A Low-cost Unmanned Aerial Vehicle Research Platform: Development, Modeling, and Advanced Control Implementation

Ony Arifianto

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Aerospace Engineering

Mazen Farhood, Chair

Craig Woolsey

Cornel Sultan

Mayuresh Patil

November 26, 2013

Blacksburg, Virginia

Keywords: Unmanned Aerial Research Vehicles, Linear Time-Varying Control, Embedded Systems, System Identification

A Low-cost Unmanned Aerial Vehicle Research Platform: Development, Modeling, and Advanced Control Implementation

Ony Arifianto

(ABSTRACT)

This dissertation describes the development and modeling of a low-cost, open source, and reliable small fixed-wing unmanned aerial vehicle (UAV) for advanced control implementation. The platform is mostly constructed of low-cost commercial off-the-shelf (COTS) components. The only non-COTS components are the airdata probes which are manufactured and calibrated in-house, following a procedure provided herein. The airframe used is the commercially available radio-controlled 6-foot Telemaster airplane from Hobby Express. The airplane is chosen mainly for its adequately spacious fuselage and for being reasonably stable and sufficiently agile. One noteworthy feature of this platform is the use of two separate low-cost open source onboard computers for handling the data management/hardware interfacing and control computation. Specifically, the single board computer, Gumstix Overo Fire, is used to execute the control algorithms, whereas the autopilot, Ardupilot Mega, is mostly used to interface the Overo computer with the sensors and actuators. The platform supports multi-vehicle operations through the use of a radio modem that enables multi-point communications.

As the goal of the development of this platform is to implement rigorous control algorithms for real-time trajectory tracking and distributed control, it is important to derive an appropriate flight dynamic model of the platform, based on which the controllers will be synthesized. For that matter, reasonably accurate models of the vehicle, servo motors and propulsion system are developed. Namely, the output error method is used to estimate the longitudinal and lateral-directional aerodynamic parameters from flight test data. The moments of inertia of the platform are determined using the simple pendulum test method, and the frequency response of each servomotor is also obtained experimentally. The Javaprop applet is used to obtain lookup tables relating airspeed to propeller thrust at constant throttle settings.

Control systems are also designed for the regulation of this UAV along real-time trajectories. The reference trajectories are generated in real-time from a library of pre-specified motion primitives and hence are not known a priori. Two concatenated primitive trajectories are considered: one formed from seven primitives exhibiting a figure-8 geometric path and another composed of a Split-S maneuver that settles into a level-turn trim trajectory. Switched control systems stemming from ℓ_2-induced norm synthesis approaches are designed for discrete-time linearized models of the nonlinear UAV system. These controllers are analyzed based on simulations in a realistic operational environment and are further implemented on the physical UAV. The simulations and flight tests demonstrate that switched controllers, which take into account the effects of switching between constituent sub-controllers, manage to closely track the considered trajectories despite the various modeling uncertainties, exogenous disturbances and measurement noise. These switched controllers are composed of discrete-time linear sub-controllers designed separately for a subset of the pre-specified primitives, with the uncertain initial conditions, that arise when switching between primitives, incorporated into the control design.

For my parents, wife, and kids.

Acknowledgments

I would never have been able to finish this dissertation without the grace of God, guidance from my committee members, help from friends, and support from my family and wife.

I would like to express my deepest gratitude to my Advisor, Dr. Mazen Farhood, who taught me the value of dilligence and excellence in research. His high standard in every aspect of his work motivated me to take the extra mile in all of the works that I had done. I would also like to thank Dr. Craig Woolsey for helping me to build my presentation skill and my confidence. I would like to thank Dr. Cornel Sultan and Dr. Mayuresh Patil for their constructive feedbacks throughout the process.

I also would like to thank the Nonlinear Systems Laboratory (NSL) crew, Artur Wolek for the sincere feedbacks, Dave Grymin for the time spent proofreading paper and dissertation scripts, Mark Palframan for the excellent videos and pictures, Ankit Ganeriwal, Hossam Abdelwahid, Tejasui Gode and Eddie Hale, for the help, support and friendship that made the past five years memorable.

I would like to thank the Nuenighoff family, mas Curt, mbak Sita, Glen, Erik, and Lexi for being great friends and providing me with a place to stay in the last few months.

I would like to thank my mom and dad for being the main reason I undertook this PhD program, this dissertation is my gift for you.

I also would like to thank my brothers for supporting the endeavour and for taking care of our parents while I am far away. I thank my children for being wonderful, I will make up

for the camps, conferences, performances and trips that I missed. For my wife, there is not enough words to express my gratitude for your support, patience in taking care of the kids, and toughness to be on my side through this hard time.

Finally, I would like to thank Dr.-Ing. B. J. Habibie, Dr. Ing. O. H. Gerlach, and Dr. S. D. Jenie for being my inspiration.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

The many appealing features of unmanned aerial vehicles (UAVs), such as persistence and versatility, have rendered these systems very useful for a wide range of military, civilian, and research applications, including real-time reconnaissance, surveillance, and target acquisition, atmospheric sciences, and disaster relief. The focus of this dissertation is miniature-sized (less than 2 meter wing span), low-cost, but highly capable fixed-wing UAVs. Such UAVs can be deployed in groups to perform complex and intricate tasks, and hence can serve as a viable alternative to dispatching larger expensive high-tech aerial vehicles in certain applications. The appealing low-cost feature, which makes these UAVs expendable, is due partly to the use of relatively cheap sensors, actuators, and processors. In addition,

the convenient small size poses restrictions on the computational and sensing capabilities. Then, the strategy for achieving the desired high capability despite the size, weight, and cost restrictions centers around the development of new principles and novel technologies in software and hardware design.

Developing a UAV research platform to validate these new principles and technologies is of great importance, and so it is no surprise that many UAV testbeds have been developed by various academic groups, e.g., [1, 2, 3, 4, 5, 6, 7, 8, 9]. The main goal in building our testbed is to develop a low-cost, open source, and reliable fixed-wing UAV platform that can be used to implement relatively complex control algorithms. Building a low-cost UAV platform is not an original idea, as other research groups in the control and robotics community have done so. For instance, the UAV research group at the University of Minnesota have developed a low-cost UAV testbed [1], which is mostly built of commercial-off-the-shelf (COTS) components and uses the radio-controlled (R/C) fixed-wing UAV, Ultra Stick 25e [10], as the main test aircraft. Another low-cost UAV platform, based on the R/C Goldberg Decathlon ARF airframe [11], has been developed at Georgia Tech [3] for educational purposes. The UAV research platform [8] uses the low-cost open source autopilot, Ardupilot Mega [12], and the Aero Testbed at the University of Illinois uses the Paparazzi autopilot [13], which is also low-cost and open source. These testbeds and others which are not mentioned here rely on COTS components as well as hardware/software developed in-house.

1.2 Low-Cost Research Platform

Our testbed also consists mostly of low-cost COTS components; the only components built in-house are the airdata probes, which are used to measure the angle of attack, sideslip, and airspeed of the UAV. The procedure for building and calibrating these sensors is based on [14, 15, 16] and discussed in sufficient details herein. The airframe used is the commercially available R/C 6-foot Telemaster from Hobby Express [17]; see Figure 1.1 and Table 1.1 for the major geometric parameters of this aircraft. This aircraft is selected primarily for its box-fuselage design, which provides ample space for onboard electronics. Furthermore, this R/C airplane is a well-known platform that is reasonably stable, yet still capable of performing certain aerobatic maneuvers. Minor modifications to the basic airframe were performed in order to place sensors both in the fuselage and on the wings. Aside from the use of standard sensors and actuators, a noteworthy feature of our platform is that the data management/hardware interfacing and control computation are performed by two separate onboard computers. Specifically, the low-cost open source single-board computer, Gumstix Overo Fire [18], is used to execute the control algorithms and ultimately compute the control commands, whereas, the COTS autopilot, Ardupilot Mega, is mostly used to interface the Gumstix computer with the senors and actuators. The use of an XBee 900-Pro radio modem onboard the platform allows for data transmission among multiple vehicles, and thus the testbed also supports multi-vehicle operations.

This UAV research platform is to be used to implement systematic and rigorous approaches,

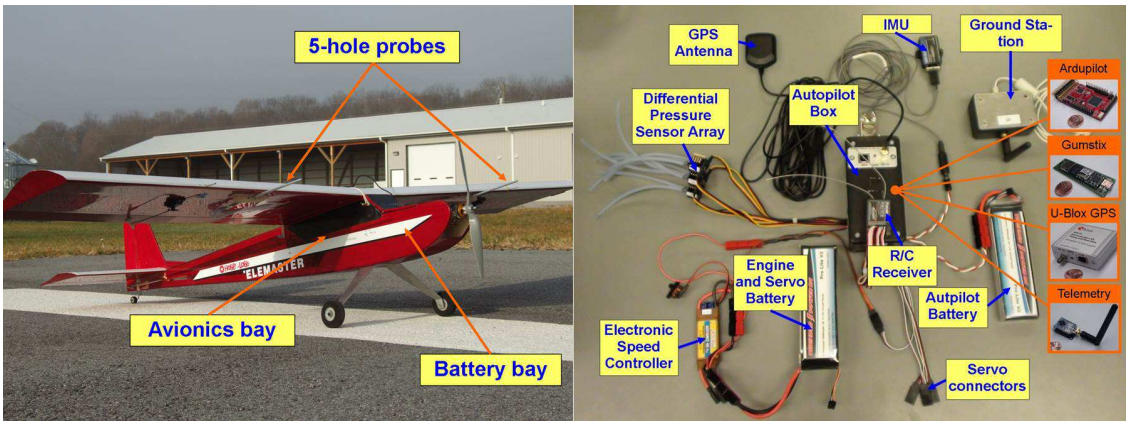

Figure 1.1: A 6-foot Telemaster UAV and its onboard instruments

Table 1.1: Main geometrical parameters of the Telemaster

Wing area	0.56	m^2
Wing span	1.83	m
Wing MAC*	0.30	m
Mass	3.24	kg

* MAC: mean aerodynamic chord

developed in our research lab, on controlled maneuvers, tracking along trajectories, and distributed control, e.g., [19, 20, 21, 22, 23], and further validate formal methods for mathematically certified control software design. As the control design methodology pursued is model based, it is important that we derive reasonably accurate models that capture the rigid-body dynamics of the vehicle and any relevant subsystems. For this application, the subsystems that directly influence the vehicle dynamics are the propulsion system and the servos that actuate the aerodynamic control surfaces. The thrust produced by the propeller is estimated using code that employs blade element theory [24]. Validation of the propulsion model is performed under static test conditions. The servo motors are modeled as single-input single-output systems based on frequency response data. The moments of inertia of the vehicle are determined experimentally by using the compound pendulum method, as de-

scribed in [25, 26, 27]. A dynamic model of the aircraft is obtained via system identification. A series of flight tests are performed, and the aerodynamic parameters of the postulated model are estimated from the flight test data using the output error method (OEM), an approach that has been used widely for aircraft system identification [28]. This approach is chosen due to its ability to handle measurement noise while maintaining a reasonable computational complexity, in contrast to the more sophisticated filter error method. Since the OEM does not account for process noise, e.g., atmospheric turbulence, it is necessary that all flight tests be performed at times when minimum disturbance is observed.

A thorough review of system identification applications for various types of UAVs is given in [29]. For fixed-wing aircraft, there are several works that are similar in scope to the approach presented here. A nonlinear mapping identification algorithm is utilized in [30] in order to estimate parameters that capture the attitude dynamics of an aircraft; the parameters corresponding to the moments acting on the vehicle are formulated as a linear model using state and input variables. In [31], the output error method is utilized to estimate stability and control derivatives that capture the roll attitude dynamics from test data. Nonlinear constrained optimization is used in [3] to estimate parameters that minimize the difference between measured and model predicted output data. Both longitudinal and lateral-directional models were obtained using the linearized dynamics for each mode, respectively. In this work, the output error method will be applied in order to estimate the parameters of a model capturing the aerodynamic forces and moments for both longitudinal and lateral-directional excitations.

A number of hybrid control systems with performance guarantees (in the ℓ_2-induced norm sense) have been implemented on this UAV platform. Figure 1.2 shows data and snapshots from a flight test in which a hybrid feedback controller designed for the UAV executes two tasks consecutively: The first task is performing a Split-S maneuver and the second is tracking a level-turn trim condition.

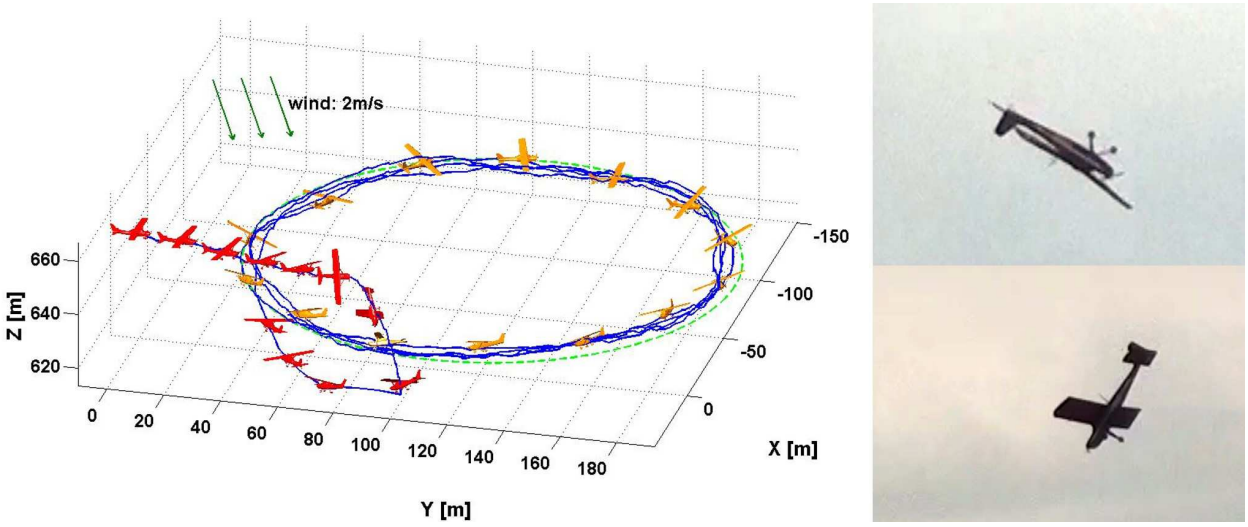

Figure 1.2: The plot on the left shows the UAV executing autonomously a Split-S maneuver followed by tracking a level-turn trim condition; the pictures on the right are snapshots of the UAV as it performs the Split-S maneuver

1.3 Advanced Control System Implementation

Motion planning and control are usually treated separately: the motion planner generates a reference trajectory online which leads to some desired goal state, and then the controller forces the system to track this reference trajectory as accurately as possible regardless of the various uncertainties and exogenous disturbances. We assume in this work that the

planner carries out primitive-based motion planning [32, 33, 34]. Specifically, the motion planner generates trajectories in real-time from a library of pre-specified motion primitives. A motion primitive is a basic dynamically-feasible trajectory, namely a state history and a corresponding control input that satisfy the nonlinear system equations over a finite or a semi-infinite time interval. These primitives can be generated by solving optimization problems involving the nonlinear system equations or by using flight simulators or actual flight data.

In addition to the development and modeling of the UAV platform, this dissertation also deals with the design and implementation of optimal controllers for the Telemaster UAV along trajectories generated in real-time by appropriately concatenating library primitives. We are mainly concerned with tracking dynamically feasible trajectories, that is, state and control histories that satisfy the system equations of motion, as opposed to just following geometric paths as in [35, 36], for instance. The motivation for this work is the control of fixed-wing UAVs in cluttered dynamic environments, where operating in the presence of moving obstacles necessitates that such complex dynamical systems with actuator limitations traverse the planned trajectories rather accurately, hence the need for high-performance controllers that can closely track dynamically feasible trajectories. In particular, we are interested in applying linear matrix inequality (LMI) based control approaches that use the ℓ_2-induced norm as the performance measure and assessing the capability of these methods in forcing a small fixed-wing UAV to closely track concatenated primitive trajectories under relatively significant wind conditions. We consider LMI-based methods for ℓ_2-induced norm

control of linear-time invariant (LTI) systems [37], linear time-varying (LTV) systems [38, 39], and linear systems with uncertain initial conditions [20].

The first step in designing a model-based controller is to derive a reasonably accurate mathematical model of the physical system. The second step is to design a library of pre-specified primitives, which provides the needed maneuverability. The library usually consists of trim conditions and transitions between trim conditions. The trim conditions can be obtained easily by solving for the equilibrium points of the nonlinear mathematical model. The transition maneuvers are not as easy to generate in general, but there are several methods that can be utilized for this purpose. One method entails formulating the problem as a nonlinear optimization problem and then using a software package such as GPOPS-II [40] to solve it. Alternatively, the maneuvers can be obtained by recording pilot inputs and corresponding system responses during an actual flight [41, 42]. Using this approach, aerobatic maneuvers such as barrel roll and hammerhead can be generated. In this dissertation, we will use simple feedback control to obtain transition maneuvers between trim conditions as well as recorded pilot data to generate the Split-S maneuver.

There are two approaches typically used for the regulation of UAVs about trajectories. The first and more common approach is to separate the control design problem into an inner loop and an outer loop [43, 44, 45]. The outer loop determines the linear and angular rates as well as the linear accelerations required to control the position and attitude. The inner loop then computes the control surface deflections and throttle in order to track the rates and accelerations commanded by the outer loop. In most cases, the inner loop is provided

by the onboard autopilot. A number of methods have been used to design the outer loop, for instance, using control Lyapunov functions [43, 44] or receding horizon control [45]. The second approach entails designing a single controller that directly controls the position and attitude by prompting corrective control surface deflection and throttle commands [46, 47]. Some of the work in the literature that falls under this approach includes [46], in which gain-scheduled H_∞ controllers are used to track trim trajectories, and [47], where receding horizon linear quadratic regulators are designed to track aerobatic maneuvers. One of the advantages of using this approach is that stability is guaranteed for the closed-loop system. The work herein uses the second approach.

When it comes to tracking trajectories generated in real-time from a library of prespecified motion primitives, one way to go about solving the control synthesis problem is to incorporate all possible connections between compatible library primitives into the control design, as discussed in [34, 21]. Such an approach comes with stability and performance guarantees across switching boundaries, but can become computationally prohibitive in the case of large primitive libraries. In this dissertation, we address the control problem using a decoupled approach, which is far less computationally intensive than the aforementioned approach because in this case the controllers associated with the library primitives are designed separately. We consider two concatenated primitive trajectories, which are assumed to be generated online using a primitive-based motion planner and hence are not known a priori. The first reference trajectory exhibits a "figure-8" geometric path and is formed by concatenating seven primitives, which include three trim trajectories and four transition

maneuvers. The second trajectory is composed of a Split-S maneuver that settles into a level-turn trim condition.

For the figure-8 trajectory, we design two discrete-time controllers and then analyze the performance of these controllers in simulation and flight tests. The first controller is a standard H_∞ controller synthesized based on a discrete-time linear plant model. This plant model is obtained by linearizing the nonlinear equations of motion of the UAV about a straight and level trim condition and then discretizing the resulting model using zero-order hold sampling. The second controller is a switching system, composed of three square ℓ_2 induced norm sub-controllers synthesized based on linear discrete-time plant models with uncertain initial conditions. These plant models are obtained by linearizing the nonlinear system equations about the three trim trajectories: straight and level, steady right turn, and steady left turn. We will refer to this controller as the switched uncertain initial condition (UIC) controller. The uncertain initial conditions are introduced into the control design to reflect the effects of switching between primitives. Based on the simulation and flight testing results, we find that the performances of the two controllers are generally comparable; however, the switched UIC controller achieves a superior performance when it comes to tracking the reference altitude trajectory.

Based on these findings, we adopt the UIC control approach for tracking the second reference trajectory. In this case, we design a standard finite horizon ℓ_2-induced norm sub-controller, with time-varying penalty weights on the state errors, for tracking the Split-S maneuver; it was not necessary to take into account in the control design the uncertain initial state since

we made sure that the system state at the beginning of the experiment was as close to the corresponding reference point as possible. In addition, a UIC sub-controller is designed to track the subsequent level-turn trim trajectory. We demonstrate in simulation as well as flight testing that the designed controller manages to closely track the reference trajectory despite the modeling uncertainties, wind disturbances, and measurement noise.

In general, the switched systems of interest in this dissertation are composed of linear time-invariant (LTI) and/or linear time-varying (LTV) subsystems obtained from linearizing the nonlinear system equations describing the vehicle dynamics about a subset of the set of prespecified primitives. As aforementioned, the switching between these subsystems and ultimately their corresponding sub-controllers is dictated by the motion planning algorithm. Related to this work is the paper [48] which provides a hybrid dynamics framework for the design of guaranteed safe switching regions using reachable sets. The paper [49] also gives a control algorithm for maneuver-based motion planning, which is robust to a certain class of perturbations.

1.4 Contributions

The contributions of this dissertation are as follows:

1. We have developed a low-cost, open source, small UAV research platform, which is built mostly of COTS components with a total price not exceeding $3000. The only non-COTS components are the airdata probes, which are manufactured in-house following

a simple process and calibrated using the procedure from [16].

2. We have derived reasonably accurate models of the UAV, servos, and propulsion system. The output error method is used to estimate both the longitudinal and lateral-directional aerodynamic parameters from flight test data, which, to the author's best knowledge, is a first for small UAVs.

3. We have applied LMI-based control approaches that use ℓ_2-induced norm as the performance measure and assessed the capability of these methods in forcing a small fixed-wing UAV to closely track concatenated primitive trajectories under relatively significant wind conditions.

The work presented in this dissertation is based on the following two submitted journal papers:

- Ony Arifianto and Mazen Farhood, "Development and Modeling of a Low-Cost Unmanned Aerial Vehicle Research Platform," submitted to Journal of Intelligent & Robotic Systems.

- Ony Arifianto and Mazen Farhood, "Optimal Control of a Small Fixed-Wing UAV about Concatenated Trajectories," submitted to IEEE Transactions on Control Systems Technology.

Other published work that is related to the work presented herein appears in the following conference paper:

- Ony Arifianto and Mazen Farhood, "Optimal Control of Fixed-Wing UAVs along Real-Time Trajectories," Proceedings of the ASME Dynamic Systems and Control Conference, 2012.

1.5 Organization

The outline of the dissertation is as follows. In Chapter 2, we present the autopilot system, along with the ground control station, and discuss the development of the airdata probes. In Chapter 3, we discuss the simple pendulum test method used to determine the moments of inertia of the vehicle, and derive the mathematical models of the UAV, servos, and propulsion system. Chapter 4 focuses on the design and validation of the control systems used for tracking the figure-8 and the Split-S/level-turn trim trajectories. A brief summary is provided in Chapter 5.

Chapter 2

Aerial Research Platform Development

This chapter consists of two sections. The first describes all the components of the platform avionics and the data flow among these components. The second section presents the development process of two airdata probes, which are manufactured and calibrated in-house; these sensors measure the airflow around the airplane and are used to determine the angle of attack, sideslip, and airspeed of the UAV.

2.1 System Description and Architecture

In this section, we discuss the electronics used onboard the platform, which consist of actuators, sensors, communication radios, and computers, as shown in Figure 1.1.

Actuators: The actuators are chosen based on the manufacturer's (Hobby Express [17]) recommendations. Specifically, Futaba S3152 servos [50] are used for aileron, elevator, and rudder control. A JETI ADVANCE 40 Pro (40 Amps programmable) electronic speed controller (ESC) [51] is used to regulate the rotational speed of an AXI 2826/12 electric motor [52], which is attached to a 13x8 APC propeller [53].

Sensors: The sensors used include a Microstrain 3DM GX3-25 Attitude and Heading Reference System (AHRS) (to measure attitude angles, angular rates, and linear accelerations) [54], a Ublox LEA-4T Global Positioning System (GPS) receiver (to determine position) [55], a VTI Technologies SCP1000 static pressure sensor (to determine altitude) [56], and two airdata probes built in-house (for airflow measurements), which use two Freescale Semiconductor MPXV7002DP [57] differential pressure sensor arrays. Details on the development of these probes are given in Section 2.2. The following are some useful comments on the use of some of the aforementioned sensors:

- The accelerometers and rate gyros of the 3DM GX3-25 AHRS give measurements of linear accelerations and angular rates about the aircraft's body reference frame. An internal Kalman filter algorithm uses these measurements, along with those of the AHRS magnetometers, to compute the Euler attitude angles. Adjustment of the Kalman filter weights may be necessary when operating in areas with severe magnetic disturbances. In such scenarios, it is advisable to place more confidence in the measurements given by the accelerometers and rate gyros than those obtained from the magnetometers.

- The Ublox GPS receiver determines position in terms of latitude, longitude, and altitude. Ground speed and course can be computed from this data, and will be utilized in the air-data probe calibration process. In general, the precision of the altitude measurement from this automotive grade GPS is insufficient for feedback control action when tracking rapid maneuvers. To circumvent this issue, a measurement of the altitude based on pressure is used instead. The pressure altitude is determined from the static pressure measurement of the SCP1000 sensor using the standard atmospheric model [58]. To account for non-standard day conditions, a bias is introduced so that the calculated pressure altitude on the ground matches the corresponding altitude determined by the GPS.

- Since the avionics bay on the Telemaster has an opening directly behind the motor and propeller, the pressure in the compartment is directly affected by the air mass exerted by the propeller. To alleviate the resulting pressure build-up, several holes have been drilled into the fuselage. To ensure that the SCP1000 sensor and the static ports of the MPXV7002DP sensor array measure the static pressure in the fuselage, felt covers have been installed on the inlet of the sensors.

- The noise levels of the sensors, based on initial calibration and assessment, are given in Table 2.1. The sensors are kept in a static condition, and measurements are recorded for a period of time, specified based on the sampling rate to obtain sufficient data. The standard deviation of the recorded data in this static condition is then used to describe the measurement noise characteristics. It is assumed in this dissertation that sensor uncertainties are only due to the presence of white noise, defined by the standard deviations

given in Table 2.1. These values will also be used in the design and simulation of feedback controllers.

Table 2.1: Measurement noise standard deviation (σ)

Variable	Unit	σ	max. rate	Variable	Unit	σ	max. rate
3DM GX3-25				LEA 4T			
X-Acceleration	m/s^2	0.008	1 kHz	Horizontal Position	m	0.30	4 Hz
Y-Acceleration	m/s^2	0.007	1 kHz	Vertical Position	m	0.55	4 Hz
Z-Acceleration	m/s^2	0.008	1 kHz	Speed over ground	m/s	0.01	4 Hz
Roll rate	deg/s	0.004	1 kHz	Course over ground	deg	0.00	4 Hz
Pitch rate	deg/s	0.004	1 kHz	SCP1000			
Yaw rate	deg/s	0.004	1 kHz	Pressure altitude	m	0.23	9 Hz
Roll angle	deg	0.042	1 kHz				
Pitch angle	deg	0.030	1 kHz				
Yaw angle	deg	0.043	1 kHz				

Communication radios: There are two radios onboard the platform: a Futaba R617FS eight-channel receiver [59] used for receiving commands from the R/C transmitter and a 900MHz XBee-Pro [60] used for transmitting data to the Ground Control Station (GCS). During multi-vehicle operations, the XBee can also be used to send and receive data among the vehicles. Specifically, this modem is capable of transmitting data to multiple modems within its range, provided that the same identification number is shared by both the sender and the receiver. Selective sending and receiving to simulate various communication topologies is achieved through the use of a specific header for the data being sent. The header consists of the origin of the information and the target. The target can be a single vehicle, multiple vehicles, or all vehicles within communication range. An example of a communication topology is given in Figure 2.1. This figure shows 3 UAVs interconnected over a communication network in a follow-the-leader type experiment, where the UAV in the red rectangle is the

leader and the other two UAVs are the followers. The leader sends data to the ground station and to both follower UAVs, and receives commands from the ground station. The follower UAVs send data to each other and to the ground station.

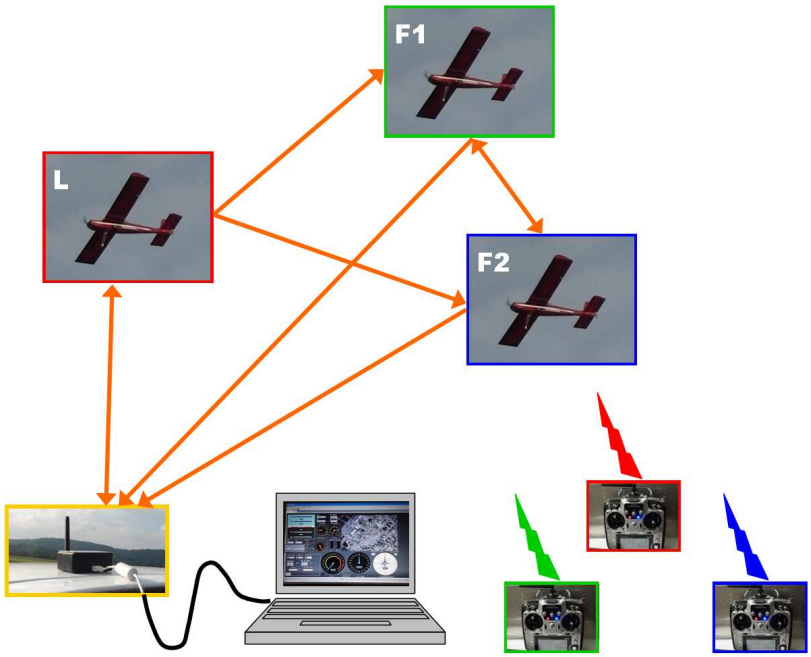

Figure 2.1: Three UAVs interconnected over a communication network through the use of the Xbee radio modems

Computers: The platform uses two low-cost, open source computers: Ardupilot Mega [12] and Gumstix Overo Fire [18]. Ardupilot is used solely for data management/hardware interfacing, which boils down to reading data from sensors, transmitting data to actuators and other instruments, and recording data for later analysis. This open source autopilot hardware has full autopilot capabilities. Its accompanying software is fully accessible and thus can be significantly altered, or, if necessary, rewritten. Ardupilot has different types of serial ports (I2C, SPI, and 4 UART) that can be used to connect to a variety of sensors. It

has 16 analog-to-digital converters to handle sensors with analog voltage outputs as well as 8 pulse-width modulation (PWM) input channels and 8 PWM output channels that can be used to drive servos. These features, along with the flexibility of the firmware, render the board adaptable to a variety of sensor configurations.

All control computations are carried out by the Gumstix Overo Fire computer, which is a single board computer powered by a 600MHz OMAP3530 microprocessor from Texas Instruments [61]. The Overo runs a Linux Angstrom distribution as its operating system, and the control algorithms are implemented in the Python programming language. This computer is primarily selected for its small size. With length, width, and height dimensions of 58 mm, 17 mm, and 4.2 mm, respectively, the Overo is advantageous for small UAV applications where space and mass are major constraints. The Overo exchanges data with Ardupilot via a serial port.

Before concluding this section, we will provide an overview of the data flow among the computers, sensors, and servos onboard the airplane. Note that the engine electronic speed controller (ESC) is considered as a servo in this scheme. A diagram of the data exchange between these components is given in Figure 2.2. As the diagram shows, Ardupilot receives data from the sensors and passes this data to the Overo computer. Overo then determines the necessary control inputs based on this data and sends the calculated control commands back to Ardupilot. Based on these commands, Ardupilot actuates the servos to drive the control surfaces. Ardupilot also composes a data package, consisting of all the measurements along with the control inputs to be sent to the onboard data logger as well as the ground

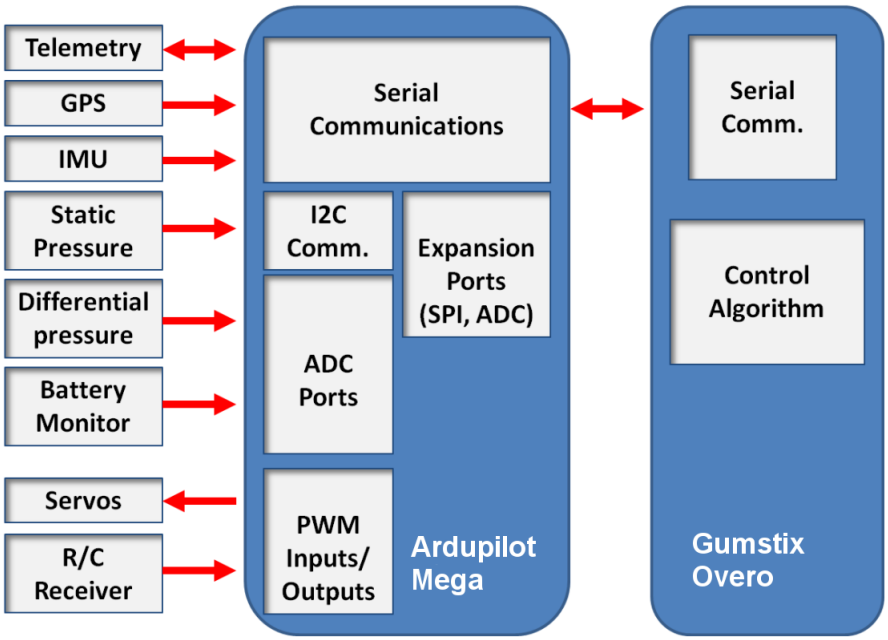

Figure 2.2: A data flow diagram of the fully programmable airborne system

station through the telemetry system. On the ground, the R/C transmitter has an on/off switch, which is used to activate/deactivate the autopilot mode. Once the autopilot mode is deactivated, i.e., the UAV is operating in manual mode, the Overo computer becomes passive, while Ardupilot listens to the manual commands from the R/C transmitter and passes them through to the servos.

2.2 Airdata Probe

One of the main goals of this work is deriving a reasonably accurate aerodynamic model of the UAV. As the aerodynamic forces and moments are highly dependent on the airflow velocity vector, it is essential for achieving this goal to be able to measure or, at least, estimate the angle of attack and sideslip. It is possible to estimate these angles rather accurately from the

inertial velocity and attitude angles in the absence of significant winds, as demonstrated in [62] and [63]. However, the condition of no significant winds is not easily met when dealing with small UAVs flying at low speeds. Thus, developing sensors that can measure the angle of attack and sideslip is instrumental for obtaining an adequate aerodynamic model of a small UAV.

2.2.1 Probe design and manufacturing

Common methods for measuring the airflow direction use mechanical vanes, differential pressure tubes, or null-seeking pressure tubes, as outlined in [64]. Due to potential design and implementation difficulties, the use of mechanical vanes and null-seeking pressure tubes is not preferable in the case of small UAVs where size and weight are major constraints. To the best of our knowledge, null-seeking pressure tubes have never been used on small UAVs. Mechanical vanes, on the other hand, have been used successfully on a small UAV in [6] to measure the airflow direction.

There are various differential probe designs that can be used to measure the airflow vector. These designs vary in the shape of the probe tip and/or the number of pressure holes. A comparison of measurement results using different designs of differential probes is given in [14]. As stated in [65], the accuracy of the differential probe measurements is proportional to the number of pressure holes, and hence comes at the expense of increased manufacturing complexity. The shape of the tip also affects the accuracy of the probe measurements. For

instance, an airdata probe with a perfectly spherical tip shape gives accurate measurements without the need for calibration because there are mathematical formulas available in the literature that can be used to directly compute the airspeed, angle of attack, and sideslip from the pressure differences in this case. Manufacturing such a probe, however, is challenging. In our case, we have opted to design a five-hole probe with a conical tip shape for the following reasons: (1) There are many publications on the calibration and testing of five-hole probes; and (2) it is possible to manufacture this probe from commercially available materials using conventional machining techniques. We have built and calibrated two probes, each mounted on a wing of the airplane at a distance of 16 inches from the plane of symmetry with the port holes situated about 6 inches from the wing leading edge; see Figure 1.1. This configuration maintains lateral balance and ensures that the effects of the propeller wash and wing interference on the airflow measurements are insignificant. Both probes are comparable in quality and are used together in our platform for redundancy. In the following, we will just focus on one probe.

The CAD model of the five-hole probe that we have designed is given in Figure 2.3, along with the final product. The figure shows the holes arrangement at the tip of the probe. The probe consists of five small aluminum tubes, glued together using JB Weld plastic steel to form the desired plus-sign configuration, and then encased in a larger aluminum tube. The tubes used are sold in most hardware stores. The length of the probe is 203.2 mm, and the tip diameter is 5.56 mm. Each of the five small tubes has an outer diameter of 1.60 mm and an inner diameter (hole diameter) of 0.89 mm. Then, the ratio of the hole diameter to the

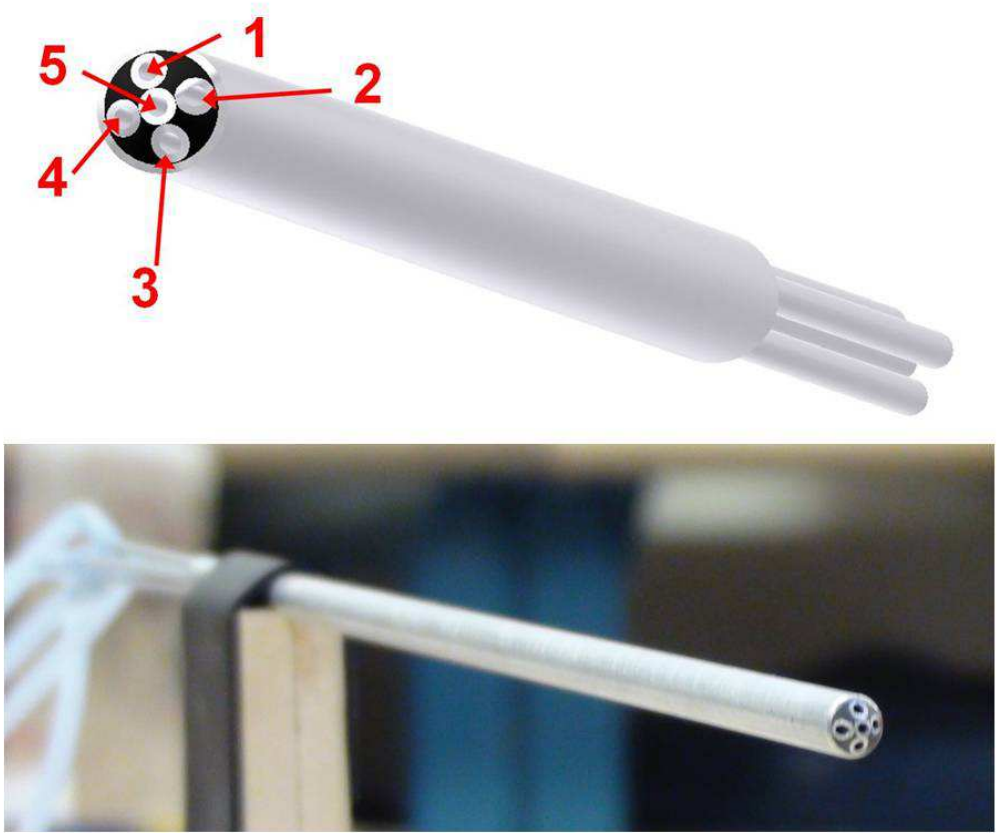

Figure 2.3: Five-hole probe and its CAD drawing

tip diameter is 0.16. Although this ratio is relatively small compared to the one in [14], the number is fairly close to newer designs, such as those in [66] and [67].

The tip design is conical with 90° angle. As observed in [14, 15], this tip design yields reasonably accurate measurements of the angle of attack and sideslip at low speeds and can be calibrated to measure these angles within the range ±22.5°. For such a tip design, there are no mathematical formulas that directly relate the pressure differences to the angle of attack and sideslip, as in the case of the perfectly spherical tip shape. Hence, the relationship between the measured pressure at each hole (port) and the airflow velocity vector can only be determined by experiment. The experiment, along with the required data processing and

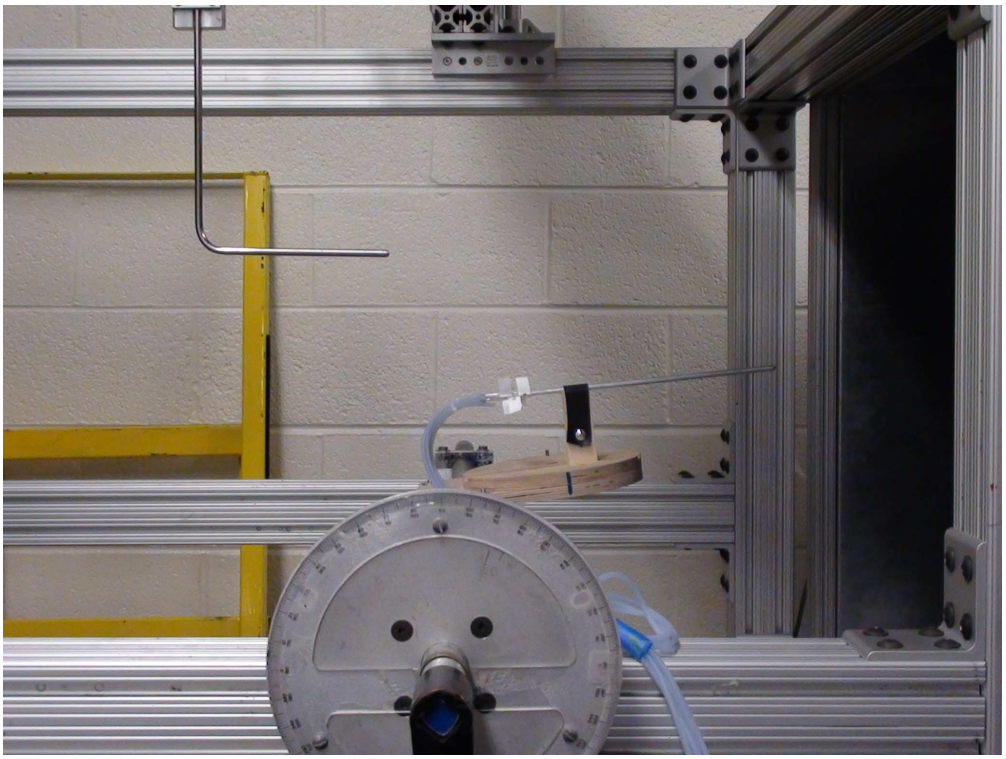

Figure 2.4: Five-hole probe calibration process

curve fitting, is henceforth referred to as probe calibration.

2.2.2 Probe calibration

The calibration of the probe was performed in the Subsonic Open Jet Wind Tunnel at Virginia Tech following the procedure given in [16]. Figure 2.4 shows the experiment setup. In this setup, the probe is placed in the middle of the test section of the wind tunnel on a turntable that can be rotated to vary the sideslip angle. The angle of attack can also be varied by rotating the rod on which the turntable is mounted. During calibration, the rotational speed of the wind tunnel fan is increased from 0 to 1180 rpm (maximum achievable speed)

in increments of 200. The airflow velocity (in m/s) is about 0.021 times the fan rotational speed (in rpm). Hence, at a fan speed of 1180 rpm, the resultant airflow velocity is about 24.78 m/s. For each considered rotational speed, the angle of attack and sideslip are varied from $-20°$ to $+20°$ in $5°$ increments, with the pressure at all ports of the probe measured in each configuration. The pressure measurements for various values of the angle of attack and sideslip at a fan speed of 400 rpm are shown in Figure 2.5.

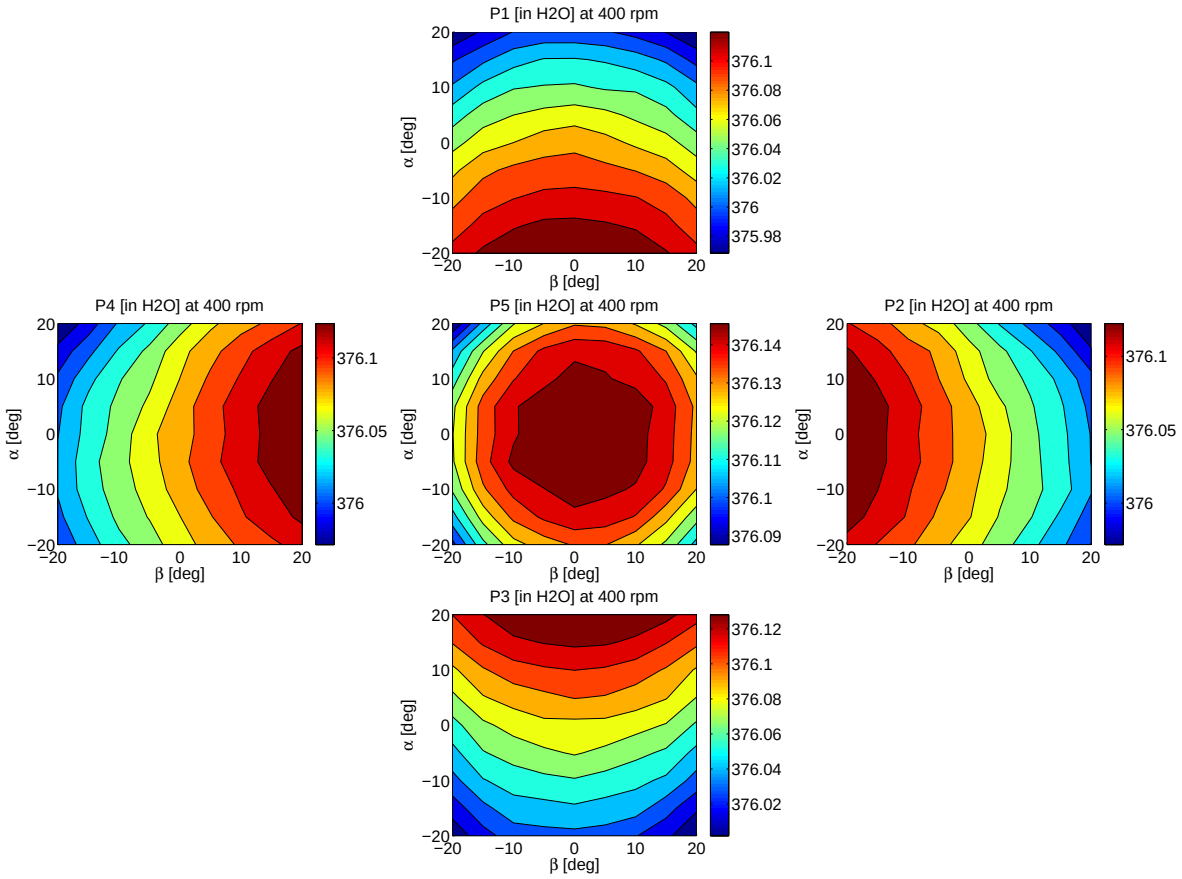

Figure 2.5: Pressure distribution at various values of the angle of attack (α) and sideslip (β) for all ports of the probe

As mentioned earlier, during calibration process, the pressures at all ports of the probe, i.e., $p_1, p_2, \ldots, p_5$, are measured for each configuration, where p_i is the pressure at port i and

the port numbering is shown in Figure 2.3. As given in [16], it is possible to approximately compute the angle of attack, α, and sideslip, β, from the pressures $p_1, \ldots, p_5$ as follows:

$$\alpha = \frac{a_0 + a_1 C_{p_\alpha} + a_2 C_{p_\beta} + a_3 C_{p_\beta}^2 + a_4 C_{p_\beta}^3}{1 + a_5 C_{p_\alpha} + a_6 C_{p_\beta} + a_7 C_{p_\beta}^2 + a_8 C_{p_\beta}^3}, \tag{2.1}$$

$$\beta = \frac{b_0 + b_1 C_{p_\alpha} + b_2 C_{p_\beta} + b_3 C_{p_\alpha}^2 + b_4 C_{p_\beta}^2 + b_5 C_{p_\alpha} C_{p_\beta}}{1 + b_6 C_{p_\alpha} + b_7 C_{p_\beta} + b_8 C_{p_\alpha}^2 + b_9 C_{p_\beta}^2 + b_{10} C_{p_\alpha} C_{p_\beta}}, \tag{2.2}$$

$$\text{where} \quad p_{av} = 0.25(p_1 + p_2 + p_3 + p_4), \quad C_{p_\alpha} = \frac{p_3 - p_1}{p_5 - p_{av}}, \quad \text{and} \quad C_{p_\beta} = \frac{p_4 - p_2}{p_5 - p_{av}}.$$

Given the data collected in the calibration process, the calibration coefficients $a_0, a_1, \ldots, a_8$ and $b_0, b_1, \ldots, b_{10}$ in equations (2.1–2.2) are obtained by solving a least-squares problem by linear regression. The fitting surface and goodness of fit for the angle of attack and sideslip are shown in Figure 2.6 and 2.7. The coefficient of determination, or R^2, value for the angle of attack data fitting is 0.9949 and that for the sideslip data fitting is 0.9972.

The airspeed V_a can be computed from the static pressure p_s and total pressure p_t, namely,

$$V_a = \sqrt{\frac{2(p_t - p_s)}{\rho}},$$

where ρ denotes the air density. While it is possible to measure p_s and p_t during calibration by using a pitot-static tube aligned with the direction of the airflow, it is impractical to replicate this setup on the actual UAV. Alternatively, as suggested in [16], we can approximately obtain the values of the static and total pressures, and hence the airspeed, from the angle of attack

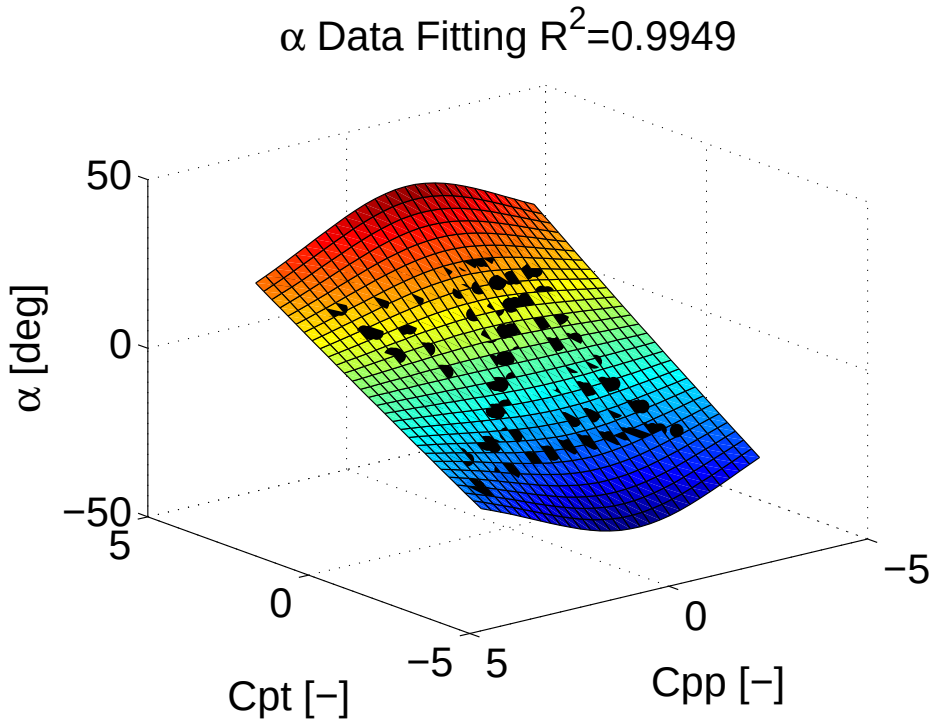

Figure 2.6: Angle of attack surface fitting

and sideslip, along with the pressures $p_1, \ldots, p_5$, as follows:

$$p_s = \frac{C_{p_5} p_{av} - C_{pav} p_5}{C_{p_5} - C_{pav}} \quad \text{and} \quad p_t = p_s + \frac{p_5 - p_s}{C_{p_5}}, \text{where}$$

$$C_{p_5} = \frac{c_0 + c_1\beta + c_2\beta^2 + c_3\beta^3 + c_4\alpha + c_5\alpha^2}{1 + c_6\beta + c_7\alpha + c_8\alpha^2 + c_9\alpha^3}, \text{and} \tag{2.3}$$

$$C_{pav} = d_0 + d_1\beta + d_2\alpha + d_3\beta^2 + d_4\alpha^2 + d_5\alpha\beta + d_6\beta^3 + d_7\alpha^3 + d_8\beta\alpha^2 + d_9\beta^2\alpha. \tag{2.4}$$

The coefficients $c_0, c_1, \ldots, c_9$ and $d_0, d_1, \ldots, d_9$ in equations (2.3–2.4) are also obtained by solving a least-squares problem by linear regression. As shown in Figure 2.8 and 2.9, the R^2 value for the C_{p_5} data fitting is 0.9788 and that for the C_{pav} data fitting is 0.9676. Our calibration results, and specifically the R^2 values, are comparable to those obtained

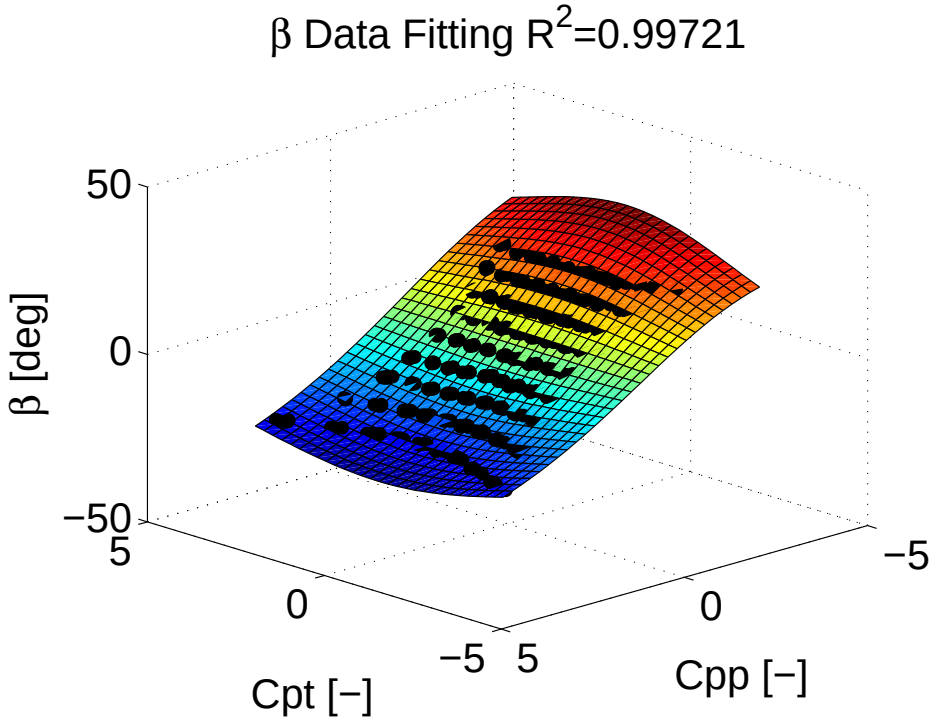

Figure 2.7: Angle of sideslip surface fitting

in [16]. This observation is noteworthy considering that the authors in [16] utilize a more sophisticated manufacturing process in building their probe, in addition to a closed-circuit wind tunnel for calibration, which generally generates less turbulent airflow compared to an open-jet tunnel.

2.2.3 Validation and analysis

The calibration results are validated by wind-tunnel testing as well as flight testing. In the wind-tunnel tests, the probe is evaluated at a low airflow speed of about 12 m/s and a high airflow speed of about 20 m/s; these speeds correspond to the lower and upper limits of typical operation of the UAV platform, respectively. For each speed, the angle of attack is

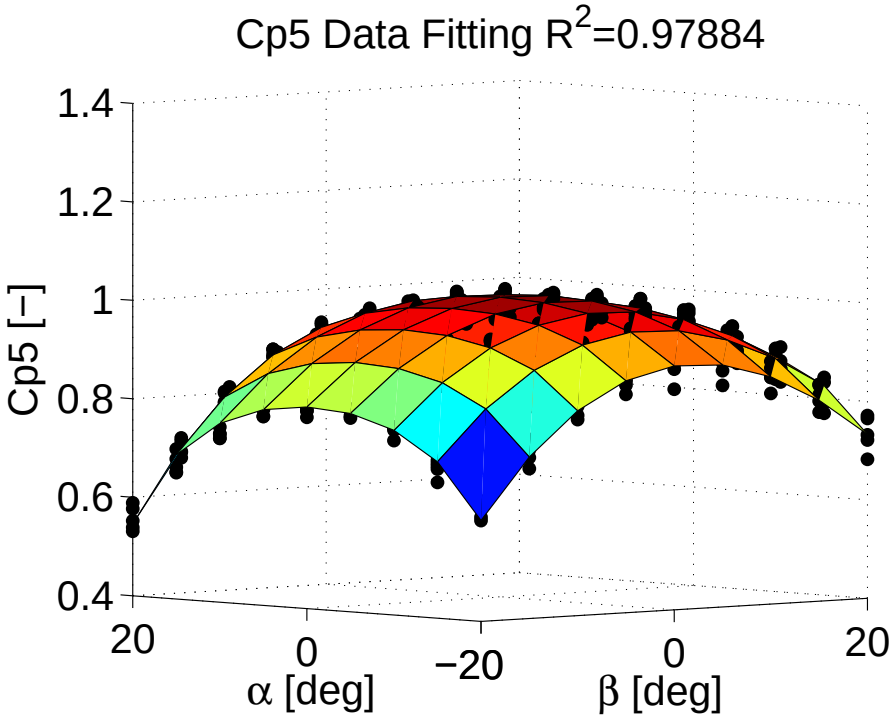

Figure 2.8: C_{p_5} surface fitting

varied from $-10°$ to $10°$ in $5°$ increments at $0°$ sideslip, and similarly, the sideslip is varied from $-10°$ to $10°$ in $5°$ increments at $0°$ angle of attack. The pressure sensors and analog-to-digital converter used in these tests are the same as those used onboard the UAV, as opposed to the high-precision pressure scanner in the wind-tunnel facility, which is used in the calibration process. In addition, a first-order low-pass filter with a cutoff frequency of 1 Hz is applied to the output of each of the pressure sensors. We choose these specific wind-tunnel validation tests as they are comparable to the flight tests carried out for aerodynamic modeling purposes.

Figure 2.10 shows the validation results for the considered values of the angle of attack with $0°$ sideslip at 12 m/s and 20 m/s airflow speeds. Specifically, at each speed and for each

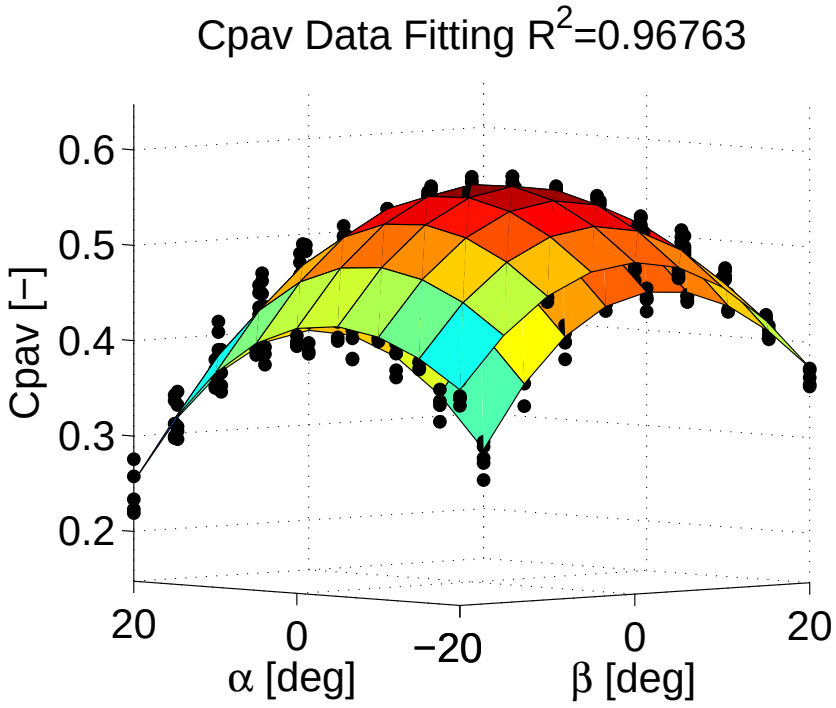

Figure 2.9: $C_{p_{av}}$ surface fitting

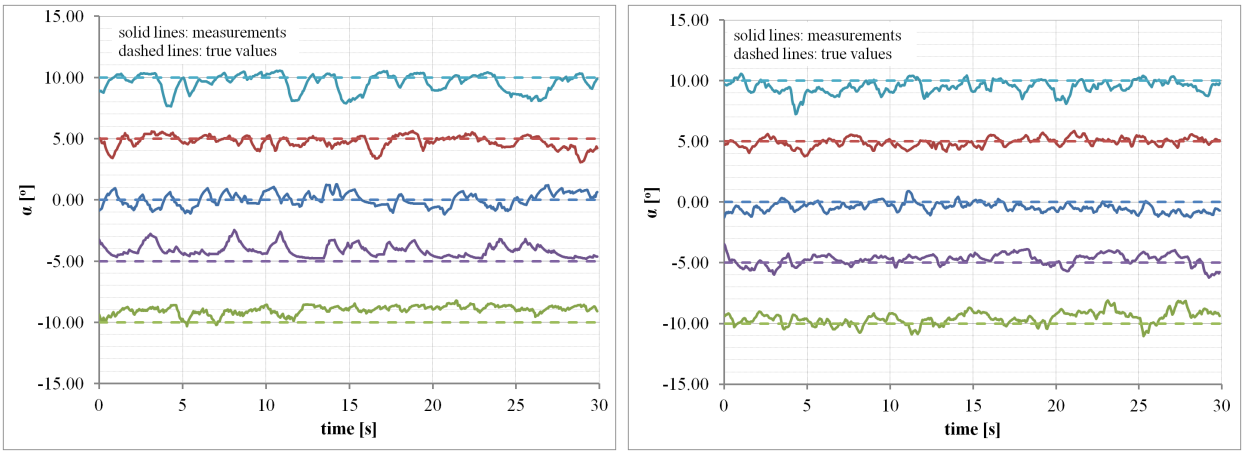

Figure 2.10: Angle of attack validation at 12 m/s airflow speed (left) and 20 m/s speed (right)

value of the angle of attack, the test is run for 30 seconds, and samples are collected at 20 Hz, i.e., we end up with 600 samples in total for each test. Thus, the total number of samples for all the angle of attack tests at each speed is 3000. The measurement error is

defined as the difference between the measured value and true value of the angle of attack. We then have 3000 samples of the measurement error at 12 m/s airflow speed (low speed) and another 3000 samples at 20 m/s (high speed). Note that the measurement discrepancies are due to calibration errors as well as instrumentation errors (because of the use of the actual, less accurate sensors). The following analysis of the aforementioned two sets of data is based on [68]. Given these data, we find that the angle of attack can be measured with an accuracy of 0.67° at low speed and 0.46° at high speed, where the accuracy is given in terms of the root-mean-square (RMS) of the measurement error. The RMS is defined as $\text{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} e_i^2}$, where N is the number of samples, which is 3000 for each speed, and e_i is the i^{th} sample of the measurement error. The RMS measure is used because the mean of the error is not zero; specifically, there is a measurement bias of 0.27 ° at low speed and −0.06 ° at high speed. Assuming the error is normally distributed about the nonzero mean value, then, for each speed, the number of samples that lie within ±RMS is equal to $\frac{1}{2}\left(\text{ERF}\left(\frac{\text{RMS}-\mu}{\sigma\sqrt{2}}\right) + \text{ERF}\left(\frac{\text{RMS}+\mu}{\sigma\sqrt{2}}\right)\right)$, where σ is the standard deviation of the error, μ is the mean value, and $\text{ERF}(\eta)$ is the error function defined as $\text{ERF}(\eta) = \frac{1}{\sqrt{2}}\int_{-\eta}^{\eta} e^{-\xi^2} d\xi$. Hence, 80.6% of the measurement error samples lie within ±0.67° at low speed and 83.9% lie in the range ±0.46° at high speed. Additionally, 95% of the error samples lie in the range ±0.99° at low speed and ±0.64° at high speed.

Figure 2.11 shows the validation results for the considered values of the sideslip with 0° angle of attack at 12 m/s and 20 m/s speeds. The samples of the measurement error are collected in the same way as in the angle of attack case. The measurement error in this case also tends

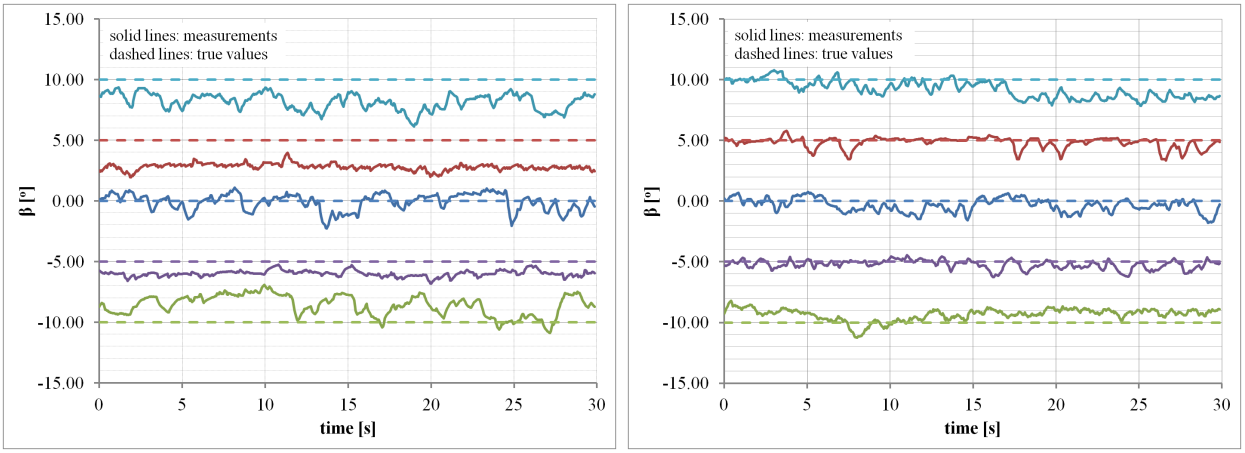

Figure 2.11: Angle of sideslip validation at 12 m/s airflow speed (left) and 20 m/s speed (right)

to decrease as the airflow speed increases. The RMS is 0.92° at low speed and 0.54° at high speed. 67.9% of the samples lie within ±0.92° at low speed and 81.9% lie within ±0.54° at high speed. The 95% confidence intervals are $[-1.37°, 1.37°]$ at low speed and $[-0.79°, 0.79°]$ at high speed.

The probe is next tested in flight. The test entails maintaining the airplane in straight (and preferably level) flight, as sensor measurements, including those from the probe, are recorded. In this scenario, as long as the wind disturbances are relatively insignificant, the angle of attack should be approximately equal to the pitch angle minus the flight path angle. In our tests, we managed to maintain the airplane in almost level flight (i.e., at a roughly zero flight path angle). Figure 2.12 shows that the measured values of the angle of attack obtained from the probe correlate well with the measurements of the pitch angle from the AHRS sensor. Note that the 4° bias between the measurements from the probe and those from the AHRS is due to the fact that the probe is installed on the wing, which has an angle of incidence of approximately 4°. Note that this type of test is not suitable for the validation

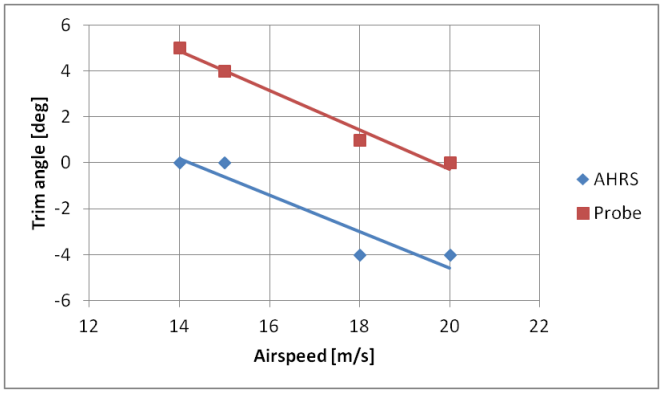

Figure 2.12: In flight measurement of the angle of attack

of sideslip measurements due to the difficulties in flying an R/C airplane at a constant angle of sideslip.

Chapter 3

Mathematical Model Development

In order to design dynamically feasible trajectories offline and develop high-performance feedback controllers, a reasonably accurate model of the aircraft dynamics is required. A combination of semi-empirical, frequency domain, and time-domain identification techniques are utilized to obtain a mathematical representation of each major component of the system. This chapter is divided into five sections: the first gives the equations of motion of the UAV; the second presents the test method used to determine the moments of inertia; the third describes the experiment conducted to obtain the frequency response of each actuator; the fourth focuses on the propeller thrust modeling; and the last section presents the parameter estimation process used for deriving the aerodynamic model of the UAV.

3.1 Aircraft Equations of Motion

The dynamics of the UAV are described by standard rigid-body six-degree-of-freedom equations of motion in the aircraft body reference frame. Before giving these equations, we define the following reference frames:

$\{I\}$:= inertial reference frame with X-axis pointing North, Y-axis pointing East, and Z-axis pointing down (NED)

$\{B\}$:= body reference frame, fixed to the center of gravity of the UAV, with X-axis pointing to nose, Y-axispointing to starboard wing, and Z-axis pointing down

The following notations are introduced to concisely present the equations of motion, as suggested by [69, 46, 70]:

P = position of the center of gravity in $\{I\} = [N,\ E,\ -H]^T$

V = linear velocity of the center of gravity relative to $\{I\}$, expressed in $\{B\}$

$= [u,\ v,\ w]^T$

V_w = linear velocity of the wind relative to $\{I\}$, expressed in $\{B\} = [u_w,\ v_w,\ w_w]^T$

$\bar{V}$ = linear velocity of the center of gravity relative to wind, expressed in $\{B\}$

$= [u - u_w,\ v - v_w,\ w - w_w]^T$

V_a = airspeed of the UAV $= \sqrt{(u-u_w)^2 + (v-v_w)^2 + (w-w_w)^2}$

Ω = angular velocity of $\{B\}$ relative to $\{I\}$, expressed in $\{B\} = [p,\ q,\ r]^T$

Λ = vector of Euler angles with respect to $\{I\} = [\phi,\ \theta,\ \psi]^T$

${}^I_B\mathcal{R}$ = rotation matrix from $\{B\}$ to $\{I\}$ in SO(3), given in terms of Λ

G = gravitational vector $= g[-\sin\theta,\ \cos\theta\sin\phi,\ \cos\theta\cos\phi]^T$,

where g is the gravitational constant

We assume that the earth is flat and the gravity field is constant ($g = 9.80665\,\mathrm{m/s^2}$). Let $\delta = [\delta_e,\ \delta_a,\ \delta_r,\ \delta_t]^T$, where δ_e, δ_a, and δ_r denote the elevator, aileron, and rudder deflections, respectively, and δ_t designates the throttle input. Note that the aforementioned control surface deflections are the actual deflections, that is, the outputs of the servomotors used to deflect the control surfaces on the UAV. The commanded deflections, which are the inputs to the servomotors, are denoted by δ_e^c, δ_a^c, and δ_r^c. The three servomotors used onboard the UAV are identical and each is modeled as a second-order system obtained by measuring the frequency response, as discussed in Section 3.3. We can then express the equations of motion as follows:

$$\dot{V} = m^{-1}F(\bar{V}, \Omega, \delta) + G - \Omega \times V \tag{3.1}$$

$$\dot{\Omega} = J^{-1}M(\bar{V}, \Omega, \delta) - J^{-1}(\Omega \times J\Omega) \tag{3.2}$$

$$\dot{P} = {}^I_B\mathcal{R}(\Lambda)V \tag{3.3}$$

$$\dot{\Lambda} = \mathcal{E}(\phi, \theta)\Omega \tag{3.4}$$

where $m = 3.24\,\mathrm{kg}$ is the mass of the airplane, J is the moment of inertia tensor, $F(\bar{V}, \Omega, \delta)$

denotes the aerodynamic and propulsion forces, $M(\bar{V}, \Omega, \delta)$ denotes the aerodynamic and propulsion moments, and

$$\mathcal{E}(\phi, \theta) = \begin{bmatrix} 1 & \sin\phi \tan\theta & \cos\phi \tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix}.$$

The ground testing procedures used to obtain the moments of inertia of the airplane, and hence the inertia tensor J, are described in Subsection 3.2. Subsections 3.4 and 3.5 give the aerodynamic and propulsion models. Note that we assume the propeller does not significantly affect the airflow over the airplane. Consequently, we approach the aerodynamic and propulsion modeling separately. We can then write F and M as

$$\begin{aligned} F(\bar{V}, \Omega, \delta) &= F_A(\bar{V}, \Omega, \delta_e, \delta_a, \delta_r) + F_P(\bar{V}, \delta_t), \\ M(\bar{V}, \Omega, \delta) &= M_A(\bar{V}, \Omega, \delta_e, \delta_a, \delta_r) + M_P(\bar{V}, \delta_t), \end{aligned} \tag{3.5}$$

where $(\cdot)_A$ and $(\cdot)_P$ denote the aerodynamic and propulsion components, respectively.

3.2 Moments of Inertia

The moments of inertia of a full-scale aircraft about the roll and pitch axes are typically determined using the compound pendulum method, whereas the moment of inertia about the yaw axis is obtained using the bifilar pendulum approach, as described in [25, 26, 27]. In

the case of a small UAV, the vehicle can be be easily tilted sideways, and so the moment of inertia about the yaw axis can also be determined using the compound pendulum method. That way, the same setup can be used to perform all the experiments. The bifilar pendulum method can also be used to determine all the moments of inertia of a small UAV, as shown, for example, in [71], provided that the testing setup is appropriately configured to avoid the inadvertent excitation of the swaying mode, which could otherwise result in significant measurement errors as discussed in [72]. Thus, either of the aforementioned test methods can be used to compute the moments of inertia of a small UAV. In our case, we have used the compound pendulum method simply for convenience.

The moments of inertia about the roll, pitch, and yaw axes are determined in three separate experiments. In each experiment, the UAV is positioned appropriately on a wire suspended cradle, as shown in Figure 3.1, in order to obtain the moment of inertia about the desired axis. The experiment entails finding the oscillation period of the pendulum and then computing from this period the moment of inertia by applying a simple mathematical formula. It is indicated in [26] and observed in our preliminary tests that the compound pendulum method is sensitive to the length of the suspending wires. Specifically, the accuracy of the measurements obtained using this method degrades as the distance from the pivot axis of the pendulum to the center of gravity of the UAV, denoted by L, increases. This finding can also be verified by examining the mathematical formula used to compute the moment of inertia and doing some sensitivity analysis. The formula for computing the moment of inertia, $\mathcal{I}$, is based on the linearized equation of motion of the compound pendulum and

Figure 3.1: Moments of inertia testing setup: wire suspended cradle (left); setup to determine the moment of inertia about the pitch axis (top middle), the roll axis (bottom middle), and the yaw axis (right)

is given by $\mathcal{I} = \dfrac{mgLT^2}{4\pi^2} - mL^2$, where T is the oscillation period of the pendulum, g is the acceleration due to gravity, and L is as defined before. As the major source of error in computing the moment of inertia is the inaccuracy in the measurement of the oscillation period, it is important to examine the local sensitivity of $\mathcal{I}$ with respect to T. Suppose that the true value of the moment of inertia is $\mathcal{I}_n$, and the corresponding oscillation period is T_n. Then, the local sensitivity of $\mathcal{I}$ with respect to T is $\dfrac{\partial \mathcal{I}}{\partial T} = \dfrac{1}{\pi}\sqrt{mgL\left(\mathcal{I}_n + mL^2\right)}$. It is clear from the preceding equation that, as L increases, any small discrepancy in the period measurement would yield a more significant error in the computation of the moment of inertia.

As mentioned before, the test setup for determining the moments of inertia is shown in Figure 3.1. The cradle that holds the vehicle is connected to the pivot points on the ceiling

by four braided steel wires. The cradle consists of a wooden frame, along with four eyescrews to attach the frame to the steel wires. The cradle is located at 1.18 m from the pivot axis and has a mass of 1.27 kg. When placing the vehicle on the cradle, it is important to make sure that the center of gravity of the vehicle is as close as possible to the center of the cradle. The test setup does not require any specific mounting apparatus, and therefore can be used for multiple airplane models of similar size. Prior to the testing of the UAV, a calibration test is performed to determine the moment of inertia of the cradle, following the standard compound pendulum procedure. The moment of inertia of the cradle about the pivot axis is found to be 1.96 kg m^2. Three experiments are then performed to determine the moments of inertia of the UAV about the roll, pitch, and yaw axes (X, Y, and Z axes in the body reference frame), which are denoted by $\mathcal{I}_{xx}$, $\mathcal{I}_{yy}$, and $\mathcal{I}_{zz}$, respectively. The location of the center of gravity of the UAV in each of these experiments slightly changes based on the orientation of the airplane with respect to the cradle. These locations, measured from the pivot axis of the pendulum, are 1.22, 1.35, and 1.21 m corresponding to the configurations used to determine $\mathcal{I}_{xx}$, $\mathcal{I}_{yy}$, and $\mathcal{I}_{zz}$, respectively. The mass of the UAV is 3.24 kg and is slightly increased to 3.31 kg when determining $\mathcal{I}_{xx}$ due to the addition of a wooden bar to hold the airplane in place during this test.

To determine the moment of inertia about a specific axis of the UAV, 3 tests are performed. In each of these tests, the time required to complete 50 oscillations is recorded. Then, the average of the three recorded values, denoted by T_{50}, is used to compute the moment of inertia by applying the following equation:

Table 3.1: Moments of inertia of the Telemaster UAV

	$\mathcal{I}_{xx}$	$\mathcal{I}_{yy}$	$\mathcal{I}_{zz}$	
T_{50}	114.89	118.25	115.36	s
ω_n	2.73	2.66	2.72	rad/s
$\mathcal{I}$	0.21	0.31	0.48	kgm^2

$$\mathcal{I}_2 = \frac{g(m_1 L_1 + m_2 L_2)}{\omega_n^2} - \mathcal{I}_1 - m_2 L_2^2,$$

where $\omega_n = 100\pi/T_{50}$ is the frequency of oscillation, m_1 is the mass of the cradle, m_2 is the mass of the UAV, L_1 is the location of the center of gravity of the cradle, L_2 is the location of the center of gravity of the UAV (both locations measured from the pivot axis), $\mathcal{I}_1$ is the moment of inertia of the cradle about the pivot axis, and $\mathcal{I}_2$ is the moment of inertia of the UAV about its body axis. Table 3.1 gives the calculated values of the moments of the inertia of the UAV. Concerning the computation of the inertia tensor J, it is reasonable to assume in our case that the "cross-product-of-inertia" terms, $\mathcal{I}_{xy}$, $\mathcal{I}_{xz}$, and $\mathcal{I}_{yz}$, are all zeros, and hence, we have

$$J = \begin{bmatrix} \mathcal{I}_{xx} & -\mathcal{I}_{xy} & -\mathcal{I}_{xz} \\ -\mathcal{I}_{xy} & \mathcal{I}_{yy} & -\mathcal{I}_{yz} \\ -\mathcal{I}_{xz} & -\mathcal{I}_{yz} & \mathcal{I}_{zz} \end{bmatrix} = \begin{bmatrix} 0.21 & 0 & 0 \\ 0 & 0.31 & 0 \\ 0 & 0 & 0.48 \end{bmatrix}.$$

3.3 Servo Model

Three servomotors are used to deflect the control surfaces on the airplane. These servos are identical and their dynamics are described by the same second-order system model. For

instance, the system equations of the servo used to deflect the elevator are:

$$\left[\begin{array}{c} \dot{x}_{1s} \\ \dot{x}_{2s} \\ \hline \delta_e \end{array}\right] = \left[\begin{array}{cc|c} 0 & 1 & 0 \\ -\omega_{ns}^2 & -2\zeta_s\omega_{ns} & \omega_{ns}^2 \\ \hline 1 & 0 & 0 \end{array}\right] \left[\begin{array}{c} x_{1s} \\ x_{2s} \\ \hline \delta_e^c \end{array}\right], \tag{3.6}$$

where x_{1s} and x_{2s} are the internal states of the servo, δ_e^c is the commanded elevator deflection, δ_e is the actual deflection, ω_{ns} is the natural frequency of the servo mechanism, and ζ_s is the damping ratio.

This model, and specifically the natural frequency and damping ratio of the servo, are obtained experimentally by measuring the frequency response. The frequency response test is performed by sending continuous sinusoidal command inputs at different frequencies to the servo, one at a time, and, in each case, recording the commanded signal and the corresponding response simultaneously. The frequencies are chosen to cover a wide range about the cutoff frequency of the servo. The test setup is given in Figure 3.2.

To measure the servo response, a potentiometer is mounted such that the shaft of the potentiometer is aligned with that of the servo. This potentiometer is supplied with 5 V voltage by an Ardupilot board and produces $0 - 5$ V output based on the angular position of its shaft. The Ardupilot is also used to generate the reference command and send it to the servo, as well as record the output from the potentiometer. The potentiometer readings for a number of commanded servo positions are recorded prior to the frequency response test in order to determine the calibration curve for this sensor. At each test frequency, the magnitude and

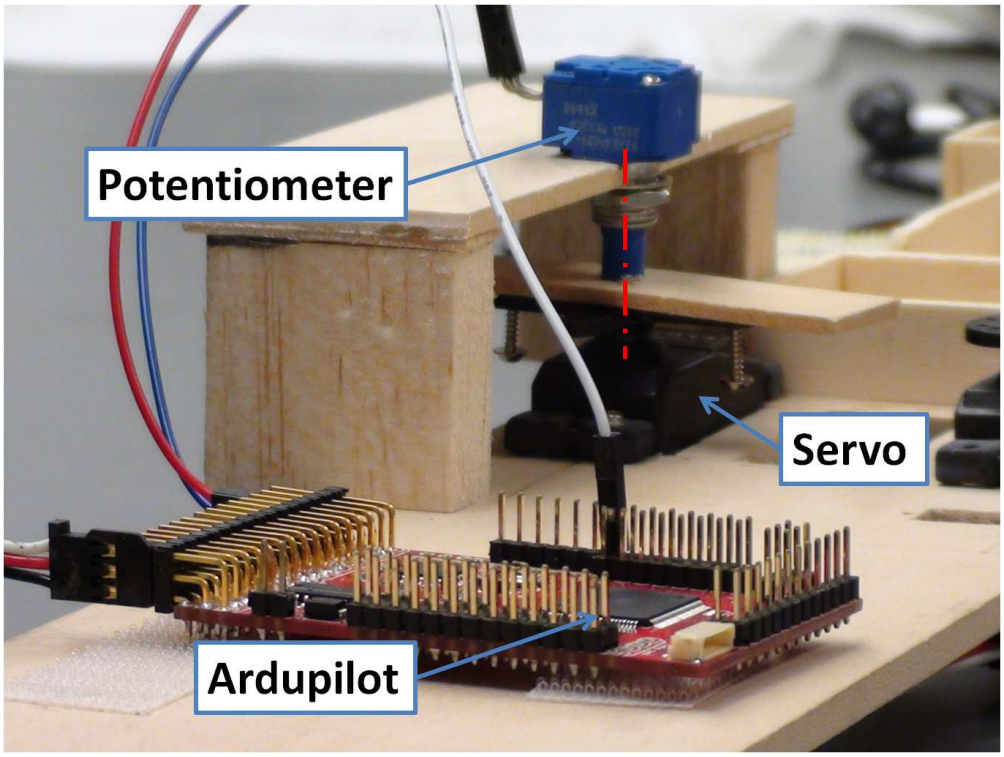

Figure 3.2: Test setup for the frequency domain identification of the Futaba S3152 servo model

phase shift of the frequency response are determined by comparing the commanded input sinusoid and the measured steady-state output signal; the values obtained are given in Table 3.2.

Table 3.2: Futaba S3152 frequency response

Frequency Hz	Magnitude -	Phase rad
0.01	1.00	0.00
0.10	1.00	-0.04
1.00	0.96	-0.57
2.00	0.81	-1.51
4.00	0.31	-3.02
5.00	0.19	-3.14

Frequency domain fitting is applied to the test data, with the natural frequency and damping

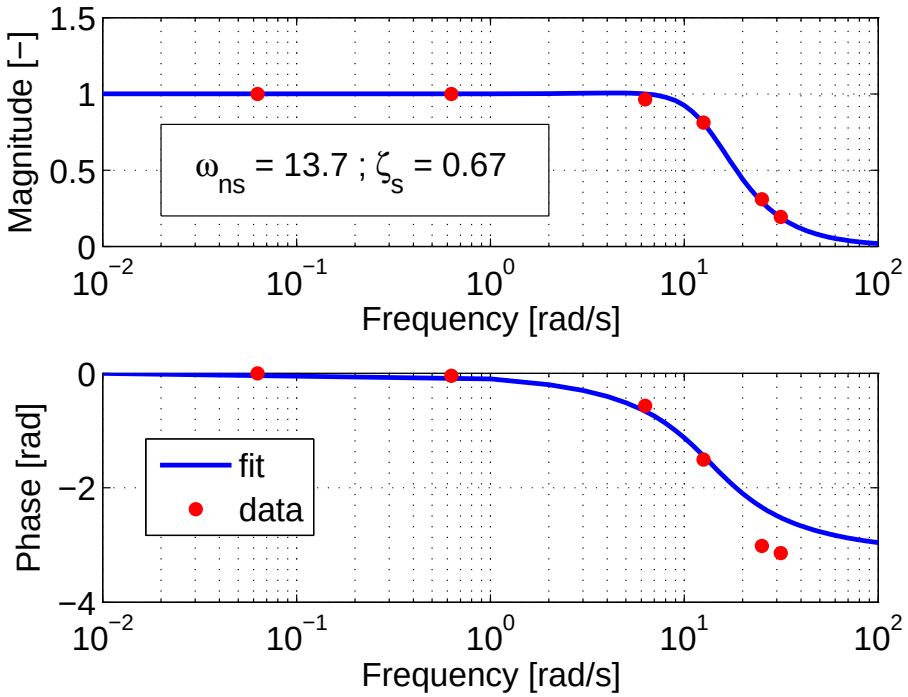

Figure 3.3: Curve fitting of the servo test data

ratio used as the fitting parameters. Since the system is fairly simple, the natural frequency and damping ratio are adjusted manually to obtain a satisfactory fit. A natural frequency of $\omega_{ns} = 13.7$ rad/s and a damping ratio of $\zeta_s = 0.67$ are obtained for the Futaba S3152 servo used in this work. The test data in Table 3.2 and the second-order system fit are shown in Figure 3.3.

3.4 Propulsion Model

The propulsion system of the Telemaster UAV consists of a 760 RPM/Volt electric motor and a 13 × 8 inch propeller. The engine is powered by a 3S LiPo battery and controlled by

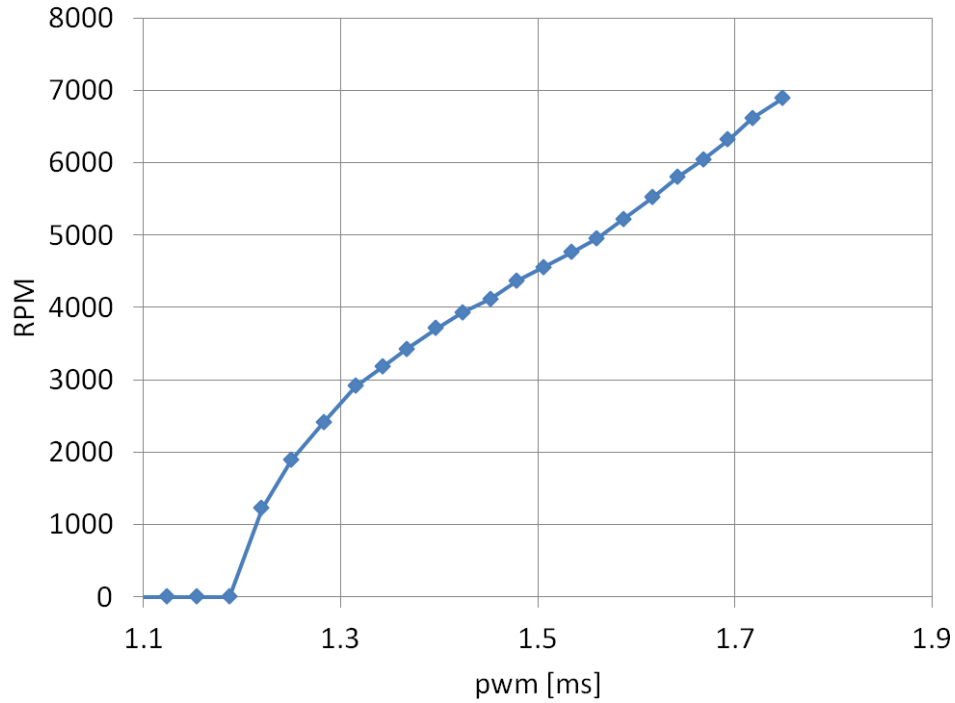

Figure 3.4: Relationship between the throttle command δ_t, which is a PWM signal, and the engine RPM

a 40 Amps speed controller. We now derive a propulsion model of our system. We assume the propeller axis to be perfectly aligned with the body X-axis. As a result, the propeller thrust will be applied along the body X-axis only, that is, $F_P(\bar{V}, \delta_t) = [T(V_a, \delta_t),\ 0,\ 0]^T$, and further, the thrust will not generate any moments, namely $M_P(\bar{V}, \delta_t) = 0$. The propulsion system model maps the throttle command δ_t, which is a PWM (pulse-width modulation) signal, to the generated propeller thrust $T(V_a, \delta_t)$, given a certain airspeed. We assume that the dynamics of the propulsion system are fast enough that we may ignore the transient behavior. We also assume that, even under nonzero airspeed conditions, the electronic speed controller (ESC) is able to ensure that, for a constant throttle command, the engine runs at a constant RPM.

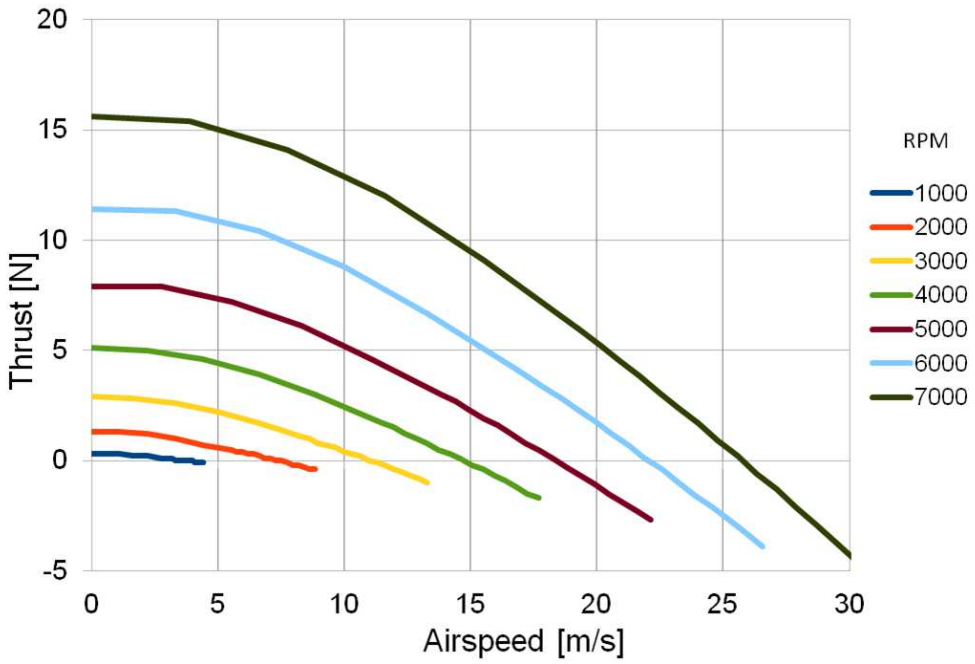

Figure 3.5: Thrust prediction as a function of airspeed and engine RPM (Javaprop)

We have obtained experimentally a look-up table, represented graphically in Figure 3.4, that can be used to map a throttle command into the corresponding engine RPM through interpolation. Then, given the engine RPM and the current airspeed, we can compute the generated propeller thrust using look-up tables obtained from the propeller analysis applet, Javaprop [24]. This applet uses blade element theory to return a thrust profile based on the propeller geometry. The complete thrust model, which relates the engine RPM and airspeed to propeller thrust, is given in Figure 3.5. Notice that, for a propeller rotating at a constant RPM, increasing the airspeed decreases the effective angle of attack of the propeller blade and, as a result, decreases the propeller thrust. Increasing the airspeed beyond a certain value will reverse the effective angle of attack, and hence cause the propeller to generate drag

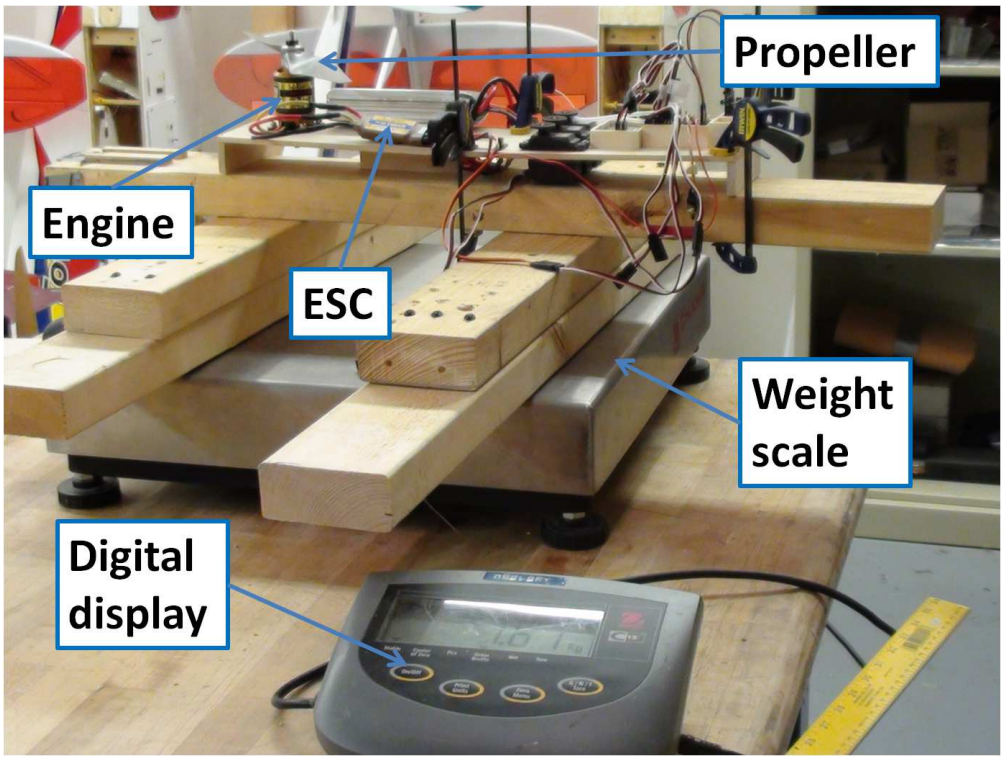

Figure 3.6: Test setup for measuring propeller thrust at zero airspeed

instead of thrust. From this figure, we also observe that the maximum thrust-to-weight ratio of the Telemaster is about 0.41, which is a typical value for a trainer class R/C airplane.

The thrust predictions generated by Javaprop at zero airspeed (static data) are verified through ground testing. The test setup is given in Figure 3.6, which shows the engine mounted on an apparatus that is attached to a weight scale, with the propulsion system set up in a pusher configuration, that is, the generated thrust pushes the whole test apparatus downward. We use the weight scale and a tachometer to obtain the values of the propeller thrust and engine RPM, respectively, for each considered throttle setting. The static thrust data returned by Javaprop and those obtained from two ground tests, performed using two different batteries, are given in Figure 3.7. The figure clearly shows that the Javaprop

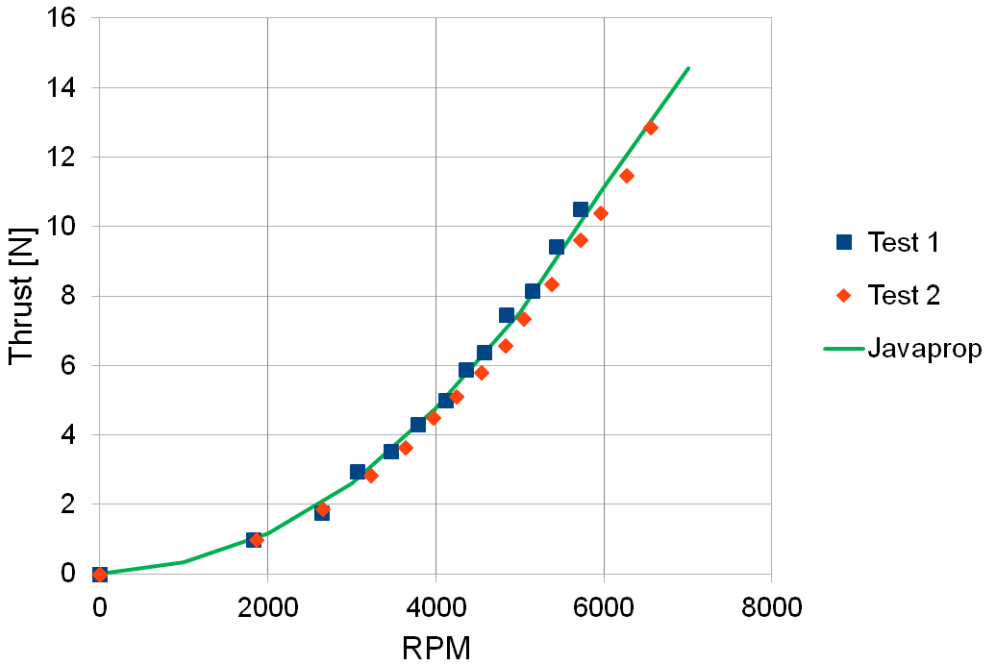

Figure 3.7: Static thrust data along with Javaprop predictions

predictions closely match the static test data.

3.5 Aerodynamic Model

The goal of aerodynamic modeling is to obtain a reasonably accurate mapping from the airplane state and control variables to the aerodynamic forces and moments in the operating envelope. Thus, we need to find formulas for $F_A(\bar{V}, \Omega, \delta_e, \delta_a, \delta_r) = [F_X,\ F_Y,\ F_Z]^T$ and $M_A(\bar{V}, \Omega, \delta_e, \delta_a, \delta_r) = [M_X,\ M_Y,\ M_Z]^T$, as defined in (3.5), where F_i is the component of the aerodynamic force in the body i-axis and M_i is the aerodynamic moment about the body i-axis. Specifically, we aim to determine the force coefficients $C_i = F_i/(Q\,S)$, for $i = X, Y, Z$,

and the moment coefficients $C_l = M_X/(Q\,S\,b)$, $C_m = M_Y/(Q\,S\,\bar{c})$, and $C_n = M_Z/(Q\,S\,b)$, where $Q = 0.5\,\rho\,V_a^2$ is the dynamic pressure, with ρ being the air density, S is the wing area, b is the wing span, and $\bar{c}$ is the mean aerodynamic chord. The values of the aforementioned geometrical parameters can be found in Table 1.1.

The aerodynamic model can be obtained using several methods, including those based on semi-empirical techniques, computational fluid dynamics (CFD), wind tunnel testing, and flight testing. A semi-empirical method typically uses basic aerodynamic theories, along with some experimental data, to generate an aerodynamic model based on some geometrical parameters of the airplane. There are available software, such as digital DATCOM [73], which use semi-empirical methods to predict the aerodynamic characteristics of airplane. DATCOM, in particular, takes some geometrical parameters as inputs and outputs aerodynamic coefficients and stability derivatives. Applying such methods to find aerodynamic models for small UAVs may be unfavorable in general because most of the experimental data used by these methods are obtained for full-scale aircraft. There are several, freely available, linear CFD codes based on vortex lattice analysis, such as Athena Vortex Lattice [74] and Tornado [75], that can be used to develop aerodynamic models for small UAVs. The CFD approach generally gives a satisfactory aerodynamic model in the linear region of the aerodynamic force and moment curves. However, this approach does not take into account the effects of gaps between the control surfaces and the wings or tails in computing the forces and moments generated by the control surface deflections. The drag prediction is also optimistic in this case, that is, the resulting model underestimates the value of drag; the reason

is the CFD method only considers the drag generated by the pressure distribution around the wings and fuselage and neglects the drag due to skin friction and appendices such as the landing gear. Wind tunnel testing offers accurate and comprehensive aerodynamic data to develop a high-fidelity model, but this comes at the expense of high cost and relatively long model development time [76]. In the case of small, low-cost UAVs, the cost and time to perform the necessary wind tunnel tests can be prohibitive. System identification and/or parameter estimation techniques can be used to develop aerodynamic models from flight test data [28, 77, 78]. The structure of the aerodynamic model may be determined based on the test data by using, for instance, a stepwise regression method (system identification), or it can be simply postulated. Parameter estimation techniques can then be used to estimate the values of the aerodynamic coefficients.

In this dissertation, the aerodynamic model is obtained from flight test data using the output error method for parameter estimation in the time domain. The output error method is a maximum likelihood estimator for data with measurement noise [79]. This method is asymptotically unbiased and consistent, that is, the estimated parameters will approach their true values as the number of measurements increases [77]. It is computationally manageable, especially when compared to the more sophisticated filter error method. But, unlike the filter error method, the output error approach is not well-suited to handling process noise, and so it is important to carry out the flight tests in the absence of significant atmospheric disturbances. Details on the use of the output error method for aerodynamic parameter estimation can be found in [28, 77]. The structure of the aerodynamic model used is given

in Eq. (3.7), as suggested in [80]. This structure is chosen so that a single model can be used to capture the UAV dynamics in the entire operating envelope of interest.

$$
\begin{aligned}
C_X &= C_{X_0} + C_{X_\alpha}\alpha + C_{X_{\alpha^2}}\alpha^2 \\
C_Y &= C_{Y_0} + C_{Y_\beta}\beta + C_{Y_p}\frac{p\,b}{2V_a} + C_{Y_r}\frac{r\,b}{2V_a} + C_{Y_{\delta a}}\delta_a + C_{Y_{\delta r}}\delta_r \\
C_Z &= C_{Z_0} + C_{Z_\alpha}\alpha + C_{Z_{\delta e}}\delta_e + C_{Z_{\alpha^2}}\alpha^2 \\
C_l &= C_{l_0} + C_{l_\beta}\beta + C_{l_p}\frac{p\,b}{2V_a} + C_{l_r}\frac{r\,b}{2V_a} + C_{l_{\delta a}}\delta_a + C_{l_{\delta r}}\delta_r \\
C_m &= C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{q\,\bar{c}}{2V_a} + C_{m_{\delta e}}\delta_e + C_{m_{\alpha^2}}\alpha^2 \\
C_n &= C_{n_0} + C_{n_\beta}\beta + C_{n_p}\frac{p\,b}{2V_a} + C_{n_r}\frac{r\,b}{2V_a} + C_{n_{\delta a}}\delta_a + C_{n_{\delta r}}\delta_r
\end{aligned}
\tag{3.7}
$$

The flight tests for parameter estimation in our case are performed by a pilot on the ground controlling the airplane remotely. Hence, only a few standard maneuvers for parameter estimation can be successfully performed. These maneuvers capture the dutch-roll, bank-to-bank, and short period motions of the aircraft. Specifically, we employ standard control input patterns for time-domain parameter estimation [28] to excite the vehicle. The input patterns used are different for each control actuator: doublet inputs are used for the rudder to capture the dutch-roll mode, 1-2-1 for the aileron to capture the bank-to-bank motion, and 3-2-1-1 for the elevator to capture the short period mode. These integers indicate the number of time steps to hold a control actuator at a specified position before moving it to the same position in the opposite direction. Figure 3.8 illustrates the shape of each control surface input to be executed during the flight tests. In this figure, the time step Δt and the

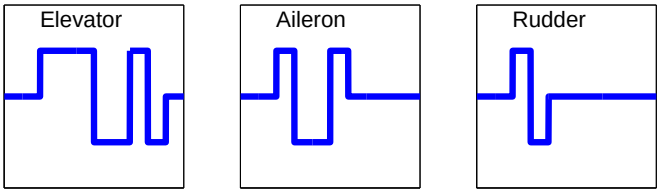

Figure 3.8: Control input shapes for parameter estimation

magnitude of the deflection are chosen such that the control input sufficiently excites the airplane (namely, generate $0.4-0.5\,g$ acceleration) without causing it to deviate significantly from the nominal flight condition. The guideline for choosing the best input is provided by [28], which is based on the frequency of the aircraft mode of interest. For the aircraft used in this work, the time step Δt is chosen to be equal to 250 ms, and an appropriate magnitude of the control surface deflection is determined during the flight tests. To maintain consistency between tests, the onboard computer is used to generate the commands for control surface deflections from the trim condition.

Prior to estimating the aerodynamic model parameters, a compatibility check is performed on the test data. The objective of this step is twofold: first, to ensure that the data satisfies the kinematic relationships dictated by the equations of motion, and second, to estimate and remove any bias and scale factor errors in the measured data. An Extended Kalman Filter (EKF) is used to obtain an estimate of the time history of the vector-valued function $\mathbf{x} = [u,\, v,\, w,\, \phi,\, \theta,\, \psi]^T$, as well as determine any sensor bias and scale factor errors. This process, referred to as flight path reconstruction (FPR) [28, 77, 81], utilizes the Eq. (3.1) and (3.2) with the adjusted measurements of the linear accelerations, a_x, a_y, and a_z, and the angular rates, p, q, and r, as inputs to the system. The notations a_x, a_y, and a_z denote the

linear accelerations at the center of gravity of the UAV along the body X, Y, and Z axes, respectively. These accelerations are not measured directly, but can be obtained from those given by the AHRS as follows: $[a_x,\, a_y,\, a_z]^T = m^{-1}F(\bar{V}, \Omega, \delta) = [\hat{a}_x,\, \hat{a}_y,\, \hat{a}_z]^T - \Omega \times (\Omega \times \Delta P) - \dot{\Omega} \times \Delta P$, where ΔP is the position of the AHRS (which is located on the body X-axis) relative to the center of gravity of the UAV, $\hat{a}_x$, $\hat{a}_y$, and $\hat{a}_z$ are measured by the AHRS and denote the linear accelerations at the location of the AHRS, expressed in $\{B\}$, and the derivative $\dot{\Omega}$ is computed in our case using the central difference method. Aside from the aforementioned inputs, the EKF uses measurements of ϕ, θ, ψ (given by the AHRS) and V_a, α, β (given by the airdata probes) to update the state estimates. Note that $\alpha = \tan^{-1}((w - w_w)/(u - u_w))$ and $\beta = \sin^{-1}((v - v_w)/V_a)$. As all of the data to be used in the parameter estimation are obtained from several tests performed during a single flight, it is sensible to process all the test data simultaneously in order to obtain a single set of values for bias and scale factor errors. To process the data concurrently, the EKF is modified to expand the number of states and measurements following the addition of flight test data to the FPR process. That is, for each flight test i, we add dynamic equations to the EKF (Eq. (3.1) and (3.4)), along with the measurements from this run, to estimate the time history of the corresponding vector-valued function $\mathbf{x}^{(i)}$. We incorporate sensor bias and scale factor errors in the measurement equations for $\hat{a}_x$, $\hat{a}_y$, $\hat{a}_z$, p, q, r, V_a, α, and β, which take the form: $Q = K_Q\, Q_m - B_Q$, where Q is the adjusted value of the variable being measured, Q_m is the measured value given by the sensor, K_Q is the scale factor error, and B_Q is the measurement bias.

A comparison of the raw and reconstructed data for an elevator 3-2-1-1 test is given in

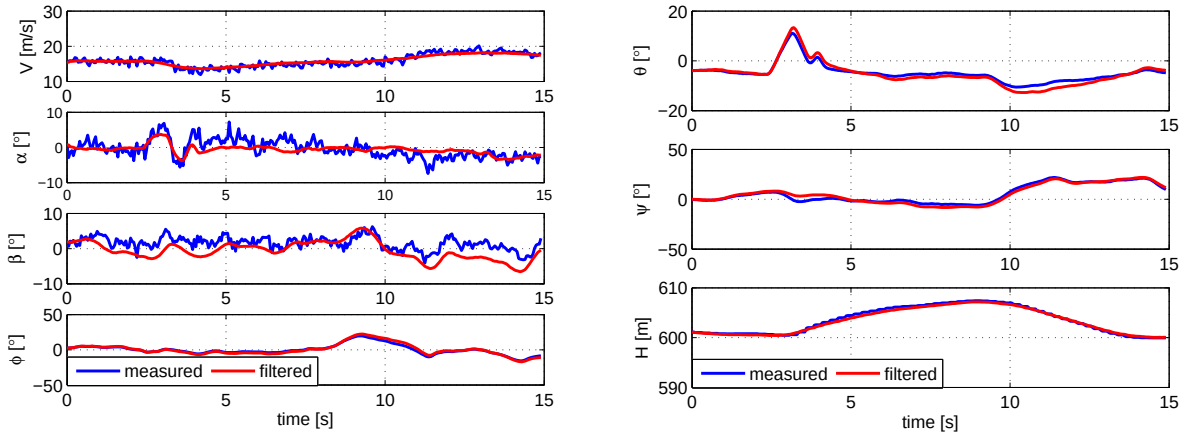

Figure 3.9: Data compatibility check using EKF prior to estimation

Figure 3.9. For the test data used to derive the aerodynamic model, the bias terms and scale factors are not significant, with the exception of the angle of attack measurement bias, B_α, as shown in Table 3.3. The larger bias in the angle of attack measurements is a consequence of the airdata probe being installed parallel to the bottom surface of the wing, which has a 4° incidence angle from the body X-axis.

Table 3.3: Estimated measurement biases and scales

bias	value	unit	scales	value
B_{ax}	-0.0434	m/s	K_{ax}	1.0000
B_{ay}	0.1200	m/s	K_{ay}	1.0000
B_{az}	-0.0097	m/s	K_{az}	1.0000
B_p	-0.0011	rad/s	K_p	1.0000
B_q	-0.0032	rad/s	K_q	1.0000
B_r	-0.0082	rad/s	K_r	1.0000
B_{Va}	-0.0199	m	K_{Va}	1.0000
B_α	0.0587	rad	K_α	1.0000
B_β	0.0118	rad	K_β	1.0000

It is convenient to decouple the estimation of the longitudinal and lateral-directional parameters [28]. This decoupling is acceptable under the assumption that, when the aircraft is in steady symmetric level flight, a small perturbation in the longitudinal motion will not affect

the lateral-direction motion and vice versa. In general, this scenario is difficult to achieve in the case of a small UAV because of the imperfections in the aircraft and the difficulty for an on-ground pilot to ensure a steady symmetric level flight. To circumvent these issues, multiple tests are performed at similar flight conditions. These tests are subsequently analyzed and those which satisfy the steady symmetric level flight requirement are then used in the parameter estimation process.

We have used three flight test runs to estimate the longitudinal aerodynamic parameters and four test runs to estimate the lateral-directional parameters. The estimated parameters are given in Table 3.4. Figures 3.10 and 3.11 show that the output of the estimated model matches well both the longitudinal and lateral-directional test data. The estimated aerodynamic parameters are validated against flight test data that are not included in the parameter estimation process. Specifically, the inputs from this data, that is, the control surface deflections and initial conditions, are used to drive a simulated model that uses the values of the aerodynamic coefficients identified in the estimation process. The simulated response is then compared to the measured response to see how well the estimated aerodynamic model parameters are able to predict the response of the actual airplane. A comparison of the predicted and measured responses is given in Figures 3.12 and 3.13 for the longitudinal and lateral-directional tests, respectively. The response of the simulated system is fairly close to the measured response, which indicates that the obtained model adequately captures the aircraft dynamics at the test conditions.

Table 3.4: Estimated parameters

parameter	value	parameter	value	parameter	value
C_{X_0}	-0.1004	C_{Y_0}	+0.0446	C_{Z_0}	-0.4522
C_{X_α}	-0.0928	C_{Y_β}	-0.5724	C_{Z_α}	-5.3550
$C_{X_{\alpha^2}}$	+1.7729	C_{Y_p}	+0.1203	$C_{Z_{\alpha^2}}$	-4.3813
		C_{Y_r}	+0.1181	$C_{Z_{\delta e}}$	+0.7406
		$C_{Y_{\delta a}}$	-0.0276		
		$C_{Y_{\delta r}}$	+0.1584		
parameter	value	parameter	value	parameter	value
C_{l_0}	+0.0128	C_{m_0}	-0.0057	C_{n_0}	-0.0068
C_{l_β}	-0.0579	C_{m_α}	-0.2402	C_{n_β}	+0.0538
C_{l_p}	-0.3590	$C_{m_{\alpha^2}}$	-0.0750	C_{n_p}	-0.0489
C_{l_r}	+0.1420	C_{m_q}	-10.6607	C_{n_r}	-0.0831
$C_{l_{\delta a}}$	+0.1519	$C_{m_{\delta e}}$	+0.5737	$C_{n_{\delta a}}$	-0.0139
$C_{l_{\delta r}}$	+0.0042			$C_{n_{\delta r}}$	-0.0465

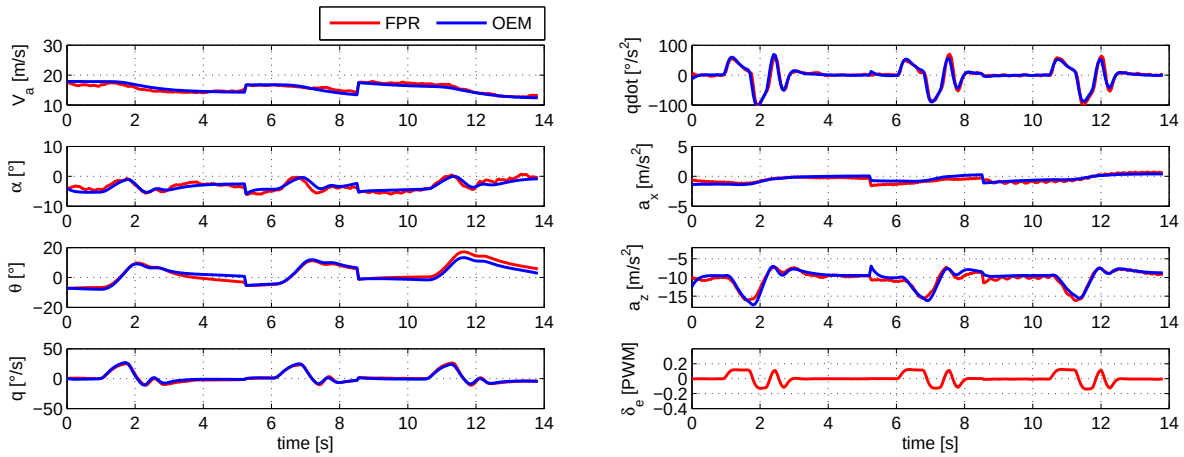

Figure 3.10: Comparison of simulated and measured responses: longitudinal parameter estimation run

3.6 Simulation Environment

A simulation environment is developed to enable us to perform aircraft motion analysis and control design. The aircraft equations of motion Eq. (3.1) to (3.4) along with the mathematical model of the aerodynamic forces and moments, servo dynamics and propulsion

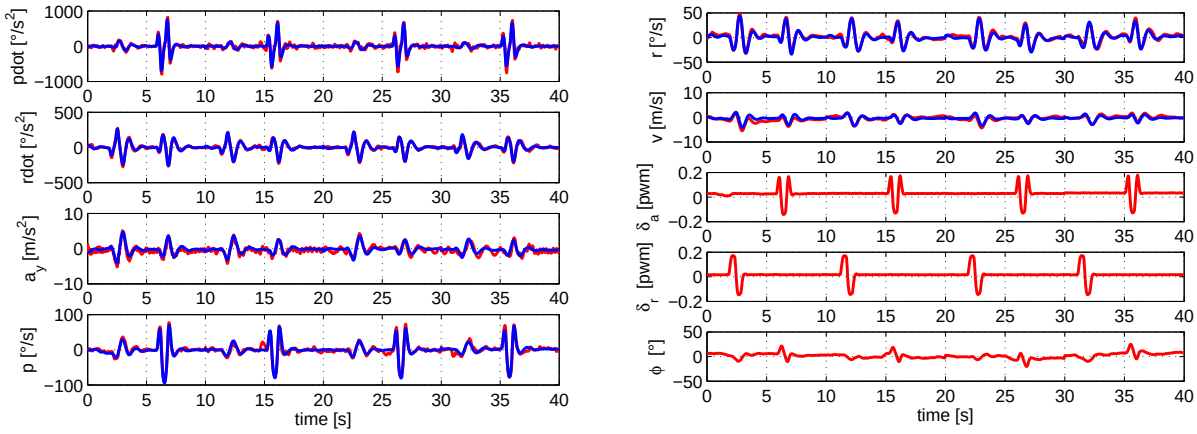

Figure 3.11: Comparison of simulated and measured responses: lateral-directional parameter estimation run

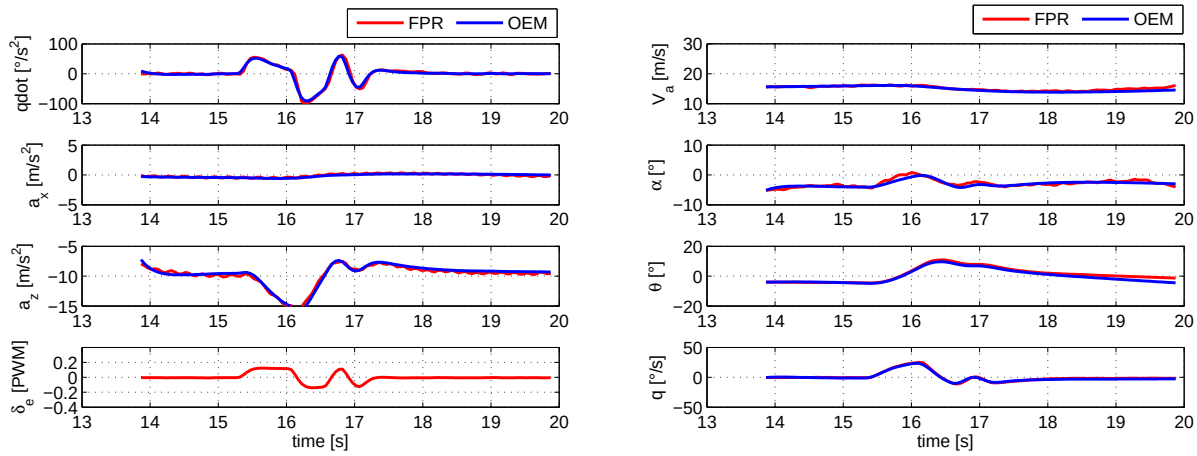

Figure 3.12: Comparison of simulated and measured responses: longitudinal model validation run

system's lookup tables are implemented in MATLAB, and will be referred to henceforth as the flight dynamic model of the UAV. A MATLAB solver for differential equations based on fourth and fifth order Runge-Kutta methods, ODE45, is used to solve for the state evolution over time, for a given initial condition and input time history.

The simulation is implemented in discrete-time, specifically the Runge-Kutta solver is utilized to solve for the state evolution over one time step, in which the input to the flight dynamic

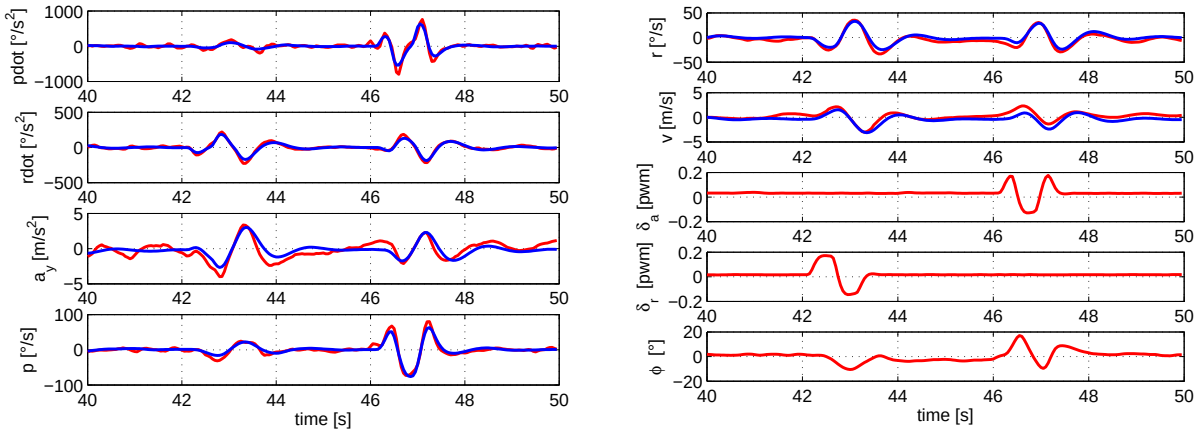

Figure 3.13: Comparison of simulated and measured responses: lateral-directional model validation run

model is held constant (zero-order-hold). The state of the dynamic model at the end of the corresponding time step is used as the initial state for the subsequent time step. The time steps are fixed throughout the simulation, and are set equal to 50ms, which corresponds to the sampling time of the autopilot system. Details about the standard workflow for control design using the developed simulation environment will be given in the next chapter.

Atmospheric disturbances, in the form of constant wind with turbulence, and one-step time delay are also included in the simulation environment. The atmospheric turbulence is generated using a low altitude continuous-time Dryden turbulence model on a linear axis. This model is obtained by passing a band-limited Gaussian white noise signal of zero mean and unit variance through linear filters defined by the following transfer functions [82, 83]:

$$H_u(s) = \sigma_u \sqrt{\frac{2V_a}{\pi L_u}} \frac{1}{s + \frac{V_a}{L_u}}, \quad H_v(s) = \sigma_v \sqrt{\frac{3V_a}{\pi L_v}} \frac{s + \frac{V_a}{\sqrt{3}L_v}}{\left(s + \frac{V_a}{L_v}\right)^2}, \quad H_w(s) = \sigma_w \sqrt{\frac{3V_a}{\pi L_w}} \frac{s + \frac{V_a}{\sqrt{3}L_w}}{\left(s + \frac{V_a}{L_w}\right)^2} \tag{3.8}$$

where the coefficients in these equations are determined based on the flight and atmospheric conditions, namely the airspeed (V_a), the altitude above surface level (h in feet), and wind speed at 6 m [20 ft] above surface level (W_{20}). Specifically, the formulas for these coefficients are given below:

$$L_u = L_w = h, \quad L_v = \frac{h}{(0.17 + 0.000823h)^{1.2}}$$

$$\sigma_w = 0.1W_{20}, \quad \sigma_u = \sigma_v = \frac{\sigma_w}{(0.17 + 0.000823h)^{0.4}}$$

where the subscripts u, v, and w indicate the gust components in the body X, Y, and Z axes, respectively. The Dryden turbulence model has appeared in numerous publications, for instance, [82] and [83].

In this section we will use the simulation tool to examine how closely the derived flight dynamic model describes the behavior of the UAV given certain commanded control inputs. This analysis is achieved by comparing the simulation results to the actual flight test data. Specifically, a couple of flight tests are chosen, and the initial conditions and time history of the inputs during each test are used to drive the simulation model.

The first test data to be used in this verification process is obtained from a randomly chosen part of the recorded flight data during a parameter estimation test. The data depicts a gentle

left turn maneuver performed by the UAV, that is the UAV transitions from a straight and level flight into a steady left turn flight. The time histories of the inputs for this maneuver are extracted and then used in the simulation to generate a similar maneuver shown in Figure 3.14. As evident from the figure, the simulated trajectory captures most of the features of the actual flight trajectory.

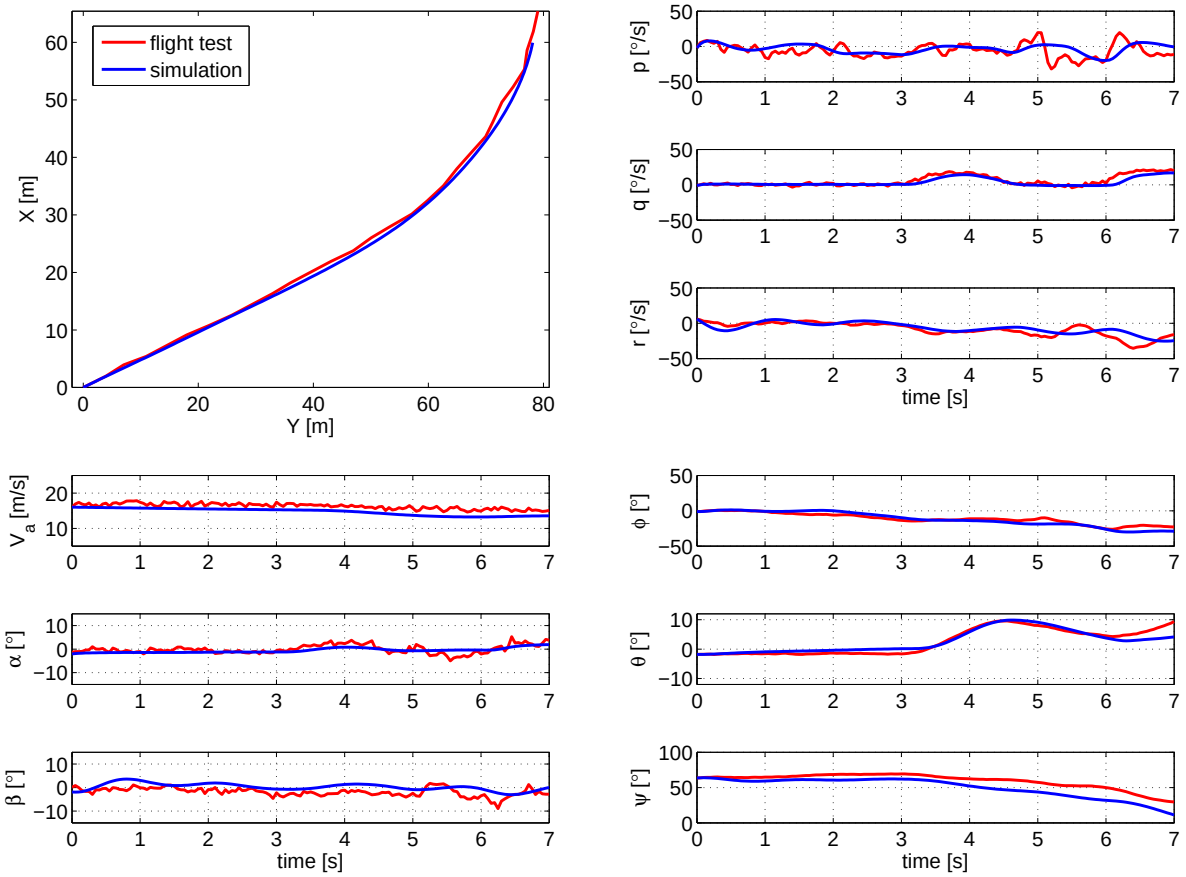

Figure 3.14: Comparison of simulated and measured responses: a gentle left turn maneuver

To further assess the capability of the developed model to simulate aggressive maneuvers, the following test is performed. The test utilizes data obtained during a Split-S maneuver. This maneuver is executed when an airplane needs to make a 180 ° turn in a short amount of

time. As in the previous case, the input history for this maneuver is extracted and used to drive the simulation. The simulation results are shown in Figure 3.15, where the solid blue lines designate the simulation results and the solid red lines indicate the flight test results. Observe that the discrepancies between the test data and the simulation results are more pronounced in this case. This is expected since the flight condition, as indicated by the measured angle of attack, is well outside the flight condition where the parameter estimation is performed.

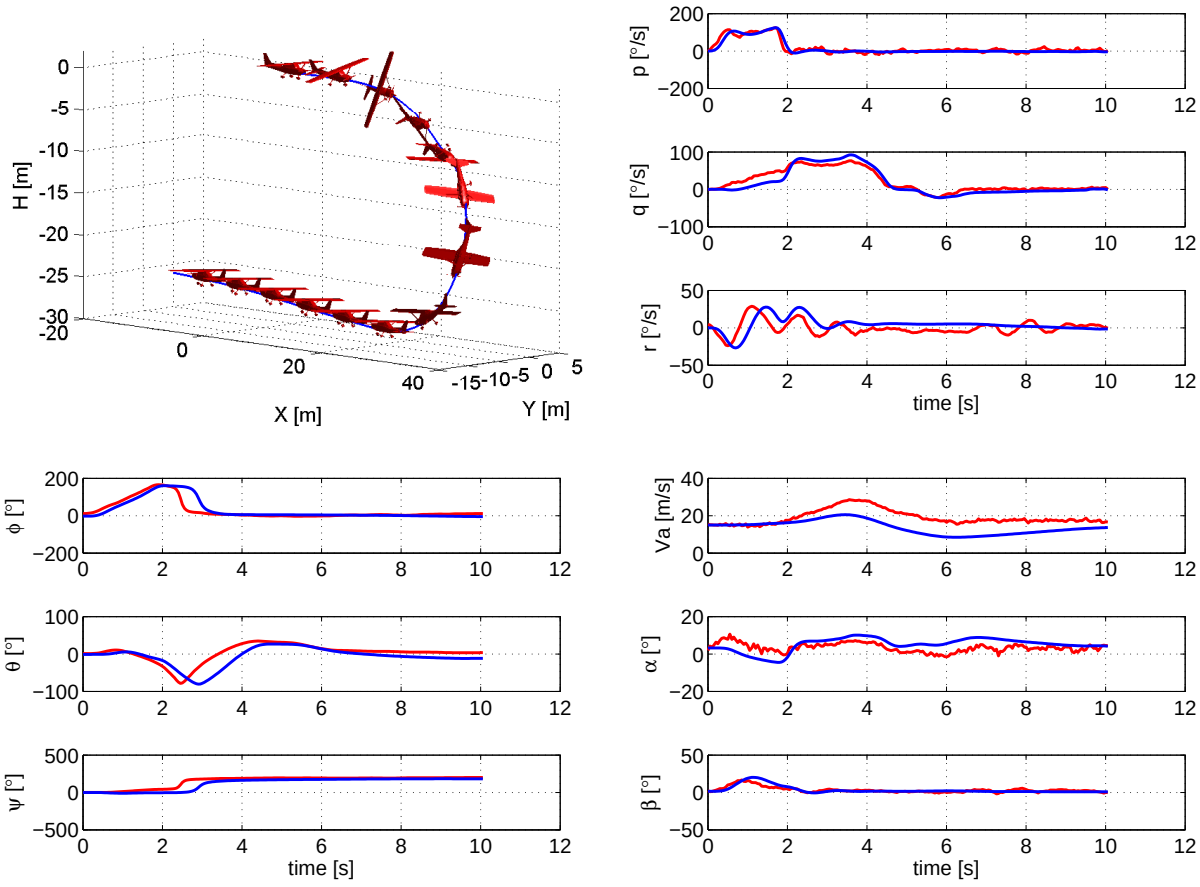

Figure 3.15: Comparison of simulated and measured responses: a Split-S maneuver

Chapter 4

Control System Design, Simulation, and Flight Testing

In this chapter, we design several controllers to be implemented on the testbed. Specifically, we are interested in using output feedback controllers with ℓ_2-induced norm performance measure to track real-time trajectories. We assume that the desired trajectories are generated from a library of pre-specified motion primitives using some primitive-based motion planner. The library usually consists of trim conditions and transitions between trim conditions. Using the flight dynamic model derived in the previous chapter, the motion primitives can be generated off-line using the aforementioned simulation environment. The trim conditions can be obtained easily by solving for the equilibrium points of the nonlinear mathematical model. The transition maneuvers are not as easy to generate in general, but there are several methods that can be utilized for this purpose. One method entails formulating the problem

as a nonlinear optimization problem and then using algorithms such as DIDO [84] to solve it. Alternatively, the maneuvers can be obtained by recording pilot inputs and corresponding system responses during an actual flight [41]. In this dissertation, we will use simple feedback control to obtain transition maneuvers between trim conditions and utilize recorded pilot input to generate the Split-S maneuver.

The controllers will be synthesized based on linearized models about motion primitives. Linearizing the nonlinear dynamics about a trim condition results in a linear time-invariant (LTI) system, whereas linearizing the nonlinear dynamics about a transition maneuver results in a finite-horizon linear time-varying (LTV) system.

4.1 Preliminaries

4.1.1 Notation

The notation is quite standard. We denote the set of real $n \times m$ matrices by $\mathbb{R}^{n\times m}$. If M_i is a sequence of matrices, then $\mathrm{diag}(S_i)$ denotes their block-diagonal augmentation. We use I_n to denote an $n \times n$ identity matrix and $0_{n\times m}$ to denote an $n \times m$ zero matrix. The transpose of a matrix X is written X^T. Given a symmetric matrix X, we use $X \prec 0$ to mean it is negative definite. The normed space of square summable vector-valued sequences is denoted by ℓ_2. It consists of elements $x = (x_0, x_1, x_2, \ldots)$, with $x_k \in \mathbb{R}^n$ for some n, having a finite 2-norm $\|x\|_{\ell_2}$ defined by $\|x\|_{\ell_2}^2 = \sum_{k=0}^{\infty} \|x_k\|_2^2 < \infty$, where $\|x_k\|_2^2 = x_k^T x_k$, that is, $\|\cdot\|_2$ denotes the

Euclidean norm. We use the notation $\|P\|_{\ell_2 \to \ell_2}$ to denote the ℓ_2 induced norm of a bounded linear mapping P on ℓ_2.

4.1.2 Control Synthesis

This work makes use of the LMI-based control synthesis techniques developed in [37, 38, 20] for discrete-time LTI systems, finite horizon systems, and linear systems with uncertain initial conditions, respectively. These techniques will be used to design switched controllers for the regulation of the UAV about concatenated trajectories. The performance measure for designing the optimal controllers is the ℓ_2 induced norm in the case of plants with zero initial conditions and the square ℓ_2 induced norm in the case of uncertain initial conditions. The first two subsections give procedures for constructing optimal controllers for finite horizon and LTI systems with zero initial conditions, whereas the last subsection considers LTI plants with uncertain initial states.

Finite horizon systems

Consider a finite horizon linear discrete-time plant G, described by the following state-space equation:

$$\begin{bmatrix} \bar{x}_{k+1} \\ z_k \\ \bar{y}_k \end{bmatrix} = \begin{bmatrix} A_k & B_{1k} & B_{2k} \\ C_{1k} & D_{11k} & D_{12k} \\ C_{2k} & D_{21k} & 0 \end{bmatrix} \begin{bmatrix} \bar{x}_k \\ d_k \\ \bar{\delta}_k \end{bmatrix}, \qquad \bar{x}_0 = 0, \tag{4.1}$$

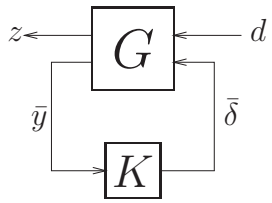

Figure 4.1: Closed-loop system

for $k = 0, 1, \ldots, h-1$, with h designating the finite horizon length. The signal $\bar{x}_k \in \mathbb{R}^n$ is the error between the actual and reference values of the state vector at discrete instant k, namely $\bar{x}_k = x_k - x_{r,k}$, where the subscript r refers to the reference data. Similarly, $\bar{\delta}_k = \delta_k - \delta_{r,k} \in \mathbb{R}^{n_\delta}$ and $\bar{y}_k = y_k - y_{r,k} \in \mathbb{R}^{n_y}$, where δ_k and y_k denote the applied control input and the measurements at time k, respectively, with $\delta_{r,k}$ and $y_{r,k}$ indicating the reference values. The signals $d_k \in \mathbb{R}^{n_d}$ and $z_k \in \mathbb{R}^{n_z}$ denote the exogenous disturbances and errors, respectively. The goal is to design a feedback controller K, defined by the state-space equation

$$\begin{bmatrix} x_{k+1}^K \\ \bar{\delta}_k \end{bmatrix} = \begin{bmatrix} A_k^K & B_k^K \\ C_k^K & D_k^K \end{bmatrix} \begin{bmatrix} x_k^K \\ \bar{y}_k \end{bmatrix}, \qquad x_0^K = 0, \tag{4.2}$$

with $x_k^K \in \mathbb{R}^n$, such that the closed-loop system in Figure 4.1 is asymptotically stable and $\|d \mapsto z\|_{\ell_2 \to \ell_2} < \gamma_{\min}$, where $\gamma_{\min}$ is the minimum achievable ℓ_2-gain performance level up to a certain tolerance.

The procedure for constructing the feedback controller K is based on the work in [37, 38] and is summarized as follows:

1. Solve the following semidefinite programming optimization problem for the matrix sequences R_k and S_k, for $k = 0, 1, \ldots, h-1$, and the scalar $\gamma_{\min}$ (which is the square

root of the optimal value):

$$\text{minimize } \gamma^2$$

subject to

$$\mathcal{F}_1(F_k, M_k, V_{1k}, V_{2k}, R_k, R_{k+1}, \gamma) = F_k^T R_k F_k - V_{1k}^T R_{k+1} V_{1k} + M_k^T M_k - \gamma^2 V_{2k}^T V_{2k} \prec 0,$$

$$\mathcal{F}_2(W_k, N_k, U_{1k}, U_{2k}, S_k, S_{k+1}, \gamma) = \begin{bmatrix} W_k^T S_{k+1} W_k - U_{1k}^T S_k U_{1k} - U_{2k}^T U_{2k} & N_k^T \\ N_k & -\gamma^2 I \end{bmatrix} \prec 0,$$

$$\mathcal{F}_3(R_k, S_k) = \begin{bmatrix} R_k & I \\ I & S_k \end{bmatrix} \succeq 0, \text{ for } k = 0, 1, \ldots, h-1, \text{ and } \begin{bmatrix} R_h & I \\ I & S_h \end{bmatrix} \succeq 0,$$

where

$$\text{Im } \begin{bmatrix} V_{1k}^T & V_{2k}^T \end{bmatrix}^T = \text{Ker } \begin{bmatrix} B_{2k}^T & D_{12k}^T \end{bmatrix}, \qquad \begin{bmatrix} V_{1k}^T & V_{2k}^T \end{bmatrix} \begin{bmatrix} V_{1k}^T & V_{2k}^T \end{bmatrix}^T = I,$$

$$\text{Im } \begin{bmatrix} U_{1k}^T & U_{2k}^T \end{bmatrix}^T = \text{Ker } \begin{bmatrix} C_{2k} & D_{21k} \end{bmatrix}, \qquad \begin{bmatrix} U_{1k}^T & U_{2k}^T \end{bmatrix} \begin{bmatrix} U_{1k}^T & U_{2k}^T \end{bmatrix}^T = I,$$

with $\text{Im}\, L$ and $\text{Ker}\, L$ denoting the image and kernel of a linear map L, respectively, and

$$F_k = A_k^T V_{1k} + C_1^T V_{2k}, \qquad M_k = B_{1k}^T V_{1k} + D_{11k}^T V_{2k},$$

$$W_k = A_k U_{1k} + B_{1k} U_{2k}, \qquad N_k = C_{1k} U_{1k} + D_{11k} U_{2k}.$$

Note that $\mathcal{F}_3(R_k, S_k) \succeq 0$ implies that $R_k \succ 0$ and $S_k \succ 0$.

2. Compute the following matrices:

$$P_k = \begin{bmatrix} 0 & I_n & 0_{n\times 2n} & 0_{n\times n_d} & 0 \\ B_{2k}^T & 0 & 0_{n_\delta \times 2n} & 0_{n_\delta \times n_d} & \frac{1}{\gamma_{\min}} D_{12k}^T \end{bmatrix}, \quad Q_k = \begin{bmatrix} 0_{n\times 2n} & 0 & I_n & 0 & 0_{n\times n_z} \\ 0_{n_y \times 2n} & C_{2k} & 0 & D_{21k} & 0_{n_y \times n_z} \end{bmatrix},$$

$$H_k = \begin{bmatrix} -R_{k+1} & -E_{k+1} & A_k & 0 & B_{1k} & 0 \\ -E_{k+1}^T & -I_n & 0 & 0 & 0 & 0 \\ A_k^T & 0 & -S_k & S_k E_k & 0 & \frac{1}{\gamma_{\min}} C_{1k}^T \\ 0 & 0 & E_k^T S_k & -I_n - E_k^T S_k E_k & 0 & 0 \\ B_{1k}^T & 0 & 0 & 0 & -I_{n_d} & \frac{1}{\gamma_{\min}} D_{11k}^T \\ 0 & 0 & \frac{1}{\gamma_{\min}} C_{1k} & 0 & \frac{1}{\gamma_{\min}} D_{11k} & -I_{n_z} \end{bmatrix},$$

where $E_k = \left(R_k - S_k^{-1}\right)^{\frac{1}{2}}$.

3. Obtain the state-space matrices of the controller, $J_k = \begin{bmatrix} A_k^K & B_k^K \\ C_k^K & D_k^K \end{bmatrix}$, by solving the following LMI:

$$H_k + Q_k^T J_k^T P_k + P_k^T J_k Q_k \prec 0,$$

for each $k \in \{0, 1, \ldots, h-1\}$. Note that the Matlab command `basiclmi` is useful in this regard.

It is also possible to derive explicit controller formulas based on the results of [37] or [85]; see, for instance, the algorithm derived in [86, 22] for nonstationary linear parameter-varying systems.

Linear time-invariant systems

Suppose the state-space matrices in (4.1) are constant, and hence the plant G in this case is a discrete-time LTI plant. The synthesis objective is the same as in the finite horizon

case in that we seek to design a feedback controller K, which is LTI in this case, defined by the state-space equation in (4.2) with the subscript k in the notation of the controller state-space matrices dropped, such that the closed-loop system is asymptotically stable and $\|d \mapsto z\|_{\ell_2 \to \ell_2} < \gamma_{\min}$. The procedure for designing the optimal H_∞ controller is given in [37] and included next for completeness:

1. Solve the following semidefinite program for the scalar $\gamma_{\min}$ and the matrices R and S:

   $$\text{minimize} \gamma^2$$

   subject to

   $$\mathcal{F}_1(F, M, V_1, V_2, R, R, \gamma) \prec 0, \quad \mathcal{F}_2(W, N, U_1, U_2, S, S, \gamma) \prec 0, \quad \mathcal{F}_3(R, S) \succeq 0.$$

   Note that the synthesis solutions are constant in this case, that is, $R_{k+1} = R_k = R$ and $S_{k+1} = S_k = S$, and so are the state-space matrices, hence the omittance of the subscript k.

2. Compute the state-space matrices of the controller by solving the following LMI for J:

   $$H + Q^T J^T P + P^T J Q \prec 0,$$

   where again, for clarity, $E_{k+1} = E_k = E$, $R_{k+1} = R_k = R$, and $S_k = S$.

LTI systems with uncertain initial states

Given the LTI plant from the previous subsection, suppose that some of the state variables have uncertain, nonzero initial values. Namely, $\bar{x}_0 \neq 0$, but rather $\bar{x}_0 = \Lambda \hat{x}_0$, where Λ is a constant $n \times m$ matrix for some m and $\hat{x}_0$ is an uncertain, norm-bounded m-entry vector. In

this case, the control synthesis objective is to find a discrete-time linear feedback controller K that results in a stable closed-loop system, which satisfies the performance inequality

$$\|(\hat{x}_0, w) \mapsto z\|_{sq} := \sup_{\|\hat{x}_0\|_2 \leq 1,\, \|w\|_{\ell_2} \leq 1} \|z\|_{\ell_2} < \gamma_{\min},$$

where $\|\cdot\|_{sq}$ denotes the square ℓ_2 induced norm, and $\gamma_{\min}$ is the minimum achievable performance level up to a certain tolerance by the considered class of linear controllers. As discussed in [20], it is conceivable that an N-eventually time-invariant controller, for some integer $N \geq 1$, could provide an improved performance over an LTI one in general. An LTV controller is N-eventually time-invariant controller if each of its state-space matrix sequences is N-eventually time-invariant; for instance, the sequence A_k^K would be of the form $A_0^K, A_1^K, \ldots, A_{N-1}^K, A_N^K, A_N^K, A_N^K, \ldots$. The procedure for constructing an optimal N-eventually times-invariant controller is given in [20] and summarized as follows:

1. Solve the following semidefinite program for the matrix sequences R_k and S_k, for $k = 0, 1, \ldots, N$, and the optimal value $\gamma_{\min}$:

minimize γ

subject to $e + f_1 + f_2 < 2\gamma, \quad \Lambda^T S_0 \Lambda \prec f_1 I,$

$$\begin{bmatrix} V_1 \\ V_2 \end{bmatrix}^T \left\{ \begin{bmatrix} A & B_1 \\ C_1 & D_{11} \end{bmatrix} \begin{bmatrix} R_k & 0 \\ 0 & pI \end{bmatrix} \begin{bmatrix} A & B_1 \\ C_1 & D_{11} \end{bmatrix}^T - \begin{bmatrix} R_{k+1} & 0 \\ 0 & eI \end{bmatrix} \right\} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \prec 0,$$

$$\begin{bmatrix} U_1 \\ U_2 \end{bmatrix}^T \left\{ \begin{bmatrix} A & B_1 \\ C_1 & D_{11} \end{bmatrix}^T \begin{bmatrix} S_{k+1} & 0 \\ 0 & tI \end{bmatrix} \begin{bmatrix} A & B_1 \\ C_1 & D_{11} \end{bmatrix} - \begin{bmatrix} S_k & 0 \\ 0 & f_2 I \end{bmatrix} \right\} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \prec 0,$$

$$\begin{bmatrix} R_k & I \\ I & S_k \end{bmatrix} \succeq 0, \quad \begin{bmatrix} p & 1 \\ 1 & f_2 \end{bmatrix} \succeq 0, \quad \begin{bmatrix} t & 1 \\ 1 & e \end{bmatrix} \succeq 0,$$

for $k = 0, 1, \ldots, N$, with $R_{N+1} = R_N$ and $S_{N+1} = S_N$.

2. Construct an N-eventually time-invariant controller (with zero initial condition) from the synthesis solutions by solving the following LMI:

$$\hat{H}_k + \hat{Q}^T J_k^T \hat{P} + \hat{P}^T J_k \hat{Q} \prec 0,$$

for $k = 0, 1, \ldots, N$, where $\hat{H}_k$, $\hat{P}$, and $\hat{Q}$ are defined as

$$\hat{P} = \begin{bmatrix} 0 & I_n & 0_{n\times 2n} & 0_{n\times n_d} & 0 \\ B_2^T & 0 & 0_{n_\delta \times 2n} & 0_{n_\delta \times n_d} & e^{-\frac{1}{2}} D_{12}^T \end{bmatrix}, \quad \hat{Q} = \begin{bmatrix} 0_{n\times 2n} & 0 & I_n & 0 & 0_{n\times n_z} \\ 0_{n_y \times 2n} & C_2 & 0 & f_2^{-\frac{1}{2}} D_{21} & 0_{n_y \times n_z} \end{bmatrix},$$

$$\hat{H}_k = \begin{bmatrix} -R_{k+1} & -E_{k+1} & A & 0 & f_2^{-\frac{1}{2}} B_1 & 0 \\ -E_{k+1}^T & -I_n & 0 & 0 & 0 & 0 \\ A^T & 0 & -S_k & S_k E_k & 0 & e^{-\frac{1}{2}} C_1^T \\ 0 & 0 & E_k^T S_k & -I_n - E_k^T S_k E_k & 0 & 0 \\ f_2^{-\frac{1}{2}} B_1^T & 0 & 0 & 0 & -I_{n_d} & (e f_2)^{-\frac{1}{2}} D_{11}^T \\ 0 & 0 & e^{-\frac{1}{2}} C_1 & 0 & (e f_2)^{-\frac{1}{2}} D_{11} & -I_{n_z} \end{bmatrix}.$$

4.1.3 Plant Model Formulation

The controllers in this chapter are synthesized based on linearized plant models. This subsection demonstrates how these linearized models are formulated. As the servomotor response is much faster than the dynamic response of the aircraft, we may neglect the effect of the servomotor dynamics in the control design process to reduce the computational complexity of the control synthesis problem. The servomotor models, however, will be incorporated in the simulation environment.

Notice that the heading angle ψ only appears in the equations of motion (3.3) corresponding to the position vector P. Then, when linearizing this equation about some reference trajectory, the resultant linearized equation will depend on the initial reference heading orientation. For example, if the reference trajectory is a straight and level trim trajectory, then we will obtain different linearized models for different heading orientations. This is not favorable for the control approaches used in this work as we will then have to design different controllers for these different linearized models when we know intuitively that the same con-

troller should work for all heading angles (assuming the same worst-case wind disturbance). To circumvent this problem, we will use $(X, Y, -H)$ to indicate the position of the center of gravity, where $X = N\cos\psi_0 + E\sin\psi_0$ and $Y = -N\sin\psi_0 + E\cos\psi_0$, with ψ_0 denoting the heading angle of the UAV at the time the controller is executed. Then, in place of (3.3), we will use the following equations:

$$\begin{aligned}
\dot{H} &= u\sin\theta - v\cos\theta\sin\phi - w\cos\theta\cos\phi \\
\dot{X} &= u\cos\theta\cos(\psi-\psi_0) + v(\sin\phi\sin\theta\cos(\psi-\psi_0) - \cos\phi\sin(\psi-\psi_0)) \\
&\quad + w(\cos\phi\sin\theta\cos(\psi-\psi_0) + \sin\phi\sin(\psi-\psi_0)) \\
\dot{Y} &= u\cos\theta\sin(\psi-\psi_0) + v(\sin\phi\sin\theta\sin(\psi-\psi_0) + \cos\phi\cos(\psi-\psi_0)) \\
&\quad + w(\cos\phi\sin\theta\sin(\psi-\psi_0) - \sin\phi\cos(\psi-\psi_0)).
\end{aligned} \tag{4.3}$$

Note that, although the Z-axis of $\{I\}$ points down, the altitude H is positive upward.

Thus, the state vector is defined as $x = [p, q, r, u, v, w, \phi, \theta, \psi, H, X, Y]^T$, and the control input in this case is $\delta = [\delta_e, \delta_a, \delta_r, \delta_t]^T$. The vector d denotes the exogenous disturbances and t is defined as $d = [u_w, v_w, w_w, d_p, d_q, d_r, d_{Va}, d_\phi, d_\theta, d_\psi, d_H, d_X, d_Y]$, where the first three components correspond to the atmospheric disturbances and the remaining terms correspond to the measurement noise. The nonlinear equations of motion are given in (3.1), (3.2), (3.4) and (4.3), with the relationships mapping the state and control variables to the aerodynamic forces and moments provided in (3.7) and the look-up tables mapping the throttle command and airspeed to the generated propeller thrust represented graphically in Figures 3.4 and

3.5. These equations of motion can be expressed in state-space form as $\dot{x} = f(x, \delta, d)$, where $f(\cdot, \cdot, \cdot)$ is defined in the obvious way. The following measurements are assumed to be available at a sampling rate of 20 Hz:

$$y = h(x, d) = [p, q, r, V_a, \phi, \theta, \psi, H, X, Y]^T + D_{21}\, d,$$

where $D_{21} = [0_{10\times 3} \;\; \mathrm{diag}(0.5, 0.5, 0.5, 2, 0.01, 0.01, 0.01, 2, 2, 2)]$ is chosen based on the sensor noise characteristics and some expected uncertainties.

Suppose that the control objective is to design a controller that would force the system to track closely the reference trajectory (x_r, δ_r, d_r), where $d_r = 0$. Then, linearizing the nonlinear system equation about this reference trajectory results in the following continuous-time LTV state-space equations:

$$\dot{\bar{x}}(t) = A_c(t)\bar{x}(t) + B_{1c}(t)d(t) + B_{2c}(t)\bar{\delta}(t),$$

$$\bar{y}(t) = C_{2c}(t)\bar{x}(t) + D_{21}d(t),$$

where t is continuous time, $\bar{x} = x - x_r$, $\bar{\delta} = \delta - \delta_r$, $\bar{y} = y - y_r$, and

$$A_c = \left.\frac{\partial f}{\partial x}\right|_{(x_r,\delta_r,d_r)} \qquad B_{1c} = \left.\frac{\partial f}{\partial d}\right|_{(x_r,\delta_r,d_r)} \qquad B_{2c} = \left.\frac{\partial f}{\partial \delta}\right|_{(x_r,\delta_r,d_r)} \qquad C_{2c} = \left.\frac{\partial h}{\partial x}\right|_{(x_r,d_r)}$$

with $d_r = 0$ as aforementioned. Note that, since $f(x, \delta, d)$ depends on the first three components of d only, namely $V_w = [u_w, v_w, w_w]^T$, the matrix-valued function $B_{1c}(t)$ is of the form $B_{1c}(t) = [B_{1w}(t) \;\; 0_{12\times 10}]$, where $B_{1w} = \left.\dfrac{\partial f}{\partial V_w}\right|_{(x_r,\delta_r,d_r)}$ with d_r and specifically V_{wr} set to zero. The Jacobian matrices are computed using the small perturbation method because of the use of the Javaprop look-up tables.

Discretizing the previous equations using zero-order hold sampling with sampling time $\tau = 0.05\,\text{s}$ yields the following discrete-time LTV model:

$$\bar{x}_{k+1} = A_k \bar{x}_k + B_{1k} d_k + B_{2k} \bar{\delta}_k,$$

$$\bar{y}_k = C_{2k} \bar{x}_k + D_{21} d_k,$$

where $\bar{x}_k = \bar{x}(k\tau)$, $\bar{\delta}_k = \bar{\delta}(k\tau)$, $\bar{y}_k = \bar{y}(k\tau)$, $C_{2k} = C_{2c}(k\tau)$ for integers $k \geq 0$, $A_k = \Phi\left((k+1)\tau, k\tau\right)$, with $\Phi(\cdot,\cdot)$ being the state transition matrix, and, for $i = 1, 2$,

$$B_{ik} = \int_{k\tau}^{(k+1)\tau} \Phi\left((k+1)\tau, s\right) B_{ic}(s) ds.$$

If the reference trajectory is a maneuver defined over a finite time interval, then the resulting linearized model will be a finite-horizon model. In the event of a trim reference trajectory, the linearized model will be LTI, in which case all the state-space matrices will be constant and the state transition matrix $\Phi\left(t_2, t_1\right)$ will simply be the matrix exponential $e^{A(t_2 - t_1)}$. The performance output z will be defined in the following sections as we consider different trajectory tracking problems.

4.2 Tracking of a Figure-8 Trajectory

We now implement the approaches discussed in Section 4.1.2 to control the Telemaster UAV about a reference trajectory generated (in real-time) from a library of pre-specified motion primitives. Specifically, a standard H_∞ controller and a switched UIC controller will be designed and tested in simulation and on the actual UAV. The section is divided into three

subsections: the first gives the motion primitives that will be used in generating the reference trajectory; the second presents the control systems; and the third provides the simulation and flight testing results, along with some interesting observations.

4.2.1 Motion primitives

A motion primitive is a dynamically feasible trajectory that can be obtained experimentally from flight tests or, as is the case in this chapter, numerically from the nonlinear equations of motion. Specifically, the transition maneuvers in our case will be obtained by simulation, as discussed later, and the trim conditions will be generated using the MATLAB command `fminsearch`.

We construct a library of motion primitives that consists of three equilibrium (trimmed) conditions and four transition maneuvers. The primitives are judiciously chosen to achieve the desired maneuverability. The trim conditions considered correspond to straight and level flight (SSLF), steady right turn (SRT), and steady left turn (SLT), all of which obtained at a nominal airspeed of 15 m/s and an altitude of 580 m (above sea level). Furthermore, the steady turns are performed at a bank angle of 30° for the clockwise direction and -30° for the counter-clockwise direction. The required bank angle and airspeed are passed on to a trimming routine in the simulation environment. The trim condition search process is based on the Nelder-Mead simplex algorithm, which can be invoked by calling the `fminsearch` command in MATLAB. Table 4.1 gives the values of the states and control inputs corre-

Table 4.1: Trim conditions used in generating the figure-8 trajectory

Trim Condition	p [rad/s]	q [rad/s]	r [rad/s]	u [m/s]	v [m/s]	w [m/s]	H [m]
straight and level	0.0000	0.0000	0.0000	15.0	1.0	-0.2	580.0
steady right turn	-0.0131	0.1882	0.3260	15.0	0.9	0.1	580.0
steady left turn	-0.0120	0.1882	-0.3259	15.0	1.1	0.1	580.0
Trim Condition	ϕ [rad]	θ [rad]	ψ [rad]	δ_e [pwm]	δ_a [pwm]	δ_r [pwm]	δ_t [pwm]
straight and level	0.0000	-0.0132	0.0000	0.0045	-0.0574	-0.0525	0.3956
steady right turn	0.5236	0.0347	0.0000	0.0471	-0.0777	-0.0851	0.4135
steady left turn	-0.5236	-0.0319	0.0000	0.0469	-0.0406	-0.0167	0.4135

sponding to the aforementioned trim conditions. Note that, in all the tables and figures herein, the control surface deflections are given in terms of the deviations from the reference pwm values. In the case of the elevator, aileron, and rudder, the reference value is 1.5 ms, which corresponds to approximately zero deflection. As for the throttle, the reference value is 1.1 ms and corresponds to the minimum possible throttle deflection.

The transition maneuvers in our case begin and end at trim conditions. As mentioned before, these maneuvers can be generated using several methods. For generating the transition maneuvers in this work, linear feedback controllers are used. Specifically, the nonlinear system (sans actuator dynamics) is linearized about the trim conditions and then the resulting LTI models are discretized using zero-order-hold sampling with a sampling time of 0.05 s, which is the sampling time of the autopilot hardware onboard the UAV. Given these linearized models, we then design a Linear Quadratic Regulator (LQR) for each of the discrete-time linear models using the MATLAB command `dlqr`. In the LQR design, we evenly penalize all the states and inputs. Then, for example, to generate the straight and level flight to

steady right turn maneuver, the simulation is set to start from the straight and level trim condition. The feedback controller is set to use the LQR that is obtained for the steady right turn. The reference (equilibrium) state and control values for this controller correspond to the steady right turn. The simulated maneuver is truncated when the values of the states are relatively close to the target ones. Note that this simple method may not work when the two trim conditions are relatively far apart.

Following this approach, we end up with four transition maneuvers, namely straight-to-right, right-to-straight, straight-to-left, and left-to-straight maneuvers. The time span of each transition is constant and equal to about 5 s (100 sampling points). To generate the figure-8 path, the amount of time the UAV flies along each trim trajectory (i.e., coasting time) is varied to result in the desired geometric path with the errors in the X and Y positions at transition-to-trim connection points being less than 0.5 m. Figure 4.2 shows the motion primitives composing the library and the desired reference trajectory generated from these primitives. As can be seen from the figure, the angle of attack at trim is not zero; the reason is that the required lift is not achieved at zero angle of attack. In addition, the non-zero sideslip follows from the asymmetry of the aircraft, which is due to manufacturing defects and/or propeller effects.

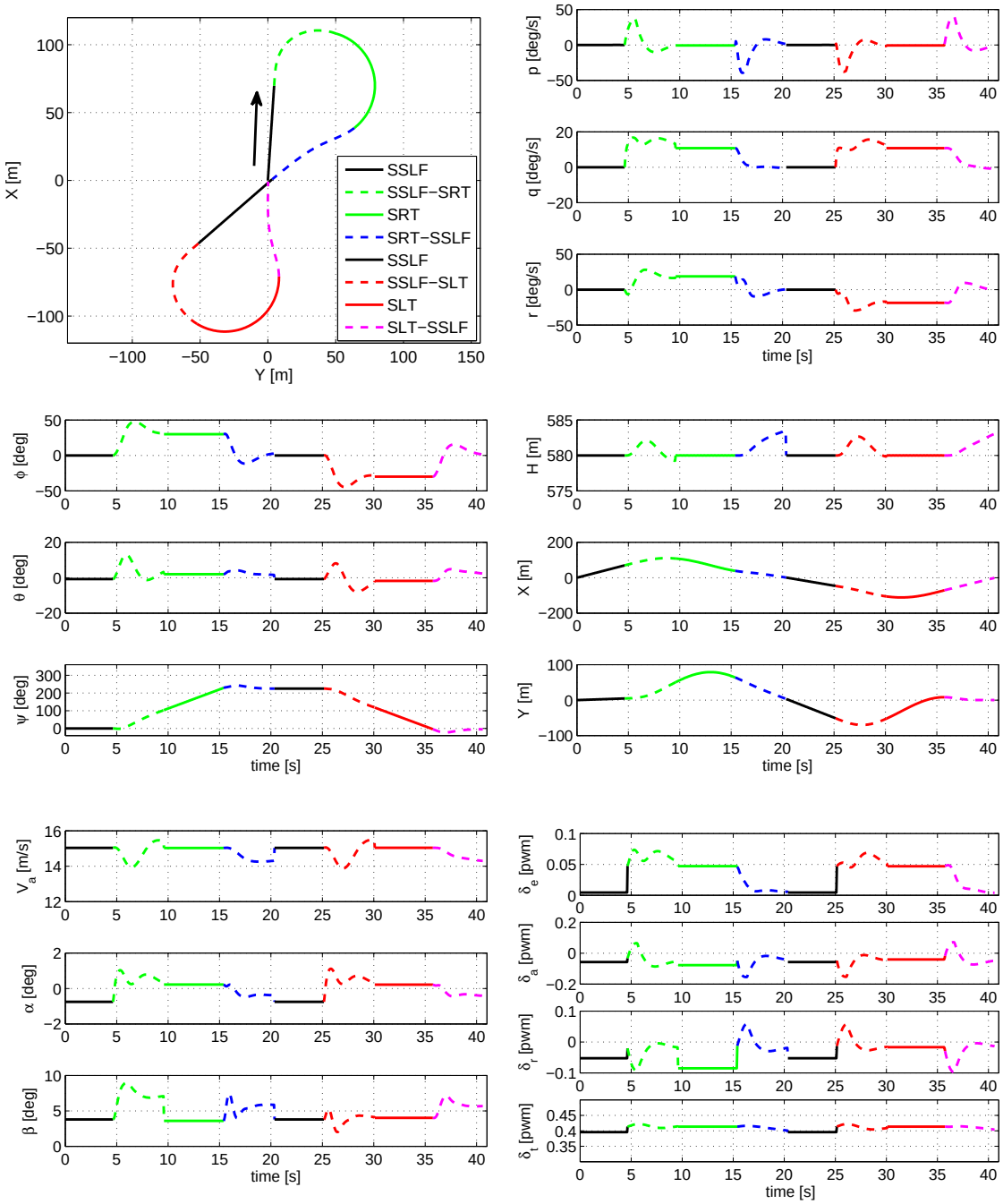

Figure 4.2: Reference trajectory of a figure-8 pattern obtained by concatenating seven motion primitives

4.2.2 Control design

We now design two controllers based on the aforementioned library of motion primitives: a standard H_∞ controller and a switched UIC controller. Although the controllers will be ultimately used to track the figure-8 trajectory, the only assumption made about the trajectory to be tracked in designing these controllers is that this trajectory will be generated in real-time by concatenating primitives from the library.

Standard H_∞ Controller

As the steady turn trim trajectories and transition maneuvers do not deviate significantly from the straight and level flight, it is conceivable that a properly designed standard H_∞ controller, synthesised based on a linearized plant model about the straight and level flight, should work reasonably well. The linearized state equation would then be $\Delta\dot{x} = A\Delta x + B_1 d + B_2\Delta\delta$, where $\Delta x = x - x_{\text{trim}}$ and $\Delta\delta = \delta - \delta_{\text{trim}}$, with $(x_{\text{trim}}, \delta_{\text{trim}})$ designating the state and control input corresponding to the straight and level trim trajectory. We may also write $\Delta\dot{x}_r \approx A\Delta x_r + B_2\Delta\delta_r$, for "small" $\Delta x_r = x_r - x_{\text{trim}}$ and $\Delta\delta_r = \delta_r - \delta_{\text{trim}}$, where (x_r, δ_r) denote the state and control input corresponding to the trajectory to be tracked. Then, as $\bar{x} = x - x_r = \Delta x - \Delta x_r$ and $\bar{\delta} = \delta - \delta_r = \Delta\delta - \Delta\delta_r$, the plant state equation can be expressed as $\dot{\bar{x}} = A\bar{x} + B_1 d + B_2\bar{\delta}$. The rest of the plant formulation follows the description given in Section 4.1.3, with the performance output z judiciously chosen as

$$z = [0.2\,\bar{p}, 0.2\,\bar{q}, 0.2\,\bar{r}, 0.1\,\bar{u}, 0.01\,\bar{\phi}, 0.01\,\bar{\theta}, 0.01\,\bar{\psi}, 0.01\,\bar{H}, 0.05\,\bar{X}, 0.01\,\bar{Y},$$
$$0.4\,\bar{\delta}_e, 0.4\,\bar{\delta}_a, 0.6\,\bar{\delta}_r, 1.0\,\bar{\delta}_t]^T.$$

The control design follows the procedure outlined in Section 4.1.2. All computations are carried out on an ASUS laptop with an Intel dual-core Pentium M2020 2.4 GHz processor, 4 GB RAM, and 64-bit Windows 8 operating system running 64-bit MATLAB R2011a. The semidefinite programs are solved using YALMIP/SDPT3-v4 [87, 88]. The elapsed (i.e., wall clock) time to solve for the minimum achievable ℓ_2-gain performance level γ is about 1 s (CPU time = 0.78 s). The optimal value, which is found to be $\gamma_{\min} = 0.33$, is relaxed to 0.35 in order to obtain a satisfactory controller performance.

Switched UIC Controller

The switched UIC controller consists of three N-eventually time-invariant subcontrollers synthesised separately based on discrete-time linear models with uncertain initial conditions, following the procedure in Section 4.1.2. The three linear plant models in question are obtained by linearizing the equations of motion about the straight and level, steady right turn, and steady left turn flights, and are formulated as described in Section 4.1.3.

Choosing the exogenous errors z that would result in a satisfactory controller performance has proved to be a bit more challenging in this case. Namely, if we are to use the same penalty weights as before, the resulting controller tends to fail under a small amount of atmospheric

disturbances, with the failure always occurring during one of the steady turn portions of the figure-8 trajectory (the specific portion at which the failure occurs varies depending on the wind condition). A closer look at the obtained value of the direct feed-through matrix, D_k^K, of the associated subcontroller for $k = N - 1$ (that is, at the end of the finite horizon, just before the subcontroller becomes time-invariant) shows that there are certain feedback responses that do not conform with some pilot intuitions. For example, to reduce the altitude tracking error, the feedback controller commands more rudder deflection than elevator deflection. Similarly, to reduce the roll rate tracking error, the controller commands more rudder deflection than aileron deflection. These arrangements, while they might work when the airplane is flying closely to the designed trim condition, will result in an undesired behavior when the deviation from the designated flight condition becomes significant, which is to be expected in the event of significant atmospheric disturbances. One solution that works in our case is that, instead of choosing the component of z that penalizes, say, the roll rate tracking error as $c\,\bar{p}$, for some scalar c, we choose this component as $c_1\bar{p} + c_2\bar{\delta}_a$, for some scalars c_1 and c_2, so that the controller is guided to use aileron corrections to reduce the error in the roll rate. This desired action is reflected in the D-matrix of the controller. As a result, we choose the performance output z in the three plant models as

$$z = [0.191\,\bar{p} + 2.0\,\bar{\delta}_a, 0.191\,\bar{q} + 0.1\,\bar{\delta}_e, 0.191\,\bar{r} + 0.1\,\bar{\delta}_r, 0.1\,\bar{u} + \bar{\delta}_t, 0.191\,\bar{\phi}, 0.191\,\bar{\theta}, 0.191\,\bar{\psi},$$

$$0.03\,\bar{H}, 0.03\,\bar{X}, 0.03\,\bar{Y}, 0.4\,\bar{\delta}_e, 0.1\,\bar{\delta}_a, 1.0\,\bar{\delta}_r, 2.0\,\bar{\delta}_t]^T.$$

For all three subcontrollers, we have found that a finite horizon length $N = 5$ gives the best performance. We assume that the initial uncertainties are mainly in the yaw rate (r) and

bank angle (ϕ). Accordingly, with $\bar{x}_0 = \Lambda\,[\bar{r}, \bar{\phi}]^T$, we define the scaling matrix Λ as

$$\Lambda = \begin{bmatrix} 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T .$$

The minimum achievable performance level is $\gamma_{\min} \approx 1.27$ for each of the three 5-eventually time-invariant subcontrollers; this value is then relaxed to 1.3 in order to obtain numerically well-conditioned solutions. The wall clock time to find the optimal solution is about 16 s (CPU time $\approx$ 15 s) for the straight and level flight case and about 9.5 s (CPU time $\approx$ 8.5 s) for each of the steady turn cases. As for the implementation of the switched controller, we start by executing the subcontroller associated with the steady straight and level flight; then, when it is time to implement the appropriate transition maneuver to make a steady right turn, the subcontroller associated with the steady right turn is activated and used to force the system to track the transition maneuver followed by the steady right turn, and so on.

4.2.3 Simulation and flight test results

The performance of each designed controller is first analyzed through simulation in a realistic operational environment to ensure that the controller can be safely implemented on the actual test platform. As opposed to merely following a geometric path, the control goal in this work is to ensure, among other things, reasonably accurate tracking of the reference time histories of the inertial X-, Y-, and H-positions. For this reason, we define two metrics to quantitatively compare the position tracking performance of the two controllers, namely

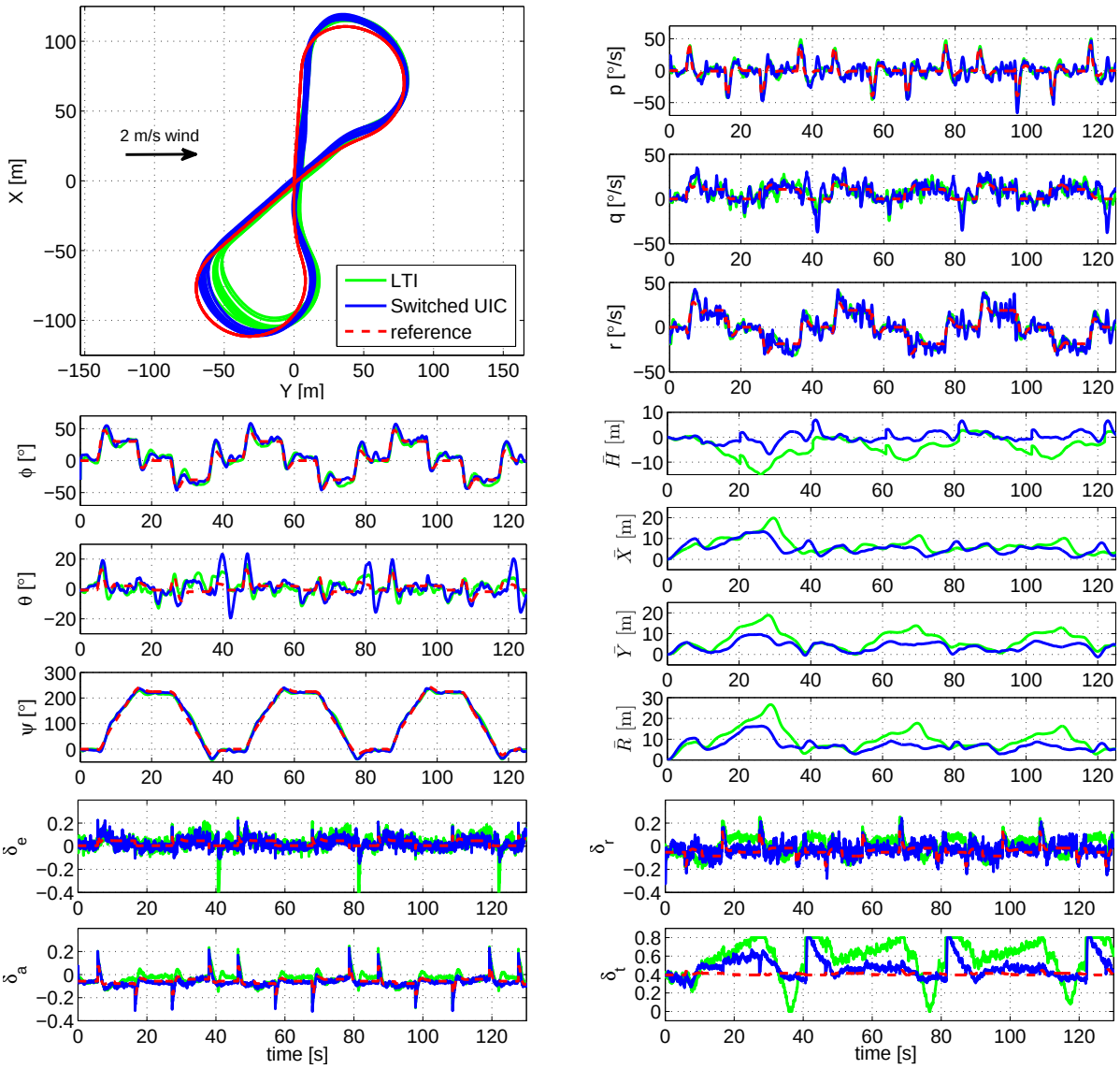

Figure 4.3: Simulation of the controllers' performance in forcing the nonlinear system, including the servomotors, to track the figure-8 trajectory under 2 m/s steady wind, light low-altitude turbulence, one-step time delay, and measurement noise

the root-mean-square (RMS) altitude error, $\sqrt{(\sum_{k=0}^{N} \bar{H}_k^2)/(N+1)}$, and the RMS planar position error, $\sqrt{(\sum_{k=0}^{N} \bar{R}_k^2)/(N+1)}$, where the discrete instant N is the time at which the simulation/test terminates and $\bar{R}_k = \sqrt{\bar{X}_k^2 + \bar{Y}_k^2}$.

The simulation is performed in MATLAB using ODE45. In the simulation, we include the

servomotor dynamics as well as a one-step time delay, whereby the control input computed at discrete time k is applied at time $k+1$. We impose lower and upper saturation limits on the four commanded control (pwm) signals of 1.1 ms and 1.9 ms, respectively. The system is also subjected to zero-mean Gaussian measurement noise, with standard deviations 0.5 rad/s for p, q, r, 2 m/s for V_a, 0.01 rad for ϕ, θ, ψ, and 2 m for H, X, Y (the same as the entries of the D_{21} matrix). As for wind disturbances, a constant wind of 2 m/s is applied, along with the corresponding turbulence generated using the low-altitude Dryden turbulence model [83] (specifically, airspeed used in Dryden model is 15 m/s, altitude above surface level is 80 m, and wind speed at 6 m [20 ft] above surface level is $W_{20} = 2$ m/s). For each controller, the same measurement noise and atmospheric turbulence are used, and the simulation is run long enough for the UAV to track the figure-8 trajectory ten times.

Figure 4.3 shows the simulation results; for clarity, we only include the state and control time histories for three traversals of the figure-8 trajectory, but the displayed results are representative of the complete simulation run. The RMS planar position error is 9.76 m for the standard H_∞ controller and 7.11 m for the switched UIC controller. The RMS altitude error is 5.12 m for the H_∞ controller and 2.07 m for the switched controller. Note that the control inputs vary rapidly because of the excessive sensor noise used in the simulation environment to account for sensor inaccuracies. In general, both controllers perform well, with the switched UIC controller clearly outperforming the standard H_∞ one in tracking the reference inertial positions, especially the altitude.

The controllers are also implemented on the Telemaster platform. The flight tests are per-

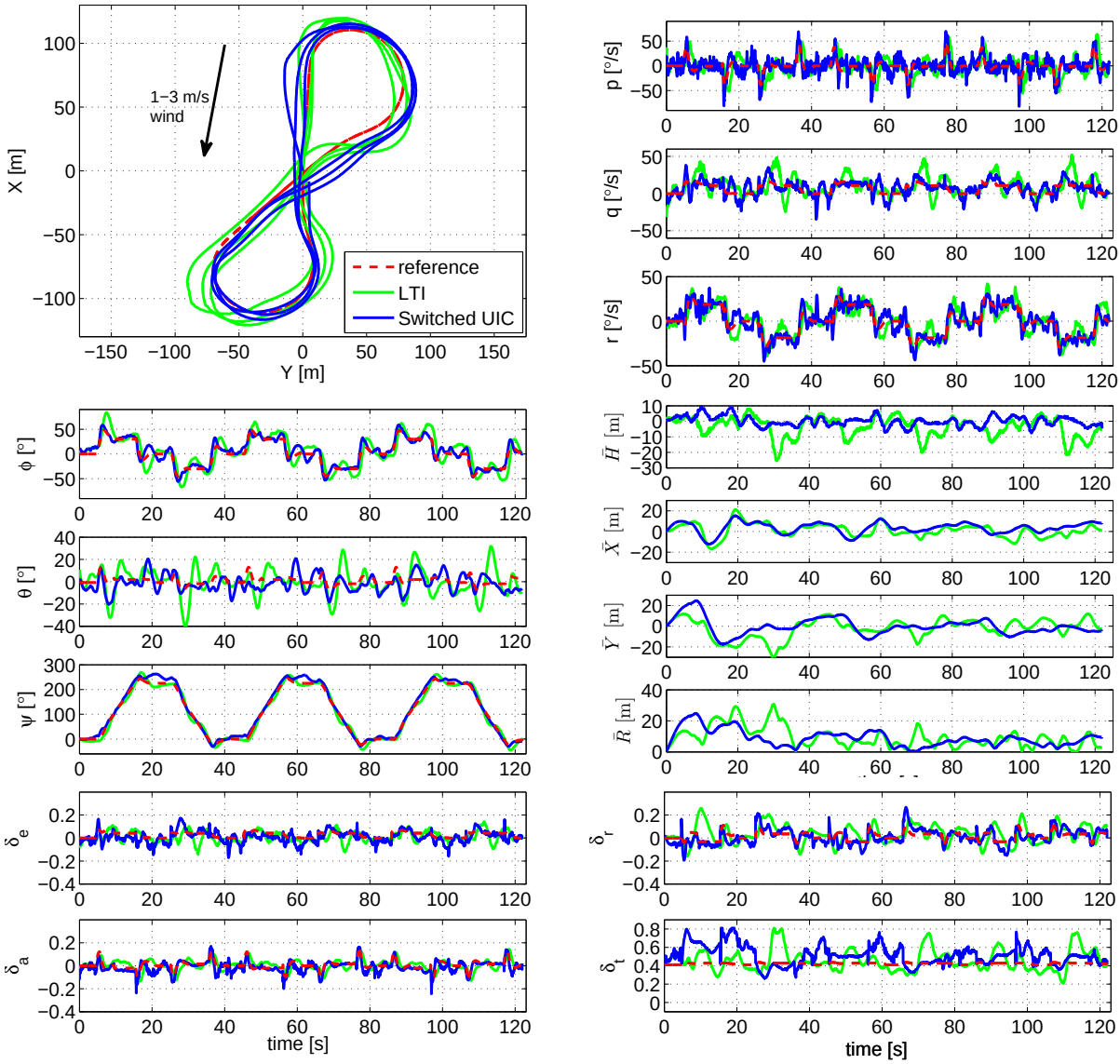

Figure 4.4: Controller tracking performance in flight tests

formed on a calm-air day, where the wind speed is 1-3 m/s based on data from a nearby on-ground station. Each test is performed in a separate flight, with a battery change taking place in-between tests. The results from both tests are given in Figure 4.4. Note that, since the tests are run in separate flights, the wind disturbances differ from test to test, and hence any performance comparison between the two controllers should be construed qualitatively.

With this said, it is observed that the results from the flight tests are in agreement with the simulation results in that both controllers perform comparably well. The values of the RMS planar position error are 11.46 m and 10.25 m, and those of the RMS altitude error are 7.44 m and 3.16 m, for the standard H_∞ and switched UIC controllers, respectively. Thus, while the advantage of the switched UIC controller in tracking the reference planar position is not as pronounced as in the simulation experiments, the switched controller still notably outperforms the standard H_∞ controller in tracking the reference altitude. The reason for the improved performance in altitude tracking is that the UIC controller is designed to anticipate relatively significant errors in the yaw rate and bank angle when switching between trim trajectories, and so it does not over-react initially to compensate for these errors, maintaining altitude as a result. All in all, even though the results are generally influenced by the way we formulate the control problems, the switched UIC control approach seems to be quite promising especially when it comes to handling tracking errors along switching boundaries.

4.3 Tracking of an Aerobatic Maneuver

In this section, we design and implement a switched controller that executes two tasks consecutively: the first task is performing an aerobatic Split-S maneuver and the second is tracking a relatively tight circular trajectory of 85.65 m radius. The controller has to ensure close tracking of the reference trajectory, with acceptable control action, despite the presence of relatively significant atmospheric disturbances and model uncertainties. It is composed

of two subcontrollers, each designed for a task. The subcontroller for the second task is activated once the Split-S maneuver is executed. The position of the UAV after running the first subcontroller is regarded as the initial reference position on the circular trajectory, that is, when it comes to tracking the circular trajectory, there is no initial error in position, but there will most probably be initial errors in attitude angles, rates, and airspeed. The control design process consists of generating the desired trajectories, namely the Split-S maneuver and level-turn trim trajectory, then linearizing the nonlinear equations of motion (sans the servomotor dynamics) about these trajectories, and finally applying the appropriate control techniques to design the subcontrollers.

4.3.1 Split-S Maneuver

The Split-S maneuver is generated from recorded pilot data, and the corresponding linearized model is a finite horizon LTV system. We do not consider an uncertain initial condition in this case since we make sure to start the experiment as close to the initial reference point as possible. Note, however, that incorporating an uncertain initial condition into the control design process can be easily achieved by applying the appropriate results from [20]. The control problem then boils down to applying the procedure outlined in Section 4.1.2 to design a standard finite horizon, ℓ_2-induced norm controller.

The reference trajectory can be generated by solving an optimization problem involving the nonlinear equations of motion of the aircraft. In this work, however, we opt to obtain the

reference trajectory experimentally. Namely, a number of Split-S maneuvers are performed by a human pilot, with each maneuver starting at roughly the same trim speed and obtained using similar control inputs. The recorded data is analyzed and several good runs are selected. The reference control input is chosen as the average of the control inputs of the usable runs. The nonlinear equations of motion are then solved numerically for the state trajectory corresponding to the reference control input, that is, the reference state trajectory is obtained in simulation. Figure 4.5 shows the used pilot-recorded data (purple dashed curves) and the resulting reference trajectories (black solid curves). Observe from this figure that, first, the obtained reference trajectories indeed correspond to a Split-S maneuver, and, second, the UAV does not reach an equilibrium at the end of the maneuver. The latter observation is not problematic as the subcontroller used to track the subsequent level-turn trim trajectory is designed to compensate for uncertain initial conditions.

The discrete-time plant model is obtained by linearizing the equations of motion about this reference trajectory, following the procedure described in Section 4.1.3. The plant model in this case is a finite horizon LTV system, with finite horizon length $h = 259$. We have found by trial and error that using the scaled disturbance input matrix $\hat{B}_{1k} = B_{1k}\,\mathrm{diag}(8, 8, 1)$ results in an improved tracking performance. Concerning the formulation of the performance output z, preliminary flight tests show that the angular rates are the most important variables to be tracked in order to successfully execute the maneuver. In addition, close tracking of the attitude angles and planar position is required at the beginning and end of the maneuver to orient the aircraft correctly and drive it to the desired X- and Y-positions. Since the control

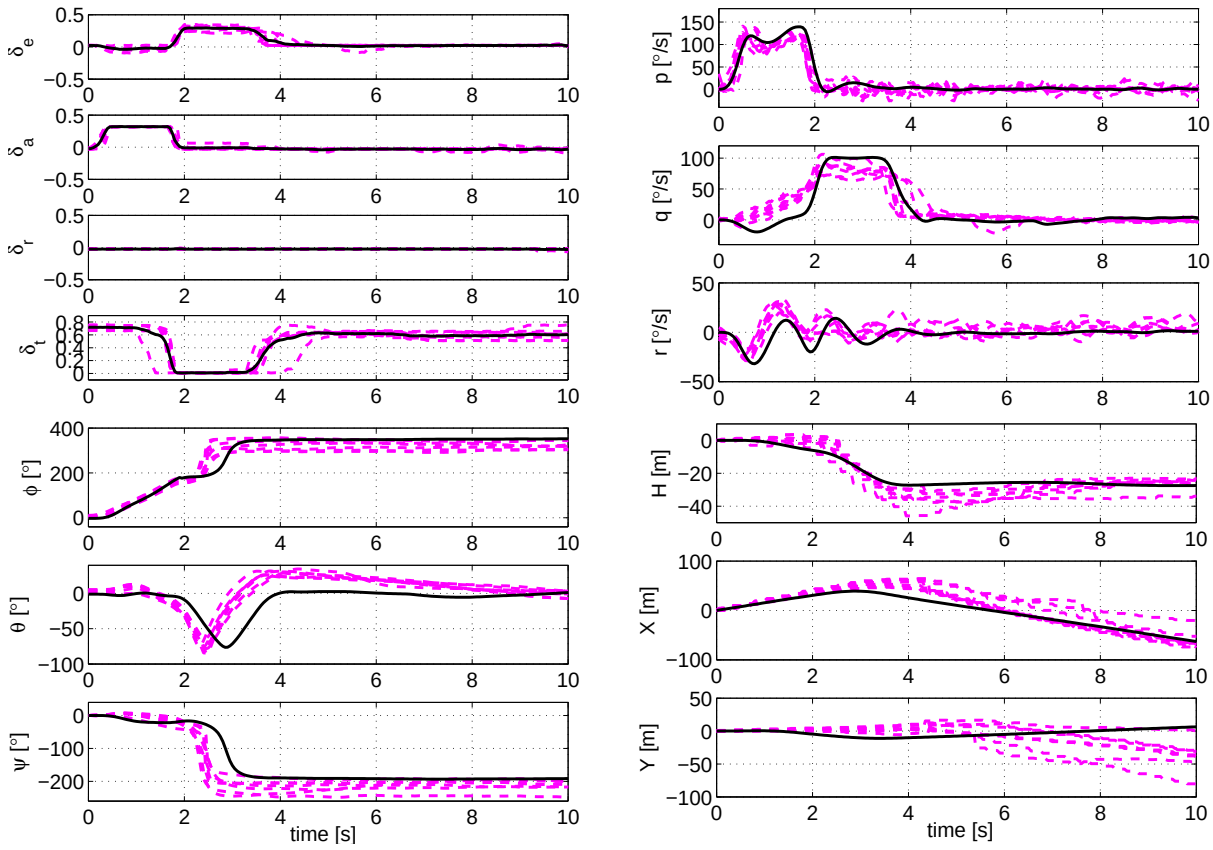

Figure 4.5: Split-S reference trajectory generation from flight test data

design is based on a nonstationary plant model, the use of time-varying penalty weights is permitted in the plant formulation and hence is exercised in choosing the penalties on the states $\bar{\phi}$, $\bar{\theta}$, $\bar{\psi}$, $\bar{X}$ and $\bar{Y}$. As for penalizing the control inputs, only the elevator and aileron are expected to be used to actively compensate for any errors in the tracking of the Split-S maneuver. The deflections of the rudder and throttle should be restricted during the maneuver and hence are heavily penalized.

The performance output is then defined as

$$z_k = [0.1\,\bar{p}_k, 0.1\,\bar{q}_k, 0.1\,\bar{r}_k, 0.4\,\alpha_k\bar{\phi}_k, 0.4\,\alpha_k\bar{\theta}_k, 0.4\,\alpha_k\bar{\psi}_k, 0.01\,\alpha_k\bar{X}_k, 0.02\,\alpha_k\bar{Y}_k,$$
$$0.2\,\bar{\delta}_{e,k}, 0.2\,\bar{\delta}_{a,k}, 2.0\,\bar{\delta}_{r,k}, 1.0\,\bar{\delta}_{t,k}]^T, \tag{4.4}$$

where $\alpha_k = \alpha(k\tau)$, and the time-varying multiplier function $\alpha(t)$ is given by

$$\alpha(t) = \begin{cases} 1 - \frac{t}{3} & 0 \leq t < 3 \\ 0 & 3 \leq t < 7 \\ \frac{t-7}{3} & 7 \leq t < 10 \\ 1 & t \geq 10. \end{cases}$$

Notice that the multiplier function takes the value of zero in the middle portion of the maneuver. The absence of penalty on the attitude angles and planar position will focus the control effort on tracking the angular rates. This is a reasonable strategy for two reasons: first, the angular rates are the most critical outputs in the middle portion of the maneuver and, second, the measurements of the attitude angles and position are not directly obtained from the corresponding sensors in this portion but are rather estimated numerically from the angular rates and accelerations using the equations of motion as discussed later. Despite not explicitly penalizing the altitude, this choice of the performance output results in a reasonably accurate tracking of the reference altitude trajectory, as will be seen in the simulation and flight tests.

The same measurements are used as before, except in this case the values of the attitude angles and position vector are not obtained directly from the AHRS and GPS. Due to the nature of the maneuver, the GPS may temporarily lose connection to the satellites and, furthermore, gimbal lock may occur rendering the AHRS measurements of the attitude angles inaccurate. To circumvent these potential problems, the attitude angles are obtained from the measured angular rates by integrating equation (3.4). The Euler method, with step size equal to the sampling time, is chosen for numerical integration since it gives sufficient accuracy over the targeted time interval and requires very low computational power. Similarly, the position is obtained from the measured accelerations and angular rates, along with the estimated attitude angles, using equations (3.1) and (4.3), with position updates incorporated whenever good GPS data is available. This process is commonly referred to as dead reckoning.

All computations are carried out in MATLAB R2009b, using YALMIP along with SDPT3 v4, on a Dell desktop with a quad-core Intel Xeon, 3.07 GHz processor and 6 GB of RAM running Windows 7. Note that, although the desktop has a quad-core processor, the solver only utilizes one of the CPUs for solving the optimization/feasibility problems. The wall clock time for solving the optimization problem is about 1003 s (CPU time $\approx$ 328 s); the minimum achievable performance level γ is found to be about 0.50. This value is then relaxed to 0.65 in order to obtain satisfactory performance in the presence of modeling uncertainties. It takes about 869 s (CPU time $\approx$ 259 s) to solve the feasibility problem for $\gamma = 0.65$, and about 82 s to construct the controller using the MATLAB command `basiclmi`.

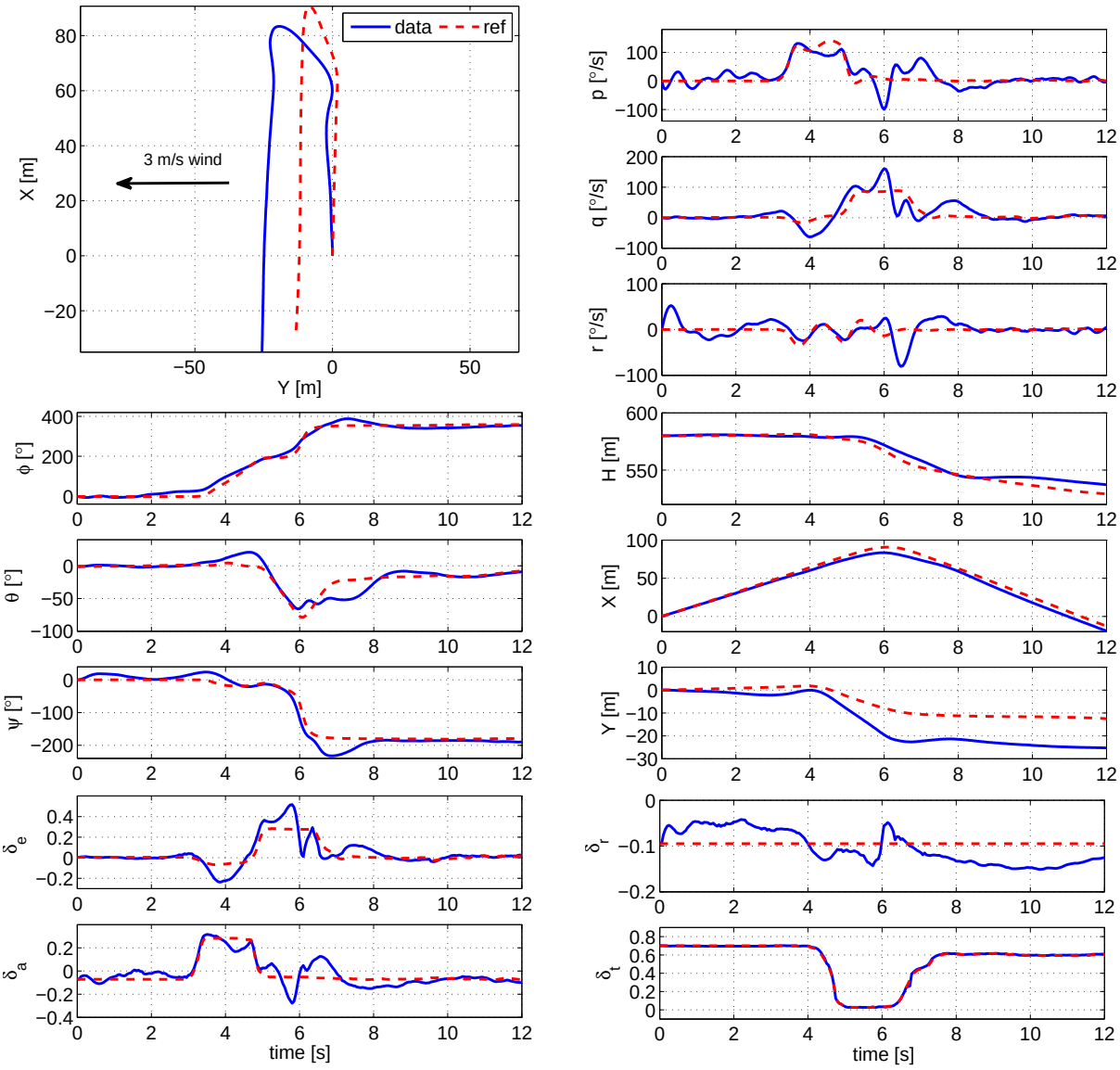

Figure 4.6: A representative simulation run for Split-S maneuver tracking under 3 m/s steady wind and medium turbulence

The designed controller is tested in the same simulation environment described before. The wind disturbance in this case consists of a steady easterly wind of 3 m/s and medium turbulence generated by the low-altitude Dryden model (with $W_{20} = 10$ m/s in this case). A representative simulation run is shown in Figure 4.6. The effect of the steady wind can be

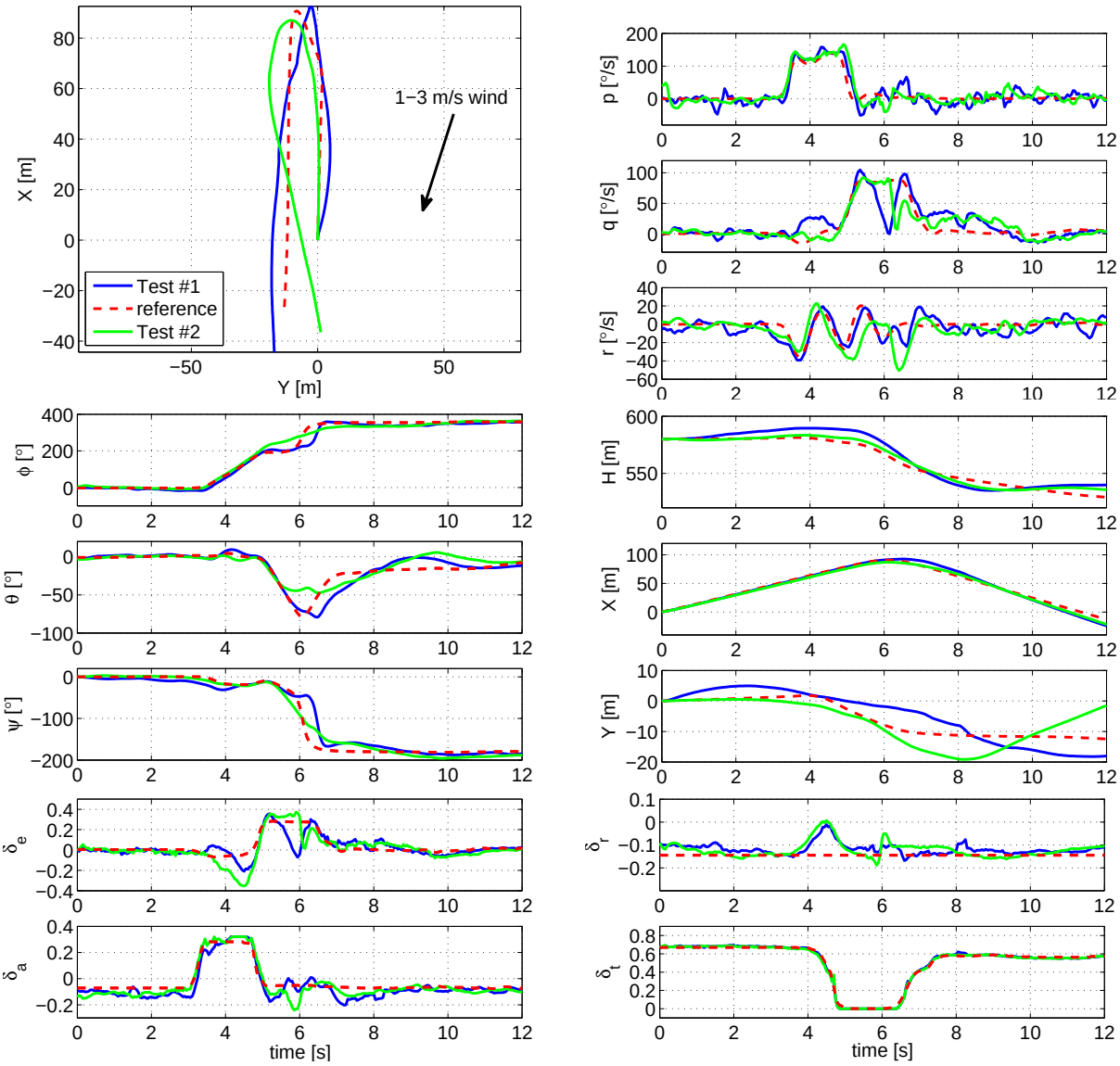

Figure 4.7: Split-S test results with penalty on position

clearly seen in the top left plot, where the traversed path is pushed to the left from the reference by around 10 m. Other than that, the controller performs well, keeping the vehicle relatively close to the reference trajectory.

The flight tests for this controller are carried out under light air to light breeze wind conditions (Beaufort scale). At the beginning of each test, the UAV is subjected to a $1-3\,\mathrm{m/s}$

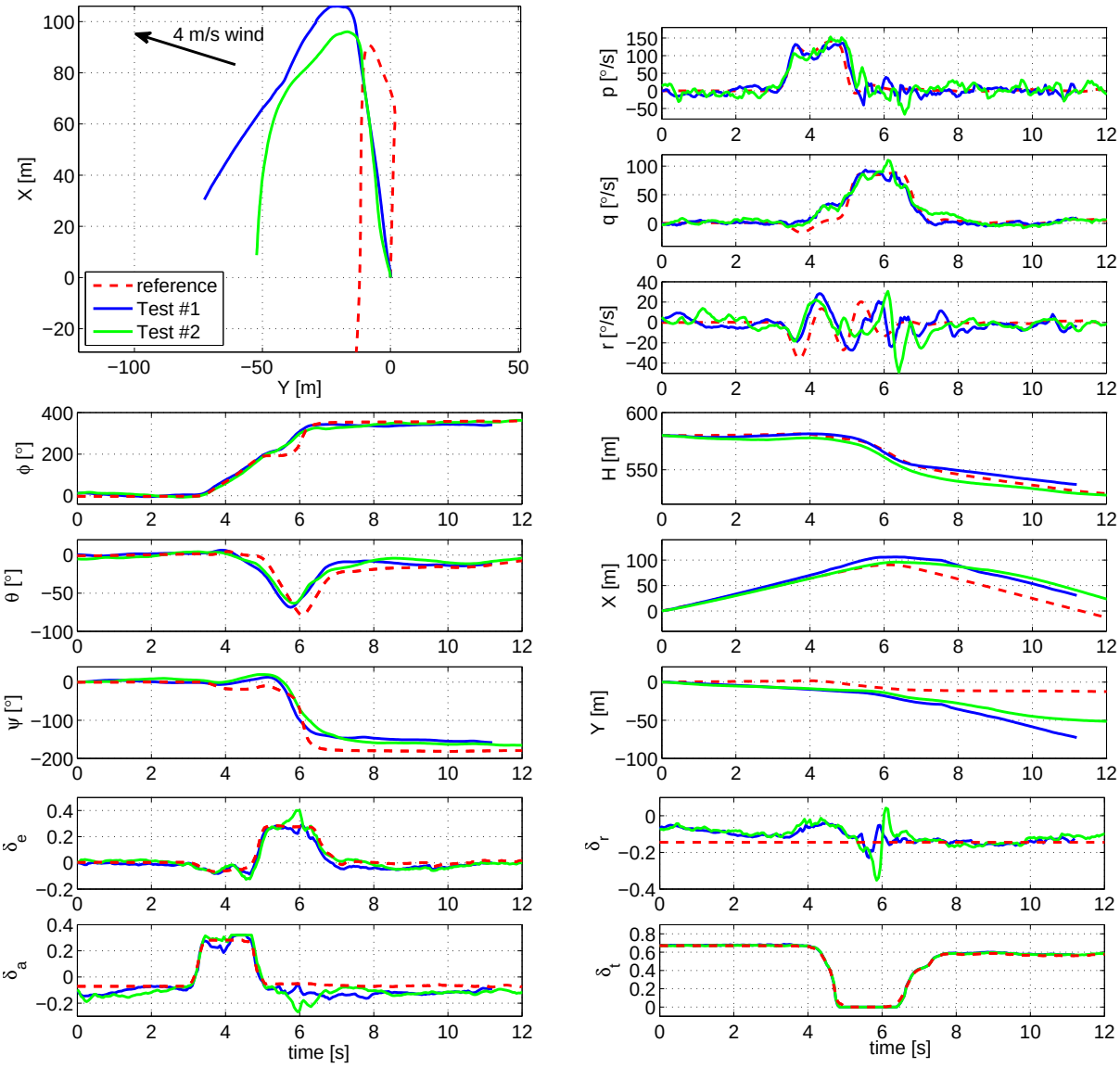

Figure 4.8: Split-S test results with no penalty on position

headwind on average. Figure 4.7 shows the results of two representative flight tests. As evident from this figure, the controller manages to track the Split-S maneuver closely in both tests despite the wind disturbances. In general, imposing penalties on the position errors increases the accuracy of position tracking. This improved tracking, however, comes at the expense of a degraded ability to compensate for relatively high wind disturbances. For

completeness, we have designed and tested a controller with the same performance output as the one in (4.4) except that in this case we do not penalize the position errors $\bar{X}$ and $\bar{Y}$. Figure 4.8 displays the results of two tests for this controller performed under gentle breeze wind conditions, where the average wind speed is around 4 m/s, with ± 2 m/s wind fluctuations, and the wind direction almost perpendicular to the flight trajectory at the onset of each test. As expected, the controller in these tests poorly tracks the reference X- and Y-positions (although the altitude tracking is still satisfactory). But this controller slightly outperforms the previous one in tracking the remaining states; specifically, it generates less oscillatory motion in tracking the attitude angles and angular rates even though the wind conditions are more severe in this case.

4.3.2 Steady Right Turn Flight

As discussed in Section 4.2.1, the level-turn trim trajectory is generated from the mathematical model of the UAV using the MATLAB command FMINSEARCH, and the corresponding linearized model is an LTI system. The trim condition considered in this case corresponds to a steady right turn at a nominal speed of 15 m/s, an altitude of 580 m above sea level, and a bank angle of 15°. The resulting geometric path is a circle of radius 85.65 m. The time it takes to traverse this circle at the chosen airspeed is about 36 s. The trim values of (p, q, r, u, v, w, ϕ, θ, ψ, H, δ_e, δ_a, δ_r, δ_t) are (-0.0014, 0.0452, 0.1688, 15.0, 0.9672, -0.1373, 0.2618, 0.0078, 0.0000, 580.0, 0.0147, -0.0676, -0.0697, 0.3996), respectively, where the units are as given in Table 4.1.

In this case, we have to consider a plant with an uncertain initial condition in the control design process because, in the flight tests, the task of tracking the circular trajectory is performed right after executing the Split-S maneuver. The uncertain initial condition is included to capture the effects of the non-smooth transition between the two trajectories. The desired subcontroller for this second task is then obtained by applying the procedure in Section 4.1.2. We have found that an adequate performance can be achieved using an eventually time-invariant UIC controller with finite horizon length $N = 20$, along with the following choices of performance output z and scaling matrix Λ:

$$z = [0.1\,\bar{p}, 0.1\,\bar{q}, 0.1\,\bar{r}, 0.05\,\bar{u}, 0.4\,\bar{\phi}, 0.4\,\bar{\theta}, 0.4\,\bar{\psi}, 0.05\,\bar{H}, 0.03\,\bar{X}, 0.03\,\bar{Y}, 0.4$$

$$\bar{\delta}_e, 0.2\,\bar{\delta}_a, 0.5\,\bar{\delta}_r, 2.0\,\bar{\delta}_t]^T,$$

$$\Lambda = \begin{bmatrix} \mathrm{diag}(0.1, 0.1, 0.1, 5) & 0_{4\times2} & 0_{4\times2} & 0_{4\times4} \\ 0_{2\times4} & 0_{2\times2} & \mathrm{diag}(0.5, 0.2) & 0_{2\times4} \end{bmatrix}^T.$$

Concerning the measurements, the turn rate in this case is not as high as that in the Split-S maneuver case, and, as a result, the measurements of the roll and pitch angles obtained from the AHRS are fairly accurate. On the other hand, due to magnetic interference from the onboard magnetometer as well as a few magnetic sources around the test site, the AHRS measurements of the yaw angle are not as accurate and hence the course angle is measured instead using data from the GPS. Note that integrating the angular rates is not an appropriate solution in this case because the time span of the test is relatively long and, as would be

expected, integrating noisy angular rate measurements over a relatively long period of time will introduce significant bias into the estimated values.

The computing system is the same as that used in the Split-S case. The wall clock time to find the optimal performance level γ is about 29 s (CPU time $\approx$ 23 s). The optimal value of γ is around 1.49, and this value is increased to 2.2 in order to achieve satisfactory performance in the presence of modeling uncertainties. Since this type of controller has already been covered in Section 4.2.

The simulation environment is as described before. The simulation is run for 360 s, which is long enough for the UAV to traverse the circular path ten times. The wind disturbance consists of a westerly steady wind of 3 m/s and moderate Dryden turbulence as used in the Split-S case. Figure 4.9 shows the performance of the controller in the absence and presence of moderate atmospheric turbulence. As expected, the atmospheric turbulence degrades the performance. For instance, the maximum value of the planar position error $\bar{R}$ is about 11 m in the case of no turbulence and becomes 19 m when there is turbulence. Similarly, the RMS planar position error increases from 8.08 m to 10.82 m when moderate turbulence is included in the simulation. The effects of turbulence can also be seen in the altitude tracking, where the RMS altitude error increases from 2.91 m in the case of no turbulence to 4.30 m when turbulence is included. It may be possible to further improve the performance by using dynamic penalty weights or incorporating the turbulence model into the plant formulation. However, these options will result in additional state variables and hence increase the computational complexity of the control problem.

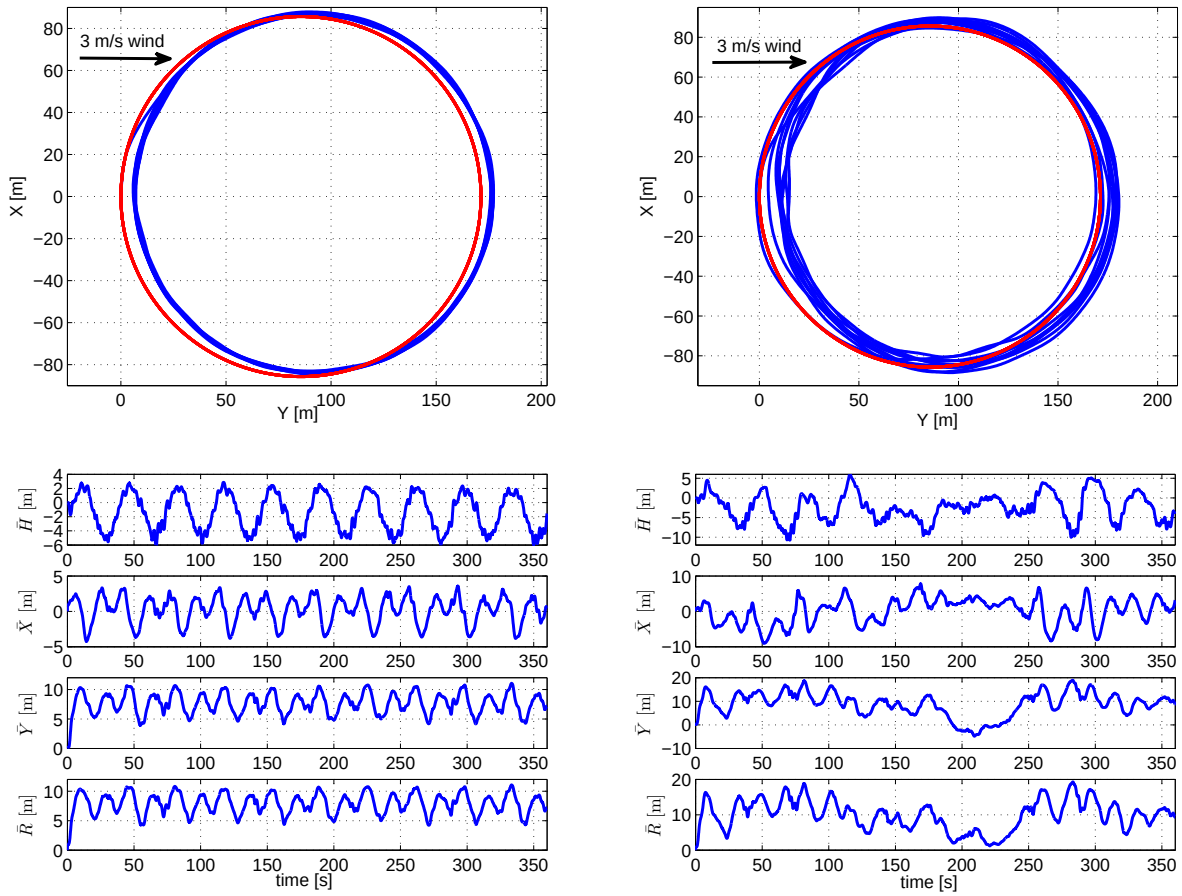

Figure 4.9: A representative simulation run for tracking of the circular trajectory under 3m/s steady wind without turbulence (left plots) and with medium turbulence (right plots)

The test results given in Figure 4.10 correspond to the circular trajectory executed right after one of the Split-S maneuvers shown in Figure 4.8. It is clear from the figure that the transition between the Split-S maneuver and the circular trajectory is reasonably smooth, for instance, the roll angle is driven to the desired value for steady right turn in about a second, with an overshoot of about 15° despite the atmospheric disturbance. The average error in planar position, that is, the mean of $\bar{R}$, for this test is around 17.7 m.

The complete runs of the tests shown in Figures 4.7 and 4.8 are given in Figure 4.11. In all

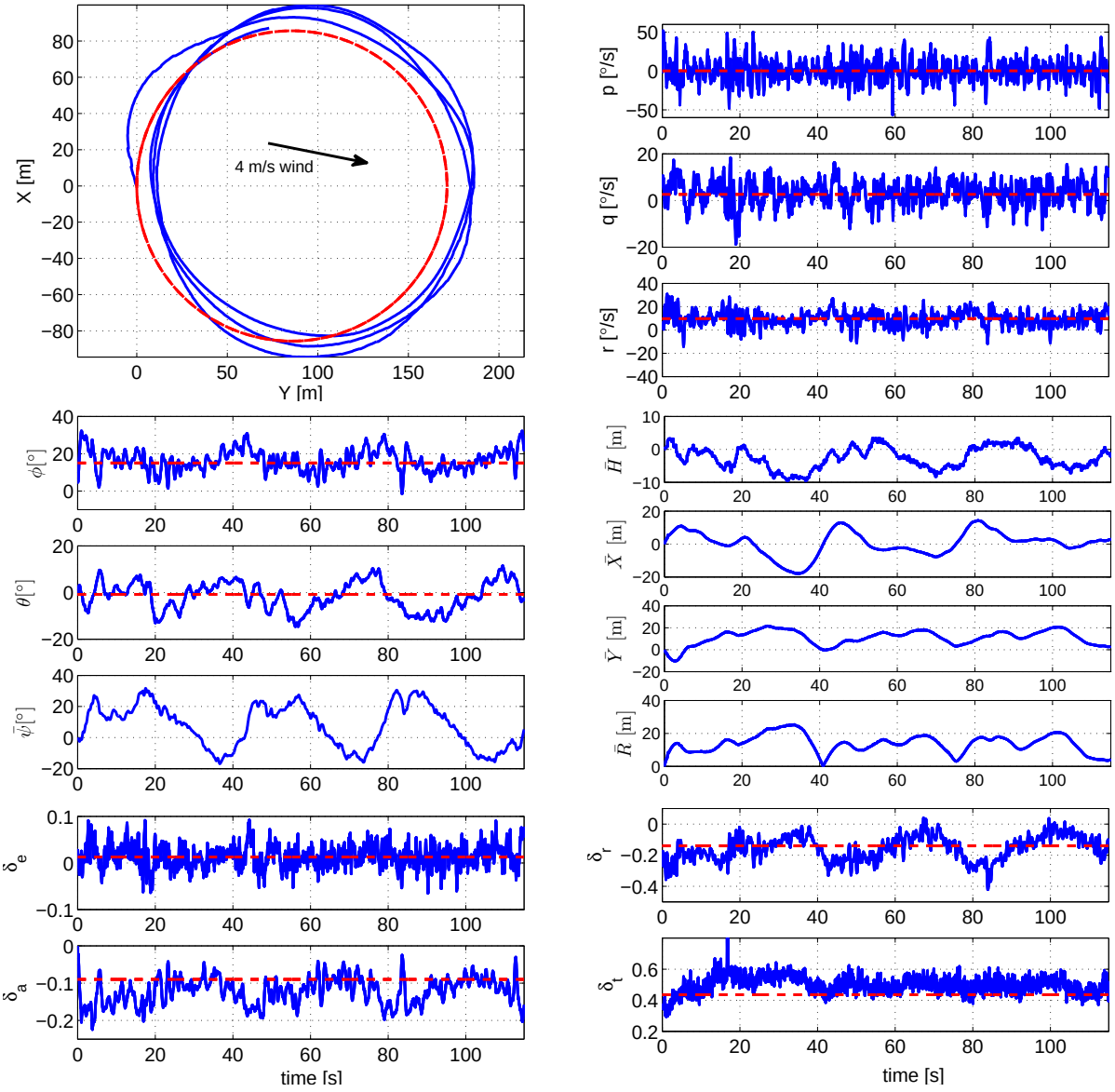

Figure 4.10: Flight test data showing the performance of the UIC subcontroller in tracking the circular trajectory

these tests, the controller performs well and achieves the desired tasks despite the significant wind disturbances. A video of one of the experiments can be found at the following address: http://www.youtube.com/watch?v=Wwwk_8WHnJ8.

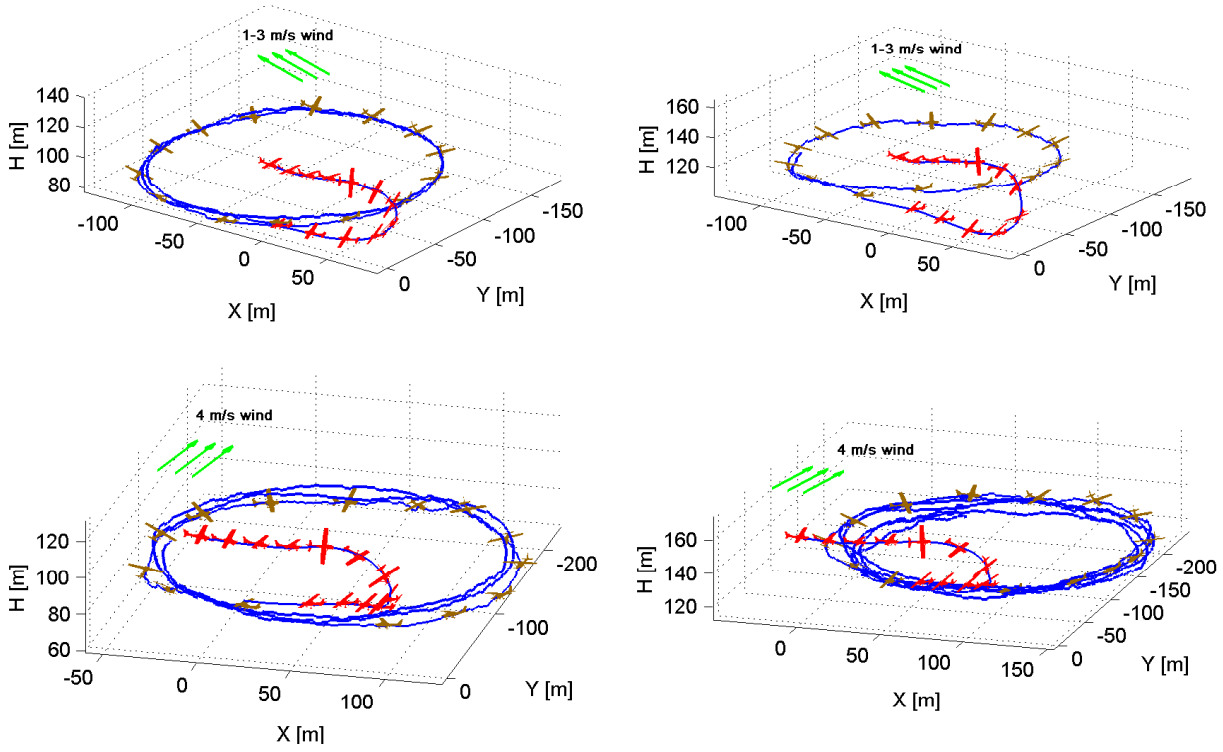

Figure 4.11: Flight test results for two switched controllers designed for tracking a Split-S maneuver that settles into a circular orbit under relatively high wind conditions; top plots correspond to the case where the planar position error is penalized in the LTV subcontroller design, and bottom plots correspond to the case where the planar position error is not penalized

Chapter 5

Conclusion

In this dissertation, the complete development process of a UAV platform for advanced control implementation is discussed. The airframe of the platform is obtained from an off-the-shelf R/C model airplane, the Telemaster, which is modified to incorporate onboard computers along with several standard and customized sensors. Two airdata probes are developed, tested, and calibrated to measure the angle of attack, sideslip, and airspeed of the UAV. Two computers are installed onboard the airplane with each assigned specific tasks. The resulting architecture enables the implementation of a control algorithm without interference from such basic functions as sensor data reading, servo actuation, and data communication.

A mathematical model of the airplane and other substantial subsystems is obtained using theoretical and empirical tools. Specifically, the output error method is employed to estimate

the longitudinal and lateral-directional aerodynamic parameters of a postulated model from flight test data. A second-order model of the servomotors is derived from the frequency response. In addition, a propulsion model is generated using in part the Javaprop applet.

The dissertation also deals with the optimal control of a small fixed-wing UAV about concatenated trajectories. The focus is on trajectory tracking rather than path following. In other words, the controllers are designed to track dynamically feasible, time-parameterized trajectories. The idea here is that a library of pre-specified motion primitives is designed a priori, and then the desired trajectories are generated in real-time by concatenating primitives from this library. A motion planner is typically used to generate the desired trajectories. The work in this dissertation concerns the design of controllers that would track the desired trajectories despite relatively significant wind disturbances and various other uncertainties. The approach adopted entails designing subcontrollers corresponding to a subset of the motion primitives. Then, as the desired trajectory is traversed, the subcontrollers will be applied in the order of the associated primitives comprising the trajectory. The main contribution of this work is that it applies systematic LMI-based control tools from robust control theory to design switched controllers for such a complex system that can track time-parameterized trajectories despite relatively significant wind disturbances, measurement noise, and other uncertainties. Specifically, the work demonstrates that recent results on control of systems with uncertain initial states are especially useful in this case as they can capture the uncertain initial conditions that come about when switching between primitives under disturbances and other uncertainties. The dissertation provides a rather complete mathematical model of

the UAV, based on which the controllers are synthesized, and presents in detail the control design process. Simulations and flight tests are carried out to demonstrate the performance of each designed controller.

Bibliography

[1] Austin M. Murch, Yew Chai Paw, Rohit Pandita, Zhefeng Li, and Gary J. Balas. A low cost small UAV flight research facility. In Florian Holzapfel and Stephan Theil, editors, *Advances in Aerospace Guidance, Navigation and Control*, pages 29–40. Springer Berlin Heidelberg, 2011. 2

[2] T. W. McLain and R. W. Beard. Unmanned air vehicle testbed for cooperative control experiments. In *Proceedings of the American Control Conference*, volume 6, pages 5327–5331, Boston, MA, June-July 2004. 2

[3] D. Jung, E. J. Levy, D. Zhou, R. Fink, J. Moshe, A. Earl, and P. Tsiotras. Design and development of a low-cost test-bed for undergraduate education in UAVs. In *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005*, pages 2739–2744, Seville, Spain, December 2005. 2, 5

[4] Mark A. Motter, Michael J. Logan, Michael L. French, and Nelson M. Guerreiro. Simulation to flight test for a UAV controls testbed. In *Proceedings of the 25th AIAA*

Aerodynamic Measurement Technology and Ground Testing Conference, San Francisco, CA, June 2006. 2

[5] T. Jordan, J. Foster, R. Bailey, and C. Belcastro. Airstar: A uav platform for flight dynamics and control system testing. In *(AIAA) 25th Aerodynamic and Measurement Technology and Ground Testing Conference*, June 2006. 2

[6] D. B. Owens, D. E. Cox, and E. A. Morelli. Development of a low-cost sub-scale aircraft for flight research: The faser project. In *(AIAA) Aerodynamic Measurement Technology and Ground Testing Conference*, June 2006. 2, 21

[7] Eloi Pereira, Karl Hedrick, and Raja Sengupta. The C3UV testbed for collaborative control and information acquisition using UAVs. In *Proceedings of the American Control Conference*, pages 1466–1471, Washington, DC, June 2013. 2

[8] R. F. Hartley and F. Hugon. Development and flight testing of a model based autopilot library for a low cost unmanned aerial systems. In *(AIAA) Guidance, Navigation and Control Conference*, August 2013. 2

[9] O. D. Dansker, M. J. Johnson, M. S. Selig, and T. W. Bretl. Development of the uiuc aero testbed: A large-scale unmanned electric aerobatic aircraft for aerodynamics research. In *(AIAA) Applied Aerodynamics Conference*, June 2013. 2

[10] E-Flite. Ultra stick 25e ARF.

URL http://www.e-fliterc.com/Products/Default.aspx?ProdID=EFL4025. 2

[11] Carl Goldberg Products Ltd. Goldberg Decathlon ARF.

URL http://www.carlgoldbergproducts.com/decathlonARF.htm. 2

[12] Ardupilot Mega.

URL http://plane.ardupilot.com/. 2, 18

[13] The Paparazzi Project, LLC. Paparazzi.

URL http://paparazzi.enac.fr/. 2

[14] D. W. Bryer and D. E. Walshe. Pressure probes selected for three-dimensional flow measurement. Repoerts and M 3037, National Advisory Committee for Aeronautics, 1955. 3, 21, 23

[15] R. G. Dominy and H. P. Hodson. An investigation of factors influencing the calibration of five-hole-probe for three-dimensional flow measurements. *Journal of Turbomachinery*, 115:513–519, 1993. 3, 23

[16] G. L. Morrison, M. T. Schobeiri, and K. R. Pappu. Five-hole pressure probe analysis technique. *Flow Measurement and Instrumentation*, 9:153–158, 1998. 3, 12, 24, 26, 28

[17] Hobby Express International. 6 Foot Telemaster Electro ARF.

URL http://www.hobbyexpress.com/telemasters_216_ctg.htm. 3, 15

[18] Gumstix. Overo Fire COM.

URL https://store.gumstix.com/index.php/products/227/. 3, 18

[19] M. Farhood and G. E. Dullerud. Control of nonstationary LPV systems. *Automatica*, 44(8):2108–2119, August 2008. 4

[20] M. Farhood and G. E. Dullerud. Control of systems with uncertain initial conditions. *IEEE Transactions on Automatic Control*, 53(11):2646–2651, December 2008. 4, 8, 64, 69, 87

[21] O. Arifianto and M. H. Farhood. Optimal control of fixed-wing uavs along real-time trajectories. In *5th Annual DSCC and 11th MOVIC*, October 2012. 4, 9

[22] Mazen Farhood. Nonstationary LPV control for trajectory tracking: a double pendulum example. *International Journal of Control*, 85(5):545–562, 2012. 4, 67

[23] M. Farhood, Z. Di, and G. E. Dullerud. Distributed control of linear time-varying systems interconnected over arbitrary graphs. *International Journal of Robust and Nonlinear Control*. 4

[24] M. Hepperle. Javaprop - design and analysis of propellers. http://www.mh-aerotools.de/airfoils/javaprop.htm. 4, 46

[25] M. W. Green. Measurement of the moments of inertia of full scale airplanes. *NACA Technical Note*, 1927. 5, 37

[26] M. P. Miller. An accurate method of measuring the moments of inertia of airplanes. *NACA Technical Note*, 1930. 5, 37, 38

[27] H. A. Soule and M. P. Miller. Experimental determination of the moments of inertia of airplanes. *NACA Technical Report*, 1933. 5, 37

[28] R. V. Jategaonkar. *Flight Vehicle System Identification: a Time Domain Methodology*. Progress in Astronautics and Aeronautics. American Institute of Aeronautics and Astronautics, 2006. 5, 50, 51, 52, 54

[29] N. V. Hoffer, C. Coopmans, A. M. Jensen, and Yang Quan Chen. Small low-cost unmanned aerial vehicle system identification: A survey and categorization. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 897–904, Atlanta, GA, May 2013. 5

[30] S. A. Salman, A. G. Sreenatha, and J. Y. Choi. Attitude dynamics identification of unmanned aerial vehicle. *International Journal of Control, Automation, and Systems*, 4(6):782–787, 2006. 5

[31] M. Manaï, A. Desbiens, and E. Gagnon. Identification of a UAV and design of a hardware-in-the-loop system for nonlinear control purposes. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2005. 5

[32] E. Frazzoli, M. A. Dahleh, and E. Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transactions on Robotics*, 21(6):1077–1091, December 2005. 7

[33] C. Goerzen, Z. Kong, and B. Mettler. A survey of motion planning algorithms from the perspective of autonomous UAV guidance. *Journal of Intelligent and Robotic Systems*, 57(1-4):65–100, 2009. 7

[34] David J. Grymin, Charles B. Neas, and Mazen Farhood. A hierarchical approach for primitive-based motion planning and control of autonomous vehicles. *Robotics and Autonomous Systems*, 62:214–228, February 2014. 7, 9

[35] Derek R Nelson, D Blake Barber, Timothy W McLain, and Randal W Beard. Vector field path following for miniature air vehicles. *IEEE Transactions on Robotics*, 23(3):519–529, 2007. 7

[36] Isaac Kaminer, Antonio Pascoal, Enric Xargay, Naira Hovakimyan, Chengyu Cao, and Vladimir Dobrokhodov. Path following for unmanned aerial vehicles using $\mathcal{L}_1$ adaptive augmentation of commercial autopilots. *AIAA Journal of Guidance, Control, and Dynamics*, 33(2):550–564, March 2010. 7

[37] Pascal Gahinet and Pierre Apkarian. A linear matrix inequality approach to h control. *International Journal of Robust and Nonlinear Control*, 4:421–448, 1994. 8, 64, 65, 67, 68

[38] G. E. Dullerud and S. G. Lall. A new approach to analysis and synthesis of time-varying systems. *IEEE Transactions on Automatic Control*, 44(8):1486–1497, August 1999. 8, 64, 65

[39] M. Farhood and G. E. Dullerud. LMI tools for eventually periodic systems. *Systems and Control Letters*, 47(5):417–432, December 2002. 8

[40] A. V. Rao, D.A. Benson, C. Darby, M.A. Patterson, C. Francolin, I. Sanders, and G.T. Huntington. Algorithm 902: GPOPS, a MATLAB software for solving multiple-phase optimal control problems using the Gauss pseudospectral method. *ACM Transactions on Mathematical Software*, 37(2):22:1–39, 2010. 8

[41] V. Gavrilets, E. Frazzoli, B. Mettler, M. Piedmonte, and E. Feron. Aggressive maneuvering of small autonomous helicopters: A human-centered approach. *International Journal of Robotics Research*, 20(10):795–807, October 2001. 8, 63

[42] M. W. McConley, M. D. Piedmonte, B. D. Appleby, E. Frazzoli, E. Feron, and M. A. Dahleh. Hybrid control for aggressive maneuvering of autonomous aerial vehicles. In *19th Digital Avionics Systems Conference*, volume 1, 2000. 8

[43] Wei Ren and Randal Beard. Trajectory tracking for unmanned air vehicles with velocity and heading rate constraints. *IEEE Transactions on Control Systems Technology*, 12:706–716, 2004. 8, 9

[44] Randal Beard, Derek Kingston, Morgan Quigley, Deryl Snyder, Reed Christiansen, Walt Johnson, Timothy McLain, and Michael Goodrich. Autonomous vehicle technologies for small fixed-wing uavs. *Journal of Aerospace Computing, Information, and Communication*, 2:92–108, 2005. 8, 9

[45] Tamas Keviczky and Gary Balas. Software-enabled receding horizon control for autonomous unmanned aerial vehicle guidance. *Journal of Guidance, Control, and Dynamics*, 29:680–694, 2006. 8, 9

[46] I. Kaminer, A. Pascoal, E. Hallberg, and C. Silvestre. Trajectory tracking forautonomous vehicles: An integrated approach to guidance and control. *Journal of Guidance, Control, and Dynamics*, 21(1):29–38, jan-feb 1998. 9, 35

[47] W. J. Hough. Autonomous aerobatic flight of a fixed-wing unmanned aerial vehicle. Master's thesis, Stellenbosch University, 2007. 9

[48] J. Gillula, H. Huang, M. P. Vitus, and C. J. Tomlin. Design of guaranteed safe maneuvers using reachable sets: Autonomous quadrotor aerobatics in theory and practice. In *2010 IEEE International Conference on Robotics and Automation*, pages 1649–1654, May 2010. 11

[49] R. G. Sanfelice. and E. Frazzoli. A hybrid control framework for robust maneuver-based motion planning. In *2008 American Control Conference*, pages 2254–2259, June 2008. 11

[50] Futaba. S3152. URL http://www.futaba-rc.com/servos/digital.html. 15

[51] Model Motors Ltd. JETI Advance Pro 40.
URL http://www.modelmotors.cz/index.php?page=63. 15

[52] Model Motors Ltd. AXI 2826/12.

URL http://www.modelmotors.cz/index.php?page=60&kategorie=2826. 15

[53] Landing Products. APC 13x8E.

URL http://www.apcprop.com/v/downloads/PERFILES_WEB/PER3_13x8E.dat. 15

[54] LORD Corporation. Microstrain 3DM GX3-25.

URL http://www.microstrain.com/inertial/3DM-GX3-25. 15

[55] u-blox AG. Antaris LEA-4T.

URL http://tinyurl.com/Ublox-LEA-4T. 15

[56] Murata Manufacturing Co. Ltd. SCP1000.

URL http://tinyurl.com/vtitechnologyscp1000. 15

[57] Freescale Semiconductor Inc. MPXV7002DP.

URL http://tinyurl.com/freescale-mpxv7002dp. 15

[58] *U.S. Standard Atmosphere*. U.S. Government Printing Office, Washington, D.C., 1976. 16

[59] Futaba. S617FS.

URL http://www.futaba-rc.com/receivers/air.html. 17

[60] Digi International Inc. XBee-PRO 900.

URL http://tinyurl.com/Digi-Xbee-Pro-900XSC. 17

[61] Texas Instruments. OMAP 3530.

URL http://www.ti.com/product/omap3530. 19

[62] Capt. USAF J. E. Zeis. Angle of attack and sideslip estimation using inertial reference platform. Master's thesis, Air Force Institute of Technology, 1988. 21

[63] E. Morelli. Real-time aerodynamic parameter estimation without air flow angle measurements. *Journal of Aircraft*, 49(4):1064–1074, 2012. 21

[64] W. Gracey. Summary of methods of measuring angle of attack. Technical Note 4351, National Advisory Committee for Aeronautics, 1958. 21

[65] D. Telionis, Y. Yang, and O. Rediniotis. Recent development in multi-hole probe (mhp) technology. In *20th International Congress of Mechanical Engineering*, November 2009. 21

[66] J. W. Naughton, L. N. Cattafesta III, and G. S. Settles. A miniature, fast-response 5-hole probe for supersonic flowfield measurements. In *(AIAA) 30th Aerospace Sciences Meeting & Exhibit*, January 1992. 23

[67] A. R. Paul, R. R. Upadhyay, and A. Jain. A novel calibration algorithm for five-hole pressure probe. *International Journal of Engineering, Science and Technology*, 3:88–95, 2011. 23

[68] B. A. Walther and J. L. Moore. The concept of bias, precision and accuracy, and their use in testing the performance of species richness estimators, with a literature review of estimator performance. *Ecography*, pages 815–829, 2005. 31

[69] J. J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1989. 35

[70] D. Cabecinhas, C. Silvestre, P. Rosa, and R. Cunha. Path-following control for coordinated turn aircraft maneuvers. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, August 2007. 35

[71] M. R. Jardin and E. R. Mueller. Optimized measurements of unmanned-air-vehicle mass moment of inertia with a bifilar pendulum. *Journal of Aircraft*, 46:63–75, 2009. 38

[72] T. R. Kane and Gan tai Tseng. Dynamics of the bifilar pendulum. *Interantional Journal of Mechanical Sciences*, 9:83–96, 1967. 38

[73] J. E. Williams and S. R. Vukelich. *The USAF Stability and Control Digital DATCOM Volume 1 Users Manual*. DTIC-MIL, 1979. 49

[74] Mark Drela. AVL. URL http://web.mit.edu/drela/Public/web/avl/. 49

[75] Tomas Melin. Tornado, URL http://www.redhammer.se/tornado/index.html. 49

[76] J.B. Barlow, W.H. Rae, and A. Pope. *Low-speed wind tunnel testing*. Aerospace engineering/mechanical engineering. Wiley, 1999. 50

[77] V. Klein and E. A. Morelli. *Aircraft System Identification: Theory and Practice*. American Institute of Aeronautics and Astronautics, 2006. 50, 52

[78] M. B. Tischler and R. K. Remple. *Aircraft and Rotorcraft System Identification: Engineering Methods with Flight Test Examples*. American Institute of Aeronautics and Astronautics, 2006. 50

[79] R. A. Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 222:pp. 309–368, 1922. 50

[80] J.R. Raol and J. Singh. *Flight Mechanics Modeling snd Analysis*. CRC Press, 2009. 51

[81] J. A. Mulder, Q. P. Chu, J. K. Sridhar, J. H. Breeman, and M. Laban. Non-linear aircraft flight path reconstruction review and new advances. *Progress in Aerospace Sciences*, 35:673–726, 1999. 52

[82] US Air Force. Flying qualities of piloted airplanes (MILSPEC-F8785 C). http://milspec.tpub.com/MIL-F/MIL-F-8785C/, 1980. 58, 59

[83] J. D. McMinn. Extension of a kolmogorov atmospheric turbulence model for time-based simulation implementation. Technical report, 1997. 58, 59, 84

[84] N. Bedrossian, S. Bhatt, M. Lammers, L. Nguyen, and Y. Zhang. Zero-prop maneuver space station demonstration. In *2007 AIAA Guidance, Navigation & Control Conference*, 2007. 63

[85] A. Packard. Gain scheduling via linear fractional transformations. *Systems and Control Letters*, 22:79–92, 1994. 67

[86] M. Farhood. LPV control of nonstationary systems: a parameter-dependent Lyapunov approach. *IEEE Transactions on Automatic Control*, 57(1):209–215, January 2012. 67

[87] J. Lofberg. YALMIP: A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004. Available from http://control.ee.ethz.ch/~joloef/wiki/pmwiki.php. 80

[88] K. C. Toh, M. J. Todd, and R. H. Tutuncu. SDPT3 — a MATLAB software package for semidefinite programming. *Optimization Methods and Software*, 11:545–581, 1999. 80