y \end{pmatrix} \leftarrow \begin{pmatrix} \dot{\epsilon}_x \\ \dot{\epsilon}_y \end{pmatrix} - \begin{pmatrix} \dot{\epsilon}_x^{\text{app}} \\ \dot{\epsilon}_y^{\text{app}} \end{pmatrix}$$

where $\dot{\epsilon}_x^{\text{app}}$ and $\dot{\epsilon}_y^{\text{app}}$ are given in (13.34).

2: Compute: $\hat{\ell}^{v2} = R_b^{v2} R_g^b R_c^g \hat{\ell}^c$ where $\hat{\ell}^c$ is given in Equation (13.9).

3: Compute: $\dot{\hat{\ell}}^{v2} = R_b^{v2} R_g^b R_c^g \dot{\hat{\ell}}^c$ where $\dot{\hat{\ell}}^c$ is given in Equation (13.55).

4: Compute: $\Omega_\perp^{v2} = \hat{\ell}^{v2} \times \dot{\hat{\ell}}^{v2}$.

5: Compute: the commanded roll angle and flight path rate

$$^c\phi = \sigma(\mu N \Omega_{\perp,z}^{v2}) \tan^{-1}\left(\frac{\Omega_{\perp,z}^{v2}}{\left|\Omega_{\perp,y}^{v2}\right|}\right) \tag{13.59}$$

$$^c\dot{\gamma} = \text{sign}(\Omega_{\perp,y}^{v2}) \mu N \sqrt{(\Omega_{\perp,y}^{v2})^2 + (\Omega_{\perp,z}^{v2})^2}. \tag{13.60}$$

6: Compute: the commanded gimbal angles $^c\alpha_{az}$ and $^c\alpha_{el}$ from Equations (13.13) and (13.14) respectively.

7: **return** The autopilot commands

$$\begin{aligned} u_1 &= k_\phi(^c\phi - \phi) \\ u_2 &= ^c\dot{\gamma}, \\ u_{az} &= k_{\alpha_{az}}(^c\alpha_{az} - \alpha_{az}) \\ u_{el} &= k_{\alpha_{el}}(^c\alpha_{el} - \alpha_{el}). \end{aligned} \tag{13.61}$$

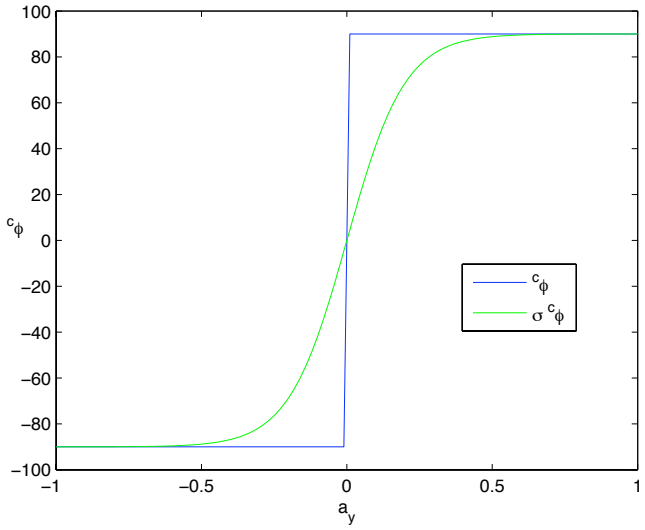


Figure 13.7: The commanded roll angle ${}^c\phi$ as a function of a_y^{v2} when $a_z^{v2} = 0$. The blue line is a plot of Equation (**??**) and the green line is a plot of Equation (**??**) multiplied by Equation (13.58) for $k = 10$.

13.7 Chapter Summary

Notes and References

Vision based guidance and control of MAVs is currently an active research topic (see for example [79, 80, 81, 82, 83, 84, 85, 74, 86, 87, 88, 89]). The gimbal pointing algorithm described in this chapter was presented [90]. Geolocation algorithms using small MAV are described in [74, 91, 92, 93, 84, 89]. The removal of apparent motion, or ego motion, in the image plane is discussed in [94, 95, 83]. Time-to-collision can be estimated using structure from motion [96], ground plane methods [97, 98], flow divergence [99], and insect inspired methods [100]. Section 13.6 is taken primarily from [101]. Proportional Navigation has been extensively analyzed in the literature. It has been shown to be optimal under certain conditions [102] and to produce zero miss distances for a constant target acceleration[103]. If rich information regarding the time-to-go is available, augmented proportional navigation [104] improves the performance by adding terms that account for target and pursuer accelerations. A three-dimensional expression of PN can be found in [77, 105].

13.8 Design Project

13.1 Homework problem 1.

Appendix A

Animations in 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.

A.1 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, and other objects that may be associated with the plot. If the plot contains drawings of several objects, then 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 only redrawing 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 A.2 we illustrate a 2D animation of an inverted pendulum using the fill command. In Section A.3 we illustrate a 3D animation of a spacecraft using lines to produce a stick figure. In Section A.4 we modify the spacecraft animation to use the vertices-faces data construction in Matlab.

A.2 Animation Example: Inverted Pendulum

Consider the image of the inverted pendulum shown in Figure A.1, where the configuration is completely specified by the position of the cart y, and the angle of the rod from vertical θ. 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 determine the

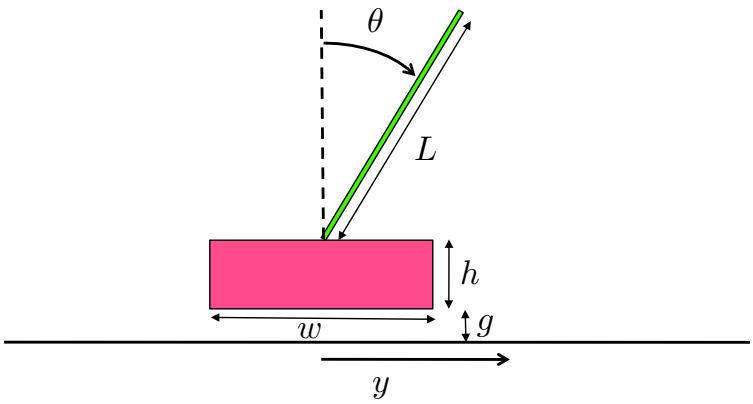


Figure A.1: Drawing for inverted pendulum. The first step in developing an animation is to draw a figure of the object to be animated and identify all of the physical parameters.

position of points that define the animation. For example, for the inverted pendulum in Figure A.1,

the four corners of the base are

$$(y + w/2, g), (y + w/2, g + h), (y - w/2, g + h), (y - w/2, g),$$

and the two ends of the rod are given by

$$(y, g + h), (y + L\sin\theta, g + h + L\cos\theta).$$

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:

```
1    function handle = drawBase(y, width, height, gap, handle, mode)
2      X = [y-width/2, y+width/2, y+width/2, y-width/2];
3      Y = [gap, gap, gap+height, gap+height];
4      if isempty(handle),
5        handle = fill(X,Y,'m','EraseMode', mode);
6      else
7        set(handle,'XData',X,'YData',Y);
8      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.

```
1    function handle = drawRod(y, theta, L, gap, height, handle, mode)
2      X = [y, y+L*sin(theta)];
3      Y = [gap+height, gap + height + L*cos(theta)];
4      if isempty(handle),
5        handle = plot(X, Y, 'g', 'EraseMode', mode);
6      else
7        set(handle,'XData',X,'YData',Y);
8      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.

The main routine for the pendulum animation is listed below.

```
1   function drawPendulum(u)
2       % process inputs to function
3       y        = u(1);
4       theta    = u(2);
5       t        = u(3);
6
7       % drawing parameters
8       L = 1;
9       gap = 0.01;
10      width = 1.0;
11      height = 0.1;
12
13      % define persistent variables
14      persistent base_handle
15      persistent rod_handle
16
17      % first time function is called, initialize plot and persistent vars
18      if t==0,
19          figure(1), clf
20          track_width=3;
21          plot([-track_width,track_width],[0,0],'k'); % plot track
22          hold on
23          base_handle = drawBase(y, width, height, gap, [], 'normal');
24          rod_handle  = drawRod(y, theta, L, gap, height, [], 'normal');
25          axis([-track_width, track_width, -L, 2*track_width-L]);
26
27      % at every other time step, redraw base and rod
28      else
29          drawBase(y, width, height, gap, base_handle);
30          drawRod(y, theta, L, gap, height, rod_handle);
31      end
```

The routine `drawPendulum` is called from the Simulink file shown in Figure A.2, where there are three inputs: the position y, the angle θ, and the time t. Lines 3-5 rename the inputs to y, θ, 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.

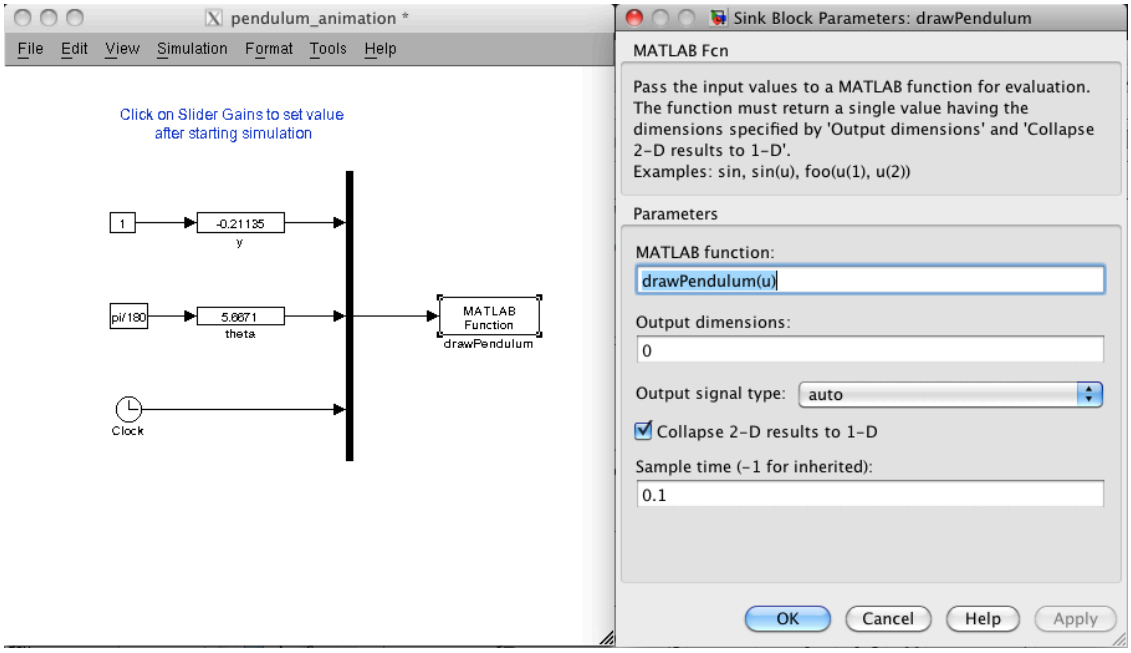


Figure A.2: Simulink file for debugging the pendulum simulation. There are three inputs to the matlab m-file `drawPendulum`: the position y, the angle θ, and the time t. Slider gains for y and θ are used to verify the animation.

A.3 Animation Example: Spacecraft using lines

The previous section described a simple 2D animation. In this section we discuss a 3D animation of a spacecraft with six degrees of freedom. Figure A.3 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.

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 A.3 and specific 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

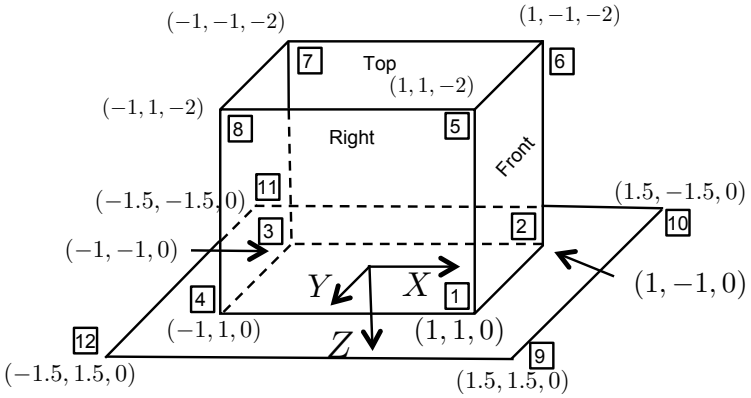


Figure A.3: Drawing used to create spacecraft animation. Standard aeronautics body axes are used, where the x-axis points out the front of the spacecraft, the y-axis points to the right, and the z-axis point out the bottom of the body.

segments. To do this as one continuous line, some of the segments will need to be repeated. To draw the spacecraft shown in Figure A.3 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 ϕ, θ, and ψ, 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.

```
1   function handle = drawSpacecraftBody(pn,pe,pd,phi,theta,psi, handle, mode)
2     % define points on spacecraft in local NED coordinates
3     NED = spacecraftPoints;
4     % rotate spacecraft by phi, theta, psi
5     NED = rotate(NED,phi,theta,psi);
6     % translate spacecraft to [pn; pe; pd]
7     NED = translate(NED,pn,pe,pd);
8      % transform vertices from NED to XYZ (for matlab rendering)
9      R = [...
10         0, 1, 0;...
11         1, 0, 0;...
12         0, 0, -1;...
13         ];
14     XYZ = R*NED;
15    % plot spacecraft
16    if isempty(handle),
17      handle = plot3(XYZ(1,:),XYZ(2,:),XYZ(3,:),'EraseMode', mode);
18    else
19      set(handle,'XData',XYZ(1,:),'YData',XYZ(2,:),'ZData',XYZ(3,:));
20      drawnow
21    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 viewers 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 accompanying CDROM. A rendering of the spacecraft is shown in Figure A.4.

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 A.5. The spacecraft points can be defined in the initialization window as shown in Figure A.5 and passed to the `drawSpacecraft` m-file as a parameter.

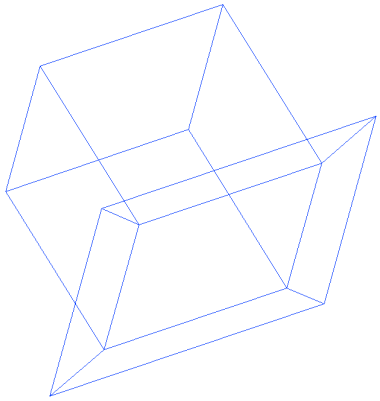


Figure A.4: Rendering of the spacecraft using lines and the `plot3` command.

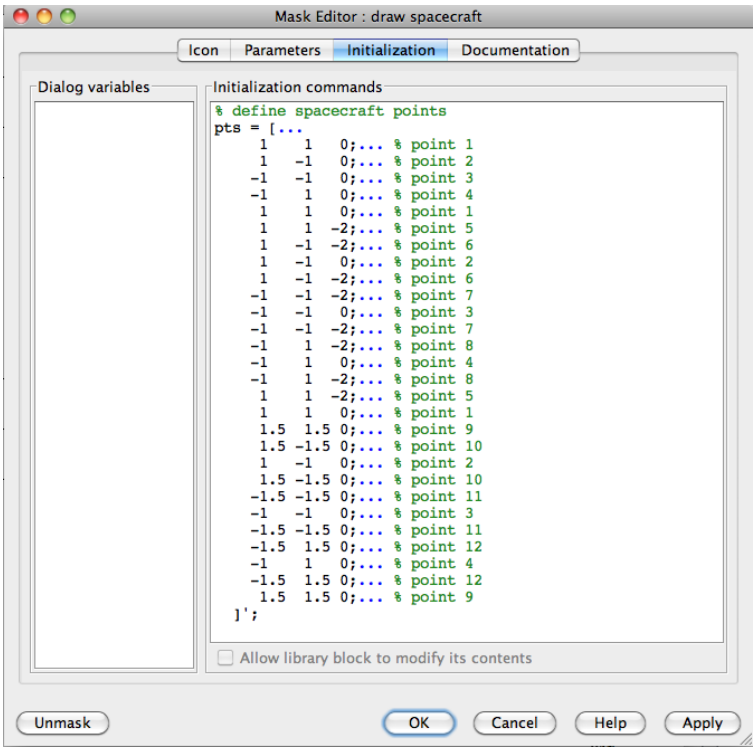


Figure A.5: The mask function in Simulink allows the spacecraft points to be initialized at the beginning of the simulation.

A.4 Animation Example: Spacecraft using vertices and faces

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

```
35          myblue;...    % right
36          myyellow;... % left
37          mycyan;...    % top
38          mycyan;...    % bottom
39          ];
```

The vertices are shown in Figure A.3 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.

```
1     function handle = drawSpacecraftBody(pn,pe,pd,phi,theta,psi, handle, mode)
2       [V, F, patchcolors] = spacecraftVFC;  % define points on spacecraft
3       V = rotate(V', phi, theta, psi)'; % rotate spacecraft
4       V = translate(V', pn, pe, pd)'; % translate spacecraft
5       R = [...
6           0, 1, 0;...
7           1, 0, 0;...
8           0, 0, -1;...
9           ];
10      V = V*R;  % transform vertices from NED to XYZ (for matlab rendering)
11      if isempty(handle),
12      handle = patch('Vertices', V, 'Faces', F,...
13                     'FaceVertexCData',patchcolors,...
14                     'FaceColor','flat',...
15                     'EraseMode', mode);
16      else
17        set(handle,'Vertices',V,'Faces',F);
18      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 A.6. Additional examples using the vertex-face format are on the accompanying CDROM.

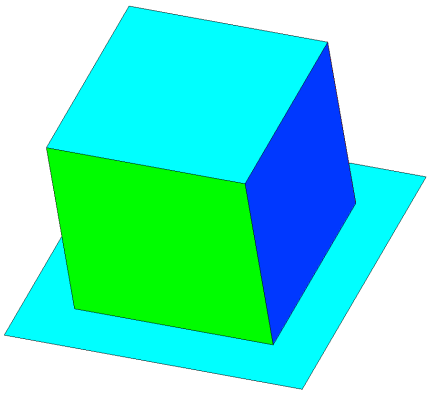


Figure A.6: Rendering of the spacecraft using vertices and faces.

Appendix B

Modeling in Simulink using S-functions

This chapter assumes basic familiarity with the Matlab/Simulink environment. For additional information, please consult the Matlab/Simulink documentation. Simulink is essentially, a sophisticated tool for solving interconnected hybrid ordinary differential equations and difference equations. Each block in Simulink is assumed to have the structure

$$\dot{x}_c = f(t, x_c, x_d, u); \qquad x_c(0) = x_{c0} \tag{B.1}$$

$$x_d[k+1] = g(t, x_c, x_d, u); \qquad x_d[0] = x_{d0} \tag{B.2}$$

$$y = h(t, x_c, x_d, u) \tag{B.3}$$

where $x_c \in \mathbb{R}^{n_c}$ is a continuous state with initial condition x_{c0}, $x_d \in \mathbb{R}^{n_d}$ is a discrete state with initial condition x_{d0}, $u \in \mathbb{R}^m$ is the input to the block, $y \in \mathbb{R}^p$ is the output of the block, and t is the elapsed simulation time. An *s-function* is a Simulink tool for explicitly defining the functions f, g, and h and the initial conditions x_{c0} and x_{d0}. As explained in the Matlab/Simulink documentation, there are a number of methods for specifying an s-function. In this Appendix, we will overview two different methods: the level-1 m-file s-function, and a C-file s-function. The C-file s-function is compiled into C-code and executes much faster than m-file s-functions. For this reason, we recommend that the blocks that model the MAV dynamics be implemented using a C-file s-function.

B.1 Example: Second Order Differential Equation

In this section we will show how to implement a system specified by the standard second order transfer function

$$Y(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} U(s) \tag{B.4}$$

using both a level-1 m-file s-function and a C-file s-function. The first step is either case, is to represent (B.4) in state space form. Using control canonical form [106] we have

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -2\zeta\omega_n & -\omega_n^2 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u \tag{B.5}$$

$$y = \begin{pmatrix} 0 & \omega_n^2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}. \tag{B.6}$$

B.1.1 Level-1 M-file S-function

The code listing for an m-file s-function that implements system (B.5) and (B.6) is shown below. Line 1 defines the main m-file function. The inputs to this function are always the elapsed time t, the state x, which is a concatenation of the continuous state and discrete state, the input u, a `flag`, followed by user defined input parameters, which in this case are ζ and ω_n. The Simulink engine calls the s-function and passes the parameters t, x, u, and `flag`. When `flag==0`, the Simulink engine expects the s-function to return the structure `sys` which defines the block, initial conditions `x0`, an empty string `str`, and an array `ts` that defines the sample times of the block. When `flag==1` the Simulink engine expect the s-function to return the function $f(t, x, u)$, when `flag==2` the Simulink engine expects the s-function to return $g(t, x, t)$, and when `flag==3` it expects the s-function to return $h(t, x, u)$. The switch statement that calls the proper functions based on the value of `flag` is shown in Lines 2–11. The block setup and the definition of the initial conditions is shown in Lines 13–27. The number of continuous states, discrete states, outputs, and inputs are defined in Lines 16-19, respectively. The direct feedthrough term on Line 20 is set to one if the output depends explicitly on the input u: for example, if $D \neq 0$ in the linear state space output equation $y = Cx + Du$. The initial conditions are defined on Line 24. The sample times are defined on Line 27. The format for this line is `ts = [period offset]`, where `period` defines the sample period, and is 0 for continuous time, or -1 for inherited, and where `offset` is the sample time offset, which is typically 0. The function $f(t, x, u)$ is defined in Lines 30–32, and the output function $h(t, x, u)$ is defined in Lines 35–36. A Simulink file that calls this m-file s-function is contained on the accompanying CDROM.

```
1   function [sys,x0,str,ts] = second_order_m(t,x,u,flag,zeta,wn)
2       switch flag,
3       case 0,
4           [sys,x0,str,ts]=mdlInitializeSizes;  % initialize block
5       case 1,
6           sys=mdlDerivatives(t,x,u,zeta,wn);  % define xdot = f(t,x,u)
```

```
7           case 3,
8               sys=mdlOutputs(t,x,u,wn);           % define xup = g(t,x,u)
9           otherwise,
10              sys = [];
11          end
12
13      %=============================================================================
14      function [sys,x0,str,ts]=mdlInitializeSizes
15          sizes = simsizes;
16          sizes.NumContStates  = 2;
17          sizes.NumDiscStates  = 0;
18          sizes.NumOutputs     = 1;
19          sizes.NumInputs      = 1;
20          sizes.DirFeedthrough = 0;
21          sizes.NumSampleTimes = 1;     % at least one sample time is needed
22          sys = simsizes(sizes);
23
24          x0  = [0; 0];  %  define initial conditions
25          str = [];  % str is always an empty matrix
26          % initialize the array of sample times
27          ts  = [0 0];      % continuous sample time
28
29      %=============================================================================
30      function xdot=mdlDerivatives(t,x,u,zeta,wn)
31          xdot(1) = -2*zeta*wn*x(1) - wn^2*x(2) + u;
32          xdot(2) = x(1);
33
34      %=============================================================================
35      function y=mdlOutputs(t,x,u,wn)
36          y = wn^2*x(2);
```

B.1.2 C-file S-function

The code listing for a C-file s-function that implements system (B.5) and (B.6) is shown below. The function name must be specified as in Line 3. The number of parameters that are passed to the s-function is specified in Line 17, and macros that access the parameters are defined in lines 6 and 7. Line 8 defines a macro that allows easy access to the input of the block. The block structure is defined using `mdlInitializeSizes` in Line 15–36. The number of continuous states, discrete states, inputs, and outputs is defined in Lines 21–27. The sample time and offset are specified in Lines 41–46. The initial conditions for the states are specified in Lines 52–57. The

function $f(t, x, u)$ is defined in Lines 76–85, and the function $h(t, x, u)$ is defined in Lines 62–69. The C-file s-function is compiled using the matlab command `>> mex secondOrder_c.c`. A Simulink file that calls this C-file s-function is contained on the accompanying CDROM.

```
1   /*  File    : secondOrder_c.c
2    */
3   #define S_FUNCTION_NAME secondOrder_c
4   #define S_FUNCTION_LEVEL 2
5   #include "simstruc.h"
6   #define zeta_PARAM(S) mxGetPr(ssGetSFcnParam(S,0))
7   #define wn_PARAM(S)   mxGetPr(ssGetSFcnParam(S,1))
8   #define U(element) (*uPtrs[element])  /* Pointer to Input Port0 */
9
10  /* Function: mdlInitializeSizes
11   * Abstract:
12   *    The sizes information is used by Simulink to determine the S-function
13   *    blocks characteristics (number of inputs, outputs, states, etc.).
14   */
15  static void mdlInitializeSizes(SimStruct *S)
16  {
17      ssSetNumSFcnParams(S, 2);  /* Number of expected parameters */
18      if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
19          return; /* Parameter mismatch will be reported by Simulink */
20      }
21      ssSetNumContStates(S, 2);
22      ssSetNumDiscStates(S, 0);
23      if (!ssSetNumInputPorts(S, 1)) return;
24      ssSetInputPortWidth(S, 0, 1);
25      ssSetInputPortDirectFeedThrough(S, 0, 1);
26      if (!ssSetNumOutputPorts(S, 1)) return;
27      ssSetOutputPortWidth(S, 0, 1);
28      ssSetNumSampleTimes(S, 1);
29      ssSetNumRWork(S, 0);
30      ssSetNumIWork(S, 0);
31      ssSetNumPWork(S, 0);
32      ssSetNumModes(S, 0);
33      ssSetNumNonsampledZCs(S, 0);
34      /* Take care when specifying exception free code - see sfuntmpl_doc.c */
35      ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
36  }
37
```

```
38      /* Function: mdlInitializeSampleTimes
39       *    S-function is comprised of only continuous sample time elements
40       */
41      static void mdlInitializeSampleTimes(SimStruct *S)
42      {
43          ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
44          ssSetOffsetTime(S, 0, 0.0);
45          ssSetModelReferenceSampleTimeDefaultInheritance(S);
46      }
47
48      #define MDL_INITIALIZE_CONDITIONS
49      /* Function: mdlInitializeConditions
50       *    Set initial conditions
51       */
52      static void mdlInitializeConditions(SimStruct *S)
53      {
54          real_T *x0 = ssGetContStates(S);
55          x0[0] = 0.0;
56          x0[1] = 0.0;
57      }
58
59      /* Function: mdlOutputs
60       *      output function
61       */
62      static void mdlOutputs(SimStruct *S, int_T tid)
63      {
64          real_T            *y    = ssGetOutputPortRealSignal(S,0);
65          real_T            *x    = ssGetContStates(S);
66          InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
67          UNUSED_ARG(tid); /* not used in single tasking mode */
68          const real_T      *wn   = wn_PARAM(S);
69          y[0] = wn[0]*wn[0]*x[1];
70      }
71
72      #define MDL_DERIVATIVES
73      /* Function: mdlDerivatives
74       *      Calculate state-space derivatives
75       */
76      static void mdlDerivatives(SimStruct *S)
77      {
78          real_T            *dx   = ssGetdX(S);
79          real_T            *x    = ssGetContStates(S);
```

```
80          InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
81          const real_T      *zeta = zeta_PARAM(S);
82          const real_T      *wn   = wn_PARAM(S);
83          dx[0] = -2*zeta[0]*wn[0]*x[0] - wn[0]*wn[0]*x[1] + U(0);
84          dx[1] = x[0];
85      }
86
87      /* Function: mdlTerminate
88       *    No termination needed, but we are required to have this routine.
89       */
90      static void mdlTerminate(SimStruct *S)
91      {
92          UNUSED_ARG(S); /* unused input argument */
93      }
94
95      #ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
96      #include "simulink.c"      /* MEX-file interface mechanism */
97      #else
98      #include "cg_sfun.h"       /* Code generation registration function */
99      #endif
```

Appendix C

Airframe Parameters

This appendix gives the parameters for several RC size airframes.

C.1 Zagi

The parameters for Zagi airframe shown in Figure C.1 are given in Table C.1.

Figure C.1: The Zagi airframe.

The following aerodynamic coefficients for an A-4D fighter aircraft [1].

Parameter	Value
m	$1.56kg$
J_x	$0.1147kg - m^2$
J_y	$0.0576kg - m^2$
J_z	$0.1712kg - m^2$
J_{xz}	$0.0015kg - m^2$
S	0.2589 m^2
b	1.4224 m
c	0.3302 m
S_{prop}	0.0314 m^2
ρ	1.2682 kg/m^3
k_{motor}	20
k_{T_p}	0
k_Ω	0

Table C.1: Parameters for the Zagi airframe.

Longitudinal Coef.	**Value**	**Lateral Coef.**	**Value**
C_{L_0}	0.28	C_{Y_0}	0
M	50	α_0	0.4712
ϵ	0.1592		
C_{D_0}	0.03	C_{l_0}	0
C_{m_0}	0	C_{n_0}	0
C_{L_α}	3.45	C_{Y_β}	-0.98
C_{D_α}	0.30	C_{l_β}	-0.12
C_{m_α}	-0.38	C_{n_β}	0.25
C_{L_q}	0	C_{Y_p}	0
C_{D_q}	0	C_{l_p}	-0.26
C_{m_q}	-3.6	C_{n_p}	0.022
$C_{L_{\delta_e}}$	-0.36	C_{Y_r}	0
$C_{D_{\delta_e}}$	0	C_{l_r}	0.14
$C_{m_{\delta_e}}$	-0.5	C_{n_r}	-0.35
C_{prop}	1.0	$C_{Y_{\delta_a}}$	0
		$C_{l_{\delta_a}}$	0.08
		$C_{n_{\delta_a}}$	0.06
		$C_{Y_{\delta_r}}$	-0.17
		$C_{l_{\delta_r}}$	0.105
		$C_{n_{\delta_r}}$	-0.032

Table C.2: Aerodynamic coefficients for design project.

Appendix D

Trim and Linearization in Simulink

RWB: Add a detailed description of how to compute trim using a Simulink model.

RWB: Can we use the linmod command and the modred command to produce reduced order transfer function models? If so, describe how to do it.

RWB: Add a detailed description of how to compute linear state space equations using a Simulink model.

RWB: The project design in Chapter 5 should point to this appendix.

Appendix E

Essentials from Probability Theory

Let $X = (x_1, \ldots, x_n)^T$ 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

$$\textit{cov}(x_i, x_j) = \Sigma_{ij} = E\{(x_i - \mu_i)(x_j - \mu_j)\}.$$

The covariance of any component with itself is the variance, i.e.,

$$\textit{var}(x_i) = \textit{cov}(x_i, x_i) = \Sigma_{ii} = E\{(x_i - \mu_i)(\xi - \mu_i)\}.$$

The standard deviation of x_i is the square root of the variance:

$$\textit{stdev}(x_i) = \sigma_i = \sqrt{\Sigma_{ii}}.$$

The covariances associated with a random vector X can be grouped into a matrix known as the covariance matrix:

$$\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} & \cdots & \Sigma_{1n} \\ \Sigma_{21} & \Sigma_{22} & \cdots & \Sigma_{2n} \\ \vdots & & \ddots & \vdots \\ \Sigma_{n1} & \Sigma_{n2} & \cdots & \Sigma_{nn} \end{pmatrix} = E\{(X - \boldsymbol{\mu})(X - \boldsymbol{\mu})^T\} = E\{XX^T\} - \boldsymbol{\mu}\boldsymbol{\mu}^T.$$

Note that $\Sigma = \Sigma^T$ so that Σ 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 μ_x is the mean of x and σ_x is the standard deviation. The vector equivalent is given by

$$f_X(X) = \frac{1}{\sqrt{2\pi \det \Sigma}} \exp\left[-\frac{1}{2}(X-\boldsymbol{\mu})^T \Sigma^{-1}(X-\boldsymbol{\mu})\right],$$

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 Σ.

Figure E.1 shows the level curves for a 2D Gaussian random variable with different covariance matrices.

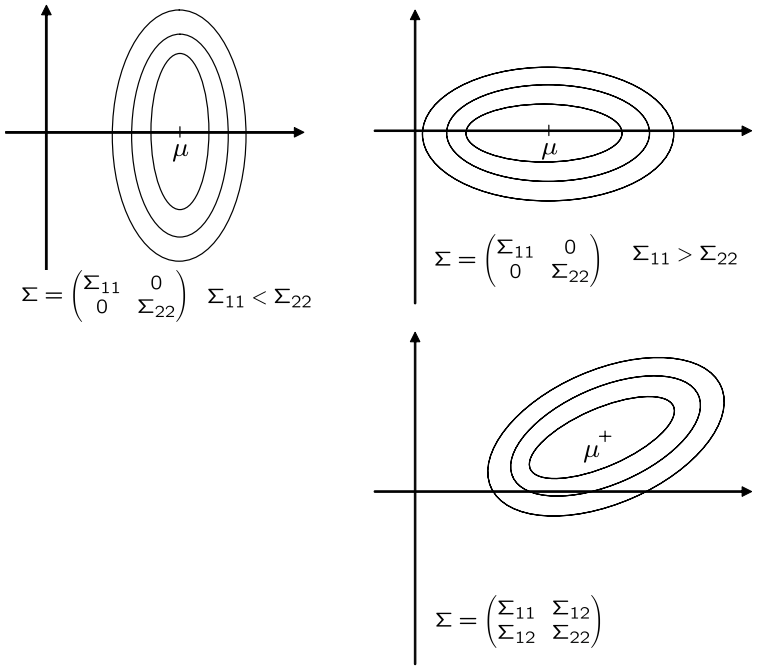


Figure E.1: Level curves for the pdf of a 2D Gaussian random variable.

Appendix F

Sensor Parameters

This appendix gives the parameters for several commercially available sensors.

F.1 Rate Gyros

RWB: Remove reference to Kestrel and replace with reference to rate gyro found on the web.

If noise density N is given in units/$\sqrt{Hz}$, and the bandwidth B is given in Hz, then the variance is

$$\sigma^2 = N^2 B$$

and the standard deviation is

$$\sigma = N\sqrt{B}.$$

The variance is in units^2 and the standard deviation is in units.

The spec sheet for the Kestrel autopilot (v2.2) lists the bandwidth of the rate gyros as $B = 9Hz$ and the noise density as $N = 0.1$ degrees/sec/sqrt(Hz). Therefore, the variance is given by

$$\sigma^2_{gyro,*} = N^2 B = 0.09(\text{degrees}^2/\text{sec}^2) = 0.00157(\text{rad}^2/\text{sec}^2),$$

and the standard deviation is given by

$$\sigma_{gyro,*} = 0.3(\text{degrees/sec}) = 0.0396(\text{rad/sec}).$$

F.2 Accelerometers

RWB: Remove reference to Kestrel and replace with reference to accelerometer found on the web.

The spec sheet for the Kestrel autopilot (v2.2) lists the bandwidth of the accelerometers as $B = 22Hz$ and the noise density as $N = 200\mu g/\sqrt{Hz}$. Therefore, the variance is given by

$$\sigma^2_{accel,*} = N^2 B = 0.00000088(g^2),$$

and the standard deviation is given by

$$\sigma_{accel,*} = 0.00094(g).$$

F.3 Pressure Sensors

RWB: Remove reference to Kestrel and replace with reference to pressure sensors found on the web. Also, get specs in terms of noise density and bandwidth.

The spec sheet for the Kestrel autopilot lists the range of the differential pressure sensor as $[-0.25, 4.7]$ kPa, and the resolution as 0.000166 kPa/LSB. The range for the static pressure sensor is listed as $[101.5, 66.5]$ kPa, and the resolution as 0.00115 kPa/LSB.

We will assume that the standard deviation is equal to twice the resolution RWB: Fix this. Therefore the standard deviation of the pressure sensors are given by

$$\sigma_{\text{static pres}} = 0.3320(Pa)$$
$$\sigma_{\text{diff pres}} = 2.3(Pa),$$

and the variances are given by

$$\sigma^2_{\text{static pres}} = 0.1102(Pa^2)$$
$$\sigma^2_{\text{diff pres}} = 5.29(Pa^2).$$

F.4 Magnetometers

RWB: Reference real magnetometer

$$\sigma_{mag,*} = .$$

F.5 GPS

Table F.1 shows an error budget for GPS which is listed at www.montana.edu/places/gps/lres357/slides/GPSaccuracy.ppt.

Effect	Ave. Horizontal Error	Ave. Vertical Error
Atmosphere	5.5 meters	5.5 meters
Satellite Geometry (Ephemeris) data	2.5 meters	15 meters
Satellite clock drift	1.5 meters	1.5 meters
Multipath	0.6 meters	0.6 meters
Measurement noise	0.3 meters	0.3 meters

Table F.1: GPS Error Budget

Parameter	Uniform distribution	Variance
$A_{n,\text{geometry}}$	$[0, 2.5\sqrt{2}]$ (m)	
$A_{e,\text{geometry}}$	$[0, 2.5\sqrt{2}]$ (m)	
$A_{h,\text{geometry}}$	$[0, 15\sqrt{2}]$ (m)	
$A_{*,\text{multipath}}$	$[0, 0.6\sqrt{2}]$ (m)	
$\omega_{*,\text{geometry}}$	$[0, 10^{-6}]$ (s)	
$\omega_{*,\text{multipath}}$	$[0, 10^{-3}]$ (s)	
$\nu_{*,\text{geometry}}$	$[-\pi, \pi]$ (rad)	
$\nu_{*,\text{multipath}}$	$[-\pi, \pi]$ (rad)	
$\eta_{n,\text{measurement}}$		$0.3\ (m^2)$
$\eta_{e,\text{measurement}}$		$0.3\ (m^2)$
$\eta_{h,\text{measurement}}$		$1.3\ (m^2)$
$\eta_{*,\text{atmosphere}}$	$[0, 5]$ (m)	
η_{clock}	$[0, 1.5]$ (m)	

Table F.2: GPS Error Budget

The GPS parameters are given by Table F.2. Sample rates for GPS can vary between $0.5 - 2$ seconds. For our purposes we will assume that the sample rate for GPS is given by

$$T_{s,GPS} = 1.0 \text{ Hz}.$$

Appendix G

Useful Formulas and other Information

G.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	100
mph	11.5	23.0	34.6	46.1	57.6	69.1	80.6	92.2	103.7	115.2

G.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 $R = 287.05$[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 N/m^2. Typical weather data reports pressure in inches of Mercury. The conversion from N/m^2 to inches of Mercury is given by

$$P_s[\text{N/m}^2] = 3376.85 P_s[\text{inHg}].$$

Bibliography

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

[2] J. Roskam, *Airplane Flight Dynamics and Automatic Flight Controls, Parts I & II*. Lawrence, Kansas: DARcorporation, 1998.

[3] J. H. Blakelock, *Automatic Control of Aircraft and Missiles*. John Wiley & Sons, 1965.

[4] J. H. Blakelock, *Automatic Control of Aircraft and Missiles*. John Wiley & Sons, second edition ed., 1991.

[5] M. V. Cook, *Flight Dynamics Principles*. New York: John Wiley & Sons, 1997.

[6] B. Etkin and L. D. Reid, *Dynamics of Flight: Stability and Control*. John Wiley & Sons, 1996.

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

[8] W. J. Rugh, *Linear System Theory*. Englewood Cliffs, New Jersey: Prentice Hall, 2nd ed., 1996.

[9] D. T. Greenwood, *Principles of Dynamics*. Englewood Cliffs, NJ: Prentice Hall, 2nd ed., 1988.

[10] T. R. Kane and D. A. Levinson, *Dynamics: Theory and Applications*. McGraw Hill, 1985.

[11] M. W. Spong and M. Vidyasagar, *Robot Dynamics and Control*. John Wiley & Sons, Inc., 1989.

[12] M. D. Shuster, "A survey of attitude representations," *The Journal of the Astronautical Sciences*, vol. 41, pp. 439–517, October–December 1993.

[13] 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.

[14] M. Rauw, *FDC 1.2 - A SIMULINK Toolbox for Flight Dynamics and Control Analysis*, February 1998. Available at http://www.mathworks.com/.

[15] H. Goldstein, *Classical Mechanics*. Addison-Wesley, 1951.

[16] A. V. Rao, *Dynamics of Particles and Rigid Bodies: A Systematic Approach*. Cambridge University Press, 2006.

[17] M. J. Sidi, *Spacecraft Dynamics and Control*. Cambridge Aerospace Series, New York: Cambridge University Press, 1997.

[18] J. B. Marion, *Classical Dynamics of Particles and Systems*. Academic Press, second edition ed., 1970.

[19] W. E. Wiesel, *Spaceflight Dynamics*. McGraw Hill, second edition ed., 1997.

[20] J. R. Wertz, ed., *Spacecraft Attitude Determination and Control*. Kluwer Academic Publishers, 1978.

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

[22] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*. McGraw-Hill, 1987.

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

[24] J. D. Anderson, *Introduction to Flight*. McGraw Hill, 1989.

[25] "U.s. standard atmosphere, 1976," tech. rep., National Oceanic and Atmospheric Administration, National Aeronautics and Space Administration, United States Air Force, Washington, D.C., 1976.

[26] Dorato, Abdallah, and Cerone, *Linear-Quadratic Control: An Introduction*. Prentice Hall, 1995.

[27] T. K. Moon and W. C. Stirling, *Mathematical Methods and Algorithms*. Englewood Cliffs, New Jersey: Prentice Hall, 2000.

[28] D. R. Nelson, D. B. Barber, T. W. McLain, and R. W. Beard, "Vector field path following for miniature air vehicles," *IEEE Transactions on Robotics*, vol. 37, pp. 519–529, June 2007.

[29] D. R. Nelson, D. B. Barber, T. W. McLain, and R. W. Beard, "Vector field path following for small unmanned air vehicles," in *American Control Conference*, (Minneapolis, Minnesota), pp. 5788–5794, June 2006.

[30] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, pp. 500–505, 1985.

[31] 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.

[32] 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*, August 2004. AIAA-2004-4900.

[33] 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.

[34] P. Aguiar, D. Dačić, J. Hespanha, and P. Kokotović, "Path-following or reference-tracking? An answer relaxing the limits to performance," in *Proceedings of IAV2004, 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles*, (Lisbon, Portugal), 2004.

[35] A. P. Aguiar, J. P. Hespanha, and P. V. Kokotovic, "Path-following for nonminimum phase systems removes performance limitations," *IEEE Transactions on Automatic Control*, vol. 50, pp. 234–238, February 2005.

[36] J. Hauser and R. Hindman, "Maneuver regulation from trajectory tracking: Feedback linearizable systems," in *Proceedings of the IFAC Symposium on Nonlinear Control Systems Design*, (Tahoe City, CA), pp. 595–600, June 1995.

[37] P. Encarnação and A. Pascoal, "Combined trajectory tracking and path following: An application to the coordinated control of marine craft," in *Proceedings of the IEEE Conference on Decision and Control*, (Orlando, FL), pp. 964–969, 2001.

[38] R. Skjetne, T. Fossen, and P. Kokotović, "Robust output maneuvering for a class of nonlinear systems," *Automatica*, vol. 40, pp. 373–383, 2004.

[39] R. Rysdyk, "UAV path following for constant line-of-sight," in *Proceedings of the AIAA 2nd Unmanned Unlimited Conference*, AIAA, September 2003. Paper no. AIAA-2003-6626.

[40] T. W. McLain and R. W. Beard, "Coordination variables, coordination functions, and cooperative timing missions," *AIAA Journal of Guidance, Control and Dynamics*, vol. 28, pp. 150–161, January 2005.

[41] 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, pp. 497–516, 1957.

[42] E. P. Anderson, R. W. Beard, and T. W. McLain, "Real time dynamic trajectory smoothing for uninhabited aerial vehicles," *IEEE Transactions on Control Systems Technology*, vol. 13, pp. 471–477, May 2005.

[43] G. Yang and V. Kapila, "Optimal path planning for unmanned air vehicles with kinematic and tactical constraints," in *Proceedings of the IEEE Conference on Decision and Control*, (Las Vegas, NV), pp. 1301–1306, 2002.

[44] P. Chandler, S. Rasumussen, and M. Pachter, "UAV cooperative path planning," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, (Denver, CO), August 2000. AIAA Paper No. AIAA-2000-4370.

[45] 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. 247–264f, A. K. Peters, 2001.

[46] 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, pp. 640–652, August 1999.

[47] R. M. Murray and S. S. Sastry, "Nonholonomic motion planning: Steering using sinusoids," *IEEE Transactions on Automatic Control*, vol. 38, pp. 700–716, May 1993.

[48] T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams," *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 926–939, December 1998.

[49] R. C. Arkin, *Behavior-based Robotics*. MIT Press, 1998.

[50] R. Sedgewick, *Algorithms*. Addison-Wesley, 2nd ed., 1988.

[51] F. Aurenhammer, "Voronoi diagrams - a survey of fundamental geometric data struct," *ACM Computing Surveys*, vol. 23, pp. 345–405, September 1991.

[52] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. McGraw-Hill, 1993.

[53] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotic Research*, vol. 20, pp. 378–400, May 2001.

[54] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[55] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. B. adn Lydia E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementation*. MIT Press, 2005.

[56] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[57] T. McLain and R. Beard, "Cooperative rendezvous of multiple unmanned air vehicles," in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, (Denver, CO), August 2000. Paper no. AIAA-2000-4369.

[58] T. W. McLain, P. R. Chandler, S. Rasmussen, and M. Pachter, "Cooperative control of UAV rendezvous," in *Proceedings of the American Control Conference*, (Arlington, VA), pp. 2309–2314, June 2001.

[59] 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.

[60] H. Choset and J. Burdick, "Sensor-based exploration: The hierarchical generalized Voronoi graph," *The International Journal of Robotic Research*, vol. 19, pp. 96–125, February 2000.

[61] 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, pp. 126–148, February 2000.

[62] D. Eppstein, "Finding the k shortest paths," *SIAM Journal of Computing*, vol. 28, no. 2, pp. 652–673, 1999.

[63] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning." TR 98-11, Computer Science Dept., Iowa State University, October 1998.

[64] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (San Francisco, CA), pp. 995–1001, April 2000.

[65] 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*, (Roma, Italy), April 2007.

[66] A. Ladd and L. E. Kavraki, "Generalizing the analysis of PRM," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Washington DC), pp. 2120–2125, May 2002.

[67] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *Journal of Guidance, Control, and Dynamics*, vol. 25, pp. 116–129, January-February 2002.

[68] E. U. Acar, H. Choset, and J. Y. Lee, "Sensor-based coverage with extended range detectors," *IEEE Transactions on Robotics*, vol. 22, pp. 189–198, February 2006.

[69] C. Luo, S. X. Yang, D. A. Stacey, and J. C. Jofriet, "A solution to vicinity problem of obstacles in complete coverage path planning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Washington DC), pp. 612–617, May 2002.

[70] Z. J. Butler, A. A. Rizzi, and R. L. Hollis, "Cooperative coverage of rectilinear environments," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (San Francisco, CA), pp. 2722–2727, April 2000.

[71] 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*, (Washington DC), pp. 1327–1332, May 2002.

[72] M. Schwager, J.-J. Slotine, and J. J. Daniela Russell, "Consensus learning for distributed coverage control," in *Proceedings of the International Conference on Robotics and Automation*, (Pasadena, CA), pp. 1042–1048, May 2008.

[73] J. H. Evers, "Biological inspiration for agile autonomous air vehicles," in *Symposium on Platform Innovations and System Integration for Unmanned Air, Land, and Sea Vehicles*, (Florence, Italy), NATO Research and Technology Organization AVT-146, May 2007. Paper no. 15.

[74] D. B. Barber, J. D. Redding, T. W. McLain, R. W. Beard, and C. N. Taylor, "Vision-based target geo-location using a fixed-wing miniature air vehicle," *Journal of Intelligent and Robotic Systems*, vol. 47, pp. 361–382, December 2006.

[75] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer-Verlag, 2003.

[76] P. Zarchan, *Tactical and Strategic Missile Guidance*, vol. 124 of *Progress in Astronautics and Aeronautics*. Washington DC: American Institute of Aeronautics and Astronautics, 1990.

[77] M. Guelman, M. Idan, and O. M. Golan, "Three-dimensional minimum energy guidance," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 31, pp. 835–841, April 1995.

[78] J. G. Lee, H. S. Han, and Y. J. Kim, "Guidance performance analysis of bank-to-turn (BTT) missiles," in *Proceedings of the IEEE International Conference on Control Applications*, (Kohala, Hawaii), pp. 991–996, August 1999.

[79] S. Ettinger, M. Nechyba, P. Ifju, and M. Waszak, "Vision-guided flight stability and control for micro air vehicles," *Advanced Robotics*, vol. 17, no. 3, pp. 617–640, 2003.

[80] 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*, 2003.

[81] 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, Barcelona, Spain*, 2000.

[82] D. Lee, K. Lillywhite, S. Fowers, B. Nelson, and J. Archibald, "An embedded vision system for an unmanned four-rotor helicopter," in *SPIE Optics East, Intelligent Robots and Computer Vision XXIV: Algorithms, Techniques, and Active Vision*, vol. vol. 6382-24, 63840G, (Boston, MA, USA), October 2006.

[83] J. Lopez, M. Markel, N. Siddiqi, G. Gebert, and J. Evers, "Performance of passive ranging from image flow," in *Proceedings of the IEEE International Conference on Image Processing*, vol. 1, pp. I–929–I–932, September 2003.

[84] M. Pachter, N. Ceccarelli, and P. R. Chandler, "Vision-based target geo-location using camera equipped mavs," in *Proceedings of the IEEE Conference on Decision and Control*, (New Orleans, LA), December 2007. (to appear).

[85] R. J. Prazenica, A. J. Kurdila, R. C. Sharpley, P. Binev, M. H. Hielsberg, J. Lane, and J. Evers, "Vision-based receding horizon control for micro air vehicles in urban environments," *AIAA Journal of Guidance, Dynamics, and Control*, (in review).

[86] I. Wang, V. Dobrokhodov, I. Kaminer, and K. Jones, "On vision-based target tracking and range estimation for small UAVs," in *2005 AIAA Guidance, Navigation, and Control Conference and Exhibit*, pp. 1–11, 2005.

[87] Y. Watanabe, A. J. Calise, E. N. Johnson, and J. H. Evers, "Minimum-effort guidance for vision-based collision avoidance," in *Proceedings of the AIAA Atmospheric Flight Mechanics Conference and Exhibit*, (Keystone, Colorado), American Institute of Aeronautics and Astronautics, August 2006. Paper no. AIAA-2006-6608.

[88] Y. Watanabe, E. N. Johnson, and A. J. Calise, "Optimal 3-D guidance from a 2-D vision sensor," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, (Providence, Rhode Island), American Institute of Aeronautics and Astronautics, August 2004. Paper no. AIAA-2004-4779.

[89] I. H. Whang, 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*, (San Francisco, CA), August 2005.

[90] 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, pp. 65–91, January 2005.

[91] M. E. Campbell and M. Wheeler, "A vision based geolocation tracking system for uavs," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Keystone, Colorado), August 2006. Paper No. AIAA-2006-6246.

[92] 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*, (Keystone, Colorado), August 2006. Paper no. AIAA-2006-6606.

[93] E. W. Frew, "Sensitivity of cooperative target geolocation to orbit coordination," *Journal of Guidance, Control, and Dynamics*, vol. 31, pp. 1028–1040, July-August 2008.

[94] D. Murray and A. Basu, "Motion tracking with an active camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, pp. 449–459, May 1994.

[95] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 651–670, October 1996.

[96] J. Oliensis, "A critique of structure-from-motion algorithms," *Computer Vision and Image Understanding (CVIU)*, vol. 80, no. 2, pp. 172–214, 2000.

[97] J. Santos-Victor and G. Sandini, "Uncalibrated obstacle detection using normal flow," *Machine Vision and Applications*, vol. 9, no. 3, pp. 130–137, 1996.

[98] R. B. L.M. Lorigo and W. Grimson, "Visually-guided obstacle avoidance in unstructured environments," in *Proceedings of IROS '97*, (Grenoble, France), September 1997.

[99] R. Nelson and Y. Aloimonos, "Obstacle avoidance using flow field divergence," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 1102–1106, October 1989.

[100] K. H. Gabbiani F and L. G, "Computation of object approach by a wide field visual neuron," *J Neuroscience*, vol. 19, pp. 1122–1141, 1999.

[101] R. W. Beard, J. W. Curtis, M. Eilders, J. Evers, and J. R. Cloutier, "Vision aided proportional navigation for micro air vehicles," in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, (Hilton Head, North Carolina), American Institute of Aeronautics and Astronautics, August 2007. Paper number AIAA-2007-6609.

[102] A. E. Bryson and Y. C. Ho, *Applied Optimal Control*. Waltham, MA: Blaisdell Publishing Company, 1969.

[103] M. Guelman, "Proportional navigation with a maneuvering target," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 8, pp. 364–371, May 1972.

[104] C. F. Lin, *Modern Navigation, Guidance, and Control Processing*. Englewood Cliffs, New Jersey: Prentice Hall, 1991.

[105] 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, pp. 556–567, July 2002.

[106] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*. Addison Wesley, 4rd ed., 2002.