Two-Step System Identification and Primitive-Based Motion Planning for Control of Small Unmanned Aerial Vehicles

David J. Grymin

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 A. Woolsey
Mayuresh J. Patil
Cornel Sultan

November 1, 2013
Blacksburg, Virginia

Keywords: Aircraft, Motion Primitives, Parameter Estimation, Path Planning, System Identification, Trajectory Generation
Copyright 2013, David J. Grymin

Two-Step System Identification and Primitive-Based Motion Planning for Control of Small Unmanned Aerial Vehicles

David J. Grymin

Abstract

This dissertation addresses motion planning, modeling, and feedback control for autonomous vehicle systems. A hierarchical approach for motion planning and control of nonlinear systems operating in obstacle environments is presented. To reduce computation time during the motion planning process, dynamically feasible trajectories are generated in real-time through concatenation of pre-specified motion primitives. The motion planning task is posed as a search over a directed graph, and the applicability of informed graph search techniques is investigated. Specifically, a locally greedy algorithm with effective backtracking ability is developed and compared to weighted A* search. The greedy algorithm shows an advantage with respect to solution cost and computation time when larger motion primitive libraries that do not operate on a regular state lattice are utilized. Linearization of the nonlinear system equations about the motion primitive library results in a hybrid linear time-varying model, and an optimal control algorithm using the ℓ_2-induced norm as the performance measure is applied to ensure that the system tracks the desired trajectory. The ability of the resulting controller to closely track the trajectory obtained from the motion planner, despite various disturbances and uncertainties, is demonstrated through simulation.

Additionally, an approach for obtaining dynamically feasible reference trajectories and feedback controllers for a small unmanned aerial vehicle (UAV) based on an aerodynamic model derived from flight tests is presented. The modeling approach utilizes the two step method (TSM) with stepwise multiple regression to determine relevant explanatory terms for the aerodynamic models. Dynamically feasible trajectories are then obtained through the solution of an optimal control problem using pseudospectral optimal control software. Discrete-time feedback controllers are then obtained to regulate the vehicle along the desired reference trajectory. Simulations in a realistic operational environment as well as flight testing with the feedback controller demonstrate the capabilities of the approach.

The TSM is also applied for system identification of an aircraft using motion capture data. In this application, time domain system identification techniques are used to identify both linear and nonlinear aerodynamic models of large-amplitude pitching motions driven by control surface deflections. The resulting models are assessed based on both their predictive capabilities as well as simulation results.

Acknowledgments

First and foremost, I would like to thank all of the faculty and staff members of the Aerospace & Ocean Engineering department at Virginia Polytechnic Institute and State University. In particular, I would like to thank my advisor, Dr. Mazen Farhood, for his guidance over the past several years. This work would not have been possible without his patience, attention to detail, and advice. I would also like to thank my committee members Dr. Craig Woolsey, Dr. Mayuresh Patil, and Dr. Cornel Sultan for their assistance and support.

In addition, I would like to thank all of the members of the Nonlinear Systems Laboratory that I had the opportunity to work with. In particular, I am grateful for the insight and assistance from Dr. Ony Arifianto. Thanks also go out to Ben Neas, Chris Rogers, Kyle Guthrie, and Himanshu Shukla for their conversations and suggestions during the often long hours in our office. I would also like to thank Matt Giarra and Artur Wolek for their support and insightful discussions on a variety of topics, whether it be optimal control, motorcycles, or even birdhouses.

Additional thanks are due to my colleagues and friends at the Air Force Research Laboratory, including, but by no means limited to Dr. David Doman, Dr. Michael Oppenheimer, Dr. Michael Bolender, Dr. Gregory Reich, Dr. Darryl Robertson, Dr. Sean Regisford, Captain Josiah VanderMey, and Isaac Weintraub. I would also like to thank all the friends I made during my time on internships, specifically Ben Grieg, Tim Pruyn, and Marcin Morys.

Finally, I would like to thank my family members and friends for their support and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motion Planning

Motion planning for unmanned vehicles involves developing feasible trajectories through an obstacle field from a given initial state to a desired goal state; see, for instance, [1, 2]. By using a discretized set of feasible motion primitives, the problem of finding a trajectory from the start to the goal becomes a graph search, a topic that has received a wealth of attention in the literature. The work presented herein takes the approach of utilizing a set of pre-specified motion primitives, i.e. state and control histories defined over finite (or semi-infinite) time intervals, to generate, in real-time, collision-free trajectories from start to goal via graph search methods. As for the execution of the motion plan, the series of motion primitives generated by the planner will correspond to a sequence of pre-designed subcontrollers to be

applied consecutively.

The notion of constructing a solution from available trajectories is a common approach for vehicle motion planning. Prior methods have used online optimization to determine the trajectory [3], concatenating trim and maneuver trajectories to form a dynamically feasible path from the start state to the goal. In certain scenarios, the solution of such an optimization problem may require more computational effort than can be allotted to the planning task. Deterministic and sampling-based searches over graphs are two broad categories that have received considerable attention related to robot and vehicle trajectory planning in obstacle environments; a comprehensive review of motion planning with respect to unmanned aerial vehicles is given in [4].

Deterministic graph search algorithms use knowledge obtained during the search as well as prior knowledge of the environment to work towards an optimal solution. A heuristic, or rule of thumb, assists in determining the order of expansion during the search. For vehicle motion planning problems, the cost-to-goal is a commonly chosen heuristic. The A* algorithm, a complete and optimal algorithm, uses the path cost to reach each node as well as the future path cost estimate from the heuristic, and traverses the graph by expanding nodes with the lowest total path cost. For vehicle motion planning, the length of the path to reach a node can be used for path cost. In some applications, finding the optimal path may become burdensome, and a suboptimal solution is accepted to reduce the computational load. Weighted A* (WA*) relaxes optimality by weighting the heuristic in relation to the cost-

to-go, effectively increasing the greediness of the algorithm, and is able to return solutions much faster with a bound on the suboptimal path cost [5, 1]. Recent work related to A* based search methods has focused on iteratively improving suboptimal trajectories towards the minimum-cost path; see, for instance, the anytime search heuristic developed in [6, 7]. Anytime search attempts to quickly return a feasible yet suboptimal path, and then improve upon this path successively in the time allotted for planning. In [7], for example, successive WA* searches are run with decreasing weight to achieve the best possible path in the given time for computation.

In sampling-based planners, such as the probabilistic roadmap (PRM) and rapidly-exploring random trees (RRTs), completeness is probabilistic; a solution will be returned, should one exist, with a probability converging to one as the number of samples tends towards infinity [8, 9]. In practice, the RRT algorithm in particular is capable of returning a path to the goal fairly quickly, even in high-dimensional search spaces and subject to differential vehicle constraints. RRTs quickly examine unexplored regions of the state space, and are able to find paths through complicated obstacle fields with relative ease. The trade-off, however, is in the solution quality, as the path is often erratic due to the random sampling which drives the expansion. Additionally, Karaman and Frazzoli showed that the probability of the RRT algorithm converging to the optimal solution was zero. Their development of the RRT* algorithm, however, provides conditions for asymptotic optimality in addition to probabilistic completeness [10]. This algorithm has since been extended to an anytime

framework, in which an initial solution is obtained quickly and then improved upon in the remaining time allotted [11].

The representation of the input and search spaces is also a factor in selecting the method to use. In [12], a discretized set of control inputs was used to compute a path for nonholonomic vehicles, with numerical integration performed during the planning process. Graph search was then utilized over a partitioning of the configuration space to determine a sequence of control inputs that brought the vehicle from its initial position to a goal region. A similar approach was taken by [13], with integration of control actions performed offline and stored for use with an online planner; solutions were obtained by performing a search over a tree. Pre-computed vehicle motions can also be developed that result in a grid-based representation of the configuration space, referred to as a state lattice. In this framework, the state lattice is represented as a directed graph, with vertices corresponding to specific reachable states of the vehicle and edges indicating the dynamically feasible motions which connect the states exactly. Such a representation is resolution complete, i.e. it is complete with respect to the resolution at which the lattice is generated [14, 15].

It is important to note that when using pre-computed control input and state histories, the ability of the vehicle to track the resulting motion plan is subject to model accuracy. Unmodeled dynamics, parametric uncertainty, and exogenous disturbances may result in deviations from the original motion plan during execution. In the work of Burridge et al., a sequence of pre-computed feedback controllers is used to bring the system to a desired

goal state in the presence of disturbances and obstacles in the robot workspace [16]. This framework has also been used for motion planning using controllers valid over regions of the free space; the vehicle is guided to the goal region by the sequence of controllers, with no path explicitly determined [17, 18].

The primary motivation for applying graph-based motion planning techniques to unmanned aerial vehicles is to reduce the computational demand required when operating in complex environments. Rather than attempting to solve for a dynamically feasible state and control history online, the motion planner instead has a library of dynamically feasible segments that can be concactenated in order to traverse the environment. As discussed in [19] and [20], a hybrid control approach provides feedback controllers in order to regulate the vehicle about each motion primitive in spite of exogenous disturbances, measurement noise, and system uncertainties. Thus a solution to the motion planning problem also immediately leads to a feedback control policy, given by a scheduled sequence of subcontrollers to be applied, that will safely bring the vehicle to the goal region.

1.2 Aircraft System Identification

The use of low-cost off the shelf radio-controlled (RC) fixed-wing aircraft for unmanned aerial vehicle (UAV) research applications has become increasingly prevalent in university settings [21], as these aircraft are relatively inexpensive to both acquire and operate. Due

to size and payload mass restrictions, these aircraft typically are not equipped with the sophisticated measurement, instrumentation, and flight computer systems onboard full-size aircraft. These systems pose challenges during both the system identification and feedback control process. In this paper, a time-domain system identification approach is applied to a small RC aircraft and subsequently utilized for deriving a dynamically feasible reference trajectory and feedback controller offline.

There are several well-regarded texts discussing system identification for aircraft applications. The text [22] focuses on time domain applications, while [23] presents frequency domain approaches; an overview and theoretical foundations for both of the aforementioned approaches is provided in [24]. With regard to time-domain system identification, maximum likelihood approaches, in particular the output-error method, have by and large seen the most widespread application for aircraft system identification. The primary disadvantage of such approaches, however, is that an appropriate model structure is assumed a priori. In constrast, the two-step method [25], which parallels the equation-error approaches discussed in [22] and [24], provides a framework for rapid investigation of various model structures.

Crucial to the application of equation-error methods for system identification are relatively noise-free estimates of the aircraft system states as well as any forces and moments acting on the vehicle during conducted flight tests. Flight path reconstruction (FPR) [26] is utilized to provide these required measurements and estimates. Depending on measurement information available, forces and moments may be able to be directly computed from measured values

[22, 24]. If this is not possible, due to measurement sampling rate or sensor noise [27], the estimation-before-modeling (EBM) approach provides a methodology to estimate time-varying histories of both the aircraft states and the aerodynamic forces and moments [28, 29]. A comparison between the output-error and two-step approaches is provided in [30], and shows that the resulting aerodynamic models are comparable bewteen the two approaches.

A survey of system identification applications for small low-cost aircraft can be found in [21]. In [31], a nonlinear mapping identification algorithm is utilized to estimate parameters capturing attitude dynamics using a linear model formulation for the aerodynamic moments acting on the aircraft. Nonlinear constrained optimization is used in [32] to estimate parameters minimizing the difference between measured flight data and model predicted data using linearized dynamic models of the longitudinal and lateral-directional modes. A number of examples using frequency-domain techniques for small UAVs are provided in [23]. Additionally, in [33], frequency-domain system identification is applied to a small low-cost UAV system.

There are a number of approaches that have been utilized for trajectory generation for UAVs, including, but not limited to, kinematic trajectories for path following [34, 35, 36, 37], hybrid systems based on maneuver classes [3, 38, 39], and optimal control methods [40, 41, 42], with the latter being of primary interest for this work. Solution of an optimal control problem to obtain a state and control history is a well-studied topic, see, for instance, [43, 44]. Recently, pseudospectral optimal control methods have been applied to solve such problems for aircraft

applications [40, 41, 42]. In these works, a kinemetic trajectory is first obtained using a point mass representation of the vehicle; the kinematic trajectory is then used as initial guess to determine the necessary control inputs based on the nonlinear aircraft equations of motion.

There are several reasons for using numerical techniques to derive reference trajectories, as opposed to, say, recording pilot-operated control inputs. Generating reference trajectories offline does not require any additional flights; accordingly, there is no 'contamination' of the reference trajectory due to exogenous disturbances, such as atmospheric turbulence and measurement noise. Additionally, specific constraints can be imposed on the reference trajectory, namely desired initial and final conditions, flight envelope restrictions, and magnitude of control deflections. The disadvantage of such an approach is that any assumptions, simplifications, or uncertainties associated with the modeling process may lead to degraded tracking of the reference trajectory for the actual system. A feedback controller using the ℓ_2 induced norm as a performance measure is also provided so that the aircraft tracks the reference trajectory despite the aforementioned model discrepancies in addition to any exogenous disturbances [45, 46].

The two-step approach, specifically using the EBM technique to determine the aircraft state, force, and moment time histories, is applied to a small low-cost UAV in this work. An appropriate model structure and associated parameters are estimated using stepwise multivariable regression [24, 47]. This aerodynamic model is then used to obtain a dynamically feasible reference trajectory as the solution of an optimal control problem. To ensure that the aircraft

tracks the reference trajectory, a feedback controller is provided. All system identification, trajectory, and controller generation is performed offline.

1.3 System Identification of Aircraft Using Motion Capture Data

The use of motion capture data for aerodynamic system identification is a recent development. For small UAVs, such systems allow flight testing to be performed in a tightly controlled environment, i.e. one absent of exogenous disturbances such as atmospheric turbulence. Using imaging systems and markers defining a known object, the inertial position and attitude of an aircraft at each sampling instant are recovered. The difficulties in such an approach, however, arise due to the size of the vehicle and test facility. The size of vehicle may preclude the use of additional instrumentation often utilized for aircraft system identification such as accelerometers, rate gyros, and airdata probes. With regard to the test facility, motion capture facilities are indoor flight environments and therefore the physical dimensions of the room limit both the type and duration of test flights performed.

Nonetheless, such systems are promising due to the ability to perform free flight tests in the absence of disturbances. The challenge is to then determine the time histories of the aircraft states as well as the forces and moments acting on the vehicle from position and attitude measurements only. Preliminary results using motion capture data for force and moment

estimation appeared around the same time in both [48] and [49]. In [48], the focus concerned estimation of conventional stability and control derivatives. The work of [49] compared flight estimated forces and moments to predictions developed considering flat plate wing surfaces for high angle of attack pitch maneuvers. The motivation of this work was to develop a feedback control policy to successfully perform a perching maneuver using a small foam aircraft with a horizontal tail as the only control surface.

The work in [48] was expanded upon in [50] to include a more formal analysis of results with respect to filtering and data processing assumptions. In [51], a more sophisticated model was developed for the same aircraft used in [49]. The improved model selected basis functions from a candidate pool of nonlinear terms to map system states to forces and moments; parameter estimates were obtained as the solution of a least squares problem and then improved upon using nonlinear optimization. In essence, this approach combined the equation-error and output-error methods, as the nonlinear optimization focused on minimization of the difference between measurements and simulation of the dynamic equations for the inertial positions and orientations. Modeling efforts using motion capture data can also be found in [52] and [53]. In the former, stability and control derivatives are estimated from flight estimated forces and moments; the latter work includes incorporation of unsteady aerodynamic effects on the motion of the vehicle. The work of [54] uses the same aircraft and test facility to the work presented in this dissertation, however a different set of test data was utilized. Specifically, flight tests in [54] use commanded step inputs with varying

magnitude. A model based on relationships between elevon effectiveness, wing efficiency, and other various aircraft parameters was then fit to the test data.

Common to the aforementioned approaches is the technique used to obtain estimates of the forces and moments acting on the vehicle. In all prior works, measurements of inertial positions and orientations are processed independently. With the exception of [54], these works utilize Savitzy-Golay filters [55] in order to obtain smooth time histories of the measurements and their respective derivatives. In [54], smoothed estimates of the measurements are obtained by fitting a spline function to each raw measurement; the spline functions are then differentiated to obtain rate information for computing body axis velocities, angular rates, et cetera. In this dissertation, an extended Kalman filter is used to obtain the time histories of the states, forces, and moments concurrently. Additionally, the aerodynamic model structure and parameter values are obtained by using time domain aircraft system identification techniques, specifically the two-step method.

1.4 Overview and Specific Contributions

The overview of the dissertation is as follows. In Chapter 2, a review of background and preliminary information is presented. This chapter introduces the application of graph-based search algorithms encompassing a variety of robotic and unmanned vehicle systems. Details regarding prior work utilizing motion primitives for path planning and various search

algorithms are discussed. The rigid body aircraft equations of motion are derived based on Newton's second law. An overview of aircraft system identification techniques, specifically focusing on time-domain approaches, is presented. The extended Kalman filter and its application in the system identification process for aircraft are also given. The feedback control approaches utilized for motion planning and control about prespecified trajectories are also discussed.

In Chapter 3, the specific graph-based motion planning approach is presented from [20]. Two graph search techniques are presented: one pre-existing technique as well as a developed algorithm based on greedy search behavior. Two types of motion primitive libraries are considered, and the implications these libraries have on the search algorithms are discussed. The search algorithms are then applied to solve motion planning problems for a four thruster hovercraft. The presented algorithm proves advantageous to a common search algorithm used for vehicle motion planning in certain scenarios.

Aerodynamic system identification is applied to a small unmanned aircraft in Chapter 4. Prior aerodynamic system identification for small aircraft has largely focused on frequency domain or linear system approximations about a reference condition. In the work presented here, the two step method is applied to determine both the model structure and parameters for a small UAV, with forces and moments estimated directly from the nonlinear equations of motion. The use of extended Kalman filters to determine the force and moment time histories as augmented states is demonstrated; this approach has not seen widespread use in aircraft

system identification, and little if any for small UAVs operating in outdoor environments. A model is identified and validated against the obtained test data. The model is then used to derive dynamically feasible trajectories as the solution to an optimal control problem. A discrete-time feedback controller is also developed and tested on the actual aircraft system.

In Chapter 5, time-domain system identification is applied to test data obtained from an unpowered glider in a motion capture facility. The use of extended Kalman filters to estimate state histories of the aerodynamic forces and moments is compared to a common approach using numerical differentiation of position and orientation measurements. The estimated forces and moments are then utilized to estimate three aerodynamic models based on quasi-steady formulations. It is shown that although the vehicle undergoes rapid and large amplitude pitching motion during flight, primarily driven by control surface deflections, a quasi-steady model with polynomial terms can capture the variation in aerodynamic forces and moments.

Chapter 2

Background

This chapter provides background information pertaining to the remainder of the dissertation. In Section 2.1, a review of graph search and its application to motion planning for autonomous vehicle systems is given. Background on the dynamic and kinematic equations of motion for rigid-body aircraft is given in Section 2.2. Section 2.3 details system identification techniques applied to full-size as well as small unmanned aircraft. Finally, in Section 2.4, details regarding the feedback control approaches utilized are reviewed.

2.1 Motion Planning

The motion planning approach considered in this work uses graph search techniques to determine a dynamically feasible collision-free path from a start location to a pre-specified

goal location. The texts by [1, 5] are standard references for understanding graph search. In the context of graph search presented here, the nodes of the graph represent *configuration* space locations of the vehicle, while edges correspond to the motion primitive (or state and control input history) applied to 'drive' the vehicle between two locations within the configuration space. With this stated, the edges of the graph are directed, i.e. it is not permissable for a vehicle to travel backwards in time along a primitive.

In order to determine a collision-free path from the start location to the goal region, it is necessary to determine if the position of the vehicle along a motion primitive will lead to contact with an obstacle. Detection of a collision between two objects is a field of research in and of itself; surveys of collision detection are availabe in [56] and [57], among others. For the graph-based search algorithms in this work, all vehicles are treated as a point mass for collision detection purposes. Additionally, all obstacles considered are convex and therefore the algorithm in [58, 59] is applied for determining whether the location of the vehicle is within an obstacle.

Depending on the location of the point mass used for collision detection with respect to the vehicle geometry, a path that is collision-free for the point mass may result in a collision for the actual vehicle. For this reason, bounding boxes are placed around each obstacle, and the point mass entering the bounding box is treated as a collision. The difference between the length and width of the bounding box and original obstacle must therefore be greater than the largest of the distances from the point mass location to all points defining the obstacle.

2.2 Aircraft Rigid-Body Equations of Motion

In this section, a brief overiew of the equations of motion for rigid-body aircraft is given. In particular, focus is given to the body-axis reference frame equations of motion using Euler angles to define the orientation of the aircraft. The notation is common for describing the dynamics of rigid-body aircraft. Readers are directed to [60, 61, 62], among others, for further details.

Denoting $\mathbf{F}$ as the net force vector, $\mathbf{W}$ as the weight vector, $\mathbf{V}$ the translational velocity vector, $\boldsymbol{\omega}$ the rotational velocity vector, $\mathbf{M}$ the net moment vector about the body-axis fixed center of gravity, m as the mass, and $\mathbf{I}$ the inertia tensor, Newton's second law for the rigid-body dynamics is given as

$$\begin{aligned} \mathbf{F} + \mathbf{W} &= \frac{d}{dt}(m\mathbf{V}) + \boldsymbol{\omega} \times (m\mathbf{V}) \\ \mathbf{M} &= \frac{d}{dt}(\mathbf{I}\boldsymbol{\omega}) + \boldsymbol{\omega} \times (\mathbf{I}\boldsymbol{\omega}) \end{aligned} \tag{2.1}$$

with the inertia tensor defined as

$$\mathbf{I} = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}$$

Defining the origin to be the center of gravity of the aircraft, the x_b axis pointing outward from the center of gravity through the nose of the aircraft, y_b outward along the right wing,

and z_b downward, the product of inertia terms can be simplified as

$$I_{xy} = I_{yx} = I_{yz} = I_{zy} = 0. \tag{2.2}$$

The conventional notation for the body-axis velocity and angular velocity components is $\mathbf{V} = [u,\ v,\ w]^T$ and $\boldsymbol{\omega} = [p,\ q,\ r]^T$, respectively. The net force and moment vectors are expressed in terms of their respective components as $\mathbf{F} = [F_x,\ F_y,\ F_z]^T$ and $\mathbf{M} = [L,\ M,\ N]^T$.

The position and orientation of the aircraft are more conveniently expressed in an Earth-fixed reference frame. Let (X_g, Y_g, z_g) denote a Cartesian coordinate system corresponding to the North, East, and down directions, respectively. The orientation of the aircraft is specified in terms of Euler angles, denoted as ϕ, θ, and ψ, referred to as the bank angle, elevation angle, and heading angle, respectively. The orientation of the aircraft relative to the inertial reference frame is obtained by applying three sequential rotations through the Euler angles as follows:

1. (X_g, Y_g, z_g) is rotated through ψ about the z_g axis to obtain (x_1, y_1, z_1)
2. (x_1, y_1, z_1) is rotated through θ about the y_1 axis to obtain (x_2, y_2, z_2)
3. (x_2, y_2, z_2) is rotated through ϕ about the x_2 axis to obtain (x_b, y_b, z_b)

These rotations can be expressed by the three rotation matrices below.

$$\mathbf{R}_\psi = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{R}_\theta = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \quad \mathbf{R}_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \tag{2.3}$$

Thus a vector in the inertial reference frame is expressed in the body-axis reference frame as $(x_b, y_b, z_b) = \mathbf{R}_\phi \cdot \mathbf{R}_\theta \cdot \mathbf{R}_\psi \cdot (X_g, Y_g, z_g)$.

In the inertial reference frame, the force due to gravity is $(0, 0, W)$, which in the body-axis reference frame is

$$\mathbf{W} = W \begin{pmatrix} \sin\theta \\ \sin\phi\cos\theta \\ \cos\phi\cos\theta \end{pmatrix} \tag{2.4}$$

With the preceding notation for the body-axis forces, moments, velocities, and angular rates, as well as the relationship given in Eq. 2.2, Eq. 2.1 is expressed as

$$\begin{aligned} \begin{Bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{Bmatrix} &= \frac{g}{W} \begin{Bmatrix} F_x \\ F_y \\ F_z \end{Bmatrix} + g \begin{Bmatrix} \sin\theta \\ \cos\theta\sin\phi \\ \cos\theta\cos\phi \end{Bmatrix} + \begin{Bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{Bmatrix} \\ \begin{Bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{Bmatrix} &= \begin{bmatrix} I_x & 0 & -I_{xz} \\ 0 & I_y & 0 \\ -I_{xz} & 0 & I_z \end{bmatrix}^{-1} \begin{Bmatrix} L + (I_y - I_z)qr + I_{xz}pq \\ M + (I_z - I_x)pr + I_{xz}(p^2 - r^2) \\ N + (I_x - I_y)pq - I_{xz}qr \end{Bmatrix}. \end{aligned} \tag{2.5}$$

The velocity of the aircraft in the inertial reference frame is obtained through the following

transformation:

$$\begin{Bmatrix} \dot{X}_g \\ \dot{Y}_g \\ \dot{z}_g \end{Bmatrix} = \mathbf{R}_\psi \mathbf{R}_\theta \mathbf{R}_\phi \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} \tag{2.6}$$

The body-axis angular rates are expressed in terms of the Euler angle rates by the following transformation:

$$\begin{Bmatrix} p \\ q \\ r \end{Bmatrix} = \mathbf{R}_\phi \begin{Bmatrix} \dot{\phi} \\ 0 \\ 0 \end{Bmatrix} + \mathbf{R}_\phi \mathbf{R}_\theta \begin{Bmatrix} 0 \\ \dot{\theta} \\ 0 \end{Bmatrix} + \mathbf{R}_\phi \mathbf{R}_\theta \mathbf{R}_\psi \begin{Bmatrix} 0 \\ 0 \\ \dot{\psi} \end{Bmatrix} \tag{2.7}$$

Combining Eq. 2.7 and taking the inverse gives the Euler angle rates in terms of the body-fixed axis rotation rates, as given below.

$$\begin{Bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{Bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{Bmatrix} p \\ q \\ r \end{Bmatrix} \tag{2.8}$$

Note that Eq. 2.5 is in terms of body-axis reference frame forces and moments. The total velocity of the aircraft with respect to the surrounding air, and its orientation with respect to the aircraft, is illustrated in Fig. 2.1. The body-axes in Fig. 2.1 are labeled as (x_b, y_b, z_b) and the stability axes are denoted by (x_s, y_s, z_s). The speed of the surrounding air is denoted by V_a, in the $x_s - y_s$ plane. The angle of sideslip, denoted as β, is the angle between the x_s

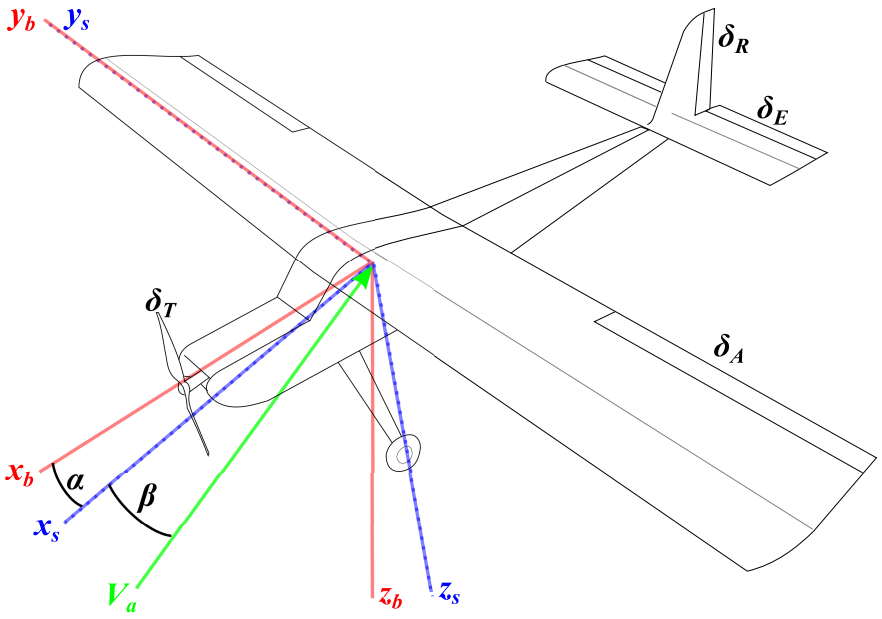


Figure 2.1: Body axes, stability axes, and relative air velocity.

stability axis and the airspeed, V_a. Angle of attack, denoted as α, is the angle between the x_b and x_s axes. The airspeed, angle of attack, and angle of sideslip are defined as

$$V_a = \sqrt{(u-u_d)^2 + (v-v_d)^2 + (w-w_d)^2} \tag{2.9}$$

$$\alpha = \tan^{-1}\frac{w-w_d}{u-u_d} \tag{2.10}$$

$$\beta = \sin^{-1}\frac{v-v_d}{V_a} \tag{2.11}$$

where (u_d, v_d, w_d) are atmospheric disturbances in the body-axis reference frame, and the disturbance vector is defined as $\mathbf{d}_a = [u_d,\ v_d,\ w_d]^T$. These disturbances are due to constant wind, gusts, and atmospheric turbulence.

The aircraft state vector is defined as $\mathbf{x} = [u,\ v,\ w,\ p,\ q,\ r,\ \phi,\ \theta,\ \psi,\ z_g,\ X_g,\ Y_g]^T$. Also

given in Fig. 2.1 are the notations for the aircraft's control surfaces. The aileron, elevator, rudder, and throttle are denoted as δ_A, δ_E, δ_R, and δ_T, respectively. The control input vector is defined as $\boldsymbol{\delta} = [\delta_A,\ \delta_E,\ \delta_R,\ \delta_T]^T$. The aerodynamic force and moment vectors can then, in general, be expressed as $\mathbf{F} = \mathbf{F_a}(\mathbf{V}, \boldsymbol{\omega}, \boldsymbol{\delta}, \mathbf{d}_a)$ and $\mathbf{M} = \mathbf{M_a}(\mathbf{V}, \boldsymbol{\omega}, \boldsymbol{\delta}, \mathbf{d}_a)$, where $\mathbf{F_a}(\cdot)$ and $\mathbf{M_a}(\cdot)$ relate the aircraft states, control inputs, and surrounding air to the forces and moments acting on the vehicle. Note that the aerodynamic force and moment relationships are commonly expressed in a non-dimensional form and are related to the force and moment vectors as

$$\mathbf{F_a} = \bar{q} S_{\text{ref}} \begin{bmatrix} C_X(\mathbf{V}, \boldsymbol{\omega}, \boldsymbol{\delta}, \mathbf{d}_a) \\ C_Y(\mathbf{V}, \boldsymbol{\omega}, \boldsymbol{\delta}, \mathbf{d}_a) \\ C_Z(\mathbf{V}, \boldsymbol{\omega}, \boldsymbol{\delta}, \mathbf{d}_a) \end{bmatrix}, \quad \mathbf{M_a} = \bar{q} S_{\text{ref}} \begin{bmatrix} b\ C_l(\mathbf{V}, \boldsymbol{\omega}, \boldsymbol{\delta}, \mathbf{d}_a) \\ \bar{c}\ C_m(\mathbf{V}, \boldsymbol{\omega}, \boldsymbol{\delta}, \mathbf{d}_a) \\ b\ C_n(\mathbf{V}, \boldsymbol{\omega}, \boldsymbol{\delta}, \mathbf{d}_a) \end{bmatrix} \tag{2.12}$$

where $C_i(\cdot)$ denote aerodynamic coefficients, b is the wing span, $\bar{c}$ the chord length, $S_{\text{ref}} = b\bar{c}$ the reference area, and $\bar{q} = 1/2\rho V_a^2$ is the dynamic pressure, with ρ denoting the density of the surrounding air.

With the preceding definitions for the state, input, and disturbance vectors, the equations of motion given in Eq. 2.5, 2.6, and 2.8 can be equivalently written as $\dot{\mathbf{x}} = \mathbf{f}\left(\mathbf{x}, \boldsymbol{\delta}, \mathbf{d}_a\right)$, where $\mathbf{f}(\cdot, \cdot, \cdot)$ is defined in the obvious way.

2.3 Aircraft System Identification

In depth treatments of system identification for aircraft applications are available in [63, 24, 22, 23, 27]. Maximum likelihood methods, including the output-error method (OEM) and filter-error method (FEM), have been applied to numerous aircraft for parameter estimation purposes. A disadvantage of these approaches, however, is that a model structure must be provided a priori. In contrast, the Two-Step Method (TSM), analagous to the EBM approach, makes no a priori assumptions regarding aerodynamic model structure [25, 30, 29]. The model structure determination and parameter estimation portion of this approach is also referred to as an equation-error method in [24, 22]. The primary advantage of the equation-error method is that a variety of model structures can be rapidly examined and assessed when processing test data offline. Further, data from other sources, e.g. wind tunnel tests or computational fluid dynamics (CFD) simulations, can be readily incorporated, if available.

A review of system identification applications for various types of small fixed-wing UAVs is given in [64]. In [31], a nonlinear mapping identification algorithm is utilized to estimate parameters capturing attitude dynamics using a linear model formulation for the aerodynamic moments acting on the aircraft. Nonlinear constrained optimization is used in [32] to estimate parameters minimizing the difference between measured flight data and model predicted data using linearized dynamic models of the longitudinal and lateral-directional modes. Similar efforts to the modeling approach presented here can be found in [65, 66, 67]. The work in both [65] and [66] focuses on full-size manned aircraft; in contrast, [67] utilizes

a subscale model of generic transport aircraft with a sophisticated instrumentation system.

2.3.1 Two Step Approach

The first step of the TSM consists of determining the forces and moments acting on the aircraft. As previously mentioned, the second step of the TSM is an equation-error approach, following the terminology of [24]. In contrast to the OEM and FEM approaches, an aerodynamic model structure is not postulated prior to initial flight test data processing. In [30], the OEM and TSM are applied for comparison purposes to flight test data for a Cessna Citation II. The same model structure was assumed for both the OEM and TSM. While the estimates of the model parameters were similar, the model obtained using the OEM was preferable when considering the time histories of longitudinal variables obtained from simulation of the aircraft using each model.

The model structure identification and parameter estimation are performed independent of the state estimation process in the TSM. This can prove to be extremely beneficial when posing the model structure in a manner that remains linear in its parameters, as these parameters can be estimated using standard linear regression techniques. Note that although models may be linear in the parameters, a variety of model structures have been previously used to capture nonlinear or large envelope aerodynamic behavior [65, 68, 66, 67].

Accelerometer and rate gyro measurements are commonly used to directly compute aerodynamic force and moment histories [22, 24, 26], however sensor limitations may prohibit

such a direct computation for certain applications. In [28, 69, 29], forces and moments are estimated as augmented states using an extended Kalman filter. While this approach is more computationally demanding, processing data offline also allows for fixed interval smoothing.

2.3.2 Extended Kalman Filter

An important step in the system identification process is FPR, also referred to as the data compatibility check [22, 24, 26]. For the OEM and FEM methods, FPR serves the purpose of correcting measurements obtained from the various sensors onboard the aircraft due to measurement noise, bias, and scale factor errors, in addition to correcting for measurement data at locations away from the center of gravity of the aircraft. In the two-step method, FPR also serves the purpose of estimating the aircraft state history, forces, and moments. The most common approach is to utilize an extended Kalman filter for FPR; a thorough treatment of the FPR process is given in [26]. Note that for purposes of measurement bias and scale factor estimation, the FPR process can also be accomplished using the OEM [22].

For the FPR process presented here, it is assumed that the atmospheric disturbances are negligible, i.e. $u_d = v_d = w_d = 0$. Let n_p denote the number of parameters to be estimated, such as bias or scale factor measurement errors. An additional n_p states are appended to the state vector, with derivative equal to zero in the dynamic equations, i.e.

$$\dot{\mathbf{x}}_{\text{aug}} = \begin{bmatrix} \mathbf{f}(\mathbf{x}, \boldsymbol{\delta}) \\ \mathbf{0}_{n_p \times 1} \end{bmatrix},$$

where $\mathbf{x}_{\text{aug}} = [\mathbf{x}^T,\ \mathbf{x}_p{}^T]^T$ is the augmented state vector and $\mathbf{x}_p \in \mathbb{R}^{n_p \times 1}$ consists of the parameters to be estimated.

The notation used for the EKF is from [70]; for all applications within this work, measurement data is sampled at a fixed-interval. The estimate of the state, $\mathbf{x}$, at time t_i based on measurement data, $\mathbf{y}$, up to time t_j is denoted $\hat{\mathbf{x}}_{i/j} = E\left[\mathbf{x}(t_i|\mathbf{y}_j, \mathbf{y}_{j-1}, \ldots, \mathbf{y}_1)\right]$. At each time step, denoted by k, the forward propagation equations for the state estimate and covariance matrix are given as

$$\begin{aligned} \hat{\mathbf{x}}_{k/k-1} =& \boldsymbol{\phi}\left(\hat{\mathbf{x}}_{k-1/k-1}, \boldsymbol{\delta}_k, t_k, t_{k-1}\right) \\ \mathbf{P}_{k/k-1} =& \boldsymbol{\Phi}_{k,k-1}\mathbf{P}_{k-1/k-1}\boldsymbol{\Phi}_{k,k-1}^T + \mathbf{Q}_{k,k-1} \end{aligned} \tag{2.13}$$

where $\boldsymbol{\phi}(\hat{\mathbf{x}}_{k-1/k-1}, \boldsymbol{\delta}_k, t_k, t_{k-1})$ is obtained through numerical integration of the equations of motion using a fourth/fifth order Runge Kutta method; $\boldsymbol{\Phi}_{k,k-1}$ is the state transition matrix over the sampling interval from $k-1$ to k, defined as

$$\boldsymbol{\Phi}_{k,k-1} = \Phi\left((k+1)T, kT\right).$$

The matrix $\mathbf{Q}_{k,k-1}$ is the process noise over the sampling interval from $k-1$ to k. After forward propagation, the estimates of the state and covariance matrix are corrected using

the measurements obtained as:

$$\begin{aligned}\hat{\mathbf{x}}_{k/k} =& \hat{\mathbf{x}}_{k/k-1} - \mathbf{K}_k \left(\mathbf{y}_k - \hat{\mathbf{y}}_k\right) \\ \mathbf{P}_{k/k} =& \mathbf{P}_{k/k-1} - \mathbf{K}_k \mathbf{H}_k \mathbf{P}_{k/k-1}\end{aligned} \tag{2.14}$$

where $\mathbf{y}_k$ is measurement data obtained from the flight test, $\hat{\mathbf{y}}_k = \mathbf{h}(\hat{\mathbf{x}}_{k/k-1})$, and

$$\begin{aligned}\mathbf{H}_k =& \frac{\partial \mathbf{h}(\hat{\mathbf{x}}_{k/k-1})}{\partial \mathbf{x}} \\ \mathbf{K}_k =& \mathbf{P}_{k/k-1}\mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_{k/k-1} \mathbf{H}_k^T + \mathbf{R}_k\right)^{-1},\end{aligned} \tag{2.15}$$

with $\mathbf{R}_k$ being the measurement covariance matrix. The EKF is applied for $k = 1, 2, \ldots, N$, where N is the number of discrete time steps in each flight test.

A common approach in the FPR process is to use measured linear and angular accelerations as inputs to Eq. 2.13 [24, 26]. If measurements of the angular accelerations are not available, the angular rate measurements can be smoothed and differentiated; the computed angular accelerations are then used as inputs for the EKF. Another approach, as presented in [28] and with further detail in [29], models the forces and moments acting on the vehicle as third-order Gauss-Markov (GM) processes. Specifically, the forces and moments are of the form

$$\begin{bmatrix} \dot{\mathcal{F}} \\ \dot{\mathcal{F}}_1 \\ \dot{\mathcal{F}}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathcal{F} \\ \mathcal{F}_1 \\ \mathcal{F}_2 \end{bmatrix} + \begin{bmatrix} w_{\mathcal{F}} \\ \omega_{\mathcal{F}_1} \\ \omega_{\mathcal{F}_2} \end{bmatrix} \tag{2.16}$$

for $\mathcal{F} = F_x, F_y, F_z, L, M, N$, where $\omega_{\mathcal{F}}$, $\omega_{\mathcal{F}_1}$, and $\omega_{\mathcal{F}_2}$ are process noise terms. For a six

degree-of-freedom rigid body aircraft, the formulation in Eq. 2.16 introduces an additional 18 states to the system. A disadvantage of the GM approach is that any exogenous force and moment disturbances, i.e. atmospheric turbulence, are included in the filter estimates of the forces and moments acting on the aircraft. Performing system identification tests in relatively calm atmospheric conditions is therefore preferable when using this approach (and for aircraft system identification in general) [24].

Rauch-Tung-Striebel Smoother

Since all state estimation is performed offline in the work presented here, fixed interval recursive smoothers can be applied to the state estimates from the EKF. In [29], a modified Bryson-Frazier (mBF) smoother is applied to obtain smooth estimates of aerodynamic forces and moments modeled as GM processes. An alternative recursion was presented in [71], known as the Rauch-Tung-Striebel (RTS) smoother.

The RTS smoother is applied as a backwards recursion for $k = N-1, N-2, \ldots, 1$, starting from $\hat{\mathbf{x}}_{N/N}$ and $\mathbf{P}_{N/N}$, as:

$$\begin{aligned} \hat{\mathbf{x}}_{k/N} =& \hat{\mathbf{x}}_{k/k} + \mathbf{G}_k \left(\hat{\mathbf{x}}_{k+1/N} - \hat{\mathbf{x}}_{k+1/k} \right) \\ \mathbf{P}_{k/N} =& \mathbf{P}_{k/k} + \mathbf{G}_k \left(\mathbf{P}_{k+1/N} - \mathbf{P}_{k+1/k} \right) \mathbf{G}_k^T \end{aligned} \tag{2.17}$$

where

$$\mathbf{G}_k = \mathbf{P}_{k/k} \mathbf{\Phi}_{k+1,k}^T \mathbf{P}_{k+1/k}^{-1}$$

Further details regarding the theory, derivation, and implementation of the EKF and RTS smoother can be found in [70, 72], among others.

2.3.3 Model Structure Determination

A crucial step in identifying an aerodynamic model from test data is model structure determination. Note that in the OEM and FEM, a model structure is assumed a priori; a poor choice of postulated model can influence the results obtained using these approaches. In contrast, the equation-error approach and TSM are able to rapidly examine a variety of candidate model structures. For the model structures considered in this work, stepwise multivariable regression is appropriate to determine the relevant model terms [47]. Note that for more complex model structures, such as those presented in [66], individual model terms cannot be added or removed.

In stepwise regression, the partial F-statistic is used to determine which regressors to add or remove from the model from a collection of candidate regressors. At each step, the partial F-values of all regressors not currently in the model are computed. The regressor with the highest partial F-value, say ξ_i, is selected, and its F-value is compared to a value, F_{enter}, to add to the model. If ξ_i is deemed statistically significant, i.e. $F_i > F_{\text{enter}}$, it is added to the model. After each addition, the partial F-values of all regressors currently in the model are also computed. The lowest F-value of all regressors currently in the model is compared to an F-value to remove a term, F_{remove}, and the regressor is removed if $F_i < F_{\text{remove}}$. The process

concludes when there are no remaining regressors above the threshold to enter, F_{enter}, and all regressors currently in the model have partial F-values greater than F_{remove}. Further details on stepwise regression can be found in [47].

For each candidate regressor, ξ_i, the F_0 statistic is computed as

$$F_0(i) = \frac{SS_R(\theta_{n+i}) - SS_R(\theta_n)}{s^2} \tag{2.18}$$

where

$$s^2 = \frac{\sum_{k=1}^{N} [y(k) - \hat{y}(k)]^2}{N - i - 1}$$

and

$$SS_R(\theta) = \theta^T \Xi^T y - N\bar{y}.$$

The regressor ξ_i with the largest $F_0 > F_{\text{in}}$ will be added to the model, with F_{in} computed as

$$F_{\text{in}} = F(\alpha; 1, N - n - 1).$$

where n is the current number of model terms and α corresponds to a given confidence level, i.e. at the 95% confidence level $\alpha = 0.05$.

After a term is added, the model is examined to determine if there are any regressors currently in the model that should be removed. That is, if n is the number of terms currently in the

model, the minimum F-statistic is computed as

$$F_0 = \min_j \frac{SS_R(\theta_n) - SS_R(\theta_{n-j})}{s^2}. \tag{2.19}$$

If $F_0 < F_{\text{out}}$, the regressor corresponding to this F_0 value, denoted ξ_j, will be removed from the model, where

$$F_{\text{out}} = F(\alpha; 1, N - n).$$

The function `finv`$(\cdot, \cdot, \cdot)$ in the MATLAB Statistics Toolbox can be used to compute F_{in} and F_{out}.

2.3.4 Analysis of Models

Several different criteria will be utilized to quantify the aerodynamic models created in later chapters. All of the criteria presented have been previously used for aircraft system identification applications [22, 24, 68, 47, 33]. The coefficient of determination, denoted as R^2, a scalar value ranging from 0 to 1, is computed from the ratio of the residual sum of squares to the total sum of squares, i.e.

$$R^2 = 1 - \frac{\sum_{k=1}^{N} (y(k) - \hat{y}(k))^2}{\sum_{k=1}^{N} (y(k) - \bar{y})^2}$$

where $y \in \mathbb{R}^{N\times 1}$ is a measured response variable, $\hat{y}$ the predicted model response, and $\bar{y}$ denotes the mean value of y.

An additional criterion, introducted by [73], characterizes the quality of fit between two series. Referred to as Theil's inequality coefficient, this criterion ranges from 0 to 1, and is computed as

$$U = \frac{\sqrt{\frac{1}{N}\sum_{k=1}^{N}\left[y(k)-\hat{y}(k)\right]^2}}{\sqrt{\frac{1}{N}\sum_{k=1}^{N}\left[y(k)\right]^2}+\sqrt{\frac{1}{N}\sum_{k=1}^{N}\left[\hat{y}(k)\right]^2}}. \tag{2.20}$$

An inequality coefficient value below 0.2-0.3 is in general indicative of a satisfactory model match [22].

Note that in [73], it is noted that the computation in Eq. 2.20 is *not* invariant to additive variation. The measured time histories should therefore be shifted towards the origin before performing computations. For analysis of linearized aircraft models this is not an issue as measurements are defined with respect to some reference value. Additionally, for system identification tests such as 3-2-1-1 or doublet inputs, where the aircraft starts at and eventually returns to a steady straight and level flight condition, the mean values of angular rate histories are essentially equal to 0. Applying the inequality coefficient to airspeed, however, can be misleading in this case.

Consider the airspeed plot given in Fig. 2.2, displaying a measured airspeed and the predicted time histories from two hypothetical models. Computing the inequality coefficient as given in Eq. 2.20, without accounting for the magnitude of the original measurement, would yield

$U_1 = 0.0190$ and $U_2 = 0.0064$. If, however, all values are shifted by subtracting the mean of the original airspeed measurement, the inequality coefficients will be $U_1 = 0.4252$ and $U_2 = 0.1681$. Without shifting the time histories, both models would be considered acceptable matches, however it is clear from both Fig. 2.2 and the corrected inequality coefficients that the second model more accurately captures the behavior of the system than the first one does.

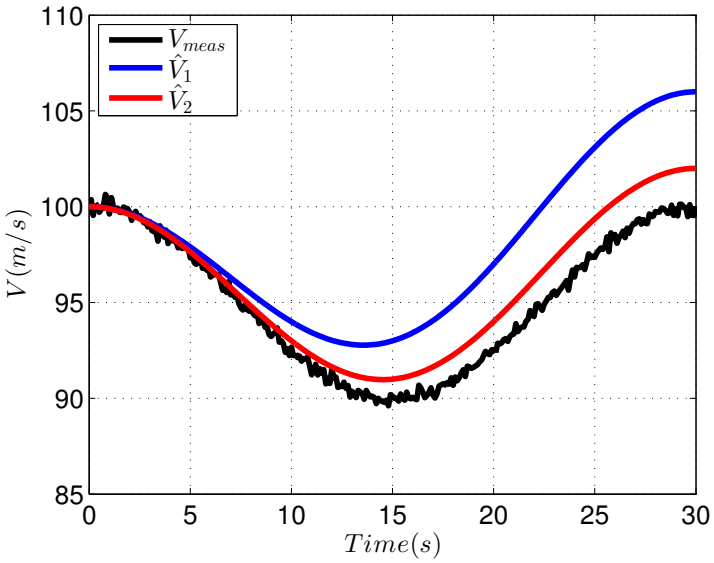


Figure 2.2: Example velocity from measured data V_{meas} and two models ($\hat{V}_1$ and $\hat{V}_2$).

The source of discrepancy between two time series can be further assessed using Theil's decomposition of fit [73, 22]. Specifically, the error is broken down into bias, variance, and covariance proportions corresponding to systematic model errors, errors due to reproducing variability, and nonsystematic errors, respectively. The sum of these values is equal to 1; in general, the bias and variance errors should each be less than 0.1. The bias (U^M), variance

(U^S), and covariance (U^C) errors are computed as

$$\begin{aligned} U^M &= \frac{\left(\bar{y} - \hat{\bar{y}}\right)}{\frac{1}{N}\sum_{k=1}^{N}\left[y(k) - \hat{y}(k)\right]^2} \\ U^S &= \frac{\left(\sigma_y - \sigma_{\hat{y}}\right)^2}{\frac{1}{N}\sum_{k=1}^{N}\left[y(k) - \hat{y}(k)\right]^2} \\ U^C &= \frac{2\left(1-\rho\right)\sigma_y\sigma_{\hat{y}}}{\frac{1}{N}\sum_{k=1}^{N}\left[y(k) - \hat{y}(k)\right]^2} \end{aligned} \tag{2.21}$$

where

$$\begin{aligned} \sigma_y &= \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left[y(k) - \bar{y}\right]^2} \\ \sigma_{\hat{y}} &= \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left[\hat{y}(k) - \hat{\bar{y}}\right]^2} \\ \rho &= \frac{1}{\sigma_y\sigma_{\hat{y}}}\frac{1}{N}\sum_{k=1}^{N}\left[y(k) - \bar{y}(k)\right]\left[\hat{y}(k) - \hat{\bar{y}}\right] \end{aligned}$$

with $\hat{\bar{y}}$ is the mean value of the predicted response.

Note that analyzing Theil's decomposition of fit according to Eq. 2.21 can be misleading. Specifically, the breakdown of fit only determines what the source of error is, and not the magnitude of error. In this work, therefore, the decomposition of fit is only performed if a deficiency has already been identified, i.e. the inequality coefficient is above the range of 0.2-0.3.

2.4 Discrete-Time H_∞ Control

In this work, all feedback control uses results developed for discrete-time linear systems. The notation for this section is mostly standard. The set of real $n \times m$ matrices is denoted by $\mathbb{R}^{n\times m}$. The adjoint of an operator X is written X^*, and $X \prec 0$ means that X is negative definite. The normed space of square summable vector-valued sequences is denoted by ℓ_2. It consists of elements $\mathbf{x} = (\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \dots)$, with each $\mathbf{x}_k \in \mathbb{R}^{n_k}$ for some n_k, having a finite 2-norm $\|\mathbf{x}\|_{\ell_2}$ defined by $\|\mathbf{x}\|_{\ell_2} = \sum_{k=0}^{\infty} |\mathbf{x}_k|^2 < \infty$, where $|\mathbf{x}_k|^2 = \mathbf{x}_k^*\mathbf{x}_k$. The system states, measurements, and feedback inputs that arise from the linearization and discretization of the nonlinear equations of motion about a finite or semi-infinite horizon are denoted $\bar{\mathbf{x}}_k$, $\bar{\mathbf{y}}_k$, and $\bar{\boldsymbol{\delta}}_k$, respectively. The exogenous errors and disturbances are denoted by $\mathbf{z}_k$ and $\mathbf{d}_k$, respectively. Let G be a linear time-varying (LTV) discrete-time system defined by the state space equation given in Eq. 2.22 and shown in Fig. 2.3. The feedback controller associated with the system G is denoted as K.

$$\begin{bmatrix} \bar{\mathbf{x}}_{k+1} \\ \mathbf{z}_k \\ \bar{\mathbf{y}}_k \end{bmatrix} = \begin{bmatrix} A_k & B_{1k} & B_{2k} \\ C_{1k} & D_{11k} & D_{12k} \\ C_{2k} & D_{21k} & 0 \end{bmatrix} \begin{bmatrix} \bar{\mathbf{x}}_k \\ \mathbf{d}_k \\ \bar{\boldsymbol{\delta}}_k \end{bmatrix}, \quad \bar{\mathbf{x}}_0 = 0 \tag{2.22}$$

The closed-loop system in Fig. 2.3 can be viewed as a map from $\mathbf{d}$ to $\mathbf{z}$, denoted as $\mathbf{d} \to \mathbf{z}$. The goal of the feedback controller design is to minimize the effects of the disturbances, $\mathbf{d}$,

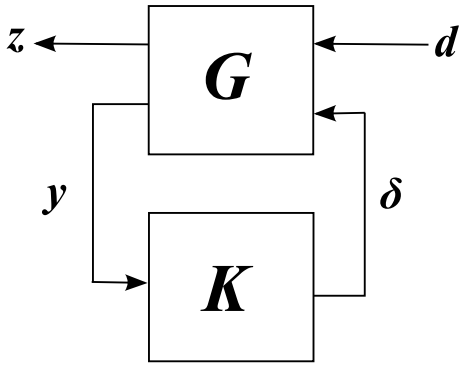


Figure 2.3: Closed loop system.

on the exogenous errors, $\mathbf{z}$. This goal can be achieved by making the map from $\mathbf{d} \to \mathbf{z}$ 'small', according to some measure; the results in [74, 45, 46] use the ℓ_2-induced norm as a performance measure, where the ℓ_2-induced norm is defined as:

$$\|\mathbf{d} \to \mathbf{z}\|_{\ell_2 \to \ell_2} = \sup_{\|\mathbf{d}\|_{\ell_2} \neq 0} \frac{\|\mathbf{z}\|_{\ell_2}}{\|\mathbf{d}\|_{\ell_2}} \tag{2.23}$$

The following definition is useful for all the systems considered in this work.

Definition 1. *An LTV system G is (h, q)-eventually periodic for some integers $h \geq 0$, $q \geq 1$ if each of its state-space matrix sequences is (h, q)-eventually periodic; for instance, the sequence A_k would be of the form*

$$\underbrace{A_0, \ldots, A_{h-1}}_{h \text{ terms}}, \underbrace{A_h, \ldots, A_{h+q-1}}_{q \text{ terms}}, \underbrace{A_h, \ldots, A_{h+q-1}}_{q \text{ terms}}, \ldots$$

For periodic systems, the finite horizon length, denoted as h in Def. 1, is 0. Finite horizon systems, on the other hand, are $(h, 1)$-eventually periodic where the periodic portions of the state-space sequences are set to zeros. It is worthwhile to note that finite horizon and periodic systems are subclasses of eventually periodic systems [45].

2.4.1 Eventually Time-Invariant Systems

In Chapter 4, the controllers designed to regulate the aircraft about a pre-specified reference trajectory utilize the results developed in [45, 46]. For this application, the system is modeled as an LTV system during a maneuver and a linear-time invariant (LTI) system about the desired final steady straight and level flight condition. Thus, the linearized system in this case is h-eventually time-invariant, following the notation of [45], where h is the number of discrete time steps required to complete the desired maneuver (the finite horizon length of the reference trajectory).

Consider the system given in Eq. 2.22. Suppose this system is h-eventually time-invariant. A synthesis is sought that would guarantee stability of the closed-loop system shown in Fig. 2.3, and additionally ensure that $\|\mathbf{d} \to \mathbf{z}\|_{\ell_2 \to \ell_2} < \gamma$, with γ being the minimum possible bound achievable by such a synthesis, to a certain tolerance. To obtain such a synthesis, the

following semidefinite programming problem must be solved:

$$
\begin{aligned}
&\text{minimize :} \quad \gamma^2 \\
&\text{subject to :} \\
&F_k^* R_k F_k - V_{1k}^* R_{k+1} V_{1k} + M_k^* M_k - \gamma^2 V_{2k}^* V_{2k} \prec 0, \\
&\begin{bmatrix} W_k^* S_{k+1} W_k - U_{1k}^* S_k U_{1k} - U_{2k}^* U_{2k} & N_k^* \\ N_k & -\gamma^2 I \end{bmatrix} \prec 0, \\
&\begin{bmatrix} R_k & I \\ I & S_k \end{bmatrix} \succeq 0, \quad R_k,\ S_k \succ 0, \quad \text{for all } k = 0, 1, \ldots, h \\
&\text{with } R_{h+1} = R_h,\ S_{h+1} = S_h
\end{aligned}
\tag{2.24}
$$

where

$$
\begin{aligned}
&\operatorname{Im}\left[V_{1k}^*\ V_{2k}^*\right] = \operatorname{Ker}\left[B_{2k}^*\ D_{12}^*\right], \quad \left[V_{1k}^*\ V_{2k}^*\right]\left[V_{1k}^*\ V_{2k}^*\right] = I, \\
&\operatorname{Im}\left[U_{1k}^*\ U_{2k}^*\right] = \operatorname{Ker}\left[C_{2k}\ D_{21}\right], \quad \left[U_{1k}^*\ U_{2k}^*\right]\left[U_{1k}^*\ U_{2k}^*\right] = I,
\end{aligned}
\tag{2.25}
$$

and

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

$$
W_k = A_k U_{1k} + B_{1k} U_{2k}, \quad N_k = C_1 U_{1k} + D_{11} U_{2k}.
$$

The solutions R_k and S_k can then be used to construct an h-eventually time-invariant controller offline, as shown in [45]. Further details on deriving the discrete-time model, G, and the solution of Eq. 2.24 will be given in Chapter 4 when the feedback approach is applied

for aircraft trajectory tracking.

2.4.2 Discrete-Time Hybrid Control

In Chapter 3, a hybrid control approach, as given in [19] and [20], is used to obtain feedback controllers for the library of motion primitives. Linearization and discretization about each motion primitive, denoted as i, result in an LTV discrete-time system $G^{(i)}$ defined by the state-space equation:

$$\begin{bmatrix} \bar{\mathbf{x}}_{k+1} \\ \mathbf{z}_k \\ \bar{\mathbf{y}}_k \end{bmatrix} = \begin{bmatrix} A_k^{(i)} & B_{1k}^{(i)} & B_{2k}^{(i)} \\ C_{1k}^{(i)} & D_{11k}^{(i)} & D_{12k}^{(i)} \\ C_{2k}^{(i)} & D_{21k}^{(i)} & 0 \end{bmatrix} \begin{bmatrix} \bar{\mathbf{x}}_k \\ \mathbf{d}_k \\ \bar{\boldsymbol{\delta}}_k \end{bmatrix}, \quad \bar{\mathbf{x}}_0 = 0 \tag{2.26}$$

As stated in [20], the system $G^{(i)}$ may be a finite-horizon, periodic, or eventually periodic system depending on the motion primitive i.

Although it is possible to develop feedback controllers for subsystems $G^{(i)}$ individually, as in Section 2.4.1, this would not guarantee stability or performance across the switching boundaries between controllers. The path planning approach in Chapter 3 concatenates motion primitives from a pre-specified library, where each motion primitive will have an associated feedback subcontroller. The motion plan is then essentially a feedback control policy composed of a sequence of subcontrollers to be applied consecutively; the following provides a hybrid control approach, as given in [19] and [20], which comes with stability

and tracking performance guarantees across the switching boundaries between compatible motion primitives.

Definition 2. *A feedback controller* $K = \left\{K^{(i)}\right\}_{i=1}^{N}$ *is a* γ*-admissible synthesis for hybrid plant* G *if the closed-loop system is asymptotically stable and the performance inequality* $\|\mathbf{d} \to \mathbf{z}\|_{\ell_2 \to \ell_2} < \gamma$ *is achieved.*

Define $\mathcal{F}_1\left(R_k, R_{k+1}, \gamma, k\right)$ and $\mathcal{F}_2\left(S_k, S_{k+1}, \gamma, k\right)$ as:

$$\mathcal{F}_1\left(R_k, R_{k+1}, \gamma, k\right) = J_k^* R_k J_k - V_{1k}^* R_{k+1} V_{1k} + M_k^* M_k - \gamma^2 V_{2k}^* V_{2k}$$

and

$$\mathcal{F}_2\left(S_k, S_{k+1}, \gamma, k\right) = \begin{bmatrix} W_k^* S_{k+1} W_k - U_{1k}^* S_k U_{1k} - U_{2k}^* U_{2k} & L_k^* \\ L_k & -\gamma^2 I \end{bmatrix},$$

respectively, where V_{1k}, V_{2k}, U_{1k}, and U_{2k} are defined as in Eq. 2.25, and

$$J_k = A_k^* V_{1k} + C_{1k}^* V_{2k} \tag{2.27}$$

$$M_k = B_{1k}^* V_{1k} + D_{11k}^* V_{2k} \tag{2.28}$$

$$W_k = A_k U_{1k} + B_{1k} U_{2k} \tag{2.29}$$

$$L_k = C_{1k} U_{1k} + D_{11k} U_{2k} \tag{2.30}$$

Additionally, define the set $\mathcal{B}_i$ as all primitives j that can succeed primitive i. With the preceding definitions, the synthesis conditions are given in Theorem 2 of [20] as:

Theorem 1. *Consider a hybrid system* $G = \left\{G^{(i)}\right\}_{i=1}^{N}$*, where each subsystem* $G^{(i)}$ *is* (h_i, q_i)*-eventually periodic. Then, there exists a* γ*-admissible hybrid synthesis* $K = \left\{K^{(i)}\right\}_{i=1}^{N}$ *to* G*, where* $K^{(i)}$ *is also* (h_i, q_i)*-eventually periodic, if, for* $i = 1, 2, \ldots, N$ *and* $k = 0, 1, \ldots, h_i + q_i - 1$*, there exist positive definite matrices* $R_k^{(i)}$ *and* $S_k^{(i)}$ *such that*

$$\mathcal{F}_1^{(i)}\left(R_k^{(i)}, R_{k+1}^{(i)}, \gamma, k\right) \prec 0 \tag{2.31}$$

$$\mathcal{F}_2^{(i)}\left(S_k^{(i)}, S_{k+1}^{(i)}, \gamma, k\right) \prec 0 \tag{2.32}$$

$$\begin{bmatrix} R_k^{(i)} & I \\ I & S_k^{(i)} \end{bmatrix} \succeq 0 \tag{2.33}$$

with

$$R_{h_i+q_i}^{(i)} = R_{h_i}^{(i)}, \quad S_{h_i+q_i}^{(i)} = S_{h_i}^{(i)}, \tag{2.34}$$

and, for all $j \in \mathcal{B}_i$*,*

$$R_{h_i}^{(i)} = R_0^{(j)}, \quad S_{h_i}^{(i)} = S_0^{(j)} \tag{2.35}$$

The proof of the above theorem can be found in [20]. The matrices $R_k^{(i)}$ and $S_k^{(i)}$ are obtained as the solution to a semidefinite feasibility problem, using solvers such as SeDuMi [75] and SDPT3 [76]. The feedback subcontroller $K^{(i)}$ can then be constructed from $R_k^{(i)}$ and $S_k^{(i)}$ according to [74].

For the synthesis conditions to have a solution, it is necessary that any linear time-invariant, periodic, or eventually periodic constituent models of the hybrid system be stabilizable and

detectable. Such requirements are not necessary for finite-horizon models (linearized about transition maneuvers) in general unless the transition maneuver can be connected to itself. As for implementation, the motion planning algorithm issues a policy which is in the form of a sequence of primitives composing a reference trajectory that leads to some desired goal state. The controller executes this policy; namely, the series of primitives corresponds to a sequence of subcontrollers to be applied consecutively. In other words, the motion planning algorithm determines the scheduling scheme of the subcontrollers constituting the hybrid control system.

As implied from the proof of Theorem 1 in [20], if a concatenated primitive trajectory is considered, the nonlinear system equations are linearized about this trajectory and then the standard ℓ_2 induced norm synthesis problem for the resulting LTV model is formulated, the hybrid synthesis solutions can be used to construct feasible solutions to this LTV synthesis program. In other words, the sequence of subcontrollers of the hybrid control system corresponding to the sequence of primitives composing the reference trajectory constitutes a stabilizing (with γ-level performance) controller to the concatenated primitive trajectory.

The aforemention control approach is utilized in Chapter 3 to regulate a vehicle about reference trajectories. These reference trajectories are generated in real-time by concatenating compatible pre-specified motion primitives. The semidefinite program in Theorem 1 is therefore comprised of temporal constraints associated with the pre-specified primitives in addition to the coupling conditions between compatible motion primitives.

Chapter 3

Graph-Based Motion Planning for Unmanned Vehicle Systems

3.1 Introduction

The approach presented in this chapter utilizes a distinct set of motion primitives and entails performing a graph search to find an appropriate dynamically feasible trajectory through an obstacle environment. The set of motion primitives, hereafter referred to as a library, is developed offline. State and control histories for each motion primitive can be obtained through a variety of methods, for instance, by solving an optimization problem involving the nonlinear system equations or by recording human operator control inputs. The task of the motion planner is to then concatenate available motion primitives to find a trajectory

from the initial state to the goal. This approach eliminates the need to solve for dynamically feasible state and control histories online. As far as the motion planner is concerned, any dynamically feasible motion primitive can be incorporated into the library. But, since these primitives can be generated experimentally, it is important that the primitive be within the state-space envelope where the derived mathematical model constitutes a reasonably accurate description of the vehicle dynamics. This requirement is imposed because the proposed control approach is model-based, as discussed later. This work examines graph-based search techniques for motion planning, where the graph does not represent a state lattice but rather exhibits a tree structure, and the edges of the graph correspond to pre-specified motion primitives. Specifically, a locally greedy algorithm with effective backtracking ability is developed and compared to a version of weighted A* based on a tree search. Both algorithms are applied in simulation to a hovercraft system and evaluated in environments composed of known, randomly constructed static obstacle fields. The greedy algorithm shows an advantage with respect to solution cost and computation time when relatively large motion primitive libraries with multiple velocities are utilized.

In addition, the hybrid control approach presented in Section 2.4.2 is applied to ensure that the system tracks the desired trajectory generated by the motion planning algorithm despite various disturbances and uncertainties. The hybrid systems of interest in this paper are composed of LTV and LTI subsystems obtained from linearizing the nonlinear system equations describing the vehicle dynamics about the library of pre-specified primitives. The

switching between these subsystems and ultimately their corresponding subcontrollers is dictated by the motion planning algorithm. The synthesis solution is provided in terms of a semidefinite program, and is based on the results of [77, 78]. Related to this work is the paper [79] which provides a hybrid dynamics framework for the design of guaranteed safe switching regions using reachable sets. The paper [80] also gives a control algorithm for maneuver-based motion planning, which is robust to a certain class of perturbations.

The work presented in this chapter serves as an extension of the results presented in the conference paper [19], and provides additional details regarding the implementation of the methodology. The intent of this work, borrowing terminology from [4], is to provide a framework for *sound* motion planning, where the devised plan guarantees a collision-free trajectory despite possible disturbances, measurement errors, and other uncertainties.

3.2 Motion Planning with a Primitive Library

The search algorithms in this work make use of a library of pre-specified motion primitives. In [3], motion primitives are defined as state and control trajectories that encompass two classes: trim trajectories and maneuvers. The trim trajectories are composed solely of steady-state motions, whereas (transition) maneuvers are trajectories that begin and end at steady-state conditions. In general, a motion primitive can be any 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. For graph search purposes, the trim trajectories in this chapter, i.e. state and control histories defined over semi-infinite time intervals, will be applied over a pre-specified number of time steps when building the search graph. With this stated, henceforth a distinction will not be made between trim and transition trajectories, and both will be referred to as motion primitives. There are two assumptions necessary for the approach presented. First, it is assumed that the motion primitives possess translational and rotational symmetry. This allows the motion planning algorithms to concatenate state and control histories during the search process. The second assumption is that the operational environment is bounded, i.e. the configuration space of the vehicle is reasonably constrained in size.

It is permissible to connect a primitive to another only when the final state of the first primitive coincides with (or at least is "very close" to) the initial state of the second one after performing the appropriate translation in position. For certain applications, it may also be necessary to ensure that the motion primitive control input histories are "compatible" across primitives, e.g. any rate limit placed on the input is respected in transitions between connectable primitives. The set of states reachable from the start node by some sequence of library primitives can be represented by a directed graph, where the vertices of the graph correspond to the reachable states and the edges indicate the motion primitives which connect the states. The expansion of nodes using this graph ultimately leads to a dynamically feasible solution path. In certain cases, where the primitives are carefully chosen, the graph

may represent a regular state lattice, as given in [15], and consequently, the A* algorithm and its variants can be applied using a lattice-based search. However, for certain systems such as a six degree-of-freedom aircraft, it may be desirable to incorporate a number of specific motion primitives into the library, which may render the task of developing a regular state lattice difficult. Specifically, requiring regularity in translational coordinates of lattice primitives will impose additional constraints on the boundary conditions when determining dynamically feasible state and control histories. Thus, it is worthwhile to study graph-based search techniques, where the graph does not represent a state lattice but rather exhibits a tree structure. For this reason, a version of WA* based on tree search is provided, in addition to an alternative algorithm where the expansion is driven by greedy behavior.

3.2.1 Weighted A* Search

In A*, each node in the graph has three values associated with it: the cost-to-go, $g(n)$, the estimated cost-to-goal, $h(n)$, and the estimated total path cost, $f(n) = g(n) + h(n)$. Note that to retain the guarantee of returning a minimum cost path, the heuristic used in A* searches must be both admissible and consistent. Admissibility requires that the heuristic never overestimates the actual cost-to-goal. A heuristic is consistent if for any nodes n and n', $h(n) \leq h(n') + c(n, n')$, where $c(n, n')$ is the edge cost between n and n'. The search is initiated from the start node and any valid (collision-free) successors are added to the queue of open nodes, $\mathcal{O}$, and the start node is added to the closed set, $\mathcal{C}$. This addition of

nodes is referred to as expansion. The search progresses by choosing the node in $\mathcal{O}$ with the lowest total path cost, $f(n)$, expanding this node and placing it in $\mathcal{C}$, and adding any valid successors to $\mathcal{O}$. This process continues iteratively until a node that reaches the goal state, or criteria, is found. A* returns the minimum cost path, should one exist, while expanding the fewest number of nodes necessary to do so [81, 1]. A relaxation of A*, developed to reduce the computational effort in the search process, is weighted A*. The optimality requirement of A* is relaxed by computing the estimated total path cost as $f(n) = g(n) + (1 + \epsilon)h(n)$, where $\epsilon > 0$ is the weighting factor. The result of this weighting is goal-driven expansion, settling for sub-optimal paths that expand towards the goal state faster [82]. The worst-case path returned by the algorithm is $(1 + \epsilon)f^*(n_{goal})$, where $f^*(n_{goal})$ is the cost of the minimum-cost path. Further information on WA* searches is contained in [83].

The WA* implementation used in this chapter is now presented. Given a library of N primitives, define the set $P = \{1, 2, \ldots, N\}$, where each element in P refers to a specific primitive in the library. Let the transition between states be denoted as $x(n') = F(x(n), p)$, where $x(n) \in X$ and $p \in P$, with $X \subset \mathbb{R}^{n_x}$ denoting the set of states reachable from the start node by some sequence of library primitives. Let X_O denote the obstacle space, $X_F = X \backslash X_O$ the obstacle-free space, and $X_G \subset X$ the goal region which is basically a user-defined neighborhood about the goal state. The motion planning problem is then to find a sequence of primitives $p_i \in P$, where $i = 1, 2, \ldots, D$ for some positive integer D, that gives an obstacle-free path x from initial state $x(1) = x_{initial}$ to $x(D + 1) \in X_G$, where

$x(i+1) = F(x(i), p_i)$. The integer D will then denote the depth of the solution path in the tree. Expansion is governed during WA* search by the value of the total path cost at each node, $f(n)$. The child node of n, denoted as n', will have a path length $g(n') = g(n)+c(n,n')$, where $c(n,n')$ is the cost associated with the primitive to reach n' from n, determined during generation of the primitive library. During motion planning, the algorithm will build and maintain a tree $\mathcal{T} = (V, E)$, where V represents the vertices of the tree in X_F and E the directed edges between vertices; each directed edge corresponds to the motion primitive necessary to transition between the associated vertices.

There are several basic functions used during the operation of the algorithm, which will now be briefly described. The function `GetSuccesors`(n,p) applies the motion primitive $p \in P_c(n)$ to n to generate a candidate node n', where $P_c(n)$ is the set of primitives that are compatible with the state $x(n)$. The function `CollisionFree`(n,p) performs collision detection along primitive p starting from $x(n)$ and terminating at $x(n')$. If collision is detected, then n' is discarded. Otherwise, the function returns n' as a valid candidate expansion node and also returns its heuristic cost $h(n')$. The implementation of collision detection used in this work considers only convex obstacle shapes, in particular rectangles defined by a base, height, and orientation angle. Obstacles are permitted to overlap, allowing for nonconvex regions of X_O. The collision detection as implemented is $O(N_c p_x n_{obs})$ in time complexity, where N_c is the number of compatible motion primitives, p_x the number of positions along each motion primitive that are checked for collision, and n_{obs} the number

of rectangular obstacles. The function `InTree`$(n', \mathcal{T})$ searches over the nodes in the tree to find those nodes, denoted n_d, which lie within an ellipsoid, $\mathcal{E}_d$, centered about $x(n')$. A similar approach was proposed by Barraquand and Latombe, however a pre-determined cell representation of the configuration space was utilized instead [12]. The choice of ellipsoid size is analogous to the choice of cell size when partitioning the configuration space; smaller $\mathcal{E}_d$ size values correspond to a finer resolution of the configuration space, and may therefore incur an increase in solution time due to a larger search tree size. The `InTree` function is $O(n_{\mathcal{T}} N_c n_{xc})$ in time complexity, where $n_{\mathcal{T}}$ is the number of nodes in the search tree, N_c the number of compatible motion primitives, and n_{xc} the number of dimensions considered for a node to be a duplicate. `Descendants`(n_d) traverses through the tree to find any nodes that were expanded from n_d and place them in the closed set if necessary. When a valid candidate is chosen for addition to the tree, the function `AddNode`$(n, n', \mathcal{T}, p)$ inserts node n' to the tree, creates an edge from n to n' storing the associated primitive p, and assigns a cost associated with this node. In WA* searches, $f(n') = g(n') + (1 + \epsilon)h(n')$, where $g(n') = g(n) + c(n, n')$ and $h(n')$ is the heuristic cost at the new node. The WA* algorithm is *resolution* complete with respect to the finite set of motion primitives utilized, as discussed in [15], contingent on the size of the ellipsoid chosen for duplicate node detection as well as the maximum allowed search depth; this will be clearly illustrated towards the end of the section.

The WA* search, given in Algorithm 1, operates as follows. The search tree $\mathcal{T}$ is initialized and n_{start}, the initial node, is added to the tree and also placed in $\mathcal{O}$ (Lines 1-2). While

Algorithm 1 Weighted A*

```
1: T ← AddNode(−, n_start, T, −)
2: O ← n_start, n ← n_start
3: while x(n) ∉ X_G or O ≠ ∅ do
4:     for p ∈ P_c(n) do
5:         n′ ← getSuccessors(n, p)
6:         if depth(n′) ≤ D_max then
7:             if CollisionFree(n, p) then
8:                 n_d ← InTree(n′, T, E_d)
9:                 if n_d ≠ ∅ then
10:                    if g(n_d) < g(n′) then
11:                        n′ not added to T
12:                    else
13:                        T ← AddNode(n, n′, T, p)
14:                        C ← Descendants(n_d)
15:                        O ← n′
16:                else
17:                    T ← AddNode(n, n′, T, p)
18:                    O ← n′
19:    C ← n
20:    n ← min(f(O))
21: return T
```

the state of the current node selected is not within the goal region, X_G, and the open list $\mathcal{O}$ is not empty, the algorithm will iterate over the while loop. The current node is set as n. Then, for any primitive $p \in P_c(n)$, the successor n' from n is obtained by applying the motion primitive p (Line 5). If the depth of the successor nodes is greater than the maximum allowed search depth, the successors will not be added to the tree (Line 6). The algorithm then checks to see if a collision occurs along the primitive p starting from state $x(n)$ (Line 7). If no collision occurs, the algorithm will proceed to check if there are nodes in the search tree that lie within some ellipsoid centered about the candidate successor (Line 8). If this

is the case and so there are already "duplicate" nodes n_d in the search tree (Line 9), the cost-to-go of the candidate and duplicates are compared. If $g(n') > g(n_d)$, n' will not be added to the search tree (Lines 10-11). Conversely, if $g(n') < g(n_d)$, any descendants of n_d in the open set will be moved to the closed set $\mathcal{C}$ and no longer considered for expansion; in addition, n' will be added to the tree and placed in the open set $\mathcal{O}$ (Lines 13-15). If no nodes are returned during the check for duplicates, the candidate node is added to the tree and placed in the open set (Lines 17-18). The current node is then placed in the closed list, $\mathcal{C}$, as all successors have been expanded (Line 19). The algorithm proceeds by choosing the node in the open set with the lowest total path cost, $f(n)$, as the next node for expansion (Line 20). The algorithm terminates when it has expanded a node within the goal region (returns the tree) or there are no more nodes remaining in the open set (returns failure).

Remark 1. *It is important to elaborate on Line 14 in Algorithm 1. The basic idea is that it is desirable to get rid of duplicate nodes up to a certain tolerance; otherwise, the search may become exhaustive and in some cases even formidable. If the tolerance is chosen to be very small, then the current node can probably inherit the descendants of the duplicate with higher cost-to-go without causing significant gaps or discontinuities in the returned trajectory. However, this may lead to scenarios where there are many nodes in the open queue that are "close" to each other with respect to their location within the configuration space, which would increase the computational cost of the algorithm. On the other hand, choosing the tolerance to be relatively large may result in trajectories with gaps, which could prove challenging to track in some agile vehicle applications. In this case, the tolerance is judiciously chosen to*

reduce the number of nodes close to each other in the open set; in addition, to avoid gaps in the returned trajectory, all descendants of a duplicate node with a higher cost-to-go, along with the duplicate node itself, are moved to the closed set. This issue does not arise when the graph represents a state lattice, in which case the duplicate nodes will match exactly, and so it is simple to update the parent node and the path cost of the descendants.

3.2.2 Greedy and Impatient Algorithm

A disadvantage of using WA* on a search tree employing the aforementioned approach occurs when updating the tree to reflect a lower cost path to within an ellipsoid of a particular node in the state space. As any descendants from the higher cost node are moved from the open list to the closed list (Algorithm 1, Line 14), computational effort expanding nodes along the higher cost path has essentially been wasted. For this reason, the application of a greedy algorithm that uses a local priority queue and quickly abandons paths deemed unfit is examined. The intent of the algorithm is to add fewer nodes into the search space, thus decreasing the number of nodes that must be examined for potential duplicates. This algorithm will henceforth be referred to as the greedy and impatient (GI) algorithm.

Expansion is governed by the incremental cost-to-go, $c(n, n')$, and the heuristic of potential successors, $h(n')$. The only nodes considered for addition to the tree are the potential successors to the current node, i.e. with a library of N total primitives, the algorithm will have at most N candidates when deciding which node to add to the tree. It should be

noted that at each iteration, the GI algorithm adds a *single* node to the search tree; this is a subtle, yet significant, difference from WA* where all valid successors are added to the tree. The inclusion of the incremental cost-to-go allows certain primitives to be penalized, if desired, by adjusting the relative cost between primitives. For a truly greedy algorithm, however, this cost would be neglected. Greedy algorithms will commit to what appear to be promising paths early on in the search, a behavior that can be problematic in complex obstacle environments; the "impatience" of the algorithm is therefore added to overcome this possible downfall. Specifically, if the estimated heuristic cost of the best successor node, n', is greater than that of the current node, n, i.e. $h(n') > h(n)$, the algorithm will become impatient and backtrack to a "watch node." Backtracking to a watch node also occurs when a node has no valid successors, i.e. all primitives applied at this node lead to a collision with an obstacle. The concept of using watch nodes is adopted from [84, 19], however the conditions for setting a watch node and the overall algorithm structure differ in this case. The watch nodes pinpoint where along the current search path specific greedy behavior has occurred. Namely, a node n is a watch node if $h(n') - h(n) < h(n) - h(n_{parent})$, where n' and n_{parent} are the successor and predecessor (parent) of n, respectively. In a physical sense, a watch node corresponds to a location in the configuration space along the current path where the algorithm selects a node for expansion that decreases the heuristic by an amount more than the decrease in heuristic resulting from the preceding motion primitive. Since there may exist other successors from the watch node that also lead to a decrease in the heuristic function, the algorithm indicates that the watch node should be re-examined if backtracking

occurs later in the search along the current path.

The notation and the functions used in the algorithm operation are the same as those used for WA*, with a few notable exceptions. If a candidate node, n', is found to be a duplicate of a node, say n_d, already within the search tree, then n' will be discarded if $g(n') > g(n_d)$. The reason is that n_d corresponds to a location in the configuration space already in the search tree. Permitting n' to be added to the tree may result in repeatedly visiting an area of the configuration space, and consequently unbounded growth of the search tree. Specifically, when nodes are checked for collision in the GI algorithm, the `InTree` check is also performed. If a potential successor is found to be a duplicate of a node already existing in the search tree and has a higher cost-to-go than the node already in the tree, then this potential successor will be treated the same as if the associated primitive led to a collision with an obstacle.

The GI algorithm is given in Algorithm 2. The algorithm begins by creating the search tree, $\mathcal{T}$, adding the start node, n_{start}, to the tree and placing it in the open set, and setting the start node as the current node (Lines 1-2). As in the WA* implementation, while the current node is not within the goal region and there are nodes remaining in the open set, the algorithm iterates over a loop (Line 3). For any primitive $p \in P_c(n)$, the successor n_p is obtained and its depth in the tree is checked to ensure that it does not exceed D_{max} (Lines 4-6). The successor n_p, along with the path from n to n_p, is then checked for collision with obstacles. If the path is collision free, n_p is added to the local priority queue, denoted $q(n)$ (Lines 7-8). If $q(n)$ is nonempty, i.e. node n has collision-free successors, the algorithm

Algorithm 2 Greedy & Impatient Algorithm

1: $\mathcal{T} \leftarrow \texttt{AddNode}(-, n_{start}, \mathcal{T}, -)$
2: $\mathcal{O} \leftarrow n_{start}$, $n \leftarrow n_{start}$
3: **while** $x(n) \notin X_G$ or $\mathcal{O} \neq \emptyset$ **do**
4: **for** $p \in P_c(n)$ **do**
5: $n_p \leftarrow \texttt{GetSuccessors}(n, p)$
6: **if** $\texttt{depth}(n_p) \leq D_{max}$ **then**
7: **if** $\texttt{CollisionFree}(n, n_p)$ **then**
8: $q(n) \leftarrow n_p$
9: **if** $q(n) \neq \emptyset$ **then**
10: $n' = \text{argmin}\{h(n_c) + c(n, n_c) \,|\, n_c \in q(n)\}$
11: $\mathcal{T} \leftarrow \texttt{AddNode}(n, n', \mathcal{T}, p)$
12: $\mathcal{O} \leftarrow n'$
13: $\text{watch}(n') \leftarrow \text{watch}(n)$
14: **if** $h(n') > h(n)$ **then**
15: $n \leftarrow \text{watch}(n)$
16: Continue while loop
17: **else**
18: $\Delta h(n') = h(n') - h(n)$
19: $\Delta h(n) = h(n) - h(n_{parent})$
20: **if** $\Delta h(n') < \Delta h(n)$ **then**
21: $\text{watch}(n') \leftarrow n$
22: $n \leftarrow n'$
23: **else**
24: $\mathcal{C} \leftarrow n$
25: **if** $n = n_{start}$ **then**
26: $n \leftarrow \min(f(\mathcal{O}))$
27: **else**
28: $n \leftarrow \text{watch}(n)$
29: **return** $\mathcal{T}$

chooses the successor, denoted n', with the minimum sum of heuristic and incremental cost (Lines 9-10). The successor n' is then added to the tree and placed in the open list (Lines 11-12). The watch node of n is also set to be the watch node of n'. The algorithm then compares the heuristic of n' to that of n. If $h(n') > h(n)$, the algorithm backtracks to the

watch node associated with n (Lines 14-16). Otherwise, the change in heuristic value between the current node, n, and the added successor n' and between the parent of the current node, n_{parent}, and n are computed (Lines 18-19). If $h(n') - h(n) < h(n) - h(n_{parent})$, the watch node of n' becomes n (Lines 20-21). n' then becomes the current node, and the iteration continues (Line 22). If a node has no successors in $q(n)$ due to collisions with obstacles (or running into duplicates), or all compatible primitives have been expanded unsuccessfully, the algorithm moves n to the closed set $\mathcal{C}$ (Line 24). If n is the start node, the node in the open set with the lowest total path cost is chosen for expansion (Line 26); otherwise, the algorithm backtracks to the watch node of n (Line 28). The algorithm terminates when no nodes remain in the open set, or a node is expanded in the goal region, X_G, in which case the search tree, $\mathcal{T}$, is returned.

Before proceeding, it is necessary to provide some commentary regarding the GI algorithm. As stated in the algorithm, a node is only removed from the open set after all primitives from the node have been expanded; it is permissible to therefore revisit any node a finite number of times, equal to the number of compatible motion primitives at the node. To avoid redundant computations, each node has a list associated with it that indicates which primitives have been expanded previously in the search process. The heuristic cost of successors, which will also indicate collision with an obstacle, is stored for each open node as well. Thus when revisiting a node during the search, the algorithm only needs to check whether the remaining available successors are potential duplicate nodes. Though in its worst-case behavior the

algorithm resorts to a brute force search over all possible primitives at each reachable node, backtracking to the start node can occur quickly during the search process. This happens, for example, when the vehicle is initially oriented away from the goal state and all primitives lead to an increase in heuristic value. In this scenario, each primitive will initially trigger the backtracking condition and the algorithm will then expand the higher heuristic cost nodes until a path with a decreasing heuristic is possible. The algorithm therefore allows for the heuristic to increase along the resulting path, but only in the event that the algorithm finds it necessary to do so. It should be noted that although the GI algorithm resorts to a WA* approach when returning to the start node via backtracking, there is no theoretically established upper bound on the path cost in this case.

Theorem 2. If X_G is reachable from $x(1)$ using a finite sequence of motion primitives, then both algorithms will find a path from the start node provided that $\mathcal{E}_d$ is set small enough, and D_{max} large enough.

Proof. The proof follows from that for Claim 1 in Barraquand and Latombe [12]. Let $\mathcal{T}_\infty \in X_F$ be the infinite tree obtained by recursively applying all motion primitives from the start node that generate collision free successors. Thus $\mathcal{T}_\infty$ denotes all reachable state space configurations in the obstacle-free space. Since the goal region, X_G, is reachable from the initial state, denote $x(D+1) \in X_G$ as the reachable state within the goal region at depth D in $\mathcal{T}_\infty$. Let $\mathcal{T}_{D_{max}}$ then be a finite subtree of $\mathcal{T}_\infty$, with depth $D_{max} \geq D$. This implies that $\mathcal{T}_{D_{max}}$ contains a collision-free path from the start node to the goal region. By choosing $\mathcal{E}_d$

small enough, it can be ensured that for any node, n, along the path from $x(1)$ to $x(D+1)$, all other nodes with state space locations within $\mathcal{E}_d$ of $x(n)$ have a cost-to-go which is greater than the cost-to-go of node n. Thus no nodes along the solution path will be removed from $\mathcal{T}_{D_{max}}$. The algorithms will then be guaranteed to find a path connecting the start node to a node within the goal region, X_G. Further, since the search tree is finite, the search will terminate in finite time. ■

As in [12], the above proof is not constructive, and thus there are several implications of Theorem 2. When using a motion primitive library constructed over a state lattice, the size of $\mathcal{E}_d$ can be set arbitrarily small, since all primitives will reach the state space location of other nodes in the tree exactly. Further, if the search space is confined to a finite region of the configuration space, the maximum depth, D_{max}, can be set arbitrarily high. By admitting only nodes with a lower cost-to-go at a location in the state space, the algorithms prevent the tree from infinite growth (repeatedly visiting areas of the state space).

For motion primitive libraries that are not generated over a state lattice, the choice of $\mathcal{E}_d$ and D_{max} may have more severe consequences. Specifically, if the algorithm terminates and returns a failure to find a path to the goal, either the size of $\mathcal{E}_d$ was too large, D_{max} was too small, or there is not a solution to the problem. Consider, for example, a set of obstacles through which there is a unique path from the start to goal, defined by the sequence of primitives $\{p_1, \ldots, p_{n-1}, \ldots, p_D\}$, where $x(1)$ is the start node, $x(n)$ some point in the state space along the path, and $x(D+1) \in X_G$. Let $\{p_1, \ldots, p_{m-1}\}$ be the sequence of motion

primitives for some other subpath to node m, where $x(n)$ lies within an ellipsoid $\mathcal{E}_d$ centered about $x(m)$. Further, let $g(m) < g(n)$, i.e. the cost-to-go to reach $x(m)$ is less than the cost-to-go for $x(n)$. In this scenario the node n and its descendants (the path to the goal region) would be discarded, and the node m retained instead. Since the path passing through n is the only path from the start node to the goal region, there is no path from m to the goal. Thus the planners would return that there is no solution when clearly this is not the case. Some smaller choice of $\mathcal{E}_d$, however, would retain node n in the search tree.

Remark 2. *Unlike best-first searches such as A* and its variants, greedy search does not provide an upper bound on the total path cost in general [81, 5, 85, 86]. Greedy search does, however, have several advantages over best-first searches in some problem domains. Due to the fact that all successors of a node are placed in the search tree, best-first search techniques may require substantial amounts of memory. The set of open nodes must be searched and sorted to determine the next node for expansion. On the other hand, in locally greedy search, there is only one active node and a potentially shorter list to sort when choosing the next node in the iteration. Thus, the computation required to choose the next node for expansion at each iteration scales linearly with the number of successors available (which is at most the size of the primitive library in our case).*

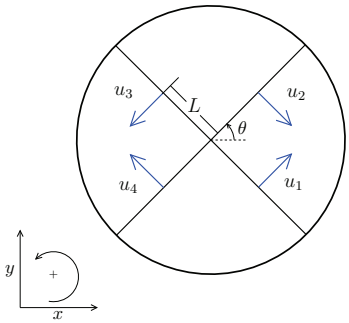


Figure 3.1: Four-thruster hovercraft

3.3 Four-Thruster Hovercraft Example

3.3.1 Motion planning

Three motion primitive libraries are developed to demonstrate the hierarchical process presented. These libraries are then used to comparatively evaluate the WA* and GI algorithms in finding suitable trajectories through environments with randomly placed obstacles. The four-thruster hovercraft model is shown in Fig. 3.1, and the equations of motion of the hovercraft are given below:

$$m\ddot{x} + b_t\dot{x} = (u_1 - u_3)\cos\theta + (u_2 - u_4)\sin\theta, \tag{3.1}$$

$$m\ddot{y} + b_t\dot{y} = (u_1 - u_3)\sin\theta + (u_4 - u_2)\cos\theta, \tag{3.2}$$

$$J\ddot{\theta} + b_r\dot{\theta} = L(u_1 - u_2 + u_3 - u_4). \tag{3.3}$$

The translational and rotational data for this system are as follows: mass $m = 1.731\,\text{kg}$, half-radius $L = 0.15\,\text{m}$, translational friction $b_t = 3.7 \times 10^{-3}\,\text{N.s/m}$, rotational friction

$b_r = 3.65 \times 10^{-4}\,\text{N.s.m/rad}$, and moment of inertia $J = 2.363 \times 10^{-2}\,\text{kg.m}^2$. Each thruster can exert a force of magnitude at most 3 N.

The libraries of motion primitives are composed of both steady-state and transition trajectories, similar to [3], with trim trajectories applied over a finite time horizon when building the search graph. For the hovercraft, the steady-state motion is forward in the body frame, where forward is defined along the $\theta = 0$ direction. The first library contains two sets of five primitives, each corresponding to final heading angle changes of $[90, 45, 0, -45, -90]$ degrees. That is, for each change of the final heading angle, there are two primitives of different lengths. The initial and final velocities of the transition primitives are all equal to 1 m/s in the direction of the vehicle heading, and the steady-state primitives also have a constant velocity of 1 m/s. Given such a velocity setup, the library is said to operate at a velocity of 1 m/s. The library primitives are judiciously chosen to form a state lattice with sampling of 0.5 m in x and y displacement. The second library has 5 primitives and is similar to the first in primitive types and velocity, however the primitives do not form a state lattice. The final library considered contains primitives with the same final heading angles as the first library and can operate at velocities of 0.5 m/s, 1 m/s, and 2 m/s. This library also contains acceleration and deceleration primitives at constant heading angle that switch between the aforementioned velocities. There are 19 total primitives in this library, and clearly not all of them are compatible with each other.

Determining dynamically feasible state and control input histories for each primitive can be

accomplished through solution of an optimal control problem [43]. Pseudospectral optimal control software tools such as DIDO [87] and GPOPS [88] specifically address the solution of such problems and have been applied to a variety of systems including autonomous ground, aerial, and surface vehicles as well as spacecraft; a review of applications of pseudospectral optimal control can be found in [89]. For the four-thruster hovercraft, the task of obtaining the state and control histories for each primitive is formulated as a convex optimization program, and then solved using the modeling system `CVX` [90], along with the solver `SDPT3` [76]. Specifically, the constraints of this program consist of the equations of motion which are discretized using zero-order hold sampling with sampling time $T = 0.05\,\mathrm{s}$, initial and final conditions on the state, a bound of $2.5\,\mathrm{N}$ on the magnitudes of the control inputs, a rate limit of $20\,\mathrm{N/s}$ on the control input histories, and affine equations relating positions to velocities based on Simpson's 1/3 rule. Note that as the equations of motion are nonlinear in the angular displacement θ, an admissible θ-trajectory is first estimated, and then, using this trajectory the equations of motion become affine in the program variables. The objective function is a weighted sum of a quadratic smoothing function of the state and control histories and a quadratic penalty function on the control inputs.

Fig. 3.2 shows a graph representing the state lattice associated with the first library. As mentioned before, the first library consists of two sets of primitives. The first set, shown in black in Fig. 3.2, can be applied to nodes with a heading angle equal to a multiple of 90 degrees. The second set can be applied to nodes with a heading angle equal to an odd

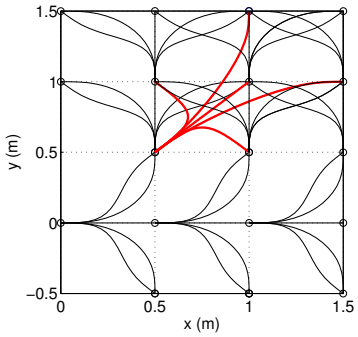


Figure 3.2: State Lattice Motion Primitives

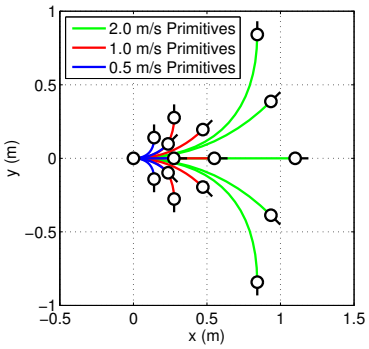


Figure 3.3: Motion Primitives at 0.5, 1, 2 m/s

multiple of 45 degrees and is shown in red in Fig. 3.2. Fig. 3.3 shows the motion primitives operating at 0.5 m/s, 1 m/s, and 2 m/s. Note that the 1 m/s primitives compose the second library. In the first library, the motion primitives with a constant heading angle are applied for 12 or 14 time steps depending on the heading of the vehicle, where a time step is equal to 0.05 s; all other motion primitives in this library are applied for 18 time steps. All primitives in the second and third libraries are applied for 12 time steps, except for the 90 degree turns at 2 m/s, which are applied for 18 time steps. The reference control inputs for a 90 degree turn at 1 m/s are shown in Fig. 3.4.

Initially, the vehicle is at the origin $(0, 0)$ with a heading angle of 45 degrees and a velocity of 1 m/s when using the first or second libraries and 0.5 m/s when using the third library. The goal region is defined by a Euclidean ball of radius 0.3 m, centered about the point $(x, y) = (12, 12)$. The performance of the GI algorithm is compared with that of WA*. The implementation of the GI algorithm will not make use of the cost-to-go, $c(n, n')$, and will be governed solely by the heuristic, $h(n)$. In the WA* case, three values for the weighting of

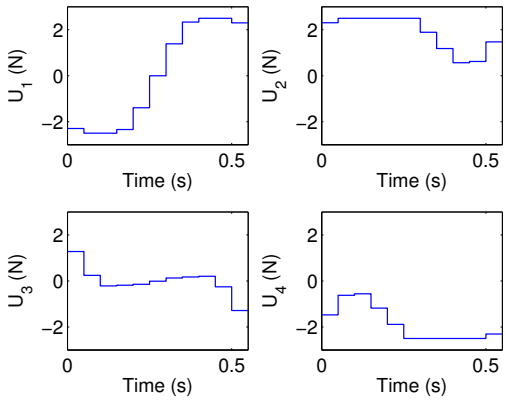


Figure 3.4: 90 degree turn control input histories

the heuristic are used, namely $\epsilon = 0.5, 1.5, 4$. The heuristic used for both algorithms is the Euclidean distance from the position of the current node in the state space to that of the goal node. More complex heuristics, which take into account obstacle edges that intersect the ray projecting from the current node to the goal node, can be used. Based on the few attempts that were carried out, however, these heuristics tend to result in an increased computation time over the Euclidean distance heuristic in this example, with a negligible improvement in the path cost. Note that the lengths of the primitives composing the third library are carefully chosen so that the vehicle can maneuver through dense obstacle areas using the lower velocity primitives and speed up in sparse regions where the higher velocity primitives do not lead to collisions.

All computations are carried out in MATLAB on a Dell Desktop with a 3.07 GHz quad-core Intel Xeon CPU and 6 GB of RAM running Windows 7. The performance measures used are the returned path cost and the wall clock time. It is probably better to use the number

of floating point operations (FLOPS) executed as a performance measure rather than the wall clock time, where the FLOP counts can be obtained using, for instance, the Lightspeed MATLAB Toolbox [91]. However, it was observed that the FLOP results are conformable with the wall clock time ones, and therefore the latter is used for simplicity. A total of 2000 tests are run in environments with random obstacle placement and orientation. In 1000 of the tests, each of the obstacles has a $1\,\text{m} \times 1\,\text{m}$ square shape, while in the remaining tests obstacles are restricted to be rectangular in shape with the length of the edges varying from $2\,\text{m}$ to $4\,\text{m}$. Although all obstacles are thus assumed to be convex, obstacles are permitted to both overlap and have arbitrary angular orientations, allowing for complex (nonconvex) configurations; collision detection is performed for each convex obstacle. For each test case, the percentage of the obstacle field coverage is the ratio of the total area of all obstacles, accounting for overlap, to the area allotted for obstacle placement. The paths are further restricted to maintain a distance of at least $0.6\,\text{m}$ from the obstacles in order to allow for some leeway when executing the motion plan in the presence of disturbances. A potential successor is considered to be a duplicate of another node in the tree if the latter node has the same heading angle and resides in a Euclidean ball of radius $0.13\,\text{m}$, centered about the potential successor; when using the third library, the velocity of the nodes must also be the same.

The mean path costs versus percentage of obstacle field coverage for each motion primitive library are given in Fig. 3.5, with error bars indicating the 95% confidence interval for the

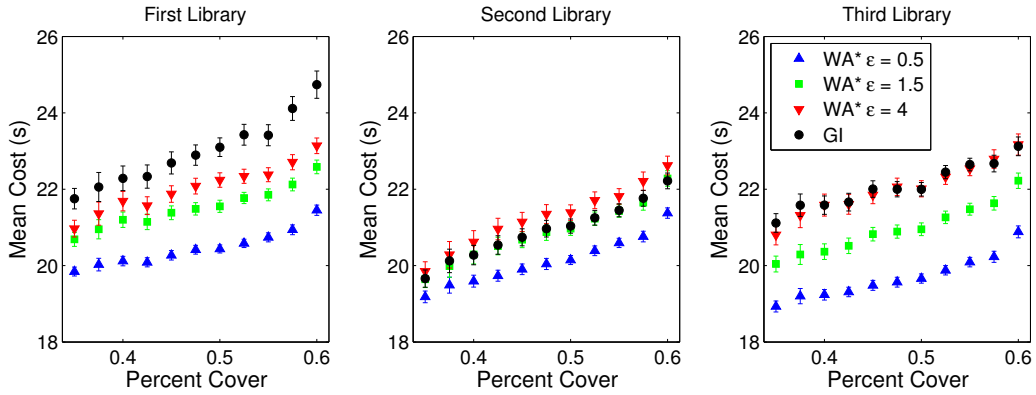


Figure 3.5: Mean path costs for all motion primitive libraries

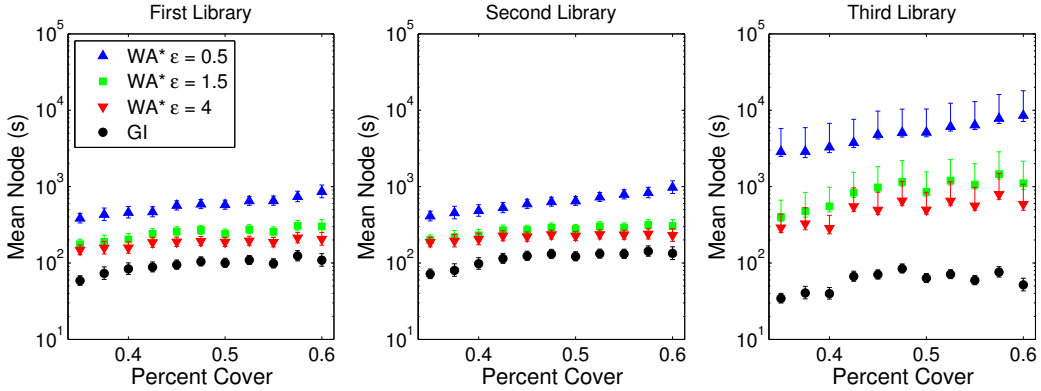


Figure 3.6: Mean number of nodes expanded for all motion primitive libraries

mean value. The path costs for the GI algorithm using the state lattice (first library) motion primitives are higher than those for WA* across the entire range of obstacle coverage. For the library with 1 m/s primitives (second library), the path costs for the GI algorithm are approximately equal to those for WA* with $\epsilon = 1.5$, verified using Student's t-test [92]. The second library also results in paths that are of lower cost than those generated using the first library. This is not an unexpected result, as the lattice library is created to operate over a uniform sampling of the translational coordinates in the configuration space, which places restrictions on the length of the motion primitives. Creating state lattice motion primitives

over a finer discretization of the translational coordinates may lead to shorter path lengths. For the library with motion primitives at multiple velocities (third library), the path costs are lower than those obtained using the second library for WA* with $\epsilon = 0.5$. The third library returns path costs that are higher than those obtained using the second library for all other choices of ϵ and for the GI algorithm as well. In this case, the GI algorithm returns path costs that are approximately equal to those for WA* with $\epsilon = 4$, and higher than those for WA* with $\epsilon = 1.5$, again verified via t-test. The mean number of nodes placed in the search tree by each algorithm for the three motion primitive libraries is given in Fig. 3.6. For the first two libraries, as expected the GI algorithm adds fewer nodes, since in this case a single node is added to the search tree per iteration. For the third motion primitive library, the WA* algorithm expands more nodes during the search process than the number expanded using the first two libraries, however the opposite is true of the GI algorithm. This is indicative of the GI algorithm utilizing the higher velocity motion primitives and finding a solution at a shallower depth in the tree, resulting in slightly higher path costs.

Fig. 3.7 shows the mean solution time versus percent obstacle coverage for each algorithm and motion primitive library, with error bars indicating the 95% confidence interval. The first library shows a slight advantage over the second library with respect to computation time. This is expected, as the motion planner using primitives constructed over a state lattice is able to update the parent of a node if a lower cost path is found to that particular node. In the case of the second library, however, duplicate nodes with lower cost result in

any descendants of the higher cost path being placed into the closed list. The GI algorithm is comparable with respect to computation time to WA* with $\epsilon = 1.5$ when using the first library; in the case of the second library, the GI algorithm requires slightly less computation time than WA* with $\epsilon = 4$. The equivalence of computation times is again verified via t-test. For WA*, the third library results in increased computation time compared to the first two libraries for all the ϵ values that we consider; in addition, the upper confidence limits on the mean computation times are significantly larger, which indicates that there are a number of cases with high solution times. The GI algorithm, on the other hand, has lower mean solution times than those obtained using the first and second libraries. Note that the number of nodes expanded in the search tree directly affects the computation time because of the computational demand of the `InTree` function. As displayed by Fig. 3.6, the GI algorithm places a far lower number of nodes in the search tree than WA* when utilizing the third motion primitive library. This leads to the WA* algorithm checking a longer list of potential duplicate nodes, resulting in increased computation time. Maximum solution times versus percent obstacle coverage are given in Fig. 3.8. From these plots, it is clear that, for the first two motion primitive libraries, the maximum computation times for the GI algorithm are comparable to those for WA* with $\epsilon = 4$. For the third library, the maximum solution times for the GI algorithm are lower than those for WA* with any of the values of ϵ considered. When using the first library, the WA* algorithm with $\epsilon = [0.5, 1.5, 4]$ returns solutions quicker than the GI algorithm in $[4.4\%, 53\%, 75\%]$ of the test cases, respectively. With the second library, WA* returns solutions in less time in $[4.7\%, 14.6\%, 26.3\%]$ of the test

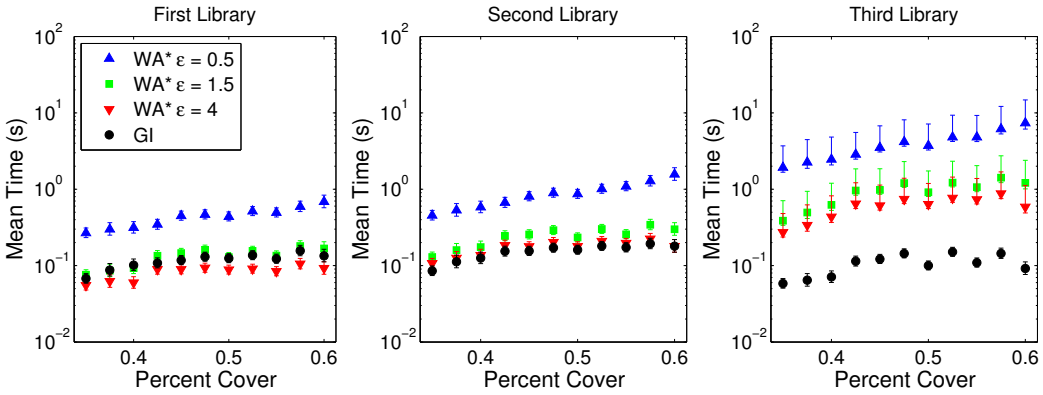


Figure 3.7: Mean solution times for all motion primitive libraries

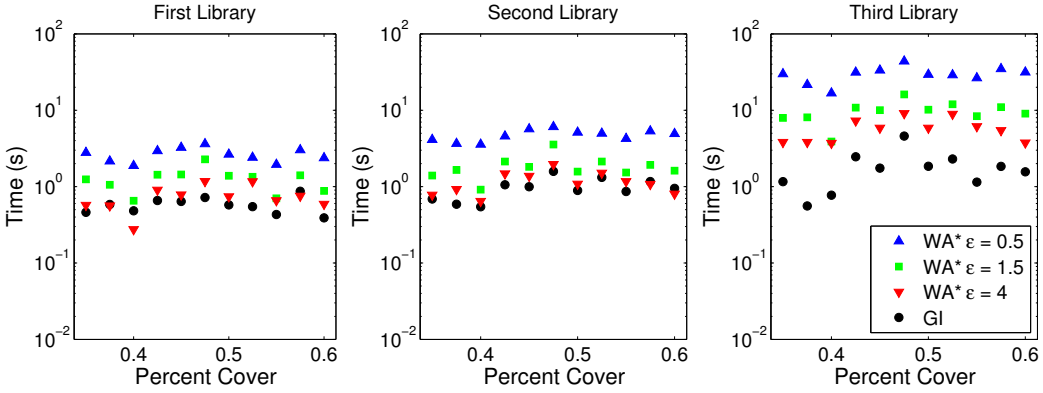


Figure 3.8: Maximum solution times for all motion primitive libraries

cases for $\epsilon = [0.5, 1.5, 4]$, respectively. Finally, WA* with $\epsilon = [0.5, 1.5, 4]$ returns solutions faster than the GI algorithm when using the third library in $[1.2\%, 2.2\%, 2.65\%]$ of the test cases, respectively.

Fig. 3.9-3.10 give paths generated by the algorithms using the three libraries for a specific obstacle environment. The (x, y) position of nodes placed in the search tree are shown in black, and the resulting solution paths are given in green. The lattice-based library planners return solutions in about 0.08 s (WA*, $\epsilon = 0.5$) and 0.04 s (GI). Computation times for

the second library are around 0.4 s (WA*, $\epsilon = 0.5$) and 0.3 s (GI), and the third library results are computed in approximately 1.6 s (WA*, $\epsilon = 0.5$) and 0.05 s (GI). The times

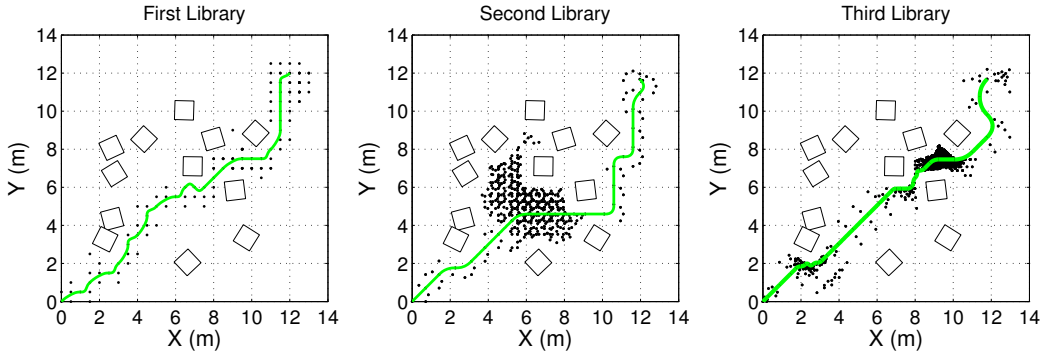


Figure 3.9: WA* paths with $\epsilon = 0.5$ for all motion primitive libraries

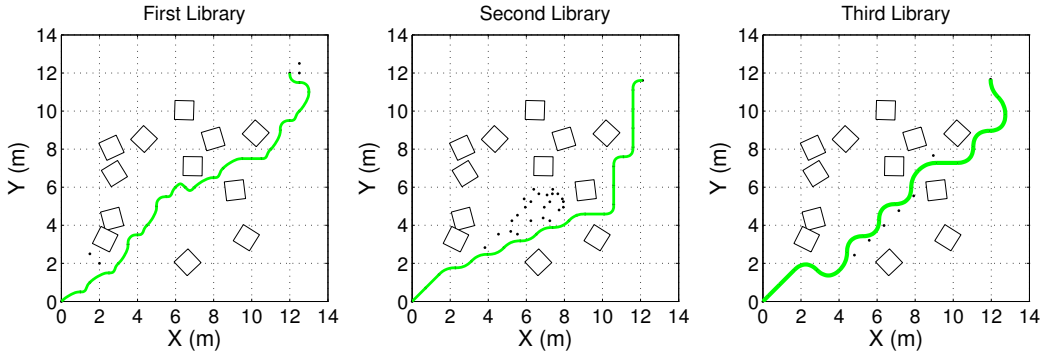


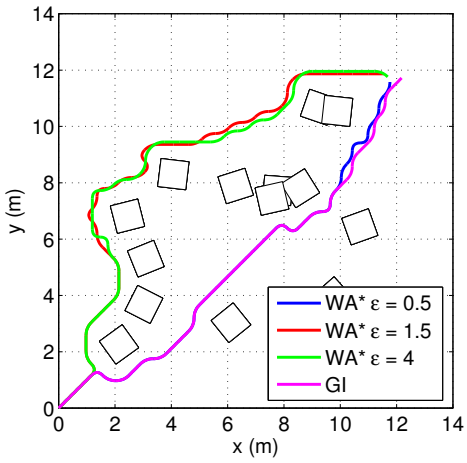
Figure 3.10: GI paths for all motion primitive libraries

needed to traverse the paths in Fig. 3.9 are 20 s, 23.4 s, and 21 s, respectively. In Fig. 3.10, the times required to traverse the paths are 20.1 s, 22.8 s, and 13.2 s, respectively. Observe that, using the third library, the GI algorithm generates a far faster trajectory than WA* (13.2 s vs 21 s). Notice also that the WA* algorithm can end up placing a large number of nodes in the tree when a state lattice is not utilized, as reflected by the density of nodes

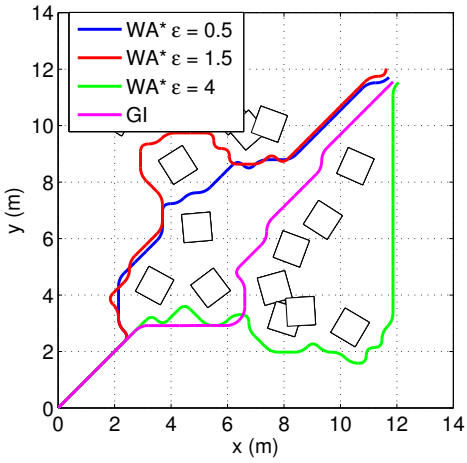
along the obstacles in Fig. 3.9 for the second and third libraries. In this particular obstacle environment, the WA* algorithm with the lowest ϵ value returns a solution faster than when using higher weighting factors in the cases of the first and third libraries, and the GI algorithm in these cases is far faster than WA* for all three weighting values. When using the second library, however, the WA* algorithm with $\epsilon = 4$ is the fastest and returns a solution in about 0.14 s. Direct comparisons of paths found using the second library are given in Fig. 3.11(a)-3.11(b), and examples of paths returned using the third library are given in Fig. 3.11(c)-3.11(d). These figures demonstrate that paths returned by the GI algorithm are comparable to those returned using WA*.

3.3.2 Hybrid control

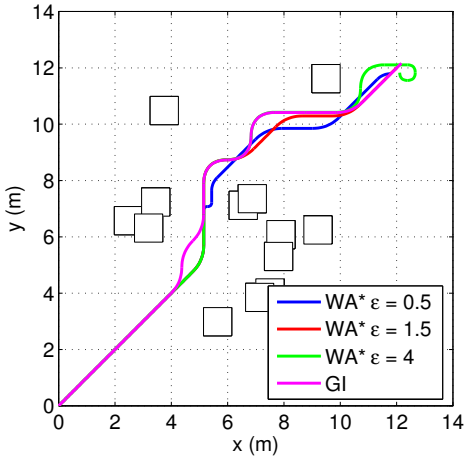
The equations of motion given in Eq. (3.1–3.2) are derived in the inertial reference frame and the dynamics subsequently vary with respect to rotations about the z-axis. Using a body-axis reference frame eliminates the dependence on θ, and the resulting transformed equations of motion are given in Eq. (3.4–3.6). The motion primitives utilized in the previous section



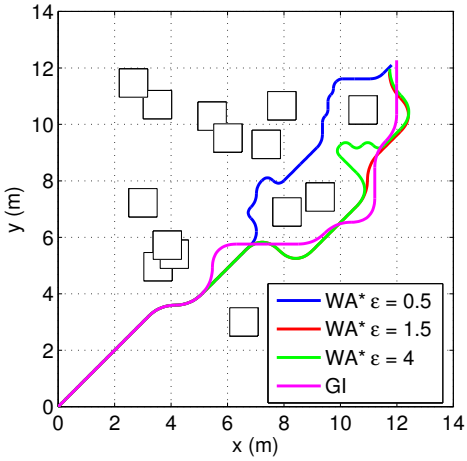
(a) Second Library Paths



(b) Second Library Paths



(c) Third Library Paths



(d) Third Library Paths

Figure 3.11: Solution Path Comparisons - Second and Third Libraries

are also transformed into the body-axis reference frame using Eq. (3.7–3.8).

$$m\ddot{x}_b + b_t\dot{x}_b = m\dot{\theta}\dot{y}_b + u_1 - u_3 \tag{3.4}$$

$$m\ddot{y}_b + b_t\dot{y}_b = -m\dot{\theta}\dot{x}_b + u_4 - u_2 \tag{3.5}$$

$$J\ddot{\theta} + b_r\dot{\theta} = L(u_1 - u_2 + u_3 - u_4) \tag{3.6}$$

$$\dot{x}_b = \dot{x}\cos\theta + \dot{y}\sin\theta \tag{3.7}$$

$$\dot{y}_b = -\dot{x}\sin\theta + \dot{y}\cos\theta \tag{3.8}$$

Defining the state column vector $\mathbf{v} = (x_b, y_b, \theta, \dot{x}_b, \dot{y}_b, \dot{\theta})$ and the input column vector $\boldsymbol{\delta} = (u_1, u_2, u_3, u_4)$, the equations of motion (3.4–3.6) can be equivalently written as $\dot{\mathbf{v}} = f(\mathbf{v}, \boldsymbol{\delta})$, where $f(\cdot,\cdot)$ is defined in the obvious way. Assuming that the input and state approximately follow the (reference) input and state trajectories associated with the finite-horizon library primitive i of length h_i, then the errors between the actual and reference variables, namely $\bar{\mathbf{v}}^{(i)} = \mathbf{v} - \mathbf{v}_r^{(i)}$ and $\bar{\boldsymbol{\delta}}^{(i)} = \boldsymbol{\delta} - \boldsymbol{\delta}_r^{(i)}$, will be small enough to satisfy the following state-space equation:

$$\dot{\bar{\mathbf{v}}}^{(i)} = A_c^{(i)}(t)\,\bar{\mathbf{v}}^{(i)} + B_{2c}^{(i)}(t)\,\bar{\boldsymbol{\delta}}^{(i)},$$

$$\text{where} \quad A_c^{(i)}(t) = \frac{\partial f}{\partial \mathbf{v}}\bigg|_{\left(\mathbf{v}_r^{(i)}, \boldsymbol{\delta}_r^{(i)}\right)} \quad \text{and} \quad B_{2c}^{(i)}(t) = \frac{\partial f}{\partial \boldsymbol{\delta}}\bigg|_{\left(\mathbf{v}_r^{(i)}, \boldsymbol{\delta}_r^{(i)}\right)}.$$

It is assumed that the hovercraft is subject to exogenous disturbances in the form of torques as well as forces in the x and y directions; also, these disturbances, like the input, are applied in discrete-time with a sampling frequency of 20Hz. In addition to these disturbances which constitute the first three components of the vector-valued signal $\mathbf{d}(t)$, errors (noise)

in the measurements of x, y, and θ are also considered, which correspond to the last three components of the signal $\mathbf{d}(t)$. Then, the state-space description of the continuous-time LTV model is

$$\dot{\bar{\mathbf{v}}}^{(i)}(t) = A_c^{(i)}(t)\,\bar{\mathbf{v}}^{(i)}(t) + B_{1c}\,\mathbf{d}(t) + B_{2c}^{(i)}(t)\,\bar{\boldsymbol{\delta}}^{(i)}(t),$$

where

$$B_{1c} = \begin{bmatrix} 0_{3\times3} & 0_{3\times3} \\ I_3 & 0_{3\times3} \end{bmatrix}.$$

The corresponding discrete-time model obtained by zero-order hold sampling is given by: $\bar{\mathbf{v}}_{k+1}^{(i)} = A_k^{(i)}\,\bar{\mathbf{v}}_k^{(i)} + B_{1k}\,\mathbf{d}_k + B_{2k}^{(i)}\,\bar{\boldsymbol{\delta}}_k^{(i)}$, where $T = 0.05\,\mathrm{s}$ is the sampling time, $\bar{\mathbf{v}}_k^{(i)} = \bar{\mathbf{v}}^{(i)}(kT)$ for integers $k \geq 0$, $A_k^{(i)} = \Phi^{(i)}\left((k+1)T, kT\right)$, $\Phi^{(i)}(\cdot,\cdot)$ being the state transition matrix, $B_{2k}^{(i)} = \int_{kT}^{(k+1)T} \Phi^{(i)}\left((k+1)T, \tau\right) B_{2c}^{(i)}(\tau)d\tau$, and B_{1k} is defined similarly to $B_{2k}^{(i)}$. It is assumed that the states x_b, y_b, and θ are measurable, and as for the exogenous errors to be controlled, in this example only $\bar{x}_b^{(i)}$, $\bar{y}_b^{(i)}$, $\bar{\theta}^{(i)}$, and $\bar{u}_j^{(i)}$ for $j = 1, \ldots, 4$ are penalized. Then the following standard discrete-time finite-horizon model is obtained:

$$\begin{bmatrix} \bar{\mathbf{v}}_{k+1}^{(i)} \\ \mathbf{z}_k \\ \mathbf{p}_k \end{bmatrix} = \begin{bmatrix} A_k^{(i)} & B_{1k} & B_{2k}^{(i)} \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{v}}_k^{(i)} \\ \mathbf{d}_k \\ \bar{\boldsymbol{\delta}}_k^{(i)} \end{bmatrix},$$

for $k = 0, 1, \ldots, h_i$, where $\mathbf{d} \in \ell_2$, $\mathbf{p}$ and $\mathbf{z}$ are the measurements and exogenous errors respectively, and, as discussed in the preceding, D_{11} and D_{22} are zeros, $C_1 = \mathrm{diag}(I_3, 0_{4\times3})$, $D_{12} = \begin{bmatrix} 0_{4\times3} & 0.1\,I_4 \end{bmatrix}^*$, $C_2 = \begin{bmatrix} I_3 & 0_{3\times3} \end{bmatrix}$, and $D_{21} = \begin{bmatrix} 0_{3\times3} & \mathrm{diag}(0.05, 0.05, 0.0436) \end{bmatrix}$.

Note that, when linearizing the system equations about the steady-state trajectory, an LTI (i.e., $(0, 1)$-eventually periodic) model is obtained. Next, given this hybrid model, a hybrid LTV synthesis is sought that would guarantee the stability of the closed-loop system and furthermore ensure that $\|\mathbf{d} \mapsto \mathbf{z}\|_{\ell_2 \to \ell_2} < \gamma$, where γ is the minimum possible bound, up to a certain tolerance, that is achievable by such a synthesis. Thus, the following semidefinite programming problem must be solved: minimize γ^2 subject to the synthesis conditions (2.31-2.35) (for a finite horizon model $G^{(i)}$, take h_i equal to the finite horizon length and $q_i = 1$ with the periodic portions of the state space sequences set to zeros; for an LTI model $G^{(j)}$, take $h_j = 0$ and $q_j = 1$). The solution is obtained using `YALMIP` [93] along with `SDPT3` [76] for this matter, with the minimum achievable $\gamma \approx 0.61$. The elapsed (i.e., wall clock) time for solving the optimization problem is about 14 s (CPU time = 11s). For the simulation, γ is relaxed to 1.3 in order to obtain satisfactory controller performance. Note that this excessive relaxation is done to accommodate the significant modeling uncertainties considered such as varying the mass and inertia by $\pm 20\%$ in simulation; however, it is possible to improve the performance while still maintaining satisfactory robustness by judiciously choosing the penalty functions on the control and state errors. The solutions of the synthesis program can then be used to construct a controller, as discussed at the end of Section 2.4; note that the MATLAB command *basiclmi* is useful for this purpose.

As for the simulation, the hovercraft is subjected to iid disturbances, which are generated by the MATLAB function *rand*; these disturbances correspond to forces in the x and y

directions, namely $\mathcal{F}_{xk}$ and $\mathcal{F}_{yk}$, as well as torques $\mathcal{T}_k$, applied on the hovercraft in discrete-time with a sampling frequency of 20 Hz, such that $|\mathcal{F}_{xk}|, |\mathcal{F}_{yk}| \leq 1$ N and $|\mathcal{T}_k| \leq 0.15$ N.m for all integers $k \geq 0$. A 3 N bound on the control inputs is imposed as well as a rate limit of 20 N/s. The mass and moment of inertia of the hovercraft is also changed by a factor of 0.8 (20% decrease) or 1.2 (20% increase). Additionally, zero mean Gaussian noise with standard deviation of 0.05 m is applied to the x and y measurements, and, similarly, zero mean Gaussian noise with 2.5-degree standard deviation is applied to the measurements of θ. Despite these disturbances and uncertainties, the controller is still able to force the hovercraft to closely track trajectories generated from the library primitives, as shown in Fig. 3.12(a)-3.12(d) for a typical run. In these figures, the different cases of considered disturbances and uncertainties are labeled as follows: force/torque disturbances only (Dist); force/torque disturbances and measurement noise (D,Meas.); force/torque disturbances, measurement noise and -20% mass and inertia (D,M,-20%); and force/torque disturbances, measurement noise, and $+20\%$ mass and inertia (D,M,$+20\%$). The reference and simulations paths through the obstacle field are given in Fig. 3.12(a). The points on the paths correspond to the locations of the center of gravity of the hovercraft. Since the radius of the hovercraft is 0.3 m which is also the difference between the edges of the concentric rectangles, the paths must not enter the rectangles with dashed edges around the obstacles. During motion planning, the width of the area between the bounding box (dashed rectangle) and the corresponding obstacle is set to 0.6 m to allow for some deviation from the reference trajectory, if necessary, due to disturbances and other uncertainties. The control input and state errors are given

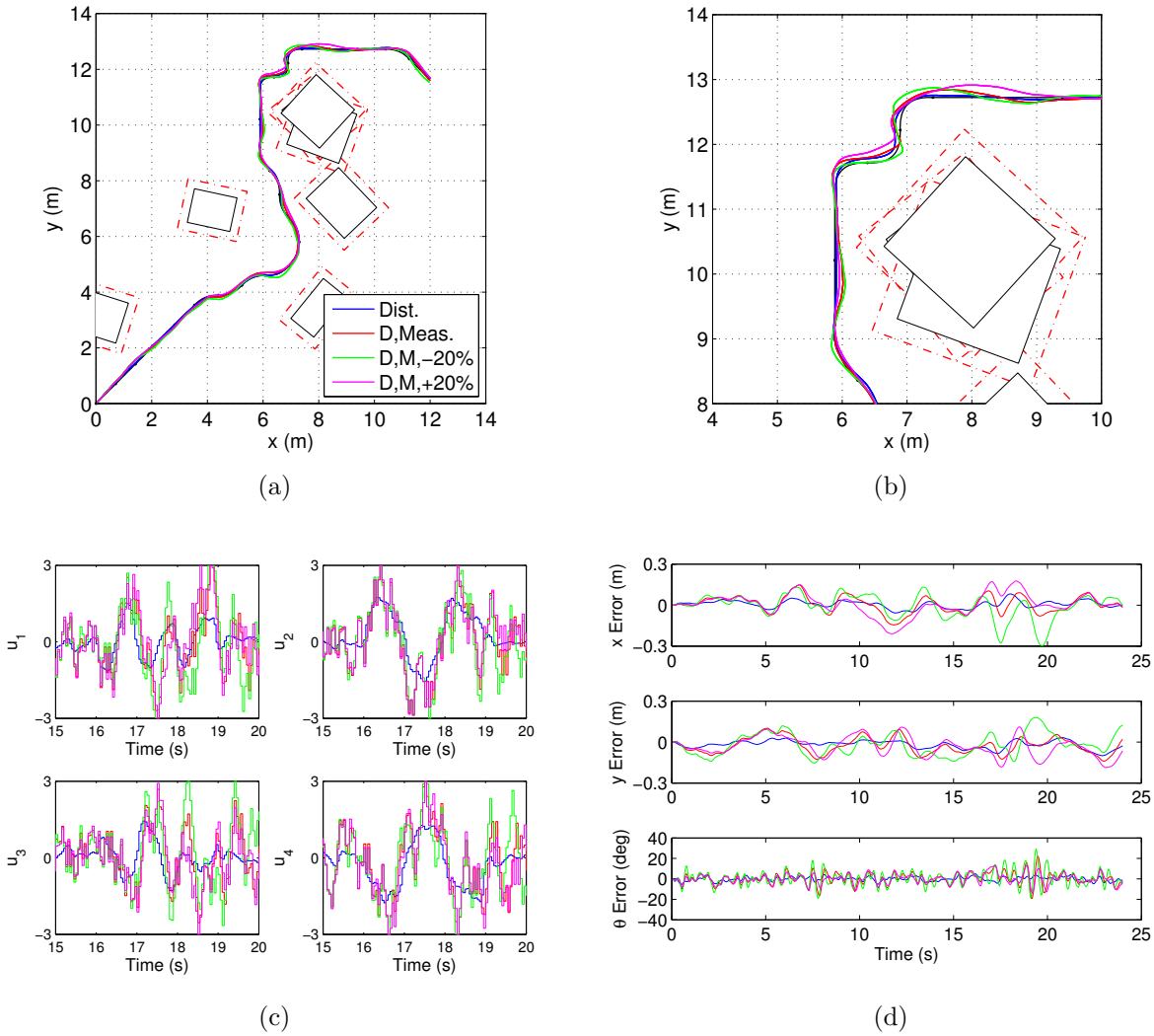


Figure 3.12: Simulation Results: (a) feedback simulation paths, (b) simulation paths - zoomed, (c) feedback control inputs, (d) errors in the states

in Fig. 3.12(c)-3.12(d). Despite the measurement noise, disturbances, and uncertainty with respect to mass and inertia, the controller is able to keep the vehicle within 0.3 m of the reference path in both x and y directions, thus avoiding collisions with obstacles in all simulation scenarios.

Chapter 4

System Identification and Control for a Small UAV

In this chapter, time-domain system identification is performed for a small commercially available radio-control (RC) aircraft operated in the Nonlinear Systems Laboratory at Virginia Tech. Specifically, the Two-Step Modeling approach, in which time histories of the aerodynamic forces and moments are first estimated through use of an extended Kalman filter and subsequently utilized for model structure determination and parameter estimation, is applied. The force and moment time histories are estimated as states of the system by modeling forces and moments as third-order Gauss-Markov processes.

The aerodynamic model obtained is then used to derive dynamically feasible trajectories for desired flight maneuvers through solution of an optimal control problem. Feedback

controllers are developed for regulation about these trajectories. These controllers are synthesized based on discrete-time plant models obtained by linearizing the nonlinear system equations about the reference trajectories and then discretizing the resulting linearized models using zero-order hold sampling. The control approach uses the ℓ_2 induced norm as the performance measure, and is based on the results of [45, 46]. Controller performance is analyzed both in simulation and flight testing of the aircraft.

4.1 Aerodynamic Modeling

4.1.1 Force & Moment Estimation

As previously stated, the TSM does not require a priori assumptions on aerodynamic model structure. It is therefore necessary to estimate the force and moment time histories during flight tests. In [66], the measurements of body-axis reference frame linear and angular accelerations, available at 1 kHz, are used to directly compute the force and moment coefficent time histories; the method used for coefficient estimation in the discussion of the equation-error method in [24] is analagous. Note that in both of the previous references, FPR is performed prior to computation of force and moment coefficients in order to remove any bias and scale factor errors from measurement data [26]. An alternative approach, presented by [29], models forces and moments as third-order GM processes driven by white noise. The latter approach is more amenable to this application, based on the lower sampling rate and

sensor noise.

The equations of motion used for force and moment estimation are the standard rigid-body dynamic equations; see [60], for example, for a more in-depth treatment and derivation. The atmospheric disturbances are assumed to be negligible, i.e. $u_d = v_d = w_d = 0$. Under the assumption that the I_{xz} product of inertia is very small relative to the moments of inertia, i.e. $I_{xz} = 0$, the rigid-body dynamic equations of motion are:

$$\begin{aligned}
\dot{u} =& \frac{F_x}{m} + rv - qw - g\sin\theta \\
\dot{v} =& \frac{F_y}{m} + pw - ru + g\cos\theta\sin\phi \\
\dot{w} =& \frac{F_z}{m} + qu - pv + g\cos\theta\cos\phi \\
\dot{p} =& \frac{1}{I_x}\left(L - (I_z - I_y)qr\right) \\
\dot{q} =& \frac{1}{I_y}\left(M - (I_x - I_z)pr\right) \\
\dot{r} =& \frac{1}{I_z}\left(N - (I_y - I_x)pq\right).
\end{aligned} \tag{4.1}$$

Note that in Eq. 4.1, the body-axis forces include contributions from aerodynamic and any thrust effects. The kinematic equations using Euler angles to define the orientation of the vehicle are given in Eq. 4.2.

$$\begin{aligned}
\dot{\phi} =& p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \\
\dot{\theta} =& q\cos\phi - r\sin\phi \\
\dot{\psi} =& q\sin\phi\sec\theta + r\cos\phi\sec\theta \\
\dot{h}_g =& u\sin\theta - v\cos\theta\sin\phi - w\cos\theta\cos\phi
\end{aligned} \tag{4.2}$$

The Gauss-Markov formulation for aerodynamic forces and moments introduces 18 additional states to the system. Each force and moment is modeled according to Eq. 2.16. Note that both lateral-directional as well as longitudinal forces and moments are estimated for every flight test analyzed.

The sensor package onboard the test aircraft provides the following measurements at a rate of 20Hz:

$$\mathbf{y} = [A_x, A_y, A_z, p, q, r, \phi, \theta, \psi, V_{T,L}, V_{T,R}, \alpha_L, \alpha_R, \beta_L, \beta_R, h_g]^T \tag{4.3}$$

where (A_x, A_y, A_z) denote the accelerations along the (X, Y, Z) body-axes, respectively, and $V_{T,i}$, α_i, and β_i are flow vector measurements obtained from pressure sensors located along the wings, defined as:

$$V_{T,i} = \sqrt{u_i^2 + v_i^2 + w_i^2}, \tag{4.4}$$

$$\alpha_i = \tan^{-1} w_i/u_i + \alpha_{b,i}, \tag{4.5}$$

$$\beta_i = \sin^{-1} v_i/V_{T,i} + \beta_{b,i}, \tag{4.6}$$

for $i = L, R$, where L and R denote left and right wings, respectively, (u_i, v_i, w_i) are the local body-axis reference frame velocities corrected for angular rotations, and $\alpha_{b,i}$ and $\beta_{b,i}$ are biases on the measurements of angle of attack and sideslip, respectively [24, 22, 26]. The body-axis reference frame velocities at each airdata probe are computed as

$$\begin{Bmatrix} u_i \\ v_i \\ w_i \end{Bmatrix} = \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} + \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \begin{Bmatrix} x_{AD,i} \\ y_{AD,i} \\ z_{AD,i} \end{Bmatrix} \quad \text{for} \quad i = L, R \tag{4.7}$$

Bias parameters on the rate gyros and accelerometers are also included. The rate gyro measurements are given as

$$\begin{Bmatrix} p_m \\ q_m \\ r_m \end{Bmatrix} = \begin{Bmatrix} p \\ q \\ r \end{Bmatrix} + \begin{Bmatrix} p_b \\ q_b \\ r_b \end{Bmatrix} \tag{4.8}$$

Additionally, since the accelerometers are not located at the center of gravity of the vehicle, the measurement equations are given as

$$\begin{Bmatrix} A_{x_m} \\ A_{y_m} \\ A_{z_m} \end{Bmatrix} = \begin{Bmatrix} A_x \\ A_y \\ A_z \end{Bmatrix} + \begin{Bmatrix} -(p^2+q^2)x_{\text{IMU}} + (pq-r)y_{\text{IMU}} + (pr+q)z_{\text{IMU}} \\ -(p^2+r^2)y_{\text{IMU}} + (pq+r)x_{\text{IMU}} + (qr-p)z_{\text{IMU}} \\ -(p^2+q^2)z_{\text{IMU}} + (pr-q)x_{\text{IMU}} + (qr+p)y_{\text{IMU}} \end{Bmatrix} + \begin{Bmatrix} A_{x_b} \\ A_{y_b} \\ A_{z_b} \end{Bmatrix}, \tag{4.9}$$

where $(x_{\text{IMU}}, y_{\text{IMU}}, z_{\text{IMU}})$ is the location of the IMU with respect to the center of gravity and $(A_{x_b}, A_{y_b}, A_{z_b})$ are the accelerometer biases. Bias terms are also included in the bank and elevation angle measurements. Due to uncertainty in both the wind direction and strength during tests, the latitude and longitude measurements are not utilized.

A continuous-discrete extended Kalman filter (EKF) is used for the state estimation process. The measurement biases are included as additional states with derivative equal to zero. With the addition of the force and moment coefficient models in Eq. 2.16 and measurement biases,

Table 4.1: Measurement Noise Characteristics

Variable	V_T	α	β	(p, q, r)
σ	0.5m/s	2.7deg	1.3deg	0.2rad/s
Variable	ϕ, θ, ψ	h_a	X_g, Y_g	A_x, A_y, A_z
σ	2deg	1.33m	0.83m	0.049m/s^2

there are a total of 39 states in the system. Equations 4.4-4.9 are clearly nonlinear functions of the state variables. The IMU corrections for accelerations not measured at the aircraft center of gravity are also nonlinear. An extended Kalman filter is therefore used for the state estimation process.

Additionally, during the lower speed longitudinal identification tests, the accelerometer measurements are corrupted due to electromagnetic interference when the engine RPM varies. The raw accelerometer measurements during one of these tests are shown in Fig. 4.1; the measurement error during this portion of the flight test is clearly not Gaussian. Instead of discarding these flight tests, the measurement vector, $\mathbf{y}$, linearized measurement matrix, $\mathbf{H}_k$, and measurement covariance matrix, $\mathbf{R_k}$, are altered during a portion of the throttle doublet to ignore accelerometer measurements. This correction was not required for the higher airspeed longitudinal flight tests. The values for the measurement covariance matrix are selected based on initial sensor calibration; standard deviations for the measurements provided by the sensors onboard the aircraft are given in Table 4.1. The process noise at each time step, $\mathbf{Q}_{k,k-1}$, required several iterations to select the appropriate order of magnitude.

Note that forces and moments, and not the aerodynamic coefficients, are estimated using

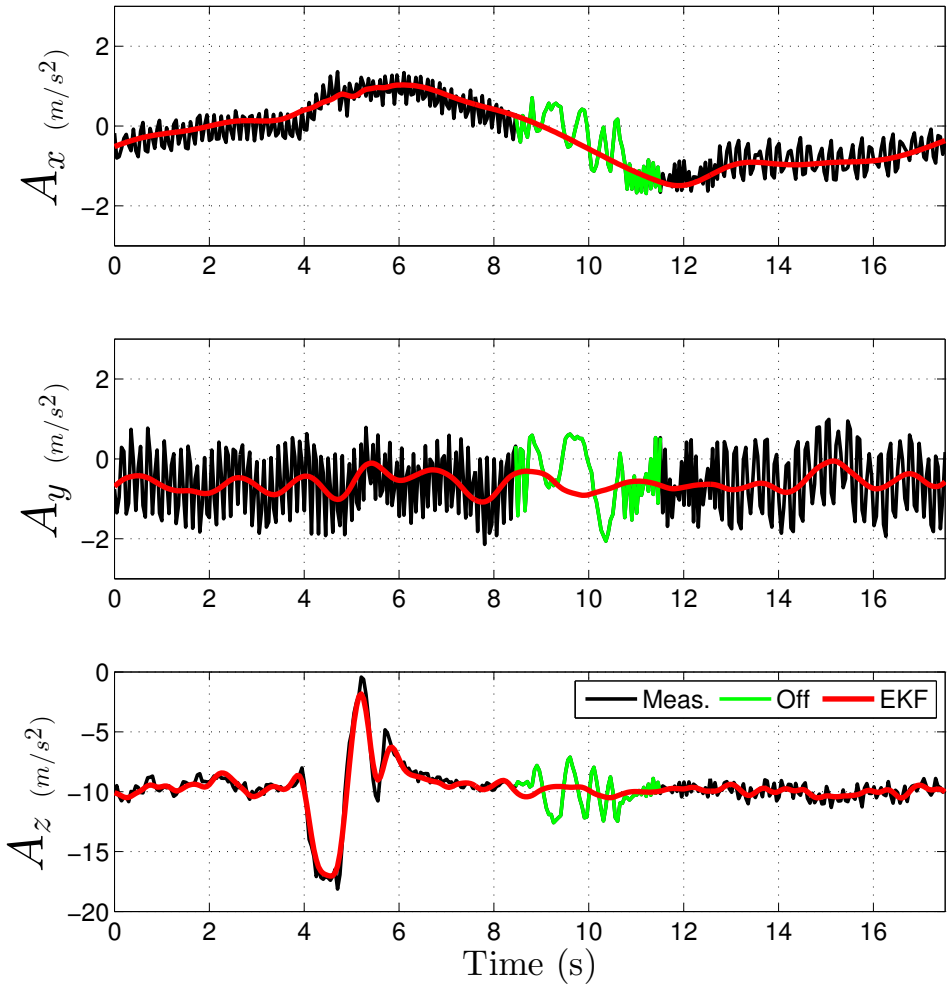


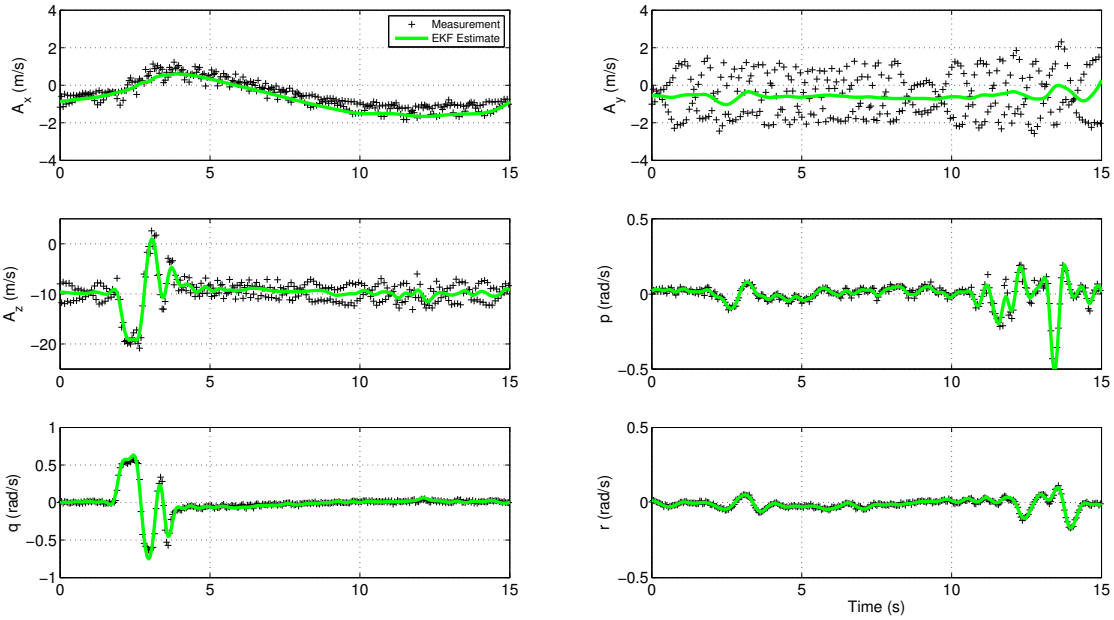
Figure 4.1: Accelerometer measurements (black) and filter estimates (red) displaying portion of test where accelerometer rate is not used (green) during a low speed longitudinal test.

the EKF. After processing each run, the nondimensionalized aerodynamic coefficients are computed from the force and moment estimates as:

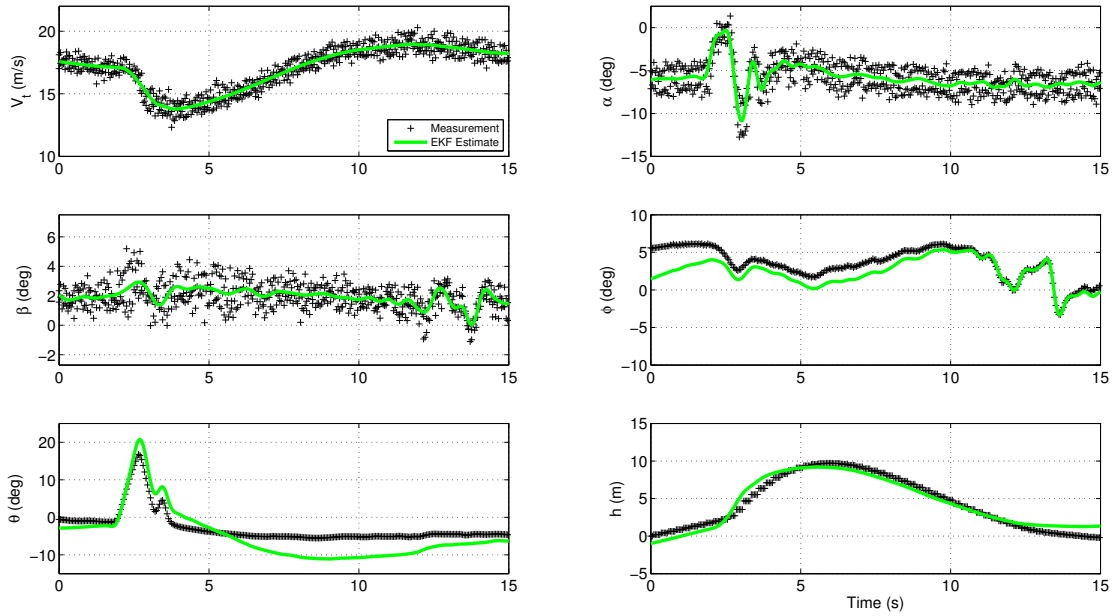
$$\begin{aligned} C_X[k] &= \frac{F_X[k]}{1/2\rho S_{\text{ref}} V[k]^2}, \quad C_Y[k] = \frac{F_Y[k]}{1/2\rho S_{\text{ref}} V[k]^2}, \quad C_Z[k] = \frac{F_Z[k]}{1/2\rho S_{\text{ref}} V[k]^2} \\ C_l[k] &= \frac{L[k]}{1/2\rho S_{\text{ref}} b V[k]^2}, \quad C_m[k] = \frac{M[k]}{1/2\rho \bar{c} S_{\text{ref}} V[k]^2} \quad C_n = \frac{N[k]}{1/2\rho b S_{\text{ref}} V[k]^2} \end{aligned} \tag{4.10}$$

where $V[k]$ in the denominator of the expressions in Eq. 4.10 is the instantaneous velocity at the center of gravity. Also, the air density, ρ, is obtained from the measurement recorded during the flight test.

An example of the measurement and state histories estimated by the EKF for a set of longitudinal test maneuvers is given in Fig. 4.2. Figure 4.2(a) gives the estimated accelerations and angular rates; the velocity, flow angles (α and β), Euler angles (ϕ, θ), and altitude are shown in Fig. 4.2(b). Note that in Fig. 4.2(b), the measured airspeed and flow angles for both five-hole probes are given as the measured data. The aerodynamic coefficient time history estimates for this set of tests are given in Fig. 4.3. From Fig. 4.3, it is clear that the lateral-directional states are not excited during the test, as evidenced by the variation primarily in the longitudinal coefficients (C_X, C_Z, C_m).



(a) Accelerations and angular rates from EKF.



(b) Flow vector and kinematic variables from EKF.

Figure 4.2: EKF state estimation results.

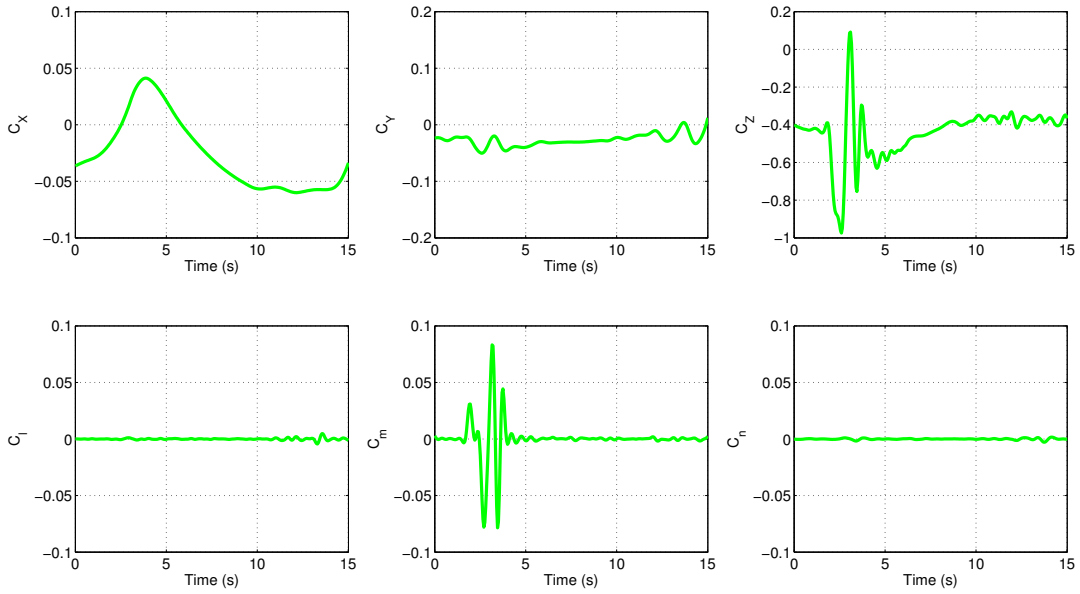
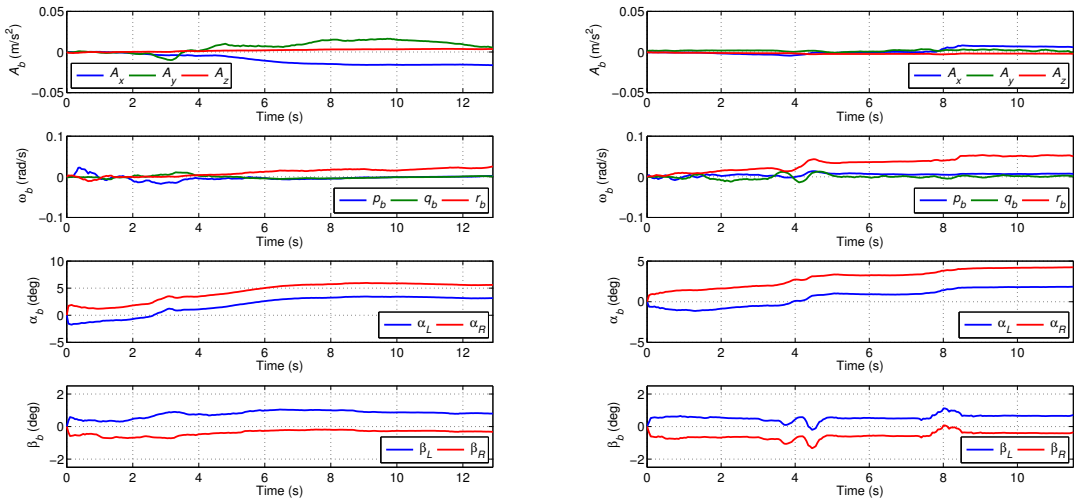


Figure 4.3: Aerodynamic coefficient estimates from elevator 3-2-1-1 and throttle doublet test.

4.1.2 Model Structure Determination

With time histories of the aerodynamic forces and moments obtained in Sec. 4.1.1, the task is then to determine an appropriate model structure and estimate any associated parameters. The aerodynamics are assumed to be quasi-steady, i.e. the forces and moments at a particular time, t, are a function of independent variables only at that time instant. In general, the forces and moments of the vehicle depend on the flow vector, Mach number, Reynold's number, and system states. Over the range of expected operation, Mach number and Reynold's number do not vary significantly and thus are not accounted for in the



(a) Biases from Longitudinal Test (b) Biases from Lateral-Directional Test

Figure 4.4: Measurement bias estimates.

aerodynamic model. The flow vector orientation is represented by angle of attack:

$$\alpha = \tan^{-1}(w/u), \tag{4.11}$$

and sideslip:

$$\beta = \sin^{-1}(v/V_t), \tag{4.12}$$

where u, v, and w are the X, Y, and Z body-axis velocties of the aircraft, and $V_t = \sqrt{u^2 + v^2 + w^2}$ is the velocity of the aircraft. The aerodynamic coefficients may also be functions of the vehicle's angular rates, non-dimensionalized as $(\omega \cdot \bar{s})/(2V_T)$, where $\bar{s}$ is a scaling parameter; the scaling parameter is chord length, $\bar{c}$, for pitch rate, and span, b, for roll

and yaw rates. These quantities, along with control surface deflections, are the independent variables in determining an appropriate aerodynamic model and structure.

The onboard flight computer records commands sent to the servos driving each control surface. A ground test is therefore performed to model the dynamics of the servos. Note that the three used servos are identical, and so only one servo needs to be modeled. Constant magnitude sinusoidal input commands across a range of frequencies are sent to the servo and the output of the servo is measured using a potentiometer. A second-order model with natural frequency $\omega_n = 21$ rad/s and damping ratio $\zeta = 0.9$ is found to adequately capture the servo dynamics.

An estimate of the thrust provided by the electric motor and propeller is available based on the dimensions of the propeller and motor RPM using the Javaprop applet [94]. This prediction of thrust is not, however, input directly into the equations of motion. Instead, the thrust coefficient is estimated as:

$$C_T = \frac{T_{JP}}{\frac{1}{2}\rho S_{\text{ref}} V_T^2}, \tag{4.13}$$

where T_{JP} is the thrust (in N) estimated using Javaprop. The computed thrust coefficient, as well as throttle command, are then included as candidate regressors for the aerodynamic forces and moments. Since it is known that the propeller shaft is not aligned with or parallel to the X-body axis of the aircraft, thrust contributions to other force and moment coefficients

are plausible.

The assumption is made that longitudinal force/moment coefficients are primarily functions of longitudinal variables, and likewise for lateral-directional coefficents. This is a common assumption made in aerodynamic modeling, particularly when flight modes are excited independently [24]. The exception is that thrust coefficient and throttle command terms are provided as candidate regressors in the lateral-directional models, thus the aerodynamic coefficient models are of the form:

$$C_i = f(\alpha, \hat{q}, \delta_E, \delta_T, C_T), \tag{4.14}$$

for longitudinal coefficients, $(i = X, Z, m)$, and of the form:

$$C_i = f(\beta, \hat{p}, \hat{r}, \delta_A, \delta_R, \delta_T, C_T), \tag{4.15}$$

for the lateral-directional coefficients $(i = Y, l, n)$.

A variety of candidate model structures are initially investigated, including global polynomial functions of the independent variables, polynomial spline formulations, and combinations of the former. As previously mentioned, the modeling approach provides for a variety of candidate model structures to be rapidly investigated. Candidate model terms are based on a standard formulation for estimating stability and control derivatives [24, 22], where polynomial splines are utilized to identify any variation in these parameters. A second

degree polynomial spline model with repsect to angle of attack, for example, is given in Eq. 4.16.

$$C_{i_{ps,2}}(\alpha) = C_{i_\alpha}\alpha + C_{i_{\alpha^2}}\alpha^2 + \sum_{j=1}^{J} C_{i_{\alpha_j}} (\alpha - \alpha_j)_+^2 \tag{4.16}$$

where J is the number of knot points, α_j the location of knots in the independent variable α, and

$$(\alpha - \alpha_j)_+^2 = \begin{cases} (\alpha - \alpha_j)^2 & \alpha \geq \alpha_j \\ 0 & \alpha < \alpha_j \end{cases} . \tag{4.17}$$

Second degree polynomial splines in α are utilized for C_X, C_Z, and C_m, with knot placement corresponding to the location of the trim angle of attack for both the low and high airspeed longitudinal tests. Additionally, the model for C_X also includes second degree polynomial splines in δ_T and C_T. Similar to angle of attack, the knot placements for throttle command and thrust coefficient also correspond to the differing trim conditions. For the lateral-directional coefficients, second degree polynomial splines in β were initially used, however it was found that including the polynomial terms β, β^2, and β^3 as candidate regressors yielded improved results. Although the candidate polynomial terms and spline functions are nonlinear in the independent variables, the aerodynamic models in this formulation remain linear with respect to the parameters, and thus the parameters can be estimated as the solution to a least-squares problem.

To determine an appropriate model structure for each aerodynamic coefficient, the estimated state histories from both longitudinal and lateral-directional tests are utilized. The reason

for utilizing all tests is that a portion of the longitudinal tests are conducted at a lower trim airspeed and have different aileron, elevator, and throttle trim settings.

Let T_M and T_V denote the number of flight tests used for aerodynamic modeling and validation, respectively. Let N_j denote the number of data points for each flight test. The total numbers of data points for modeling and validation are therefore

$$N_M = \sum_{j=1}^{T_M} N_j \quad \text{and} \quad N_V = \sum_{j=1}^{T_V} N_j, \tag{4.18}$$

respectively. Denoting each coefficient computed from the estimated forces and moments in the previous section as $\mathbf{y}_{c,j}$, each coefficient for the tests selected for aerodynamic modeling is combined into a single column vector as

$$Y_c = \left[y_{c,1}^T,\ y_{c,2}^T, \ldots, y_{c,T_M}^T\right]^T$$

for $c = (X, Y, Z, l, m, n)$. Similarly, let $\xi_{c_{i,j}}$ denote a regressor in the postulated model for a particular coefficient, i.e. a linear or nonlinear combination of the independent variables, for $i = 1, 2, \ldots, n_{p,c}$, where $n_{p,c}$ is the total number of regressors for a particular coefficient. Each regressor is then combined for all modeling flight tests into a column vector as

$$\xi_{c_i} = \left[\xi_{c_{i,1}}^T,\ \xi_{c_{i,2}}^T,\ \ldots, \xi_{c_{i,T_M}}^T\right]^T.$$

It is clear from the above that $Y_c \in \mathbb{R}^{N_M}$ and $\xi_{c,i} \in \mathbb{R}^{N_M}$. Denoting the row vector of model parameters as $\Theta_c \in \mathbb{R}^{1\times n_{p,c}+1}$, each aerodynamic coefficient model can be expressed as

$$Y_c = \theta_{c,0} + \xi_{c,1}\theta_{c,1} + \xi_{c,2}\theta_{c,2} + \cdots + \xi_{c,n_{p,c}}\theta_{c,n_{p,c}}, \tag{4.19}$$

where $\theta_{c,0}$ is a bias term. Equation 4.19 can be expressed in matrix form as

$$Y_c = \left[\mathbf{1},\ \xi_{c,1},\ \xi_{c,2},\ \ \ldots,\ \xi_{c,n_{p,c}}\right]\Theta_c \tag{4.20}$$

where $\mathbf{1} \in \mathbb{R}^{N_M}$ is a vector of ones, or, more compactly, as

$$Y_c = \ \Xi_c\Theta_c \tag{4.21}$$

with $\Xi_c \in \mathbb{R}^{N_M\times n_{p,c}+1}$ defined in the obvious way. The ordinary least-squares solution for the parameters, Θ_c, is then obtained from Eq. 4.21 as

$$\hat{\Theta}_c = \left(\Xi_c^T\Xi_c\right)^{-1}\Xi_c^T Y_c \tag{4.22}$$

Note that Eq. 4.22 determines the parameter row vector Θ_c for a given model structure. Stepwise multivariable regression is therefore used to determine the relevant regressors to include in each aerodynamic coefficient model from a pool of candidate regressors.

4.1.3 Aerodynamic Modeling Results

In total, there are eight longitudinal and three lateral-directional tests used to obtain the aerodynamic coefficient models. One test from each set is used for validation purposes, thus $T_M = 9$ and $T_V = 2$, with $N_M = 2285$ and $N_V = 532$, according to Eq. 4.18. The candidate regressors for all six aerodynamic coefficients, in the form of Eq. 4.21, are given in Eq. 4.23. Initial candidate regressors included additional knots for the second degree spline terms in α, however these terms were removed prior to the stepwise selection procedure due to having high pairwise correlation with other regressors [24].

$$\begin{aligned}
Y_X &= \big[\mathbf{1},\ \alpha,\ \alpha^2,\ (\alpha \geq -6^\circ)_+^2,\ (\alpha \geq -3^\circ)_+^2,\ \delta_T,\ \delta_T^2, C_T, C_T\alpha, (\delta_T \geq 1.50)_+^2,\ \ldots, \\
&\quad (\delta_T \geq 1.80)_+^2,\ (\delta_T \geq 1.85)_+^2,\ (C_T \geq 0.15)_+^2,\ (C_T \geq 0.25)_+^2,\ (C_T \geq 0.35)_+^2\big]\,\Theta_X \\
Y_Y &= \big[\mathbf{1},\ \beta,\ \beta^2,\ \beta^3,\ p,\ r,\ \delta_A,\ \delta_R,\ \delta_T, C_T\big]\,\Theta_Y \\
Y_Z &= \big[\mathbf{1},\ \alpha,\ \alpha^2,\ (\alpha \geq -6^\circ)_+^2,\ (\alpha \geq -3^\circ)_+^2,\ q, \delta_E, \delta_T\big]\,\Theta_Z \\
Y_l &= \big[\mathbf{1},\ \beta,\ \beta^2,\ \beta^3,\ p,\ r,\ \delta_A,\ \delta_R,\ \delta_T, C_T\big]\,\Theta_l \\
Y_m &= \big[\mathbf{1},\ \alpha,\ \alpha^2,\ (\alpha \geq -6^\circ)_+^2,\ (\alpha \geq -3^\circ)_+^2,\ q, \delta_E, \delta_T\big]\,\Theta_m \\
Y_n &= \big[\mathbf{1},\ \beta,\ \beta^2,\ \beta^3,\ p,\ r,\ \delta_A,\ \delta_R,\ \delta_T, C_T\big]\,\Theta_n
\end{aligned} \tag{4.23}$$

After performing the stepwise regression procedure for each of the aerodynamic coefficients in Eq. 4.23, the resulting aerodynamic model structures are

$$\begin{aligned}
Y_X &= \big[\mathbf{1},\ \alpha,\ \alpha^2,\ (\alpha \geq -6^\circ)_+^2,\ \delta_T^2,\ C_T,\ C_T\alpha,\ (\delta_T \geq 1.80)_+^2,\ (\delta_T \geq 1.85)_+^2, \ldots, \\
&\quad (C_T \geq 0.25)_+^2,\ (C_T \geq 0.35)_+^2\big]\,\Theta_X \\
Y_Y &= [\mathbf{1},\ \beta,\ p,\ r,\ \delta_R]\,\Theta_Y \\
Y_Z &= \big[\mathbf{1},\ \alpha^2,\ (\alpha \geq -6^\circ)_+^2,\ q,\ \delta_E,\ \delta_T\big]\,\Theta_Z \\
Y_l &= \big[\mathbf{1},\ \beta,\ \beta^2,\ \beta^3,\ p,\ r,\ \delta_A,\ \delta_R,\ \delta_T\big]\,\Theta_l \\
Y_m &= [\mathbf{1},\ \alpha,\ q,\ \delta_E,\ \delta_T]\,\Theta_m \\
Y_n &= \big[\mathbf{1},\ \beta,\ \beta^2,\ \beta^3,\ p,\ r,\ \delta_A,\ \delta_R,\ C_T\big]\,\Theta_n
\end{aligned} \tag{4.24}$$

with parameters, Θ_c for $c = (X, Y, Z, l, m, n)$, given in Tables 4.2 and 4.3 for the aerodynamic force and moment coefficients, respectively. Note that thrust terms corresponding to the commanded throttle appear in the models for C_Z, C_m, and C_l, while the thrust coefficient contributes to the yawing moment coefficient, C_n.

The coefficient of determination (R^2 value) is included for all aerodynamic coefficients, in addition to the root mean square error (RMSE). In order to assess the predictive capability of the model, aerodynamic coefficients using the validation data and Eq. 4.24 are computed for comparison with the EKF force and moment coefficient estimates. The R^2 and RMSE values for the validation data should not vary significantly from their counterparts computed from the modeling data. A significant increase in RMSE, for example, may be indicative of overfitting during the modeling process.

The R^2 and RMSE values for modeling and validation, for each aerodynamic coefficient, are given in Table 4.4. It is clear from the results in Table 4.4 that there is a reduction in R^2 value for the longitudinal coefficients, with the pitching moment, C_m, showing the largest decrease. The RMSE values for the longitudinal validation data, however, are in agreement with those computed from the modeling data. The lateral-directional coefficients on the other hand show an increase in R^2 for the validation data, with RMSE values in close agreement between both modeling and validation data. Computing the coefficient of determination for the lateral-directional modeling tests only yields $R^2 = [0.9633,\ 0.8824,\ 0.9761]$ for $[C_Y,\ C_l,\ C_n]$, respectively. Model quality is further assessed through forward simulation from the initial

Table 4.2: Aerodynamic Force Coefficient Parameters

Term	Value	Rel. σ	Term	Value	Rel. σ	Term	Value	Rel. σ
C_{X_0}	-0.3524		C_{Y_0}	0.0066		C_{Z_0}	-0.8668	
C_{X_α}	0.8628	0.1355	C_{Y_β}	-1.1760	0.0120	$C_{Z_{\alpha^2}}$	23.7591	0.0094
$C_{X_{\alpha^2}}$	3.1972	0.1582	C_{Y_p}	0.5401	0.0651	$C_{Z_{(\alpha \geq -6^\circ)_+^2}}$	-28.0433	0.0083
$C_{X_{(\alpha \geq -6^\circ)_+^2}}$	-2.4651	0.2351	C_{Y_r}	0.7287	0.0758	C_{Z_q}	-10.9171	0.0680
$C_{X_{\delta_T^2}}$	0.0964	0.0180	$C_{Y_{\delta_R}}$	0.3957	0.0468	$C_{Z_{\delta_E}}$	1.1278	0.0242
$C_{X_{C_T\alpha}}$	-0.4098	0.1728				$C_{Z_{\delta_T}}$	0.1224	0.0920
$C_{X_{C_T}}$	0.6880	0.0117						
$C_{X_{(\delta_T \geq 1.8)_+^2}}$	-4.3729	0.1159						
$C_{X_{(\delta_T \geq 1.85)_+^2}}$	6.9898	0.3635						
$C_{X_{(C_T \geq 0.25)_+^2}}$	-2.7334	0.0526						
$C_{X_{(C_T \geq 0.35)_+^2}}$	8.2866	0.0849						

Table 4.3: Aerodynamic Moment Coefficient Parameters

Term	Value	Rel. σ	Term	Value	Rel. σ	Term	Value	Rel. σ
C_{l_0}	-0.0113		C_{m_0}	-0.0733		C_{n_0}	-0.0002	
C_{l_β}	-0.0500	0.0409	C_{m_α}	-0.2654	0.0239	C_{n_β}	0.0702	0.0153
$C_{l_{\beta^2}}$	-0.0500	0.2324	C_{m_q}	-11.4204	0.0135	$C_{n_\beta^2}$	-0.0441	0.1408
$C_{l_{\beta^3}}$	0.4725	0.3055	$C_{m_{\delta_E}}$	0.6412	0.0089	$C_{n_{\beta^3}}$	0.8909	0.0862
C_{l_p}	-0.2550	0.0208	$C_{m_{\delta_T}}$	0.0380	0.0611	C_{n_p}	-0.0405	0.0713
C_{l_r}	0.1744	0.0287				C_{n_r}	-0.0846	0.0321
$C_{l_{\delta_A}}$	0.1609	0.0150				$C_{n_{\delta_A}}$	-0.0266	0.0486
$C_{l_{\delta_R}}$	0.0140	0.1192				$C_{n_{\delta_R}}$	-0.0590	0.0152
$C_{l_{\delta_T}}$	0.0045	0.0605				$C_{n_{C_T}}$	-0.0043	0.0430

Table 4.4: Modeling and Validation Results

Modeling

Coef.	C_X	C_Y	C_Z	C_l	C_m	C_n
R^2	0.9650	0.8270	0.9657	0.7362	0.8609	0.9201
RMSE	0.0083	0.0133	0.0370	0.0013	0.0077	0.0006

Validation

Coef.	C_X	C_Y	C_Z	C_l	C_m	C_n
R^2	0.9226	0.9475	0.9277	0.8028	0.7898	0.9656
RMSE	0.0073	0.0088	0.0290	0.0014	0.0057	0.0005

conditions estimated by the EKF with the commanded deflections as inputs to the system. Specifically, the nonlinear equations of motion are integrated forward in time and relevant states are compared to the filter estimated state histories. For all simulations, both the longitudinal and lateral-directional models are used to determine aerodynamic coefficients, i.e. forward simulation does not use decoupled equations of motion.

The forward simulation performance is quantified using the goodness-of-fit (GOF) criterion as well as Theil's inequality coefficient (TIC) [30, 22]. The metrics have been previously utilized to anlayze the discrepancy between measured (filter estimated) and predicted time series for aerodynamic modeling purposes. Note that for longitudinal tests, GOF and TIC are computed for the states (u, w, q), while for lateral-diretional tests, the relevant states are (v, p, r). From [30], GOF is computed as

$$\text{GOF} = 1 - \frac{(\mathbf{z}_\text{m} - \mathbf{z}_\text{p})^\text{T}(\mathbf{z}_\text{m} - \mathbf{z}_\text{p})}{(\mathbf{z}_\text{m} - \text{z}_{\text{m}_0})^\text{T}(\mathbf{z}_\text{m} - \text{z}_{\text{m}_0})} \tag{4.25}$$

where $\mathbf{z}_m$ is the measured time history for the variable of interest, z_{m_0} the initial value of $\mathbf{z}_m$, and $\mathbf{z}_p$ the model predicted time history. From Eq. 4.25, it is clear that a GOF value of 1 indicates a perfect time history match between the measured and model predicted state. Theil's inequality coefficient is computed as given in Eq. 2.20. Note that due to the implications when the time series have non-zero means, as discussed in Section 2.3.4, all inequality coefficient values are computed after subtracting the mean of the measured (EKF estimated) time history from both the reference and simulation time histories.

Table 4.5: GOF and TIC Values for Modeling & Validation Simulations

	Modeling		**Validation**			**Modeling**		**Validation**	
Variable	GOF	TIC	GOF	TIC	Variable	GOF	TIC	GOF	TIC
u	0.9585	0.1326	0.9831	0.0071	v	0.9751	0.0834	0.9492	0.0909
w	0.9521	0.1389	0.9576	0.0332	p	0.9551	0.1087	0.9312	0.1382
q	0.9688	0.0961	0.9683	0.0913	r	0.9461	0.1139	0.9609	0.1056

The GOF and TIC values for simulation of both modeling and validation tests are given in Table 4.5. Note that for the modeling tests, the average GOF and TIC values are reported. The inequality coefficient values are computed according to Eq. 2.20 with the mean values of the estimated time history removed. Based on the GOF values in the table, it is clear that the aerodynamic model is able to reproduce the dynamic behavior of the aircraft for both modeling and validation test simulations. Further, the TIC values indicate that the models are not deficient.

The forward simulation states of the longitudinal and lateral-directional validation flight tests are given in Fig. 4.5 and Fig. 4.6, respectively. The longitudinal test consists of an elevator 3-2-1-1 control input sequence followed by a throttle doublet; the lateral-directional test is a rudder doublet followed by an aileron 1-2-1 sequence. From both Table 4.5 and Fig. 4.5 and 4.6, the identified model is in good agreement with the validation data.

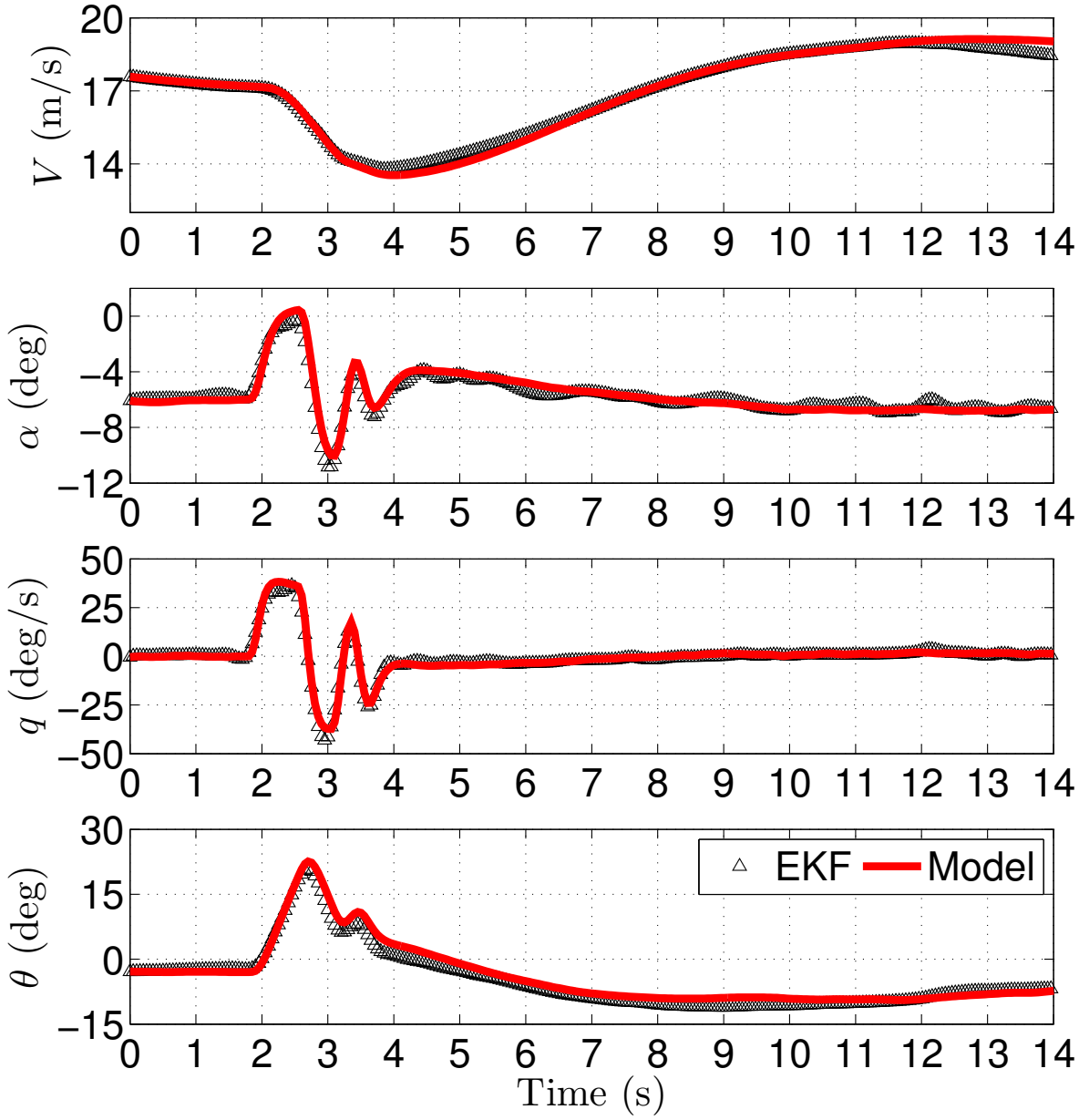


Figure 4.5: Longitudinal validation simulation plots.

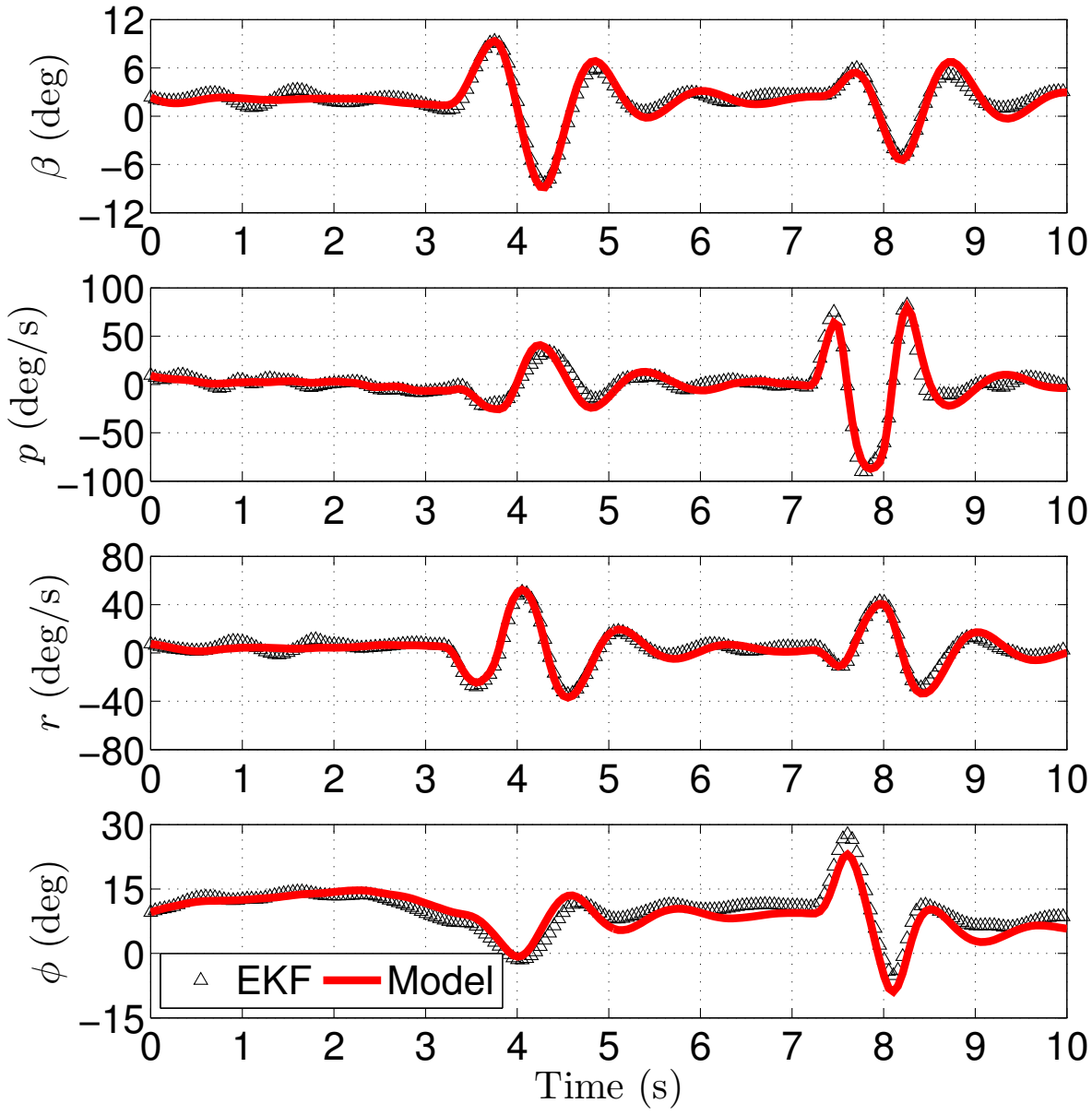


Figure 4.6: Lateral-directional validation simulation plots.

4.2 Trajectory Generation

As previously stated, deriving dynamically feasible reference trajectories based on the nonlinear equations of motion is advantageous over obtaining trajectories from flight testing, primarily due to the ability to enforce specific constraints on the vehicle motion and avoiding contamination of the reference trajectory due to exogenous disturbances.

4.2.1 Problem Formulation

In this section, dynamically feasible reference trajectories for the Telemaster aircraft are obtained as the solution to an optimal control problem of the form

$$\begin{aligned} \text{Minimize} \quad & J = \boldsymbol{\Psi}\left(\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f\right) + \int_{t_0}^{t_f} \boldsymbol{\mathcal{L}}\left(\mathbf{x}(t), \boldsymbol{\delta}(t), t\right) dt \\ \text{Subject to} \quad & \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \boldsymbol{\delta}) \\ & \mathbf{C}_L \leq \mathbf{C}(\mathbf{x}(t), \boldsymbol{\delta}(t), t) \leq \mathbf{C}_U \\ & \mathbf{B}_L \leq \mathbf{B}(\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f) \leq \mathbf{B}_U \end{aligned} \tag{4.26}$$

where $\mathbf{x} \in \mathbb{R}^{\mathbb{N}_x}$, $\boldsymbol{\delta} \in \mathbb{R}^{\mathbb{N}_\delta}$, $t \in \mathbb{R}$ define the state, control, and time respectively. The functions $\boldsymbol{\Psi}(\cdot)$, $\boldsymbol{\mathcal{L}}(\cdot)$, $\mathbf{f}(\cdot)$, $\mathbf{C}(\cdot)$, and $\mathbf{B}(\cdot)$ in Eq. 4.26 define the endpoint cost, running cost, dynamic constraints, path constraints, and boundary condition constraints, respectively, with the mappings given below:

$$\begin{aligned}
\boldsymbol{\Psi} &: \mathbb{R}^{\mathbb{N}_x} \times \mathbb{R} \times \mathbb{R}^{\mathbb{N}_x} \times \mathbb{R} \to \mathbb{R} \\
\boldsymbol{\mathcal{L}} &: \mathbb{R}^{\mathbb{N}_x} \times \mathbb{R}^{\mathbb{N}_\delta} \times \mathbb{R} \to \mathbb{R} \\
\mathbf{f} &: \mathbb{R}^{\mathbb{N}_x} \times \mathbb{R}^{\mathbb{N}_\delta} \times \mathbb{R} \to \mathbb{R}^{\mathbb{N}_x} \\
\mathbf{C} &: \mathbb{R}^{\mathbb{N}_x} \times \mathbb{R}^{\mathbb{N}_\delta} \times \mathbb{R} \to \mathbb{R}^{\mathbb{N}_c} \\
\mathbf{B} &: \mathbb{R}^{\mathbb{N}_x} \times \mathbb{R} \times \mathbb{R}^{\mathbb{N}_x} \times \mathbb{R} \to \mathbb{R}^{\mathbb{N}_b}
\end{aligned}$$

where $\mathbb{N}_c$ and $\mathbb{N}_b$ denote the number of the path and boundary constraints, respectively. The dynamic contraints, $\mathbf{f}(\mathbf{x}(t), \boldsymbol{\delta}(t))$, are given by the aircraft's equations of motion; the remaining constraints, and their associated bounds, are chosen based on the desired aircraft motion.

Solutions to the optimal control problem are obtained using GPOPS-II, a commercial nonlinear optimal control software package written to interface with MATLAB. Specifically, GPOPS-II uses an *hp*-adpative Radau pseudospectral method with Legendre-Gauss-Radau collocation points, transcribing the optimal control problem into a nonlinear programming problem (NLP). Further details regarding the theory and implementation of the pseudospectral method employed by GPOPS-II can be found in [95, 96, 97, 98]. In this work, the NLP solver IPOPT included with GPOPS-II is utilized [99].

Before proceeding, it is worthwhile to comment on the operation of GPOPS-II with respect to the aerodynamic and thrust models utilized in this work. Let $\mathbf{x} \circ \mathbf{y}$ denote the element-wise

multiplication of the vectors $\mathbf{x} \in \mathbb{R}^a$ and $\mathbf{y} \in \mathbb{R}^a$, i.e.

$$\begin{aligned} \mathbf{z} =& \mathbf{x} \circ \mathbf{y} \\ z_j =& x_j \cdot y_j \quad \text{for} \quad j = 1, \ldots, a \end{aligned}$$

In transcribing the optimal control problem to an NLP, GPOPS-II requires that the dynamics, constraint, and cost functions are evaluated at each collocation point. The velocity dynamic equations, for example, are therefore written as

$$\begin{aligned} \dot{u}_i =& -g_0 \sin\theta_i + r_i v_i - q_i w_i + \frac{1}{m} F_{x,i} \\ \dot{v}_i =& g_0 \cos\theta_i \sin\theta_i + p_i w_i - r_i u_i + \frac{1}{m} F_{y,i} \\ \dot{w}_i =& g_0 \cos\theta_i \cos\theta_i + q_i u_i - p_i v_i + \frac{1}{m} F_{z,i} \end{aligned} \tag{4.27}$$

for $i = 1, \ldots, N_c$, where N_c is the number of collocation points. Alternative, Eq. 4.27 can be expressed as

$$\begin{aligned} \dot{\mathbf{u}} =& -g_0 \sin\boldsymbol{\Theta} + \mathbf{r} \circ \mathbf{v} - \mathbf{q} \circ \mathbf{w} + \frac{1}{m} \mathbf{F_x} \\ \dot{\mathbf{v}} =& g_0 \cos\boldsymbol{\Theta} \circ \sin\boldsymbol{\Phi} + \mathbf{p} \circ \mathbf{w} - \mathbf{r} \circ \mathbf{u} + \frac{1}{m} \mathbf{F_y} \\ \dot{\mathbf{w}} =& g_0 \cos\boldsymbol{\Theta} \circ \sin\boldsymbol{\Phi} + \mathbf{q} \circ \mathbf{w} - \mathbf{r} \circ \mathbf{u} + \frac{1}{m} \mathbf{F_z} \end{aligned} \tag{4.28}$$

where, $\mathbf{u} \in \mathbb{R}^{N_c \times 1}$, for example. The angular dynamic equations and inertial position and orientation kinematic equations are computed in the same manner.

Vectorized Thrust Model

The aerodynamic model developed in Section 4.1.2 is in a form well-suited for evaluation using vectors of the independent variables, as in Eq. 4.28, with the exception of the Javaprop predicted thrust. The output of the Javaprop applet provides tables at constant motor RPM values that give the predicted thrust at different airspeed values. In order to determine the thrust for a given PWM throttle command, the propeller RPM must first be determined based on a calibration relating PWM and propeller RPM. Two-dimensional interpolation is then performed in order to determine the thrust based on the airspeed and computed propeller RPM. As the number of collocation points increases, so will the number of function calls to the Javaprop model. A vectorized model of the thrust predicted by Javaprop is therefore created, which returns accurate thrust predictions significantly faster than using multivariable interpolation.

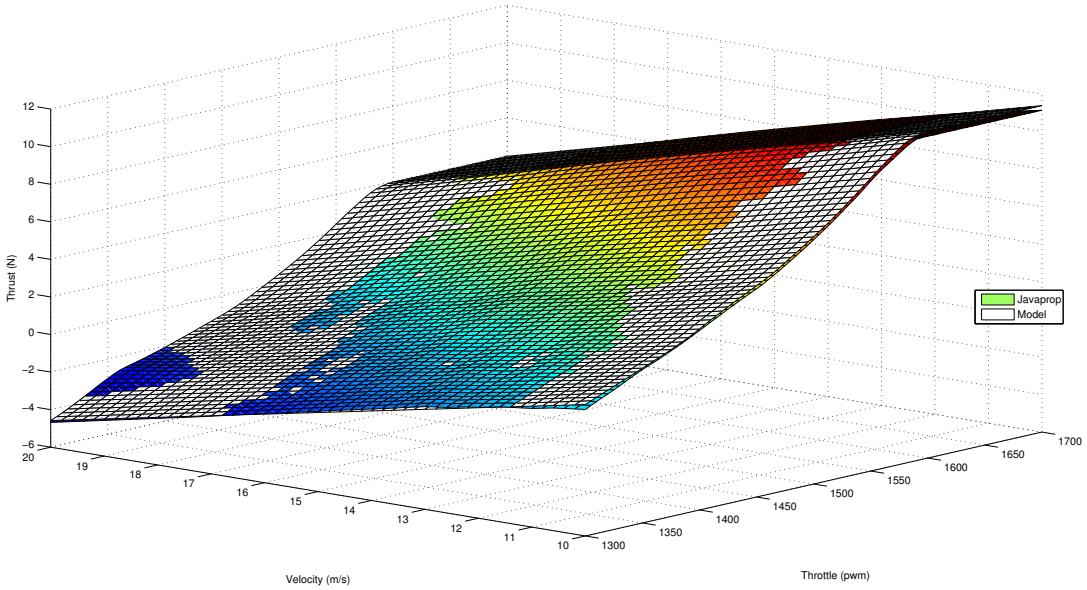
A meshgrid of throttle command values, in PWM, and airflow velocities is created covering the desired flight envelope. The interpolation-based Javaprop model is then called for each throttle and velocity value. By examining the output from the Javaprop model, insight can be gained towards initial model creation. Specifically, the candidate thrust model is of the form:

$$\begin{aligned} f(\delta_T, V_T) =& T_0 + a_1\delta_T + a_2 V_T + a_3\delta_T^2 + a_4 V_T^2 + \sum_{j=1}^{k} b_j \left(\delta_T \geq \delta_{T,j}\right)\left(\delta_T - \delta_{T,j}\right)^2 \\ &+ V_T \sum_{m=1}^{n} c_m \left(\delta_T \geq \delta_{T,m}\right)\left(\delta_T - \delta_{T,m}\right)^2 \end{aligned} \tag{4.29}$$

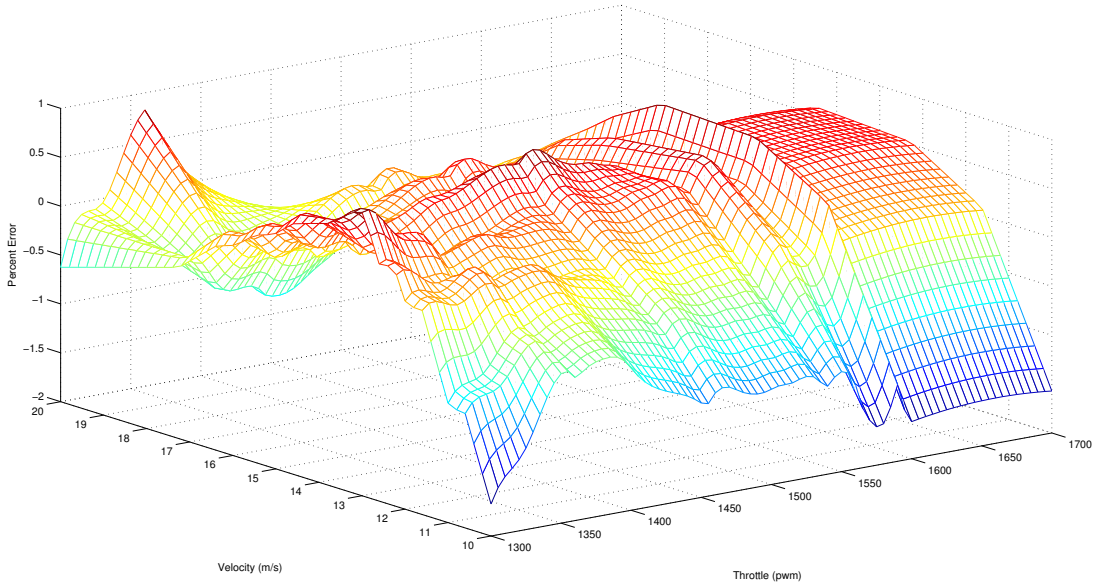
where T_0 is a bias term, $(a_1, a_2, \ldots, a_4)$, $(b_1, b_2, \ldots, b_j)$, and $(c_1, c_2, \ldots, c_j)$ are coefficients to be estimated, and k and n define the number of piecewise polynomial knot points. Stepwise multiple regression is performed in order to determine the relevant independent variables and their associated coefficient values.

The resulting model has a coefficient of determination (R^2) value of 0.9996, with a normalized RMS error of 0.04810%. The thrust predicted by both Javaprop and the model across the flight envelope is given in Fig. 4.7(a), with the percent error over the flight envelope shown in Fig. 4.7(b).

To quantify the improvement in computation time, 200 test points are drawn from a uniform distribution over the throttle and velocity flight envelope bounds, and the CPU clock time for each method to return the thrust prediction is obtained. This process is repeated a total of 100 times. Since the thrust model is vectorized, it is reasonable to expect a decrease in computation time. Comparing the mean computation times for each set, the vectorized thrust model is roughly 900 times faster than performing multivariable interpolation, with the mean computation times being 0.18 seconds for the interpolation based model and 2×10^{-4} seconds for the vectorized model. While the maximum error in predicted thrust is roughly 1.7%, as can be seen in Fig. 4.7(b), such errors are deemed acceptable considering the significant decrease in computation time.



(a) Javaprop and thrust model - predicted thrust.



(b) Percent error - thrust model and Javaprop prediction.

Figure 4.7: Vectorized thrust model.

4.2.2 Reference Trajectories

Desired trajectory motions are obtained through the choice of running cost, path constraints, and boundary conditions. The desired trajectory for the example presented is an aggressive heading angle change that starts and ends at a steady straight and level flight condition. Specifically, a minimum time 180° heading angle change starting and finishing at a specified steady straight and level flight condition is sought. Further, the trajectory should minimize altitude loss between the initial and terminal states. It is obvious that the desired initial and final conditions for the states and controls correspond to the states and controls at the trim flight condition.

The dynamic constraints are given by Eq. 2.5, 2.6 and 2.7. Additionally, to ensure that the aircraft initial and final conditions correspond to a pre-specified steady straight-and-level flight condition, the second-order actuator models are included in the dynamic constraints. Use of second-order models for the actuators allows the running cost to penalize both actuator commands as well as actuator rates. Note that for aerodynamic control surface deflections, the natural frequency and damping ratio from the prior section are used; a second-order system is also used for the commanded throttle input with a natural frequency $\omega_{n,T} = 100$ rad/s and damping ratio $\zeta_T = 0.9$. Thus for trajectory generation, $\mathbf{x} \in \mathbb{R}^{18\times 1}$ is defined as

$$\mathbf{x} = \left[u,\ v,\ w,\ p,\ q,\ r,\ \phi,\ \theta,\ \psi,\ h_g,\ \delta_A,\ \dot{\delta}_A,\ \delta_E,\ \dot{\delta}_E,\ \delta_R,\ \dot{\delta}_R,\ \delta_T,\ \dot{\delta}_T\right]^T.$$

The control inputs, $\boldsymbol{\delta} \in \mathbb{R}^{4\times 1}$, are then defined as $\boldsymbol{\delta} = [\delta_{A,\text{cmd}},\ \delta_{E,\text{cmd}},\ \delta_{R,\text{cmd}},\ \delta_{T,\text{cmd}}]^T$, where the subscript cmd denotes a commanded input.

The path constraint function, $\mathbf{C}(\mathbf{x}(t), \boldsymbol{\delta}(t))$, restricts the aircraft to operate within a desired flight envelope throughout the trajectory. Specifically, airspeed, angle of attack, and angle of sideslip are constrained, and thus the path constraint function is given as

$$\mathbf{C}(\mathbf{x}(t)) = \begin{bmatrix} \sqrt{u^2+v^2+w^2} \\ \tan^{-1}\left(\frac{w}{u}\right) \\ \sin^{-1}\left(\frac{v}{\sqrt{u^2+v^2+w^2}}\right) \end{bmatrix}, \tag{4.30}$$

with lower and upper bounds, $\mathbf{C}_L$ and $\mathbf{C}_U$ respectively, selected depending on the type of reference trajectory desired. To ensure that the reference trajectory begins and ends at the specified trim condition, the boundary constraint function uses inequality constraints on each system state, i.e. $\mathbf{B}(\mathbf{x}(t)) \in \mathbb{R}^{18\times 1}$.

Over the expected operational altitude for the aircraft, density does not vary significantly and thus altitude does not influence the dynamics. Additionaly, it is clear from Eq. 2.5, 2.6 and 2.7 that the dynamic and kinematic equations (excluding the X_g and Y_g inertial positions) are not functions of ψ, and thus both the final altitude and final heading angle can be selected freely.

At a steady straight and level flight condition, it is clear that actuator derivative states must be zero, as well as the angular rates (p, q, r). The remaining states are determined through solution of a constrained nonlinear optimization problem. Specifically, the MATLAB func-

tion `fmincon` is used to minimize a cost function penalizing the derivatives of the equations of motions. The optimization variables are airspeed, angle of attack, sideslip, control surface deflections, Euler angles, and commanded thrust. This trim condition will serve as the desired initial and final conditions for the reference trajectory. It is then straightforward to determine the body-axis velocities at the trim condition based on the following relationships to the airspeed, angle of attack, and sideslip:

$$\begin{aligned} u_{\text{Trim}} &= V_{\text{Trim}} \cos\alpha_{\text{Trim}} \cos\beta_{\text{Trim}} \\ v_{\text{Trim}} &= V_{\text{Trim}} \sin\beta_{\text{Trim}} \\ w_{\text{Trim}} &= V_{\text{Trim}} \sin\alpha_{\text{Trim}} \cos\beta_{\text{Trim}} \end{aligned} \tag{4.31}$$

Denoting the reference trim condition as $\mathbf{x}_{\text{trim}} \in \mathbb{R}^{18\times 1}$, it is clear that the desired initial condition for the optimization problem is $\mathbf{x}(t_0) = \mathbf{x}_{\text{trim}}$. The terminal condition is expressed as

$$\mathbf{x}(t_f) = \left[\mathbf{x}_{\text{trim}}(1:8)^T,\ \psi(t_f),\ h_g(t_f),\ \mathbf{x}_{\text{trim}}(11:18)^T\right]^T.$$

The example trajectory derived is a short 180° heading angle change minimizing deviation in altitude and airspeed at the initial and final states in addition to penalizing the total time to complete the maneuver.

The 180° heading angle change trajectory obtained using GPOPS-II is given in Fig. 4.8. For

this trajectory, the following cost function is used:

$$\begin{aligned} J =& 20 \cdot t_f + 5\left(z(t_0) - z(t_f)\right)^2 + \left(V(t_0) - V(t_f)\right)^2 + \\ & \int_{t_0}^{t_f} \dot{\delta}_T^2 + \dot{\delta}_R^2 + 10 \cdot \left(\delta_R - \delta_{R,\text{trim}}\right)^2 + 1/2\left(\dot{p}^2 + \dot{q}^2 + \dot{r}^2\right) dt \end{aligned} \tag{4.32}$$

The cost function in Eq. 4.32 is selected after several preliminary iterations. Specifically, the total time and deviations in altitude and airspeed from the initial condition are penalized. The integrand penalizes both throttle rate and rudder deflection rate, in addition to heavily penalizing the deviation of the rudder from the trim position. The penalizing of angular accelerations is included after initial examination in order to obtain smoother angular rate histories. Note that for this trajectory, the control surfaces are constrained to an at most deflection of $\pm 75\%$ of their respective maximum deflections; these constriants are imposed to avoid input saturation in the reference trajectory.

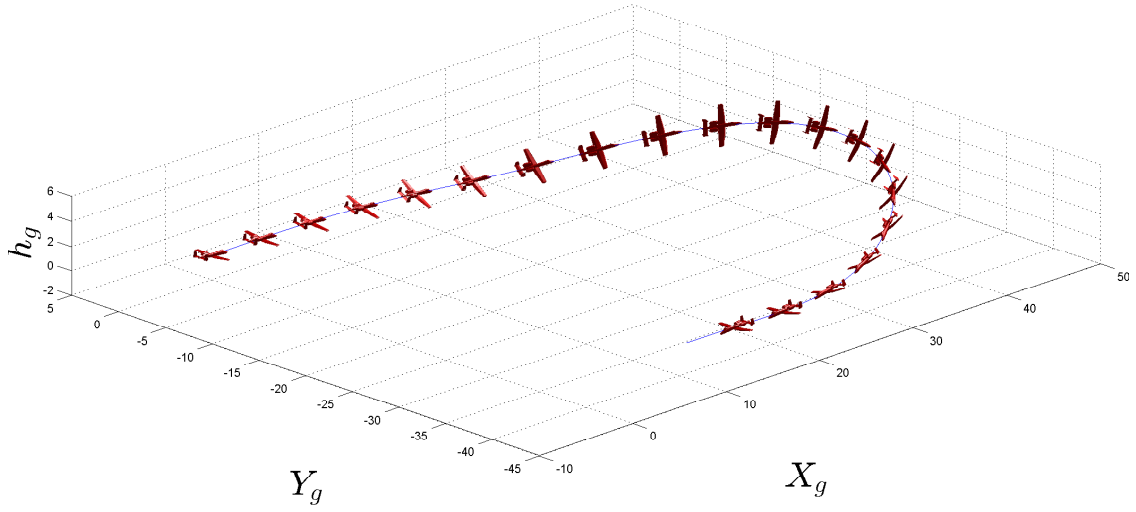


Figure 4.8: Aggressive 180° heading angle change trajectory

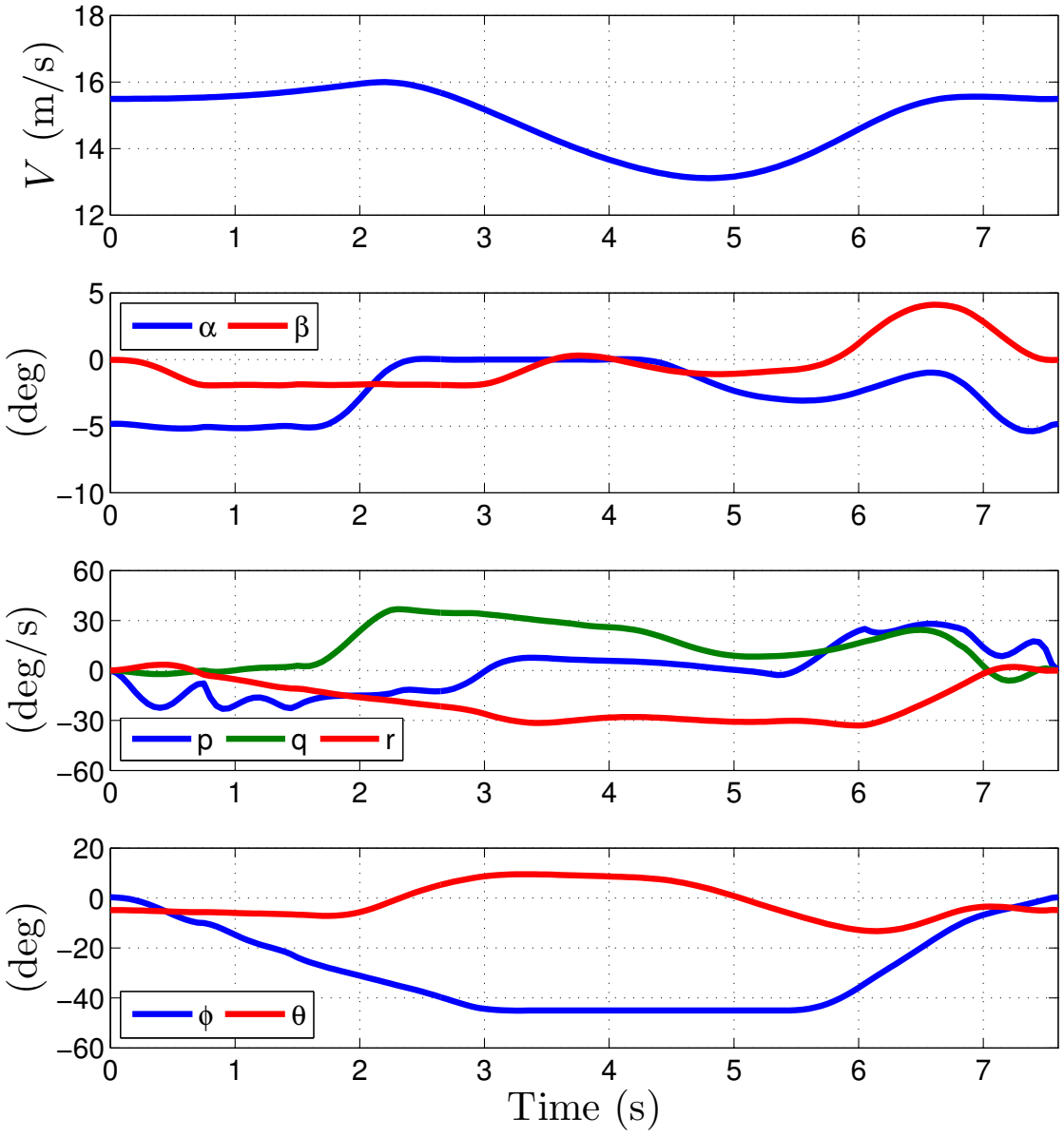


Figure 4.9: State histories for 180° high bank angle trajectory.

4.3 Feedback Control

The controllers developed to regulate the aircraft about the reference trajectory utilize the results developed in [45, 46]. The reference trajectory is composed of the banking maneuver followed by a straight and level trim trajectory. As stated in Section 2.4.1, the system is modeled as a finite horizon LTV system during the banking maneuver and an LTI system about the desired final state. Thus, the linearized system in this case is eventually time-invariant, following the notation of [45].

The state vector is defined as $\mathbf{x} = [u, v, w, p, q, r, \phi, \theta, \psi, h_g, X_g, Y_g]^T$, and the input vector is defined as $\boldsymbol{\delta} = [\delta_A, \delta_E, \delta_R, \delta_T]^T$. It is assumed that the following measurements are available at a sampling rate of 20Hz:

$$\mathbf{y} = \mathbf{h}(\mathbf{x}, \mathbf{d}) = [V_T, \alpha, \beta, p, q, r, \phi, \theta, \psi, h_g, X_g, Y_g]^T + D_{21c}\mathbf{d}, \tag{4.33}$$

where α and β are as defined in Eq. 2.10 and 2.11, respectively, and $\mathbf{d} = \left[\mathbf{d}_a^T, \mathbf{d}_m^T\right]^T$ denotes the exogenous disturbances, which include the atmospheric disturbances $\mathbf{d}_a$ that affect the reference dynamic pressure, angle of attack, and sideslip, as detailed in [100], as well as the measurement noise $\mathbf{d}_m$. The eventually time-invariant plant model is derived by linearizing the nonlinear system equations, $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \boldsymbol{\delta}, \mathbf{d})$, $\mathbf{y} = \mathbf{h}(\mathbf{x}, \mathbf{d})$, about the reference trajectory $(\mathbf{x}_r, \boldsymbol{\delta}_r, \mathbf{d}_r)$, where $\mathbf{d}_r = 0$. Specifically, under the assumption that the input and state approximately follow the reference input and state trajectories, the errors $\bar{\mathbf{x}} = \mathbf{x} - \mathbf{x}_r$,

$\bar{\boldsymbol{\delta}} = \boldsymbol{\delta} - \boldsymbol{\delta}_r$, and $\bar{\mathbf{y}} = \mathbf{y} - \mathbf{y}_r$ will be small enough to satisfy the following state-space equations:

$$\begin{aligned} \dot{\bar{\mathbf{x}}} =& A_c(t)\bar{\mathbf{x}}(t) + B_{1c}(t)\mathbf{d}(t) + B_{2c}(t)\bar{\boldsymbol{\delta}}(t), \\ \bar{\mathbf{y}} =& C_{2c}(t)\bar{\mathbf{x}}(t) + D_{21c}(t)\mathbf{d}(t) \end{aligned} \tag{4.34}$$

where $A_c(t) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}|_{(\mathbf{x}_r, \boldsymbol{\delta}_r, \mathbf{d}_r)}$, $B_{1c}(t) = \frac{\partial \mathbf{f}}{\partial \mathbf{d}}|_{(\mathbf{x}_r, \boldsymbol{\delta}_r, \mathbf{d}_r)}$, $B_{2c}(t) = \frac{\partial \mathbf{f}}{\partial \boldsymbol{\delta}}|_{(\mathbf{x}_r, \boldsymbol{\delta}_r, \mathbf{d}_r)}$ and $C_{2c}(t) = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}|_{(\mathbf{x}_r, \mathbf{d}_r)}$, with $\mathbf{d}_r = 0$ as aforementioned.

A corresponding discrete-time model is obtained utilizing zero-order hold sampling, with sampling time $T = 0.05$sec, and given as:

$$\bar{\mathbf{x}}_{k+1} = A_k\bar{\mathbf{x}}_k + B_{1k}\mathbf{d}_k + B_{2k}\bar{\boldsymbol{\delta}}_k \tag{4.35}$$

where $\bar{\mathbf{x}}_k = \bar{\mathbf{x}}(kT)$, $\bar{\boldsymbol{\delta}}_k = \bar{\boldsymbol{\delta}}(kT)$ for all integers $k \geq 0$, $A_k = \Phi((k+1)T, kT)$, with $\Phi(\cdot, \cdot)$ being the state transition matrix, and, for $i = 1, 2$,

$$B_{ik} = \int_{kT}^{(k+1)T} \Phi((k+1)T, \tau)B_{ic}(\tau)d\tau \ . \tag{4.36}$$

The discrete-time system matrices for the measurement error are equal to their continuous-time counterparts, i.e. $C_{2k} = C_{2c}(kT)$ and $D_{21k} = D_{21c}(kT)$. The discrete-time matrix C_{2k} varies with time, while D_{21k} does not, and thus is hereafter denoted as D_{21}.

Thus, the resulting discrete-time h-eventually time-invariant plant model, denoted by G, is

defined by the following state-space equations:

$$\begin{bmatrix} \bar{\mathbf{x}}_{k+1} \\ \mathbf{z}_k \\ \bar{\mathbf{y}}_k \end{bmatrix} = \begin{bmatrix} A_k & B_{1k} & B_{2k} \\ C_1 & D_{11} & D_{12} \\ C_{2k} & D_{21} & D_{22} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{x}}_k \\ \mathbf{d}_k \\ \bar{\boldsymbol{\delta}}_k \end{bmatrix}, \ \bar{\mathbf{x}}_0 = 0 \tag{4.37}$$

where the finite horizon length $h = 153$ is the number of sampling points (at 20Hz sampling rate) along the banking maneuver obtained in the prior section, and A_k, B_{1k}, B_{2k}, C_{2k}, and D_{21} are as previously defined. Additionally, $\mathbf{z}_k$ denotes the exogenous error at discrete time k. Further, the matrices C_1 and D_{12} are used to assign relative weights to the error states and feedback control inputs, while the D_{11} and D_{22} matrices are all zeros.

There are three controllers that are analyzed in both simulation and flight testing. The elements of the matrix D_{21} are chosen based upon the sensor noise characteristics of the system. That is, D_{21} is block diagonal in structure, with the diagonal terms corresponding to the standard deviations of the sensor measurements. The D_{21} matrix is the same for all three controllers, and chosen as

$$D_{21} = \text{diag}(0_{3\times3}, 0.05, 0.0471, 0.0227, 0.2I_{3\times3}, 0.0349I_{3\times3}, 1.33, 0.83I_{2\times2})$$

The C_1 and D_{12} matrices are formulated to penalize certain states for trajectory tracking purposes as well as specific control inputs. The error states chosen to be penalized in the

case of the first controller are u, p, q, r, ϕ, θ, ψ, h_g, δ_A, δ_E, δ_R, and δ_T. The C_1 and D_{12} matrices are selected as

$$C_1 = \text{diag}(0.25, 0.1I_{3\times3}, 0.143I_{3\times3}, 0.01, 0_{4\times4}) \cdot 10^{-2}$$

$$D_{12} = [0_{8\times4} \;\; \text{diag}(0.0588, 0.0417, 0.0588, 0.0100)]^T$$

In the case of the second controller, the position Y_g is also penalized. Additionally, the control inputs are penalized in conjunction with certain states. Specifically, the performance output in this case is defined as

$$\mathbf{z} = \left[u + 20\delta_T, p + 10\delta_A, q + 10\delta_E, r + 10\delta_R, 0.1\phi, 0.1\theta, 0.1\psi, 5h_g, 1 \times 10^{-3}Y_g\right]^T \tag{4.38}$$

The C_1 and D_{12} matrices corresponding to Eq. 4.38 are defined as

$$C_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \times 10^{-3} & 0 \end{bmatrix}, \; D_{12} = \begin{bmatrix} 0 & 0 & 0 & 20 \\ 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.39}$$

From Eq. 4.38 and 4.39, the penalty on the u body-axis velocity is penalized in conjunction with the throttle input, p with δ_A, q with δ_E, and r with δ_R.

The third controller is similar to the second controller, except that X_g tracking is penalized instead of Y_g. The exogenous errors for this controller are defined as

$$\mathbf{z} = \left[u + 10\delta_T, p + 10\delta_A, q + 10\delta_E, r + 10\delta_R, 0.1\phi, 0.1\theta, 0.1\psi, 1h_g, 1 \times 10^{-2} X_g\right]^T \tag{4.40}$$

Note that in comparison to Eq. 4.38, the exogenous errors in Eq. 4.40 have lower weighting on both δ_T and h_g. These weights are decreased in order to attain improved tracking of X_g. The formulation of the C_1 and D_{12} matrices for the exogenous errors in Eq. 4.40 follows that of Eq. 4.38 and 4.39.

The control synthesis problem, as given by Eq. 2.24, is a convex optimization problem and is solved using YALMIP [93] along with SeDuMi [75]. The solutions R_k, S_k, and γ can then be used to construct a controller, as detailed in [101, 74, 102]. All computations are carried out in MATLAB on a Lenovo W530 laptop with a 2.30 GHz quad-core Intel i7 CPU and 20 GB of RAM running Windows 8. A solution is obtained for the first controller with minimum achievable $\gamma = 0.0251$. In order to obtain satisfactory controller performance for simulation and flight test purposes, γ is relaxed to 0.05. Solutions are obtained for the second and third controllers with minimum achievable $\gamma = 1.2444$ and $\gamma = 0.7489$, respectively. Again, the γ values are relaxed to 1.4 and 0.9, for the second and third controllers, respectively.

4.3.1 Simulation

Controller performance is assessed through simulation in a realistic operational environment. Specifically, the system is subjected to measurement noise, atmospheric turbulence, and constant direction and magnitude wind fields. Additionally, the actuator dynamics, which are not accounted for in the control design process, are included in the nonlinear simulation. In these simulations, the reference trajectory after completing the banking maneuver has an additional 25.35sec of steady straight and level flight.

Atmospheric turbulence is modeled according to the Dryden turbulence model as descibed in [103, 104]. The Dryden model generates a time history of turbulence velocities in the aircraft body-axis reference frame based on the aircraft velocity, altitude, and a reference wind speed. Guidelines for selecting the reference wind speed to obtain light, moderate, and severe turbulence disturbances are given in [103]. Simulations are used to provide an initial assessment of controller performance. The measurement vector available for determining feedback input corrections is as given in Eq. 4.33. All measurement noise is assumed to be zero-mean Gaussian with standard deviations as specified in Table 4.1.

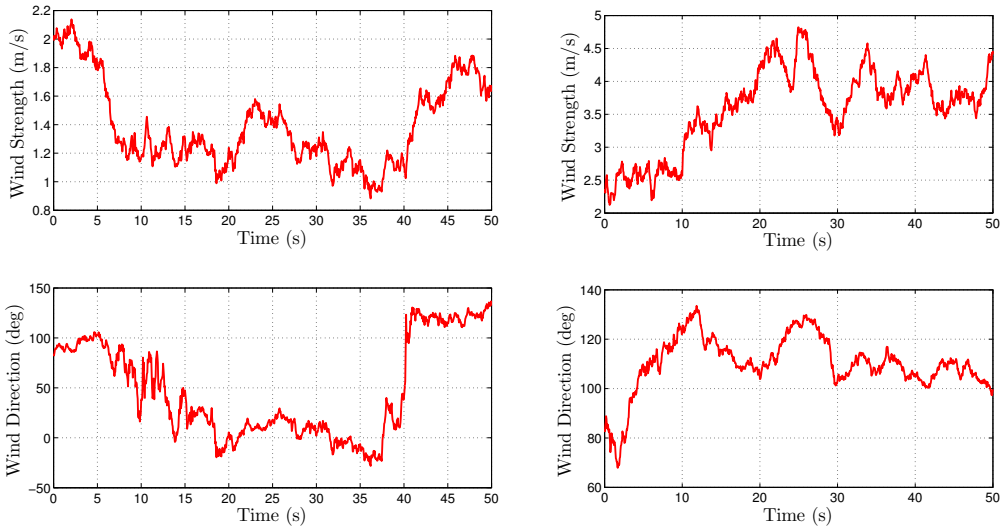
Figure 4.10 gives an example of the total wind disturbance magnitude and direction for light and moderate turbulence cases with a 2m/s constant cross wind (90° direction from the X_g axis). During the preliminary controller design, both light and moderate turbulence cases are utilized to assess controller performance. Specifically, a variety of disturbance scenarios are used to analyze the effects of changes in both performance output structure

and relative weighting of the states in the performance output. For the simulation analysis, the same measurement noise and atmospheric turbulence are used for simulation with all three controllers. The plots presented show controller performance with light turbulence and a 2 m/s constant wind oriented at 90° direction from the X_g axis.

The tracking performance for 20 simulation flight tests using the first controller, which does not penalize the X_g and Y_g positions, is given in Fig. 4.11. Note that in each of these plots, the reference trajectory is given by the black curve. From the attitude histories given in Fig. 4.11(b), it is clear that the controller tracks the reference trajectory very well with the exception of a single simulation, where there is a departure from the reference trajectory during the banking maneuever. From Fig. 4.11(c), it is evident that the departure from the reference trajectory is due to a disturbance in the roll rate, p. The remainder of the simulations have very good tracking for the angular rate histories. The control inputs, which include the reference control history and feedback corrections, are shown in Fig. 4.11(d). The feedback control commands are not excessive, with the exception of the rudder input during the same simulation with the aforementioned large deviations in the attitude and angular rate histories. Note that despite the errors occurring in the roll and pitch rates prior to the errors in the other states, specifically ψ, the controller commands a large rudder deflection. In Fig. 4.11(a), the position tracking performance is given. While the altitude error, Δh_g, remains small, the deviations from the X_g and Y_g positions are clearly significant.

The tracking performances for the same 20 simulation flight tests using the second and

third controllers, with exogenous errors given by Eq. 4.38 and 4.40, respectivley, are given in Fig. 4.12 and 4.13. The attitude tracking errors in both cases are shown in Fig. 4.12(a) and 4.12(b). Both the bank angle (ϕ) and heading angle (ψ) histories for these controllers have degraded performance in comparison to the first controller. This is expected, however, as the aircraft must perform rolling and yawing motions in order to correct the error in the Y_g position history. Between these two controllers, the third controller admits larger bank angle excursions in order to correct the position. The angular rate tracking histories are given in Fig. 4.12(c) and 4.12(d). Note that in comparison to Fig. 4.11(c), the same measurement errors and atmospheric disturbances histories do result in a large excursion from the reference trajectory. The feedback control inputs for the second and third controllers are given in Fig. 4.13(a) and 4.13(b), respectively. Note that unlike the first controller, the rudder control surface does not saturate. Additionally, the third controller has significantly large throttle feedback inputs in order to track the X_g position. The position errors are shown in Fig. 4.13(c). While the errors in altitude, Δh_g, are larger for the third controller than those for the second controller, the X_g tracking is significantly improved in the case of the third controller. Additionally, despite not penalizing the Y_g position, the third controller has tracking performance of Y_g that is comparable to the second controller.



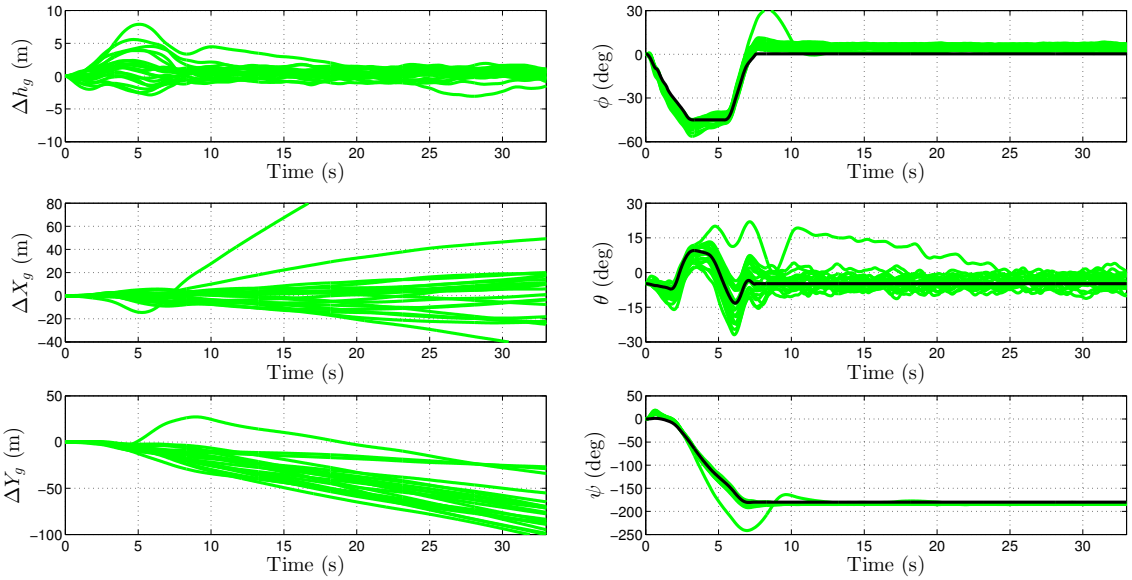
(a) Wind disturbance components - light turbulence.

(b) Wind disturbance components - moderate turbulence.

Figure 4.10: Total wind disturbances - strength and direction.

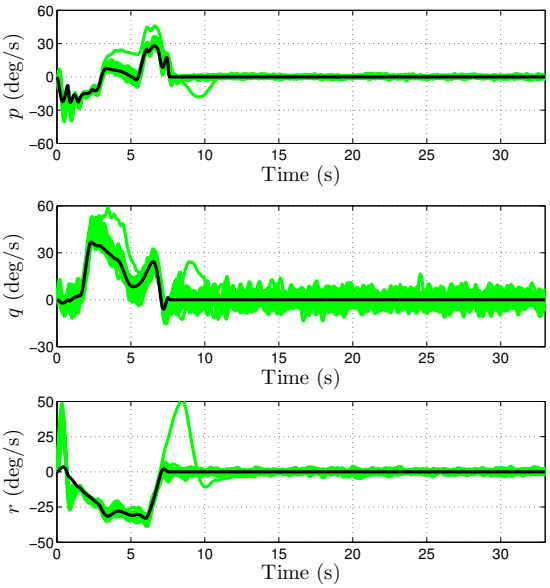
4.3.2 Flight Testing

The controllers from the previous section are also analyzed for performance in flight on the Telemaster platform. Details regarding the hardware and software implementation of the discrete-time feedback controllers are given in [105]. Four flight tests with the first controller are given in Fig. 4.14. These tests are performed in calm wind conditions, with approximately 0.5 m/s constant wind at runway level in the East-Southeast direction measured using a handheld anemometer. The tracking peformance of the angular rates and attitude angles is given in Fig. 4.14(a) and 4.14(b), respectively, and the position tracking is shown in Fig. 4.14(c). Although the controller tracks the desired angular rates, attitude angles, and altitude very well, the X_g and Y_g position errors are rather poor for two of the tests. The

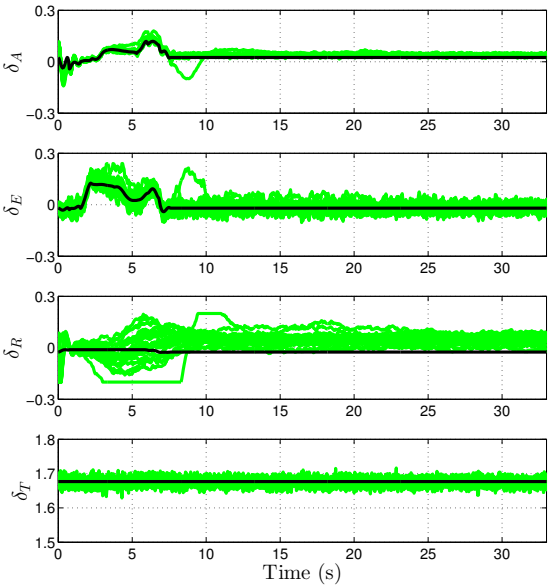


(a) Position errors.

(b) Attitude history.

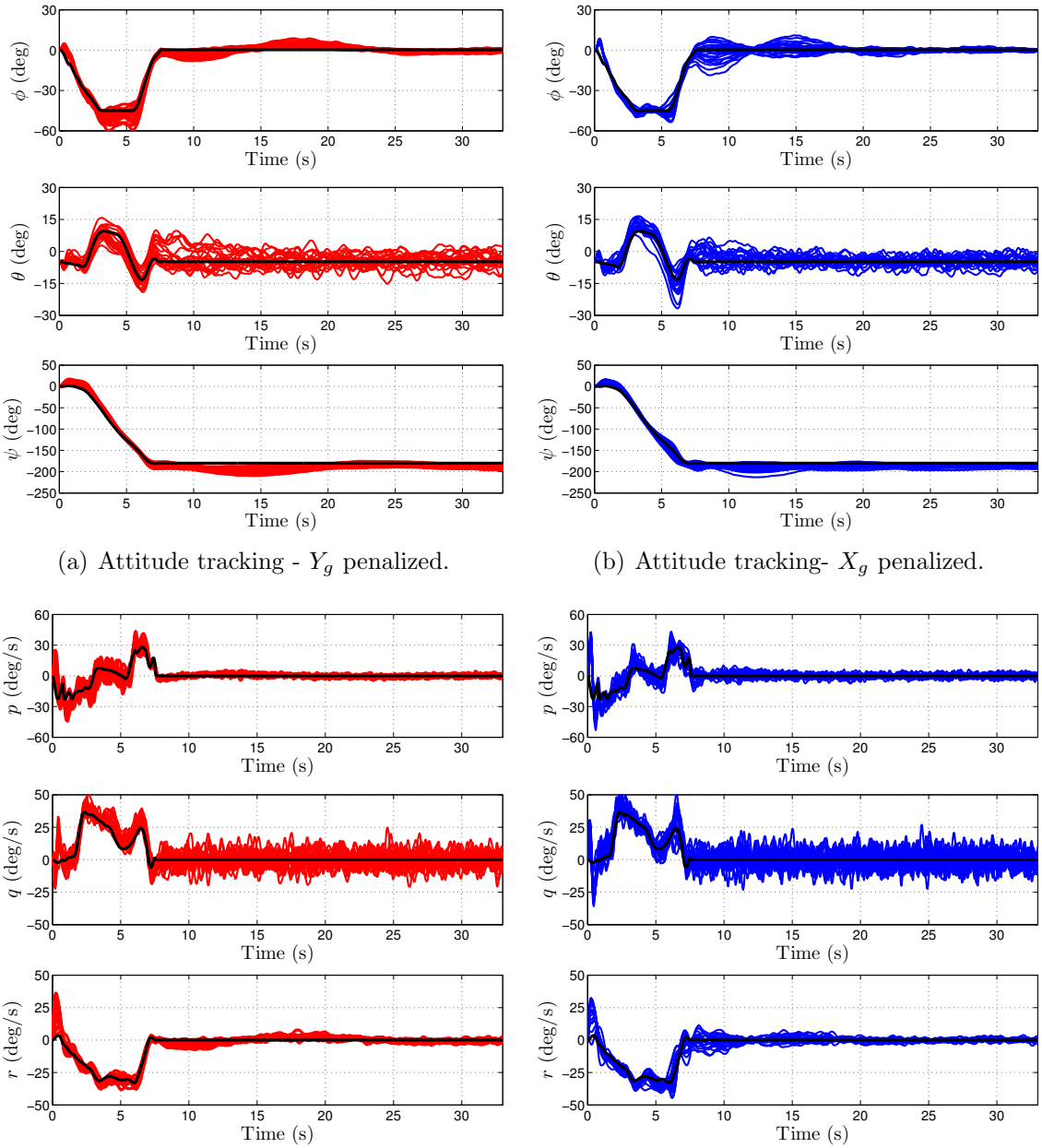


(c) Angular rates.



(d) Feedback inputs.

Figure 4.11: Trajectory tracking performance without X_g and Y_g in the exogenous errors.



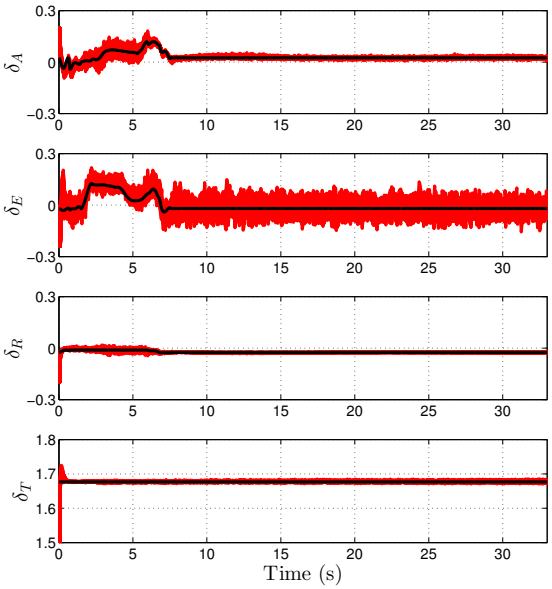
(a) Attitude tracking - Y_g penalized.

(b) Attitude tracking- X_g penalized.

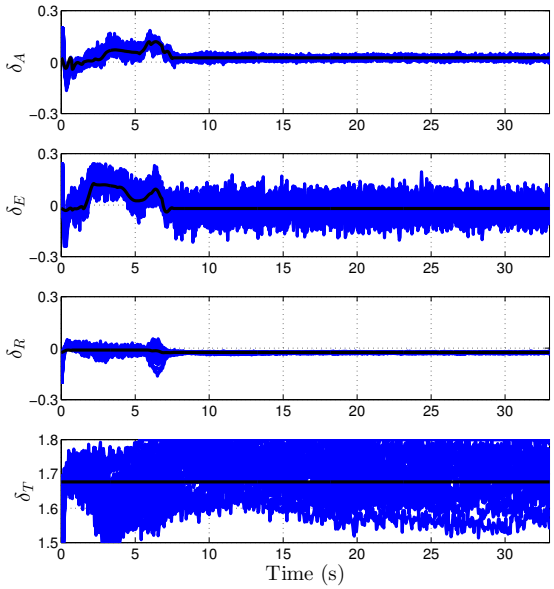
(c) Angular rate tracking- Y_g penalized.

(d) Angular rate tracking- X_g penalized.

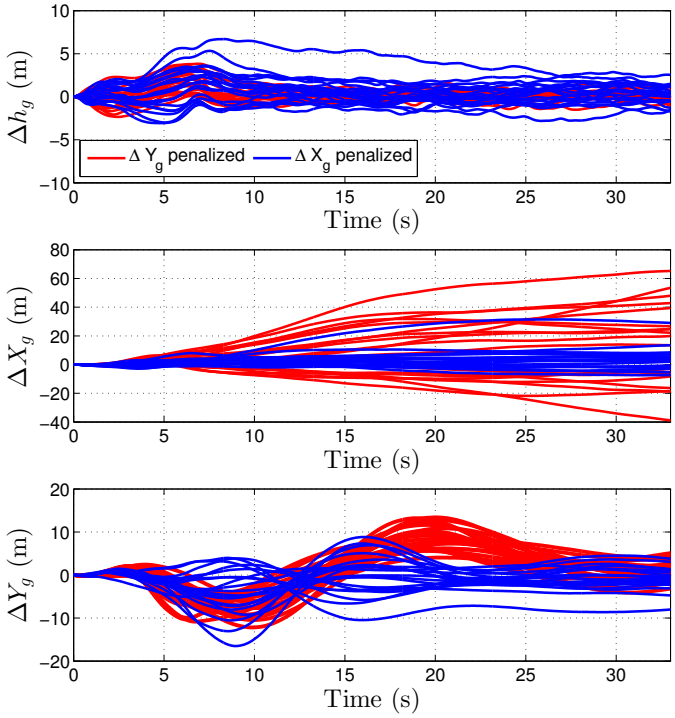
Figure 4.12: Tracking performance for attitude angles and angular rates with inertial position penalties.



(a) Feedback inputs - Y_g penalized.



(b) Feedback inputs - X_g penalized.

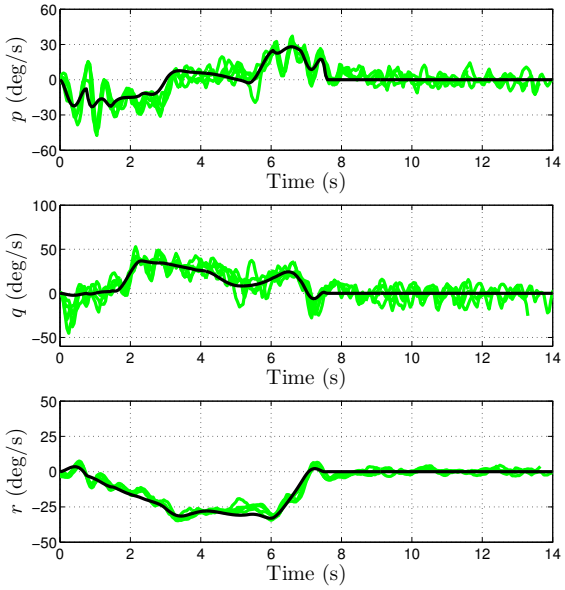


(c) Position errors.

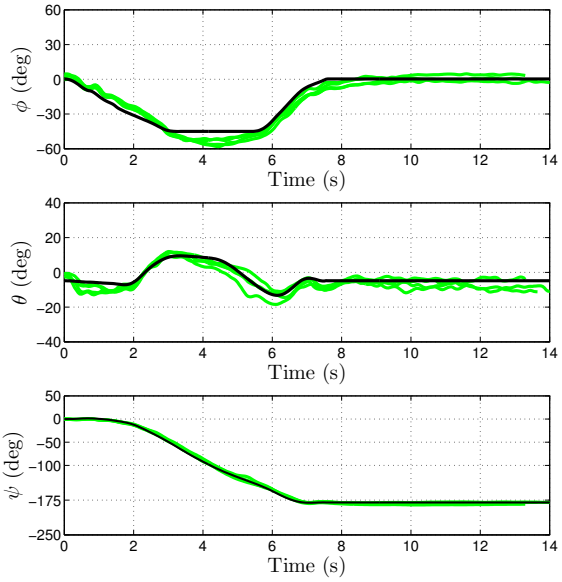
Figure 4.13: Feedback control inputs and positions errors with intertial position penalties.

controller also prompts throttle commands significantly larger than the reference values, as shown in Fig. 4.14(d).

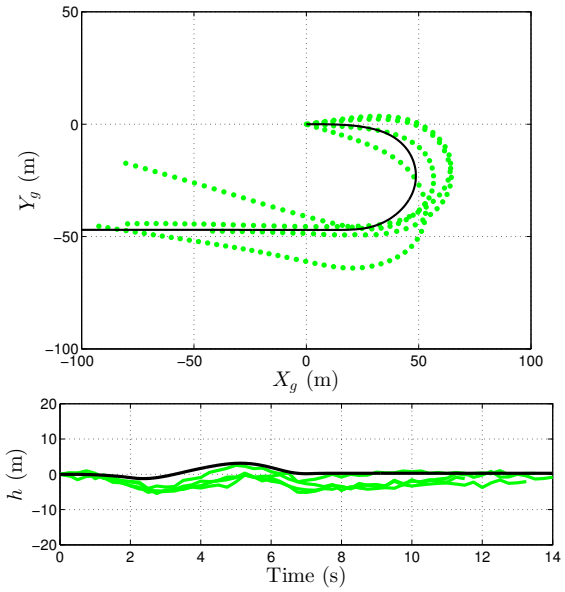
Flight test results for the second and third controllers are given in Fig. 4.15 and 4.16. The second controller was flown seven times and the third controller three times. Also note that the third controller is run for a longer time period in order to analyze its peformance in reducing the error in the X_g position. The second controller is flown on a calm day with constant wind measured using a handheld anemometer at the runway level varying from $0.5 - 0.8$ m/s in the East-Southeast direction; the third controller is also flown on a calm day, however the wind measured at runway level is approximately $0.8-1$ m/s varying between the East and East-Southeast directions. For both of these controllers, the aircraft does not diverge from the reference angular rate and attitude histories as shown in Fig. 4.15(a)-4.15(d). The feedback control inputs for the second and third controllers are given in Fig. 4.16(a) and 4.16(b). While the aileron inputs are comparable between these two controllers, the third controller uses significantly more elevator and throttle corrections, in particular during the straight and level trim trajectory. The result of these applied feedback corrections is that the altitude and X_g position tracking are better with the third controller when compared to the second controller, as evidenced by Fig. 4.16(c) and 4.16(d). Additionally, despite not penalizing the Y_g position, the third controller has comparable tracking peformance to the second controller.



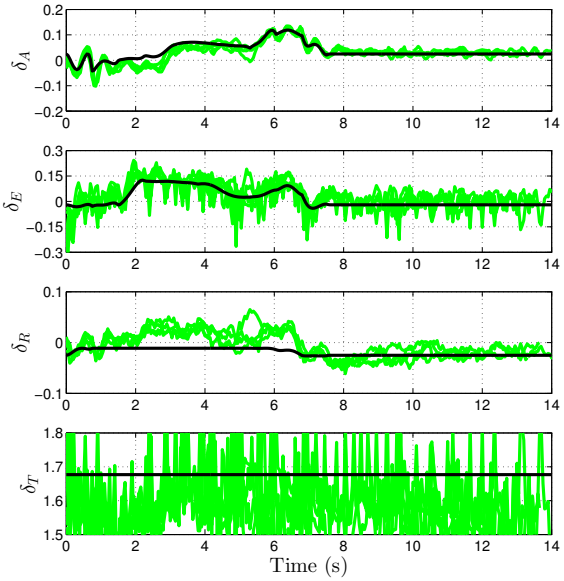
(a) (p, q, r) - no position feedback.



(b) (ϕ, θ, ψ) - no position feedback.

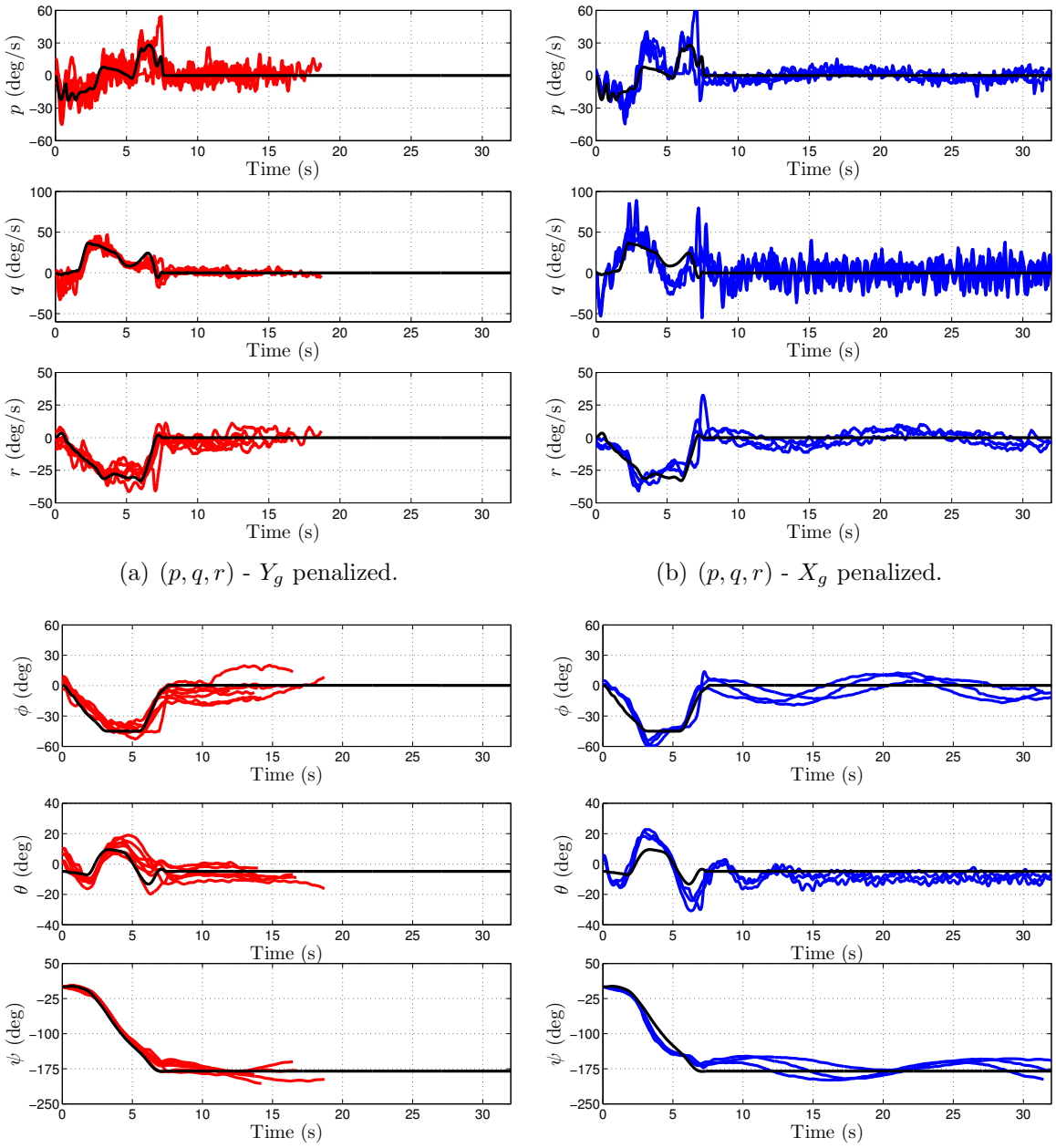


(c) Position error - no position feedback.



(d) Feedback control inputs - no position feedback.

Figure 4.14: Tracking performance from flight test data - no inertial position feedback.



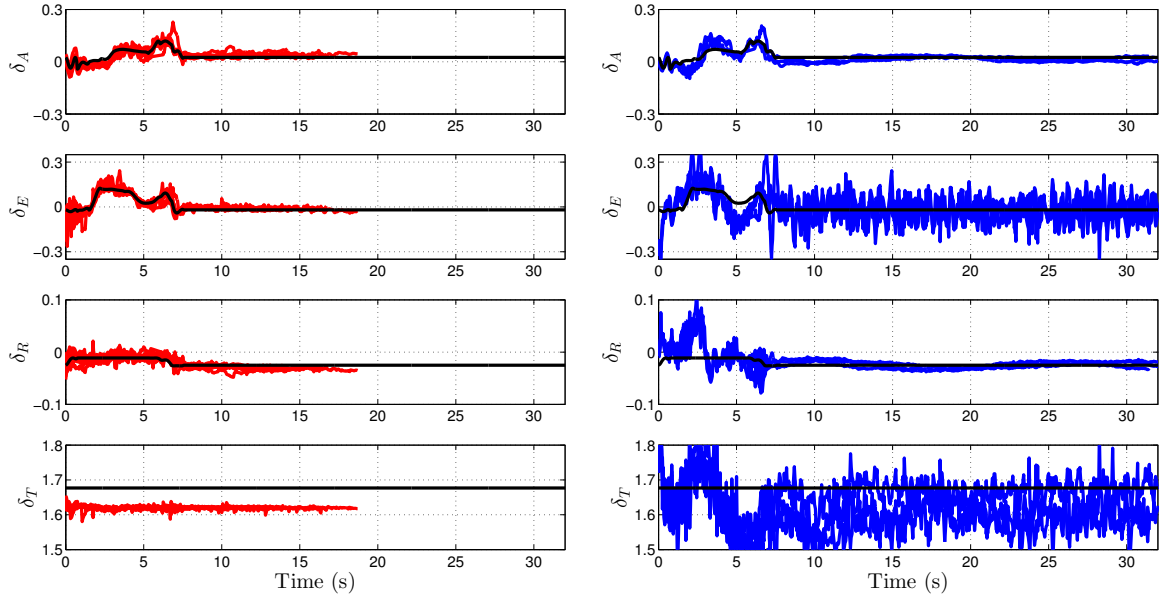
(a) (p, q, r) - Y_g penalized.

(b) (p, q, r) - X_g penalized.

(c) (ϕ, θ, ψ) - Y_g penalized.

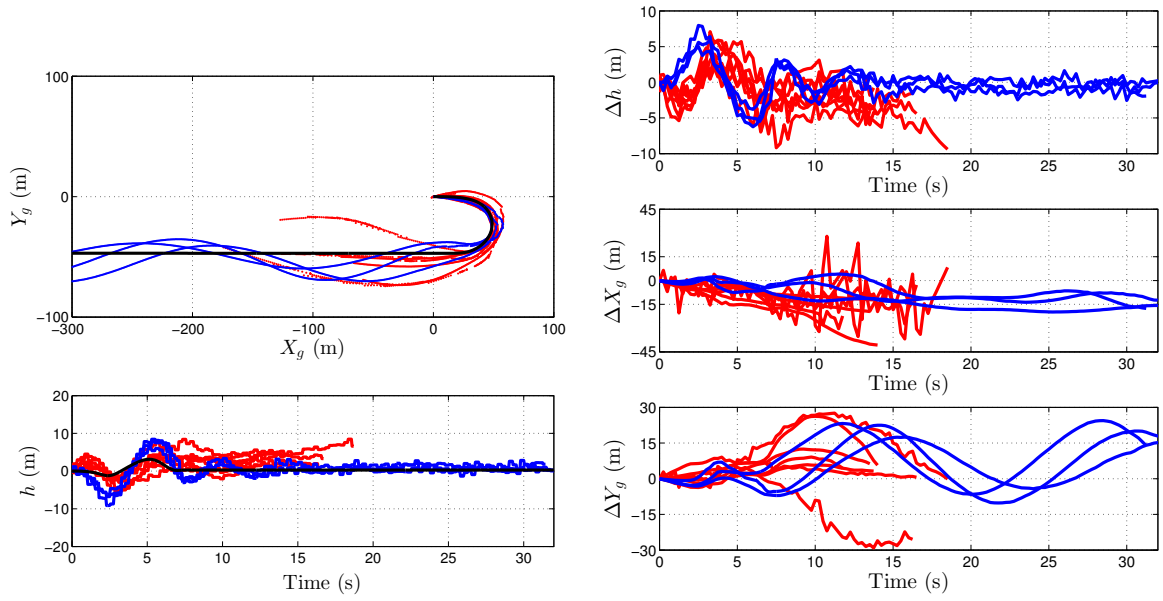
(d) (ϕ, θ, ψ) - X_g penalized.

Figure 4.15: Angular rate (p, q, r) and attitude (ϕ, θ, ψ) tracking from flight tests.



(a) $(\delta_A, \delta_E, \delta_R, \delta_T)$ - Y_g penalized.

(b) $(\delta_A, \delta_E, \delta_R, \delta_T)$ - X_g penalized.



(c) (X_g, Y_g, h)

(d) $(\Delta X_g, \Delta Y_g, \Delta h)$

Figure 4.16: Feedback inputs and position tracking from flight tests.

Chapter 5

Aircraft System Identification from Motion Capture Data

This chapter presents the application of the two-step aerodynamic modeling approach to an unpowered glider flown in a motion capture facility. Though the atmospheric conditions are favorable due to test flights occurring indoors, i.e. there are no wind gusts or turbulence, the particular aircraft and instrumentation system utilized present their own unique challenges. Specifically, the size of the facility and lack of thrust limit all flights to short-duration tests with initial airspeed provided by a slingshot device. Additionally, the only measurements available are the inertial reference frame position and attitude of the vehicle. The airspeed, angle of attack, and angle of sideslip therefore must be estimated from the measurement time history.

There are several assumptions made prior to the estimation of the aerodynamic forces and moments. First, it is assumed that the aircraft is a rigid body; this assumption was made in Chapter 2 during presentation of the equations of motions. Additionally, if the aircraft undergoes a deformation, this can lead to errors in the VICON system measurements of the aircraft's position and orientation. As in Chapter 4, it is also assumed that the curvature of the earth is neglible. Since the aircraft is operating in a controlled indoor test facility, a further assumption can be made that there are no exogenous atmospheric disturbances, i.e. $u_d = v_d = w_d = 0$. Finally, since the aircraft does not undergo changes in altitude of more than a few meters, it is reasonable to assume that variations in density as well as the magnitude of the gravitational vector are negligible for all tests.

5.1 Test Setup

All flight tests were performed in the μ-Aviary facility at the Air Force Research Laboratory, Wright-Patterson Air Force Base, Ohio. The μ-Aviary facility provides position and angular orientation measurements at a sampling rate of up to 150Hz using a VICON motion capture system [109, 110]. Specifically, markers are placed on the object, or multiple objects, of interest and the marker configuration is recorded while the object is stationary. During tests, the VICON cameras determine the location of each marker within the room, and then provide an estimate of the vehicle's position and attitude based on the location of the individual markers in comparison to the original object definition. Figure 5.1 provides an

illustration of the test aircraft with reflective markers.

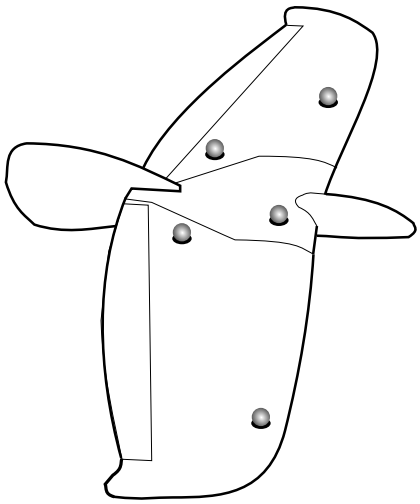


Figure 5.1: Alula glider with marker locations for illustration purposes.

The particular aircraft utilized is the Alula glider, a commercially available EPO foam aircraft produced by the Dream-Flight Corporation, Santa Barbara, CA [108]. The dimensions, mass, and inertial properties for the Alula are given in Table 5.1. All parameters in Table 5.1 were obtained based on the actual aircraft flown. The inertia parameters were computed from a CAD model created using a three-dimensional surface scan of the aircraft performed at AFRL; also included in the mass and inertia is a hook attached to the nose of the aircraft for the slingshot device, which was produced using a rapid prototyping printer.

The aircraft is a flying wing design with a vertical tail for passive yaw stability. The location of the center of gravity along the x_b body-axis can be varied by adding ballast at the battery location in the nose of the aircraft, however, with the addition of the catapult hook, additional ballast was not necessary. The Alula has two active control surfaces that provide combined

Table 5.1: Alula Glider Dimensions

Span	0.892m	I_x	$4.50 \cdot 10^{-3}$kg m^2
Chord	0.2036m	I_y	$1.50 \cdot 10^{-3}$kg m^2
Wing area	0.167m^2	I_z	$5.90 \cdot 10^{-3}$kg m^2
Mass	0.212kg	I_{xz}	$1.35 \cdot 10^{-4}$kg m^2

elevator and aileron control, referred to as elevons. Each elevon surface is individually actuated using digital servomotors with commands sent from a ground station computer.

The servomotors are modeled as second-order systems with a constant time delay and are given by the transfer function

$$\frac{\delta_i(s)}{\delta_{i,c}(s)} = \frac{\omega_n^2 e^{-\tau_d s}}{s^2 + 2\zeta\omega_n s + \omega_n^2}, \quad \text{for } i = L, R, \tag{5.1}$$

where $\delta_{i,c}$ is the commanded deflection and δ_i is the actual deflection. The natural frequency, damping ratio, and time-delay were determined experimentally and found to be $\omega_n = 60$ rad/s, $\zeta = 1$, and $\tau_d =$ 0.035 sec, respectively, as reported in [54].

5.1.1 Flight Testing & Data Pre-Processing

Though a large number of test flights were performed throughout the program [54, 106, 107], only a specific subset of tests are utilized for system identification purposes presented here. Over the course of the various tests performed, a number of aircraft with slightly different configurations (e.g. ballast, damage from crashes) were used. To eliminate any variation in

mass and inertial properties between tests, all flights utilized for modeling are from flights performed using a single aircraft.

It should also be noted that these flight tests were not specifically intended for system identification purposes; commanded control surface deflections were computed using a proportional-integral-derivative (PID) control law to maintain a commanded altitude, and resulted in varying amplitude and frequency control surface deflection histories. Though a total of 42 flights were performed on this specific day, not all tests are used for system identification.

The measurements provided by the VICON motion capture system are the sample time, inertial position (X_g, Y_g, z_g), Euler angle orientation (ϕ, θ, ψ), and commanded control surface deflections (δ_L, δ_R). Due to the size limitations of the test facility, flights frequently terminate in a collision with either the protective netting or wall. To avoid force contributions from both the launch mechanism and these collisions, each flight is analyzed for abrupt changes in the inertial position and orientation angle histories. An example of processed raw data to remove exogenous force effects is given in Fig. 5.2, where "Flight" indicates the portion of the time history that will be used for modeling purposes. Additionally, there are several flights which resulted in an impact with the ground shortly after leaving the launcher; these flights will not be utilized for system identification.

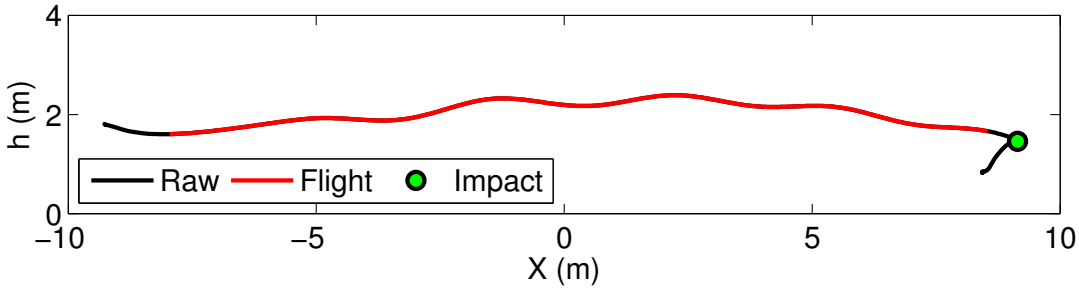


Figure 5.2: Inertial downrange and altitude time history from VICON motion capture system.

5.2 Force & Moment Estimation

In order to apply the TSM for system identification from motion capture data, it is first necessary to estimate the aerodynamic forces and moments acting on the aircraft during flight tests. As previously mentioned, prior works estimating aerodynamic forces and moments from motion capture data utilize numerical filtering and differentiation of inertial position and orientation measurements [50, 49, 54, 53]. In contrast, the approach presented here demonstrates that the time histories of the aerodynamic forces and moments can be estimated using an EKF.

5.2.1 Smoothing & Numerical Differentiation

A straight-forward approach to estimate the aerodynamic force and moment histories utilizes Savitzky-Golay filters for smoothing and differentiation of the position and orientation

measurements [55]. Specifically, the measurements are first filtered and then differentiated in order to compute the body-axis velocities and angular rates. After differentiation of the body-axis velocity and angular rate histories, the force and moment histories are then computed according to the equations of motion.

The inertial velocity components of the aircraft in the Earth-fixed reference frame are given as

$$\left\{ \begin{array}{c} \dot{X}_g \\ \dot{Y}_g \\ \dot{z} \end{array} \right\} = \mathbf{R}_\phi \mathbf{R}_\theta \mathbf{R}_\psi \left\{ \begin{array}{c} u \\ v \\ w \end{array} \right\},$$

with $\mathbf{R}_\phi$, $\mathbf{R}_\theta$, and $\mathbf{R}_\psi$ as defined in Chapter 2. Let k denote the index of the sampling time for each test, such that $t = kT$, where T is the sampling time. The time histories of the body-axis velocities, $(u(k), v(k), w(k))$, are then computed from the derivatives of the measured positions, $(\dot{X}_g(k), \dot{Y}_g(k), \dot{z}(k))$, as

$$\left\{ \begin{array}{c} u(k) \\ v(k) \\ w(k) \end{array} \right\} = \mathbf{R}_\psi(k) \cdot \mathbf{R}_\theta(k) \cdot \mathbf{R}_\phi(k) \left\{ \begin{array}{c} \dot{X}_g(k) \\ \dot{Y}_g(k) \\ \dot{z}_g(k) \end{array} \right\}. \tag{5.2}$$

Similarly, the numerical derivatives of the Euler angle histories are used to compute estimates

of the body-axis angular rates as

$$\begin{bmatrix} p(k) \\ q(k) \\ r(k) \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi}(k) \\ \dot{\theta}(k) \\ \dot{\psi}(k) \end{bmatrix}, \tag{5.3}$$

where the matrix in Eq. 5.3 is the inverse of that in Eq. 2.8. Note that during all flights utilized for aerodynamic modeling, $|\theta(k)| \leq 45°$, and thus the Euler angle representation does not encounter gimbal lock (at $\theta = \pm 90°$). The estimates of $(u(k), v(k), w(k))$ and $(p(k), q(k), r(k)$ are then differentiated to obtain the body-axis accelerations and angular accelerations, respectively.

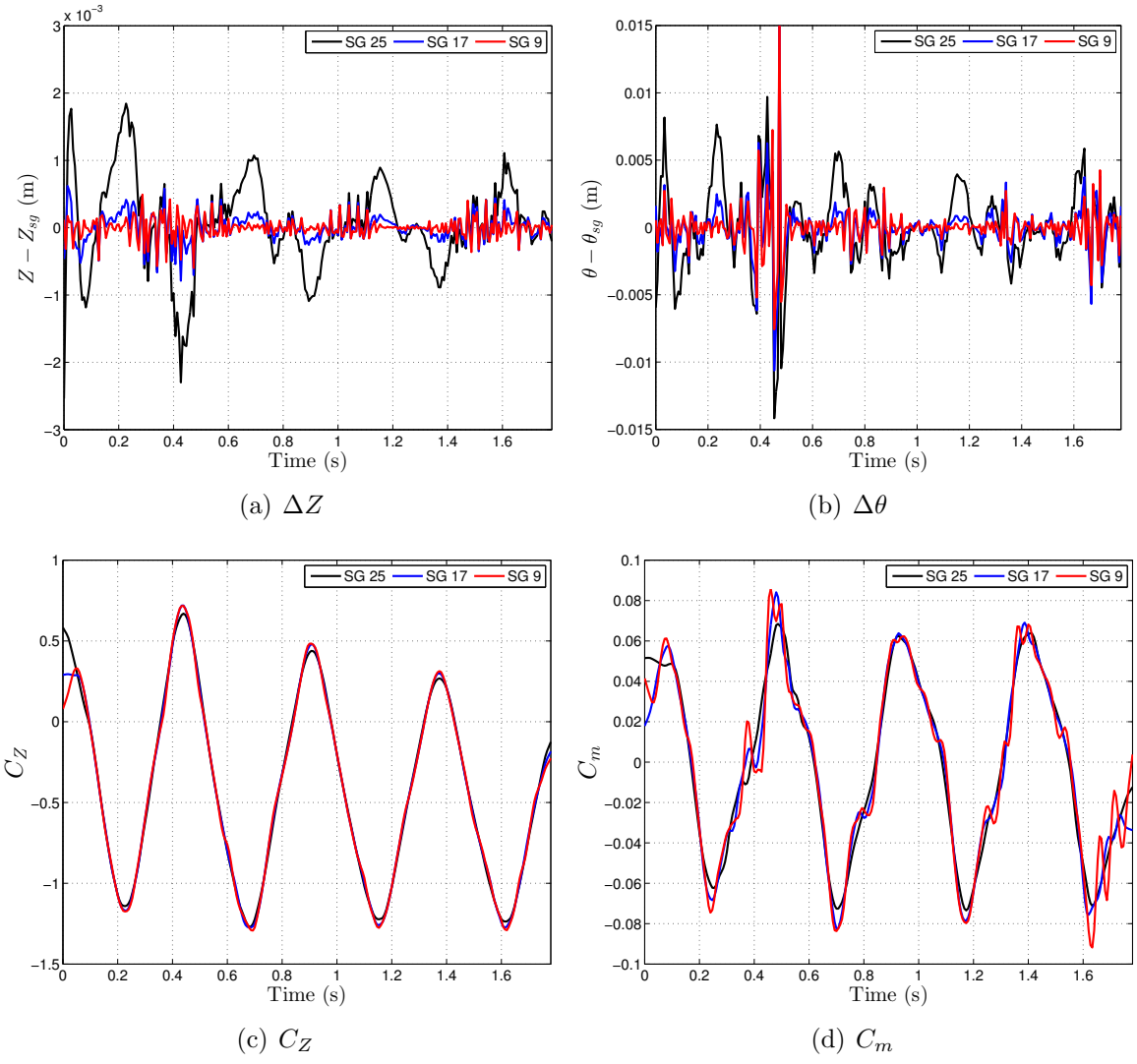
With estimates of the body-axis velocities and angular rates, in addition to their respective derivatives, the aerodynamic forces and moments are then computed according to Eq. 5.4 and Eq. 5.5, respectively.

$$\begin{aligned} F_x(k) &= [\dot{u}(k) + g\sin\theta(k) - r(k)v(k) + q(k)w(k)]\, m \\ F_y(k) &= [\dot{v}(k) - g\cos\theta(k)\sin\phi(k) - p(k)w(k) + r(k)u(k)]\, m \\ F_z(k) &= [\dot{w}(k) - g\cos\theta(k)\cos\phi(k) - q(k)u(k) + p(k)v(k)]\, m \end{aligned} \tag{5.4}$$

$$
\begin{aligned}
L(k) &= \left[\dot{p}(k) - \frac{I_{xz}}{I_x}\dot{r}(k) + \frac{(I_z - I_y)}{I_x}q(k)r(k) - \frac{I_{xz}}{I_x}q(k)p(k)\right] I_x \\
M(k) &= \left[\dot{q}(k) + \frac{(I_x - I_z)}{I_y}p(k)r(k) + \frac{I_{xz}}{I_y}\left(p^2(k) - r^2(k)\right)\right] I_y \\
N(k) &= \left[\dot{r}(k) - \frac{I_{xz}}{I_z}\dot{p}(k) + \frac{(I_y - I_x)}{I_z}p(k)q(k) + \frac{I_{xz}}{I_z}q(k)r(k)\right] I_z
\end{aligned}
\tag{5.5}
$$

The aerodynamic force and moment histories are then non-dimensionalized for model identification and paramter estimation purposes.

In order to determine the coefficients of the Savitzky-Golay filters, the window size and filter order must be selected. As the size of the window increases, the number of data points that are not filtered at the beginning and end of the time series increases. Figure 5.3 shows the effects of window length for the Savitzky-Golay filter on the measurements and computed aerodynamic coefficient histories. The difference between the measured values and smoothed values for inertial vertical displacement, z, and pitch angle, θ, are given in Fig. 5.3(a) and Fig. 5.3(b), respectively. Based on the differences in measurements, it is apparent that a window size of 25 time steps has a much more pronounced effect on the smoothed estimates of z and θ than the smaller window size choices. As a result, in Fig. 5.3(c) and Fig. 5.3(d), the estimates of the C_Z and C_m aerodynamic coefficients are degraded significantly compared to the smaller window sizes. With a window size of 9 time steps, the estimated pitching moment, C_m, is not as smooth as when it is computed with a window size of 17 steps, while C_Z is fairly robust to choice of window size. Note that the effects of window size presented here are specific to the particular test shown, and may not necessarily generalize to other applications.



(a) ΔZ (b) $\Delta\theta$

(c) C_Z (d) C_m

Figure 5.3: Window size choice for Savitizky-Golay filters and its effects on measurement variables and resulting aerodynamic coefficients.

5.2.2 Estimation-Before-Modeling

As an alternative to the use of Savitzky-Golay filters, the aerodynamic forces and moments acting on an aircraft can be estimated using an EKF. Since all data is processed offline, an RTS smoother is also utilized. The six degree-of-freedom equations of motion are used for the forward state propagation in the EKF. Similar to the force and moment estimation in Chapter 4, the forces and moments are modeled as third-order GM processes. It is assumed that the measurements from the VICON motion capture system do not have bias or scale factor errors, and thus $\mathbf{x} \in \mathbb{R}^{30\times 1}$ with the addition of the force and moments and their derivatives, where

$$\mathbf{x} = [u,\ v,\ w,\ p,\ q,\ r,\ \phi,\ \theta,\ \psi,\ X_g,\ Y_g,\ z,\ F_x,\ F_{x_1},\ F_{x_2},\ \ldots,\ N,\ N_1,\ N_2]^T . \tag{5.6}$$

The measurement equation is

$$\mathbf{y} = \mathbf{h}(\mathbf{x}) = [\phi,\ \theta,\ \psi,\ X_g,\ Y_g,\ z]^T . \tag{5.7}$$

It is clear from Eq. 5.7 that the measurement equation is linear. At each time step, with the state as defined in Eq. 5.6, the linearization of Eq. 5.7 results in

$$\mathbf{H}_k = [0_{6\times 6},\ I_{6\times 6},\ 0_{6\times 18}] .$$

The measurement noise covariance matrix, $\mathbf{R}_k$, is chosen to be a constant block diagonal matrix for all time steps. Specifically, the measurement covariance matrix is $\mathbf{R}_k = \mathbf{R}$, where

$$\mathbf{R} = \begin{bmatrix} 1 \times 10^{-3} I_{3\times 3} & 0_{3\times 3} \\ 0_{3\times 3} & 1 \times 10^{-4} I_{3\times 3} \end{bmatrix}.$$

The process noise covariance matrix is denoted as $\mathbf{Q} \in \mathbb{R}^{6\times 6}$ and defined as

$$\mathbf{Q} = \begin{bmatrix} \frac{1}{(T)^2} I_{3\times 3} & 0_{3\times 3} \\ 0_{3\times 3} & \frac{0.1}{(T)^2} I_{3\times 3} \end{bmatrix}.$$

where T is the sampling time. Note that since $\mathbf{Q} \in \mathbb{R}^{6\times 6}$, then $\mathbf{G}_k = \mathbf{G} \in \mathbb{R}^{30\times 6}$. The process noise terms in this application are chosen to only affect the second derivatives of the forces and moments, and thus

$$\mathbf{G} = \begin{bmatrix} 0_{24\times 6} \\ I_{6\times 6} \end{bmatrix}.$$

The initial conditions for the Euler angles and inertial positions are simply the respective measurements at the first time step for each flight, while the initial estimates for the body-axis velocities and angular rates are computed via numerical differentiation. Initial estimates of the forces and moments, and their derivatives, are not computed; the initial conditions for all aerodynamic forces and moments are set equal to 0. The state covariance matrix at the first time step, $\mathbf{P}_0$, is a diagonal matrix with values reflecting the uncertainty for each

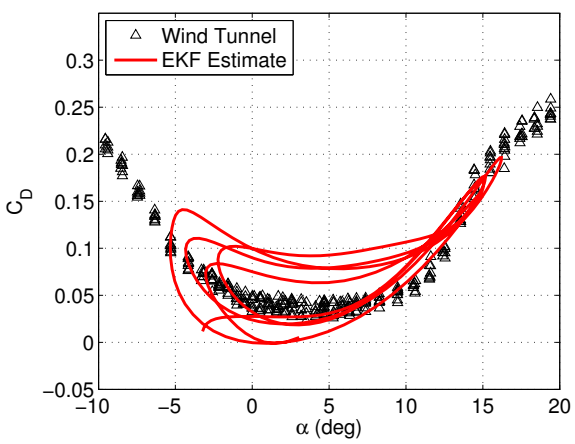
state. The same state covariance matrix is used for all flight tests and is chosen as

$$\mathbf{P}_0 = \text{diag}\left(0.5I_{3\times3},\ 1\times10^{-2}I_{3\times3},\ 1\times10^{-4}I_{3\times3},\ 1\times10^{-3},\ 1\times10^{-4}I_{2\times2},\ldots\right.$$
$$\left.2I_{3\times3},\ 0.1I_{3\times3},\ \frac{2}{(T)}I_{3\times3},\ \frac{0.1}{(T)}I_{3\times3},\ \frac{2}{(T)^2}I_{3\times3},\ \frac{0.1}{(T)^2}I_{3\times3}\right).$$

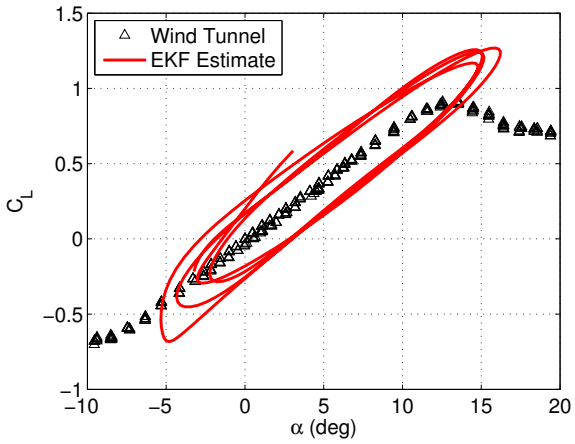
Figure 5.4 shows the estimated longitudinal coefficients from a single flight test in comparison with the static wind tunnel data. Note that the wind tunnel data shows the static aerodynamic coefficients for zero control surface deflection, i.e. $\hat{q} = \delta_E = 0$. The estimates of the coefficients using the EKF, however, include contributions from pitch rate, elevon deflection, and any other phenomena.

Comparison to Savitzky-Golay Smoother Approach

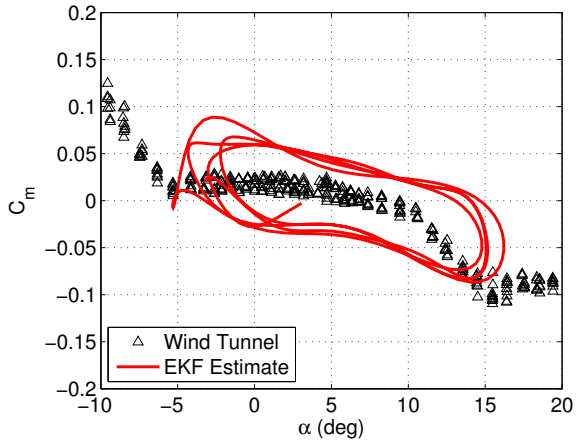
Aerodynamic coefficient histories for two different flights estimated using the Savitzky-Golay filters for smoothing and differentiation are shown in comparison with the EKF in Fig. 5.5. The Savitzky-Golay filters are 3$^{\text{rd}}$-order with window sizes of 7, 17, and 25 time steps. The C_Z force coefficient estimates, in Fig. 5.5(a) and Fig. 5.5(c), are comparable between the two approaches for all window lengths. The C_X and C_Y coefficients estimated using the Savitzky-Golay filters are more erratic than those provided by the EKF. As shown in Fig. 5.5(b) and Fig. 5.5(d), the pitching moment coefficients estimated with a window size of 25 time steps are comparable to the estimates from the EKF, except at peak values. For the lateral-directional moment estimates, the EKF in general returns smoother time histories.



(a) Drag coefficient (C_D)



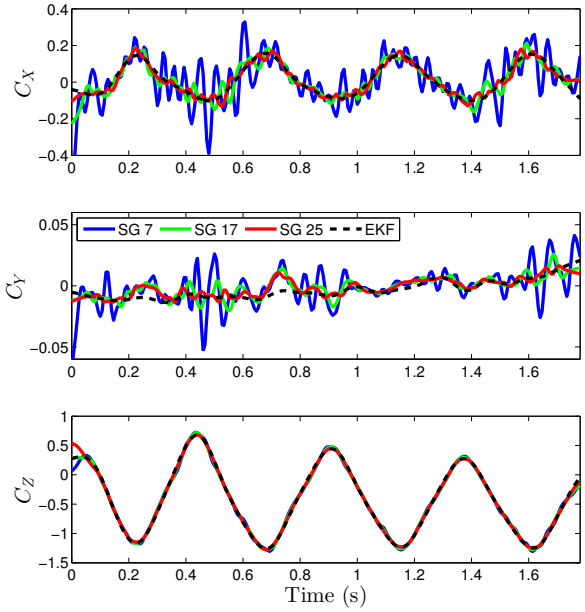
(b) Lift coefficient (C_L)



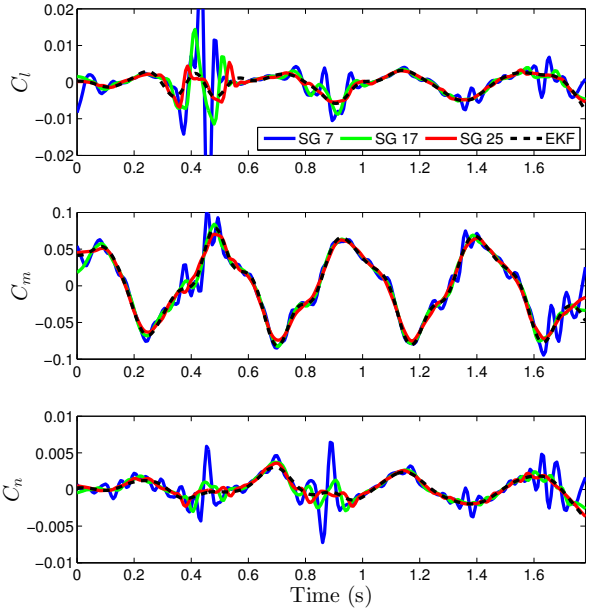
(c) Pitching moment coefficient (C_m)

Figure 5.4: Aerodynamic coefficients estimated using the EKF compared to static wind tunnel data.

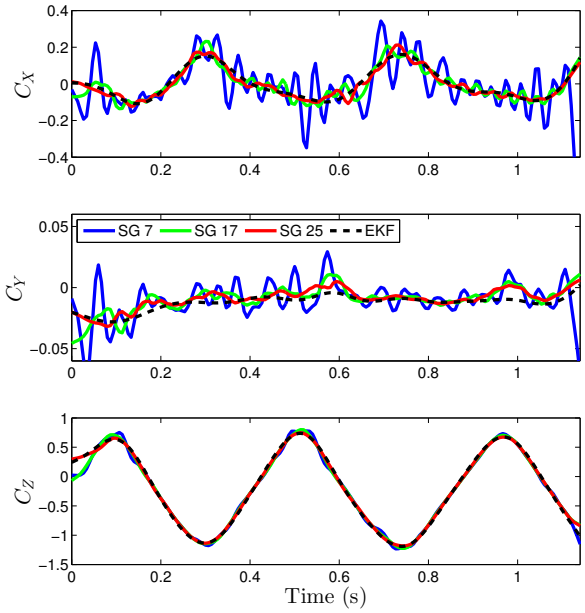
From these plots, it is clear that a window size of 7 time steps is too small, as all coefficient estimates have significant noise. Note that for the results presented in Fig. 5.5(a)-5.5(d), the same window size is used for all measurement variables. Improvement in the coefficients estimated is certainly a possibility using different window sizes, or filter orders, for each measurement variable. Based on the results from the EKF, however, further examination of window size and filter orders is not pursued; all further results utilize state estimates obtained from the EKF.



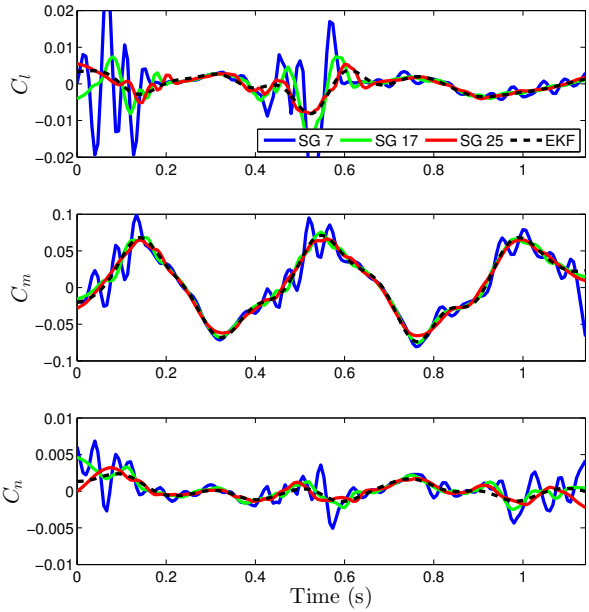
(a) Force coefficients - Test 3



(b) Moment coefficients - Test 3



(c) Force coefficients - Test 12



(d) Moment coefficients - Test 12

Figure 5.5: Savitzky-Golay filter and EKF force & moment coefficient estimation comparison.

5.3 Aerodynamic Model Identification

Due to both the size limitations of the flight test facility as well as duration of flight, it was not feasible to perform lateral-directional system identification tests. The primary focus, therefore, is identification of a longitudinal aircraft model. Three longitudinal aerodynamic model structures are presented. The first model structure is a relatively simple model with stability and control derivative terms to capture pitch dynamics about a nominal flight condition [22, 24]. The second model structure uses piecewise spline functions to capture the variation in stability and control derivatives over the range of independent variables, namely angle of attack and elevon deflection [24, 67]. The third model structure uses polynomial terms of the independent variables in order to capture aerodynamic behavior over the entire test flight envelope [111]. While the second and third model structures are nonlinear with respect to the independent variables, note that they remain linear with respect to the parameters.

A cross-plot of the angle of attack and angle of sideslip computed from the state histories estimated using the EKF is given in Fig. 5.6. From the figure, it is clear that the test data primarily consists of large amplitude pitching motions, with $-10^\circ \leq \alpha \leq 17^\circ$. Note that although there is some variation in the angle of sideslip, β, this is the result of the glider leaving the slingshot device with a non-zero sideslip angle.

Since all flight tests have predominantly longitudinal motion, it is assumed that the longitudi-

nal aerodynamic coefficients do not depend on lateral-directional variables. The longitudinal independent variables are therefore the angle of attack, rate of change of angle of attack, pitch rate, and average elevon deflection. The average elevon deflection is denoted as δ_E and $\dot{\hat{\alpha}}$ is the nondimensionalized rate of change of angle of attack. These quantities are computed as given in Eq. 5.8 and Eq. 5.9, respectively.

$$\delta_E = \frac{\delta_L + \delta_R}{2} \tag{5.8}$$

$$\dot{\hat{\alpha}} = \frac{\dot{\alpha}\bar{c}}{2V} \tag{5.9}$$

Cross plots of the independent variables used for the longitudinal aerodynamic model, namely α, $\dot{\hat{\alpha}}$, $\hat{q}$, and δ_E are shown in Fig. 5.7(a)-5.7(d). From these plots, it is clear that there is coverage over a wide range of the indepedent variables. Note, in particular, from Fig. 5.7(d), that $\hat{q}$ and $\dot{\hat{\alpha}}$ are not highly correlated. For quasi-steady modeling purposes, it is often assumed that $q = \dot{\alpha}$, i.e. the rate of change of angle of attack is equivalent to the pitch rate of the aircraft. Under such an assumption, variation in aerodynamic coefficients due to $\dot{\alpha}$ is essentially "absorbed" into the pitch rate contributions. From examination of Fig. 5.7(d), it is clear that $q \neq \dot{\alpha}$ in this case.

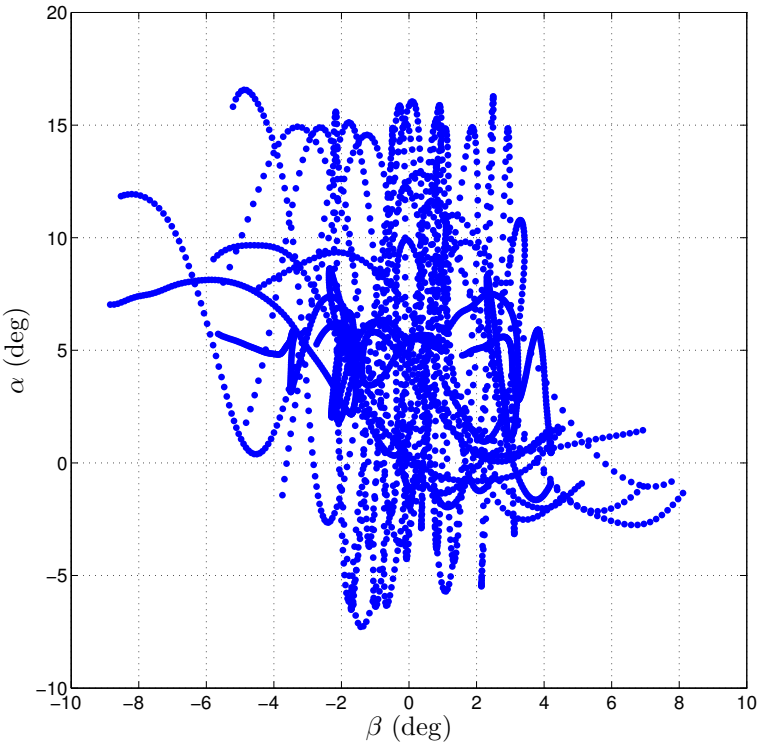
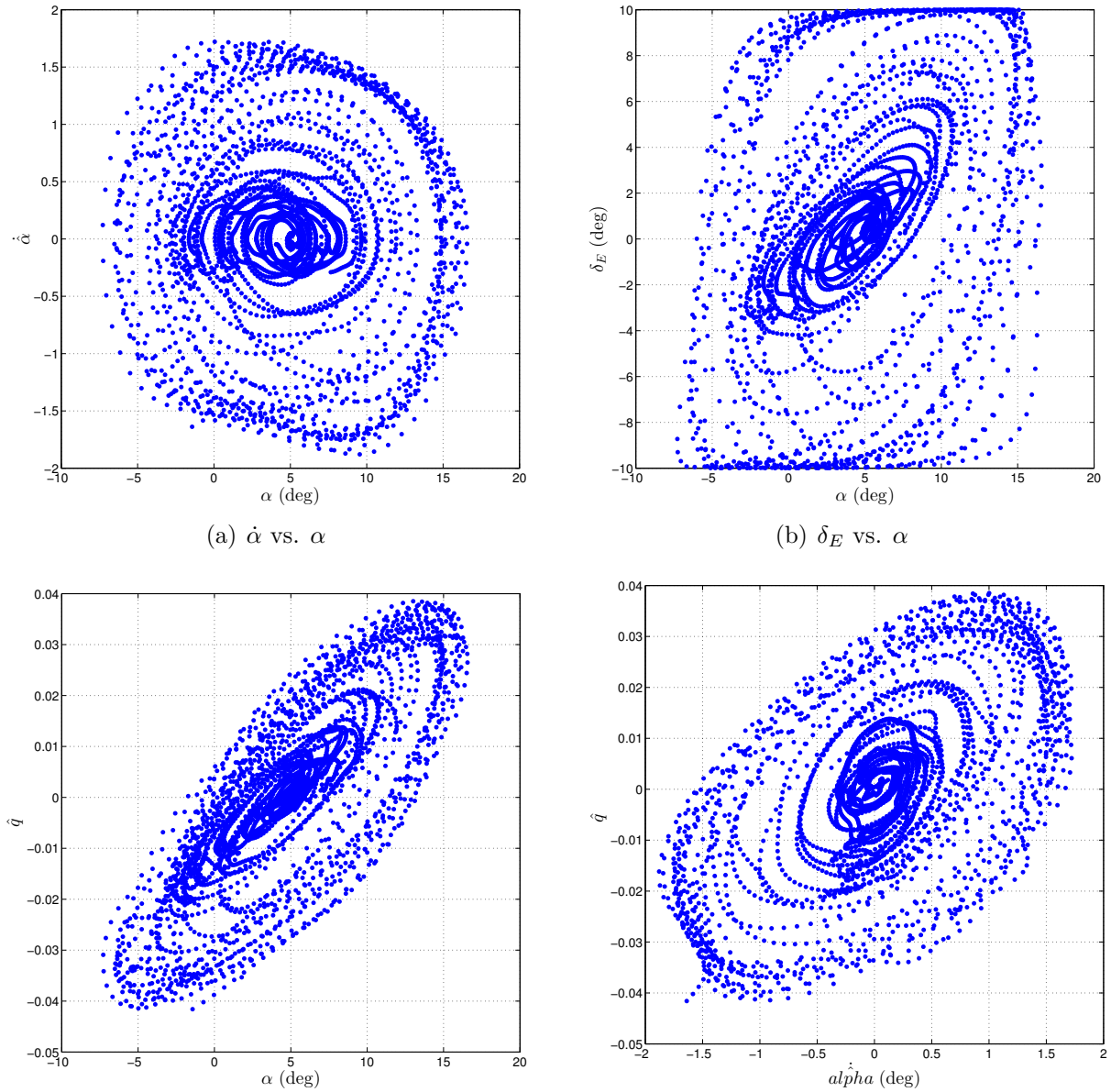


Figure 5.6: α-β cross plot of test data.



(a) $\dot{\alpha}$ vs. α

(b) δ_E vs. α

(c) $\hat{q}$ vs. α

(d) $\hat{q}$ vs. $\dot{\alpha}$

Figure 5.7: Cross plots of independent variables for aerodynamic modeling.

5.3.1 Stability and Control Derivative Model

As previously mentioned, the first model investigated is a stability and control derivative model that is linear with respect to the independent variables with the exception of α^2 terms in the C_X and C_Z force coefficient models. Since angle of attack ranges from approximately $-10°$ to $17°$, such a model may be able to capture the time series variation in the aerodynamic coeficients. For this model structure, it is assumed that angle of attack rate effects can be modeled in conjunction with pitch rate effects for the C_X and C_Z coefficients; the model for pitching moment, C_m, considers the separate contributions of pitch rate and angle of attack rate of change. The model structures for the three longitudinal coefficients are therefore given as

$$C_X = C_{X_0} + C_{X_\alpha}\alpha + C_{X_{\alpha^2}}\alpha^2 + C_{X_{\delta_E}}\delta_E \tag{5.10}$$

$$C_Z = C_{Z_0} + C_{Z_\alpha}\alpha + C_{Z_{\alpha^2}}\alpha^2 + C_{Z_q}\frac{\bar{c}}{2V}q + C_{Z_{\delta_E}}\delta_E \tag{5.11}$$

$$C_m = C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{\bar{c}}{2V}q + C_{m_{\delta_E}}\delta_E + C_{m_{\dot{\alpha}}}\frac{\bar{c}}{2V}\dot{\alpha} \tag{5.12}$$

As in Section 4.1.2, data is combined from all flight tests into a single matrix of independent variables for each coeficient. Least squares regression is then applied to obtain estimates of the parameters in Eq. 5.10-5.12. The R^2 and RMSE values for the modeling and validation data are given in Table 5.2. Despite this rather simple model structure, the R^2 and RMSE for the modeling data indicate that the models are able to capture the variation in the

aerodynamic coefficents very well, particularly for the C_X and C_Z coefficients. The R^2 and RMSE values for the validation data indicate that the model is able to predict the coefficient histories accurately as well. The estimated parameter values and their respective relative standard deviations are given in Table 5.3.

Table 5.2: R^2 and RMSE values for stability and control derivative aerodynamic models

	Modeling		**Validation**	
Coef.	R^2	RMSE	R^2	RMSE
C_X	0.9701	0.0113	0.9760	0.0112
C_Z	0.9942	0.0351	0.9934	0.0434
C_m	0.9576	0.0062	0.9659	0.0066

Table 5.3: Parameters for longitudinal aerodynamic models in Eq. 5.10-5.12

Coef.	**Value**	**Rel. σ**	**Coef.**	**Value**	σ	**Coef.**	**Value**	σ
C_{X_0}	-0.0493	0.0070	C_{Z_0}	0.0417	0.0524	C_{m_0}	0.0021	0.2453
C_{X_α}	0.3129	0.0166	C_{Z_α}	-5.1834	0.0052	C_{m_α}	-0.0518	0.1106
$C_{X_{\alpha^2}}$	1.6365	0.0150	$C_{Z_{\alpha^2}}$	1.0916	0.0641	$C_{m_{\hat{q}}}$	-0.8857	0.0387
$C_{X_{\delta_E}}$	-0.0045	0.0098	$C_{Z_{\hat{q}}}$	2.0615	0.0701	$C_{m_{\delta_E}}$	0.0020	0.0326
			$C_{Z_{\delta_E}}$	0.0159	0.0184	$C_{m_{\dot{\alpha}}}$	1.4745	0.0255

5.3.2 Spline Model

For each stability and control derivative, second-order spline functions with α as the independent variable are created at specified values of angle of attack. Denoting k_N as the total number of knot points, the effect of pitch rate on the C_Z coefficient, for example, is of the

form in Eq. 5.13.

$$C_{Z_q}(\alpha) = \left(C_{Z_q} + C_{Z_{q,\alpha}}\alpha + C_{Z_{q,\alpha^2}}\alpha^2 + \sum_{k=1}^{k_N} C_{Z_{q,\alpha\geq\alpha_{\text{knot}}(k)}} \left(\alpha - \alpha_{\text{knot}}(k)\right)_+^2 \right) \tag{5.13}$$

The other stability and control derivatives, e.g. $C_{m_q}(\alpha)$, are expressed in a similar form. For this model, spline functions of both α and δ_E are generated as candidate regressors. The aerodynamic models for each coefficient are therefore given as

$$\begin{aligned} C_X =& C_{X_0} + C_X(\alpha) + C_{X_q}(\alpha)\frac{q\bar{c}}{2V} + C_{X_{\delta_E}}(\alpha)\delta_E + C_X(\delta_E) \\ C_Z =& C_{Z_0} + C_Z(\alpha) + C_{Z_q}(\alpha)\frac{q\bar{c}}{2V} + C_{Z_{\delta_E}}(\alpha)\delta_E + C_Z(\delta_E) \\ C_m =& C_{m_0} + C_m(\alpha) + C_{m_\alpha}(\alpha)\frac{q\bar{c}}{2V} + C_{m_{\delta_E}}(\alpha)\delta_E + C_{m_{\dot{\alpha}}}(\alpha)\frac{\dot{\alpha}\bar{c}}{2V} + C_m(\delta_E) \end{aligned} \tag{5.14}$$

A preliminary second-degree spline function with knot points in α ranging from $-8°$ to $12°$ and evenly spaced at $1°$ intervals is created. Similarly, a second-degree spline function in δ_E ranging from $-10°$ to $10°$ with knot points evenly spaced at $2.5°$ intervals is also created. These preliminary spline functions are examined for pairwise correlation between the candidate regressors. Specifically, the correlation coefficient between two regressors is computed as

$$\rho(x,y) = \frac{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)\left(y_i - \bar{y}\right)}{\sqrt{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2 \sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2}} \tag{5.15}$$

where $\bar{x}$ denotes the mean value of x [24, 22]. The spline function of α is found to have high pairwise correlation ($\rho \geq 0.95$) between most of the regressors with this spacing of

knot points; the spline function of δ_E does not. The number of knot points in α is therefore reduced to remove the high correlation between the candidate regressors. Specifically, the second-degree spline function of α has knot points at $-7.5°$, $0°$, and $7.5°$.

In total, there are 25 candidate regressors for both C_X and C_Z, and 31 for C_m. The stepwise regression procedue is then performed. The identified model has $(14, 15, 19)$ terms corresponding to the models for (C_X, C_Z, C_m), respectively. The identified spline models are of the form

$$\begin{aligned}
C_X = &\big[1, \alpha, \alpha^2, (\alpha - 7.5)_+^2, \hat{q}, \hat{q}\alpha, \hat{q}(\alpha - -7.5)_+^2, \hat{q}(\alpha - 0)_+^2, \hat{q}(\alpha - 7.5)_+^2, \dots \\
&\delta_E, \delta_E\alpha, \delta_E(\alpha - 0)_+^2, (\delta_E - -7.5)_+^2, (\delta_E - 0)_+^2, (\delta_E - 7.5)_+^2\big]\,\Theta_X \\
C_Z = &\big[1, \alpha, (\alpha - -7.5)_+^2, (\alpha - 0)_+^2, \hat{q}, \hat{q}\alpha, \hat{q}\alpha^2, \hat{q}(\alpha - 0)_+^2, \delta_E\alpha, \delta_E\alpha^2, \dots \\
&\delta_E(\alpha - -7.5)_+^2, \delta_E(\alpha - 7.5)_+^2, (\delta_E - 0)_+^2, (\delta_E - 2.5)_+^2, (\delta_E - 5)_+^2, (\delta_E - 7.5)_+^2\big]\,\Theta_Z \\
C_m = &\big[1, \alpha, (\alpha - -7.5)_+^2, (\alpha - 0)_+^2, (\alpha - 7.5)_+^2, \hat{q}, \hat{q}\alpha, \hat{q}\alpha^2, \hat{q}(\alpha - 0)_+^2, \hat{q}(\alpha - 7.5)_+^2, \dots \\
&\delta_E, \delta_E\alpha^2, \delta_E(\alpha - -7.5)_+^2, \delta_E(\alpha - 0)_+^2, \delta_E(\alpha - 7.5)_+^2, (\delta_E - -2.5)_+^2, (\delta_E - 7.5)_+^2, \dots \\
&\dot{\alpha}, \dot{\alpha}(\alpha - -7.5)_+^2, \dot{\alpha}(\alpha - 0)_+^2\big]\,\Theta_m
\end{aligned} \tag{5.16}$$

where Θ_X, Θ_Z, and Θ_m are the estimated model parameters.

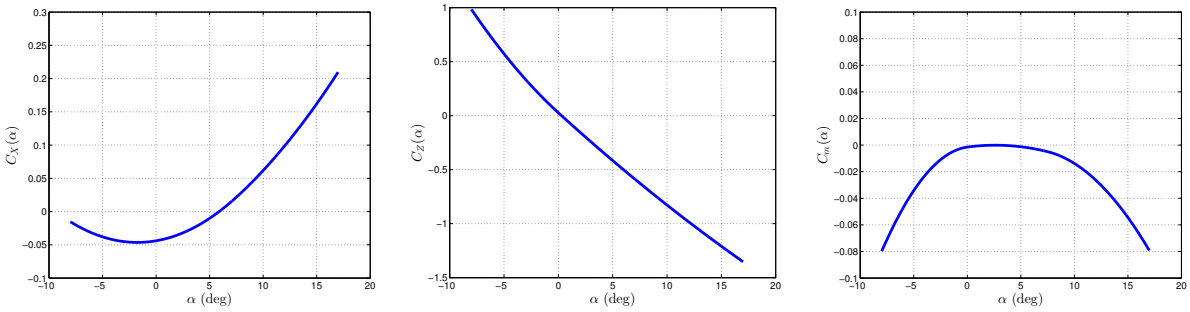
The R^2 and RMSE values for the modeling and validation data are given in Table 5.4. From the table, it is clear that the model captures the variation in the longitudinal aerodynamic

coefficients. Further, there are no discrepancies in the R^2 and RMSE values between the modeling and validation data that would be indicative of overfitting.

Table 5.4: R^2 and RMSE values for spline aerodynamic models

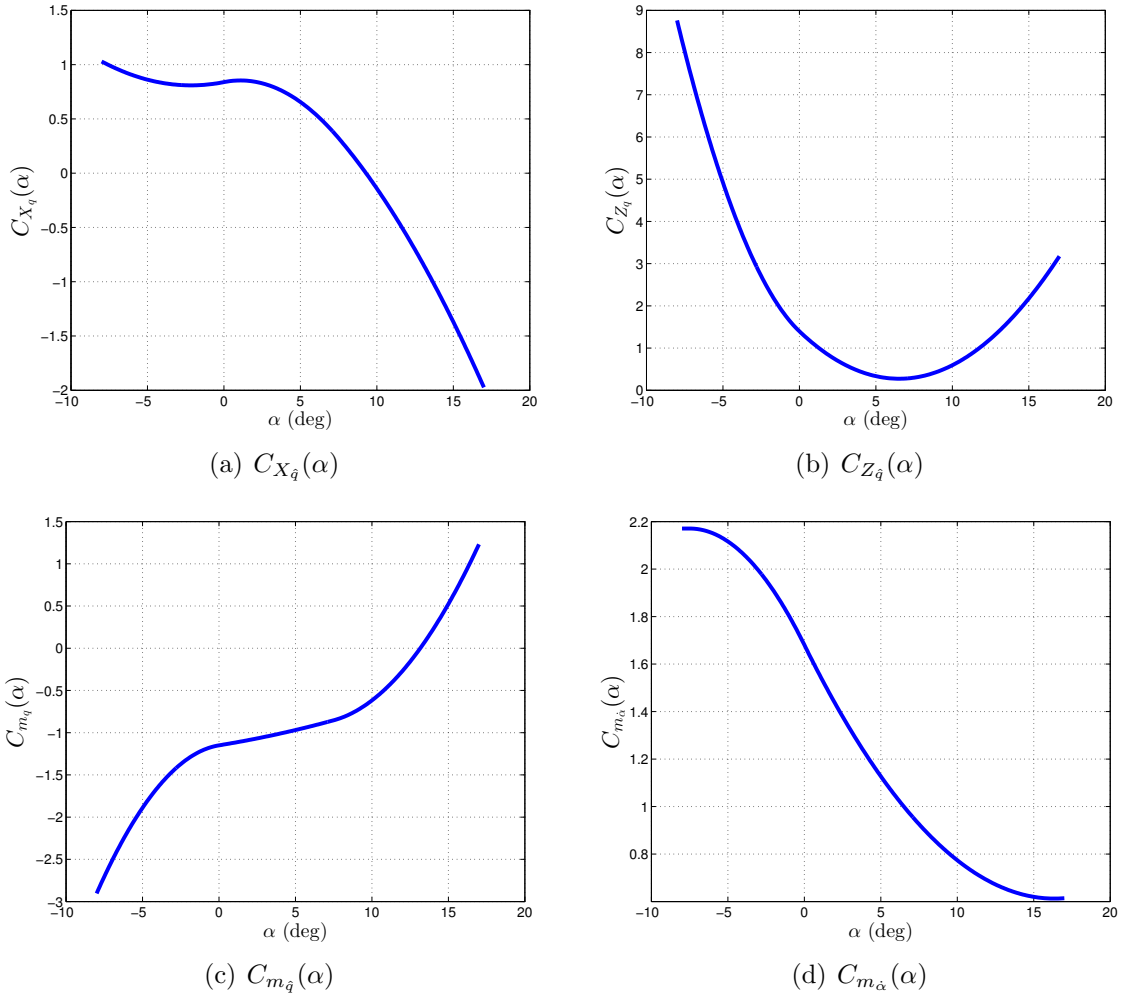
	Modeling		**Validation**	
Coef.	R^2	RMSE	R^2	RMSE
C_X	0.9840	0.0083	0.9816	0.0098
C_Z	0.9967	0.0263	0.9958	0.0348
C_m	0.9774	0.0045	0.9812	0.0049

A graphical presentation of the aerodynamic coefficients for the spline model is provided instead of a table with individual parameter values. The variations of the model parameters for C_X, C_Z, and C_m are given in Fig. 5.8(a), 5.8(b) and 5.8(c), respectively. Note that in Fig. 5.8(c), the variation of the pitching moment coefficient with respect to α only indicates that the aircraft is unstable at negative angle of attack, i.e. $C_m(\alpha) \leq 0$ for $\alpha \leq 0$. The variations of the dynamic derivatives $C_{X_{\hat{q}}}$ and $C_{Z_{\hat{q}}}$ with respect to angle of attack are shown in Fig. 5.9(a) and 5.9(b). The variation of the pitching moment dynamic derivatives, i.e. pitch rate and angle of attack rate contributions, denoted as $C_{m_{\hat{q}}}$ and $C_{m_{\dot{\alpha}}}$, are given in Fig. 5.9(c) and 5.9(d), respectively. The variations of the longitudinal models with repsect to α and δ_E, with $\hat{q} = \dot{\alpha} = 0$, are given in Fig. 5.10(a)-5.11.



(a) $C_X(\alpha)$ (b) $C_Z(\alpha)$ (c) $C_m(\alpha)$

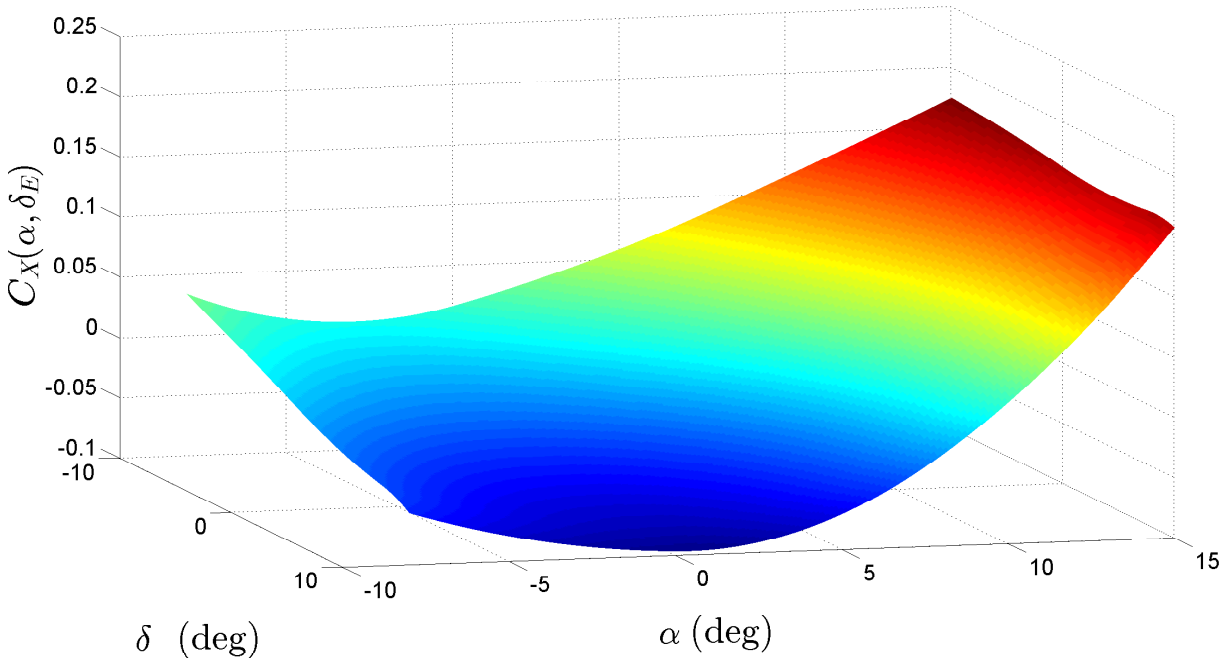
Figure 5.8: Variation of coefficients with respect to angle of attack only.



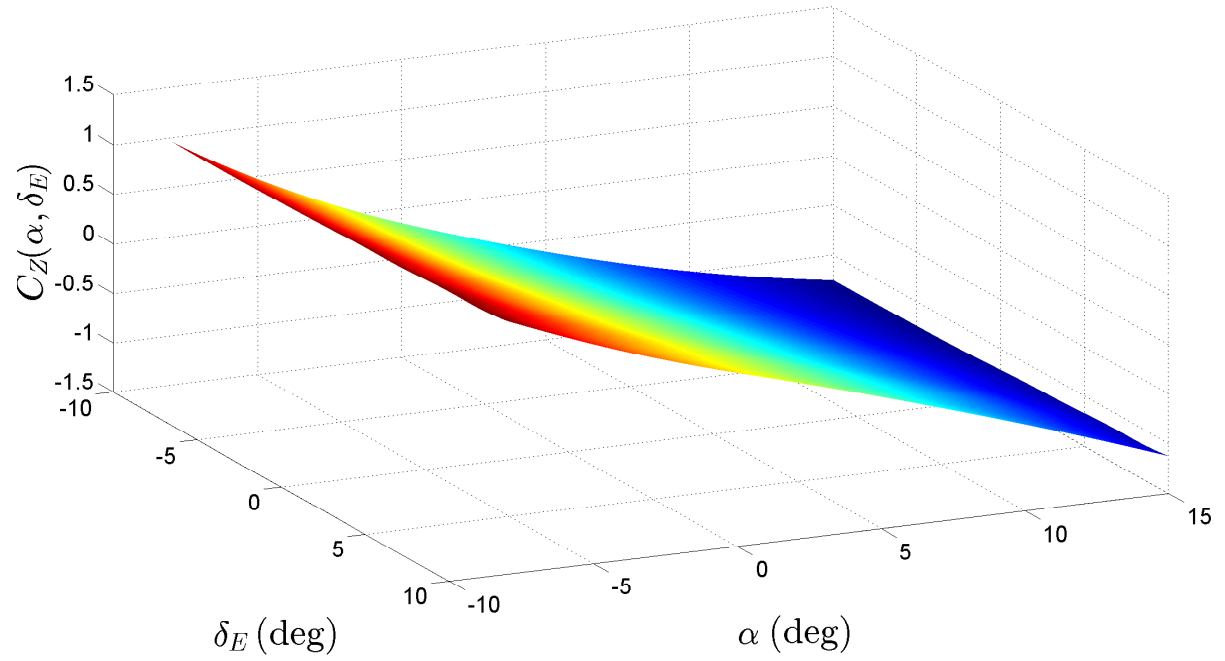
(a) $C_{X_{\hat{q}}}(\alpha)$ (b) $C_{Z_{\hat{q}}}(\alpha)$

(c) $C_{m_{\hat{q}}}(\alpha)$ (d) $C_{m_{\dot\alpha}}(\alpha)$

Figure 5.9: Dynamic derivative variation with respect to angle of attack (α).



(a) $C_X(\alpha, \delta_E)$ for spline model.



(b) $C_Z(\alpha, \delta_E)$ for spline model.

Figure 5.10: Variation of force coefficients with respect to α and δ_E for spline models.

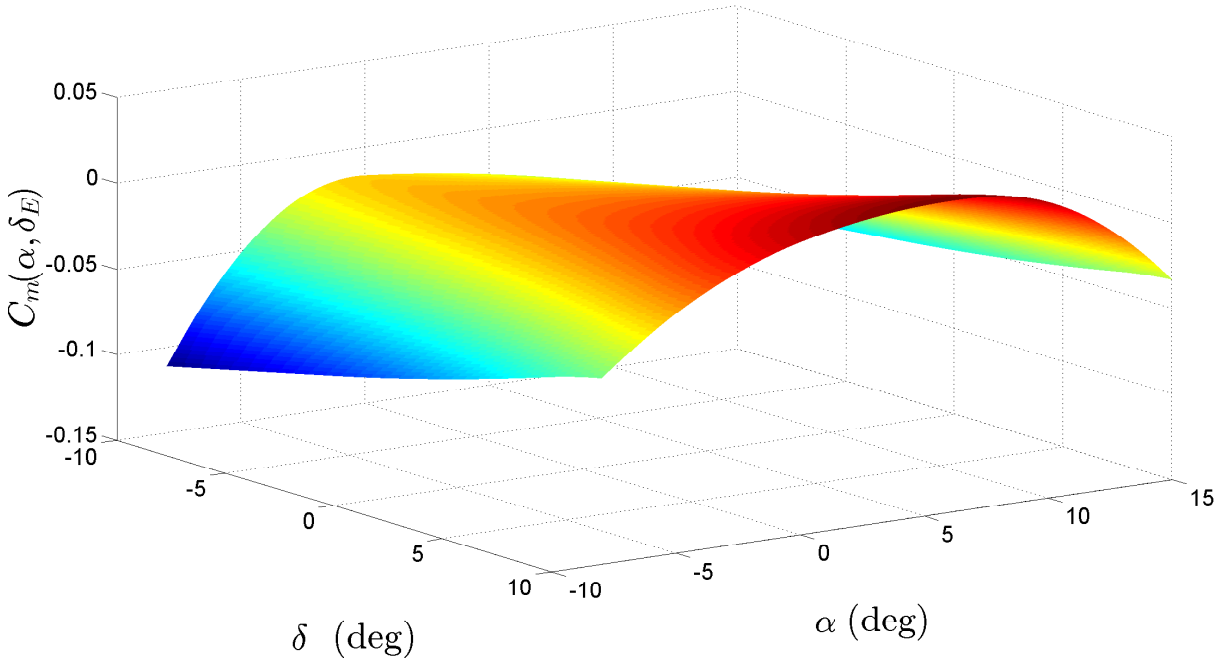


Figure 5.11: Variation of pitching moment coefficient with respect to α and δ_E for spline model.

5.3.3 Polynomial Model

An aerodynamic model using polynomial terms of the independent variables is also investigated. Specifically, the subroutine for multivariate orthogonal function generation in SIDPAC is utilized [111]. For each aerodynamic coefficient, the polynomials are functions with the following independent variables.

$$\begin{aligned} C_X &= f(\alpha, \hat{q}, \delta_E) \\ C_Z &= f(\alpha, \hat{q}, \delta_E) \\ C_m &= f(\alpha, \hat{q}, \delta_E, \dot{\hat{\alpha}}) \end{aligned} \tag{5.17}$$

Table 5.5: Polynomial degree for each individual variable and model term for MOF longitudinal models.

Model	Maximum Polynomial Degree				
–	α	$\hat{q}$	δ_E	$\dot{\alpha}$	All Terms
C_X	3	3	3	–	3
C_Z	3	3	3	–	3
C_m	3	3	3	3	4

For each model in Eq. 5.17, the maximum order of each independent variable and total degree of each term in the final model must be specified; the maximum degree for each indepedent variable and total for all terms permitted in the candidate models are given in Table 5.5. The multivariate orthogonal function (`mof`) tool in SIDPAC generates orthogonal polynomials of the independent variables via Gram-Schmidt orthogonalization [24, 111]. The predicted response, degree of independent variables for each individual model term, and corresponding parameter values are returned.

Prior to arriving at the structure given in Eq. 5.17, other sets of indepedent variables and maximum polynomial term orders were examined. Notably, the inclusion of $\dot{\alpha}$ in the models for C_X and C_Z is not identified as a significant explanatory variable. Additionally, the inclusion of piecewise spline functions, similar to the second model form, is also investigated. These variables, however, do not provide any improvement to the model.

The resulting polynomial models for C_X, C_Z, and C_m are given in Eq. 5.18, Eq. 5.19, and Eq. 5.20, respectively. Note that there are 11, 4, and 14 parameters in the models corresponding to C_X, C_Z, and C_m, respectively. The R^2 and RMSE values for the polynomial

model are given in Table 5.6. Based on the values in the table, the polynomial model captures the variation in the aerodynamic coefficients without overfitting. The parameter values and relative standard deviations are given in Table 5.7.

$$\begin{aligned} C_X =& C_{X_0} + C_{X_\alpha}\alpha + C_{X_{\hat{q}}}\hat{q} + C_{X_{\delta_E}}\delta_E + C_{X_{\alpha^2}}\alpha^2 + C_{X_{\alpha^2\hat{q}}}\alpha^2\hat{q} + C_{X_{\alpha^2\delta_E}}\alpha^2\delta_E \\ &+ C_{X_{\alpha\delta_E}}\alpha\delta_E + C_{X_{\hat{q}\delta_E}}\hat{q}\delta_E + C_{X_{\hat{q}^2\delta_E}}\hat{q}^2\delta_E + C_{X_{\alpha\hat{q}\delta_E}}\alpha\hat{q}\delta_E \end{aligned} \tag{5.18}$$

$$C_Z = C_{Z_\alpha}\alpha + C_{Z_{\delta_E}}\delta_E + C_{Z_{\alpha^2\delta_E}}\alpha^2\delta_E + C_{Z_{\alpha\hat{q}}}\alpha\hat{q} \tag{5.19}$$

$$\begin{aligned} C_m =& C_{m_{\hat{q}}}\hat{q} + C_{m_{\dot{\alpha}}}\dot{\alpha} + C_{m_{\delta_E}}\delta_E + C_{m_{\alpha^2}}\alpha^2 + C_{m_{\alpha^2\hat{q}}}\alpha^2\hat{q} + C_{m_{\delta_E\dot{\alpha}}}\delta_E\dot{\alpha} + C_{m_{\alpha^2\delta_E}}\alpha^2\delta_E \\ &+ C_{m_{\alpha^3}}\alpha^3 + C_{m_{\alpha\hat{q}}}\alpha\hat{q} + C_{m_{\hat{q}\delta_E}}\hat{q}\delta_E + C_{m_{\alpha\dot{\alpha}}}\alpha\dot{\alpha} + C_{m_{\alpha\hat{q}\delta_E}}\alpha\hat{q}\delta_E + C_{m_{\alpha\delta_E}}\alpha\delta_E \\ &+ C_{m_{\alpha\delta_E\dot{\alpha}}}\alpha\delta_E\dot{\alpha} \end{aligned} \tag{5.20}$$

Table 5.6: R^2 and RMSE values for polynomial aerodynamic models.

	Modeling		**Validation**	
Coef.	R^2	RMSE	R^2	RMSE
C_X	0.9839	0.0083	0.9810	0.0100
C_Z	0.9955	0.0309	0.9946	0.0393
C_m	0.9773	0.0045	0.9798	0.0051

Table 5.7: Polynomial parameter estimates and relative standard deviations.

Param.	Value	Rel. σ	Param.	Value	Rel. σ	Param.	Value	Rel. σ
C_{X_0}	-0.0491	0.0327	C_{Z_α}	-4.7122	0.0014	$C_{m_{\hat{q}}}$	-1.0753	0.0134
C_{X_α}	0.2577	0.0138	$C_{Z_{\delta_E}}$	0.0219	0.0065	$C_{m_{\dot{\alpha}}}$	1.0873	0.0312
$C_{X_{\hat{q}}}$	0.5402	0.0934	$C_{Z_{\alpha^2\delta_E}}$	-0.2200	0.0245	$C_{m_{\delta_E}}$	0.0048	0.0347
$C_{X_{\alpha^2}}$	2.2389	0.0119	$C_{Z_{\alpha\hat{q}}}$	10.6763	0.0328	$C_{m_{\alpha^2}}$	0.1260	0.2400
$C_{X_{\delta_E}}$	-0.0047	0.0219				$C_{m_{\alpha^2\hat{q}}}$	25.8571	0.0503
$C_{X_{\alpha^2\hat{q}}}$	-30.7704	0.0282				$C_{m_{\delta_E\dot{\alpha}}}$	-0.0571	0.0924
$C_{X_{\alpha^2\delta_E}}$	0.2191	0.0363				$C_{m_{\alpha^3}}$	-3.0101	0.0565
$C_{X_{\alpha\delta_E}}$	-0.0373	0.0435				$C_{m_{\alpha\hat{q}}}$	-1.7124	0.0993
$C_{X_{\hat{q}\delta_E}}$	0.1143	0.0600				$C_{m_{\alpha\dot{\alpha}}}$	-0.6072	0.4239
$C_{X_{\alpha\hat{q}\delta_E}}$	-0.8824	0.0728				$C_{m_{\alpha\hat{q}\delta_E}}$	-0.5763	0.0994
$C_{X_{\hat{q}^2\delta_E}}$	2.0591	0.0956				$C_{m_{\alpha\delta_E}}$	-0.0215	0.0930
						$C_{m_{\alpha\delta_E\dot{\alpha}}}$	0.3570	0.1247

5.4 Comparison of Models

Based on a direct comparison between the R^2 and RMSE values given in Tables 5.2, 5.4, and 5.6 for the stability and control derivative, spline, and polynomial models, respectively, it is difficult to distinguish which, in any, of these models is preferable over the others. A more in-depth comparison of the models is therefore performed using Theil's inequality coefficient for both the predicted aerodynamic coefficient time histories as well as the longitudinal state variables obtained from simulation with each model.

Theil's inequality coefficient is computed for each modeling and validation test for all flights analyzed. The average and maximum TIC values are given in Table 5.8, computed for both the modeling and validation tests. Note that the average inequality coefficient values for all tests are below the acceptable range of 0.2-0.3 for all three coefficients. Further, only

the stability and control derivative model has maximum inequality coefficient values that are high enough to be questionable, specifically the coefficient models for C_X and C_m. The spline and polynomial models are equivalent in performance based on the inequality coefficient.

Table 5.8: Theil's Inequality Coefficient - Aerodynamic coefficient time history predictions.

		Modeling			**Validation**		
Model	–	S/C	Spline	Poly	S/C	Spline	Poly
C_X	average	0.1025	0.0772	0.0777	0.0815	0.0692	0.0715
	maximum	0.2153	0.1561	0.1661	0.1009	0.0744	0.0815
C_Z	average	0.0384	0.0291	0.0359	0.0398	0.0304	0.0366
	maximum	0.0591	0.0546	0.0637	0.0463	0.0383	0.0427
C_m	average	0.1065	0.0803	0.0810	0.0917	0.0697	0.0689
	maximum	0.1853	0.1177	0.1192	0.1147	0.0812	0.0946

Analysis of Theil's decomposition of fit reveals that for the predicted pitching moment coefficients, the contribution of error that is systematic is unacceptable (above 0.1) for five of the time history predictions using the stability and control derivative model, one of the flights being the test with the maximum inequality coefficient. For the spline and polynomial models, the decomposition of fit indicates that the systematic error between the estimated and predicted pitching moment coefficient histories is unacceptable for three tests, however, the inequality coefficient is small enough for these tests that such errors are deemed acceptable. The decomposition of fit also indicates that there are six flights where the systematic error in C_Z is unacceptable for the stability and control derivative model; again for the spline and polynomial models, only three flights have unacceptable systematic error. For all three model structures, the inequality coefficients are low enough that these systematic errors are not problematic. The C_X coefficient, however has systematic error for seven flight

tests for the spline and polynomial models. Again, however, both the spline and polynomial predicted coefficients for these tests have inequality coefficients that are acceptable. The decomposition of fit reveals that there are five flights where the systematic error is above the acceptable recommended threshold; two of these flights have inequality coefficients near the lower bound of the acceptable range (0.2).

Model Simulation

To further assess each aerodynamic model, the forward simulation performance is analyzed. As previously stated, flight tests did not have sufficient excitation of the lateral-directional states to derive aerodynamic models for the side-force, rolling moment, and yawing moment. The forward simulation therefore uses longitudinal equations of motion that are decoupled from the lateral-directional equations through substitution of measured values. This approach is useful for the analysis of highly coupled as well as large amplitude maneuvers [24]. Specifically, the estimates of the lateral-directional states obtained using the EKF are treated as inputs in the equations of motion, i.e. $p = p_E$ where p_E is the filter estimate. Note that since the aerodynamic models contain terms corresponding to $\dot{\alpha}$, during simulation, this is computed as

$$\dot{\alpha} = \frac{d}{dt}\left\{\tan^{-}1\left(\frac{w(t)}{u(t)}\right)\right\} = \frac{u(t)\dot{w}(t) - w(t)\dot{u}(t)}{u^2(t) + w^2(t)}. \tag{5.21}$$

The decoupled longitudinal equations of motion for forward simulation of the aerodynamic model are as given in Eq. 5.22. As in Chapter 4, the initial conditions are the EKF state esti-

mates. Each flight test is simulated using Eq. 5.22 and linear interpolation for all substituted state and elevon time histories.

$$\begin{aligned}
\dot{u} =& -g_0 \sin\theta + r_E v_E - qw + \frac{F_x}{m} \\
\dot{w} =& g_0 \cos\theta \cos\phi_E + qu - p_E v_E + \frac{F_z}{m} \\
\dot{q} =& \frac{1}{I_y}\left[M - (I_x - I_z)p_E r_E - I_{xz}(p_E^2 - r_E^2)\right] \\
\dot{\theta} =& q\cos\phi_E - r_E \sin\phi_E
\end{aligned} \tag{5.22}$$

Recall that in Section 2.3.4, it was briefly mentioned that for time histories where the reference value is non-zero, Theil's inequality coefficient can give a false indication of model fidelity and a simple example was provided to illustrate this notion. Here, data from simulation with the aerodynamic models is utilized in order to further reinforce this point. Consider the plot of the u body-axis velocity component from simulation of a single flight using the three aerodynamic models as shown in Fig. 5.12. Theil's inequality coefficient is computed as given in Eq. 2.20 as well as with the mean of the estimated u velocity (from the EKF) removed. When computing Theil's inequality coefficient without accounting for the

Table 5.9: Theil's Inequality Coefficient for Fig. 5.12 (without and with mean removed)

	S/C	Spline	Polynomial
TIC	0.0134	0.0196	0.0046
TIC (MR)	0.1494	0.2029	0.0576

magnitude of u, it is clear from Table 5.9 that all three models indicate an excellent match of the time series, despite the stability and control derivative and spline models having a poor visual fit after 0.8 seconds, as shown in Fig. 5.12. After accounting for the mean value of the original time series, Theil's inequality coefficient indicates that the first two models are not nearly as good of a match as the polynomial model. For all further computations of Theil's inequality coefficient presented, the mean value of the original time series is subtracted from all time series compared.

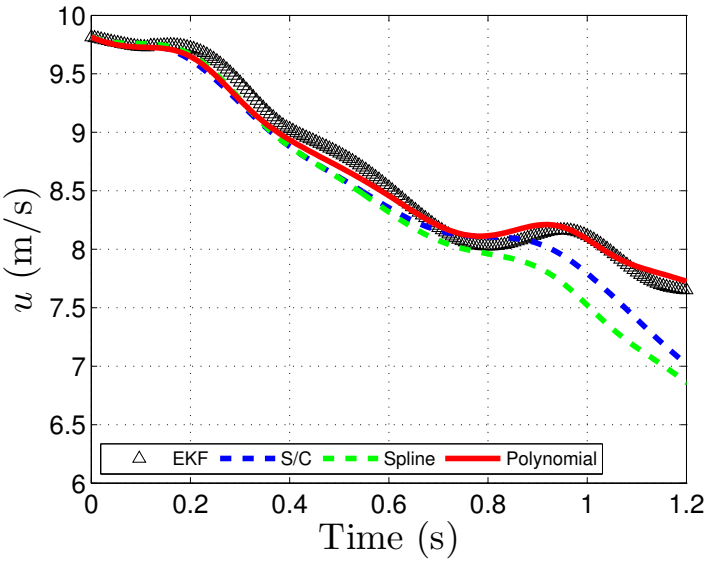


Figure 5.12: u body-axis velocity component from model simulation for a single flight test.

Examples of the comparison between the EKF estimated longitudinal state histories and those obtained from forward simulation of the aerodynamic models are given in Fig. 5.13 and Fig. 5.14. The Theil inequality coefficient values for the time histories of the longitudinal states from simulation are given in Table 5.10. Note that both validation and modeling flight

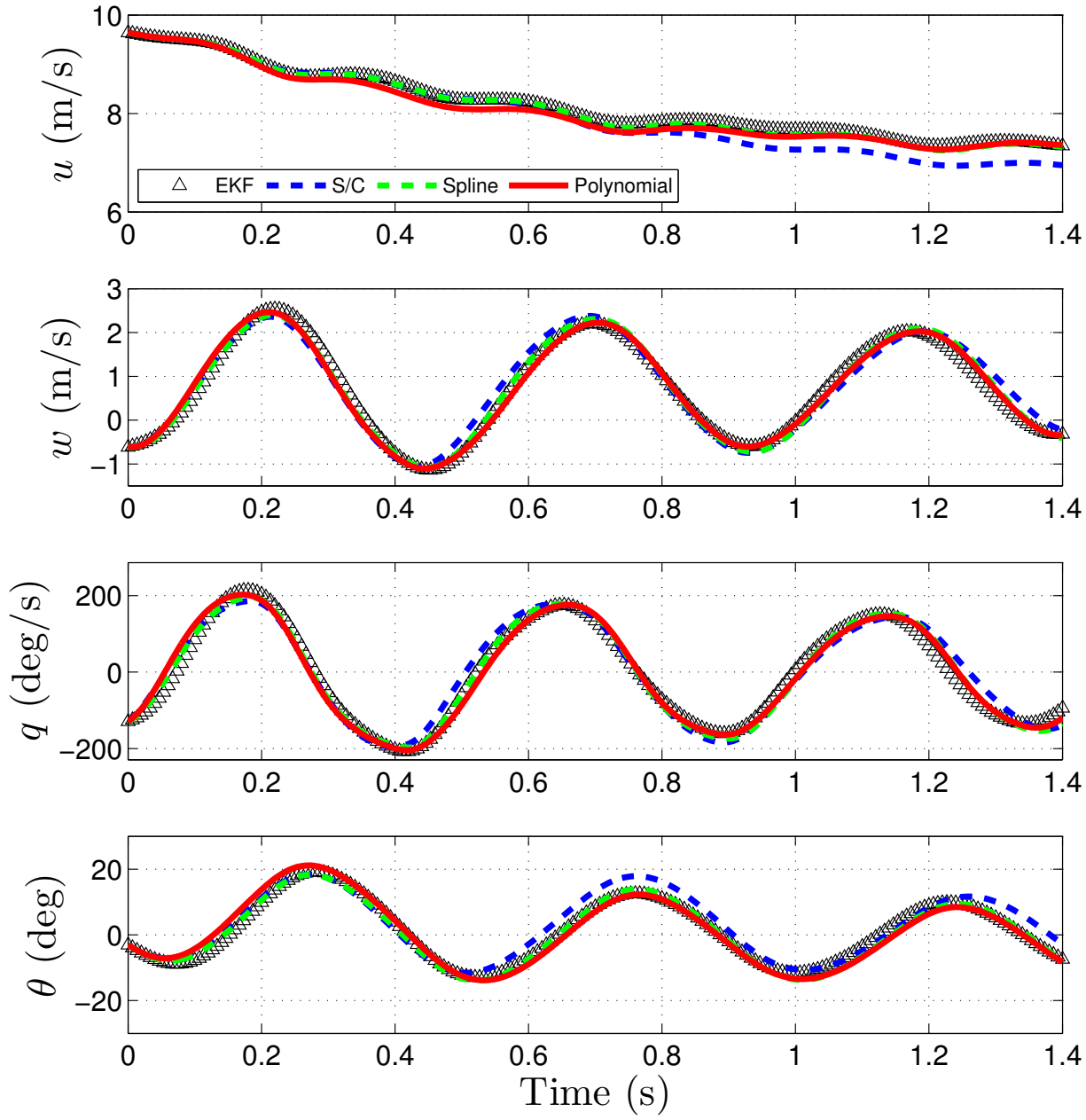


Figure 5.13: Forward simulation of validation Test 4.

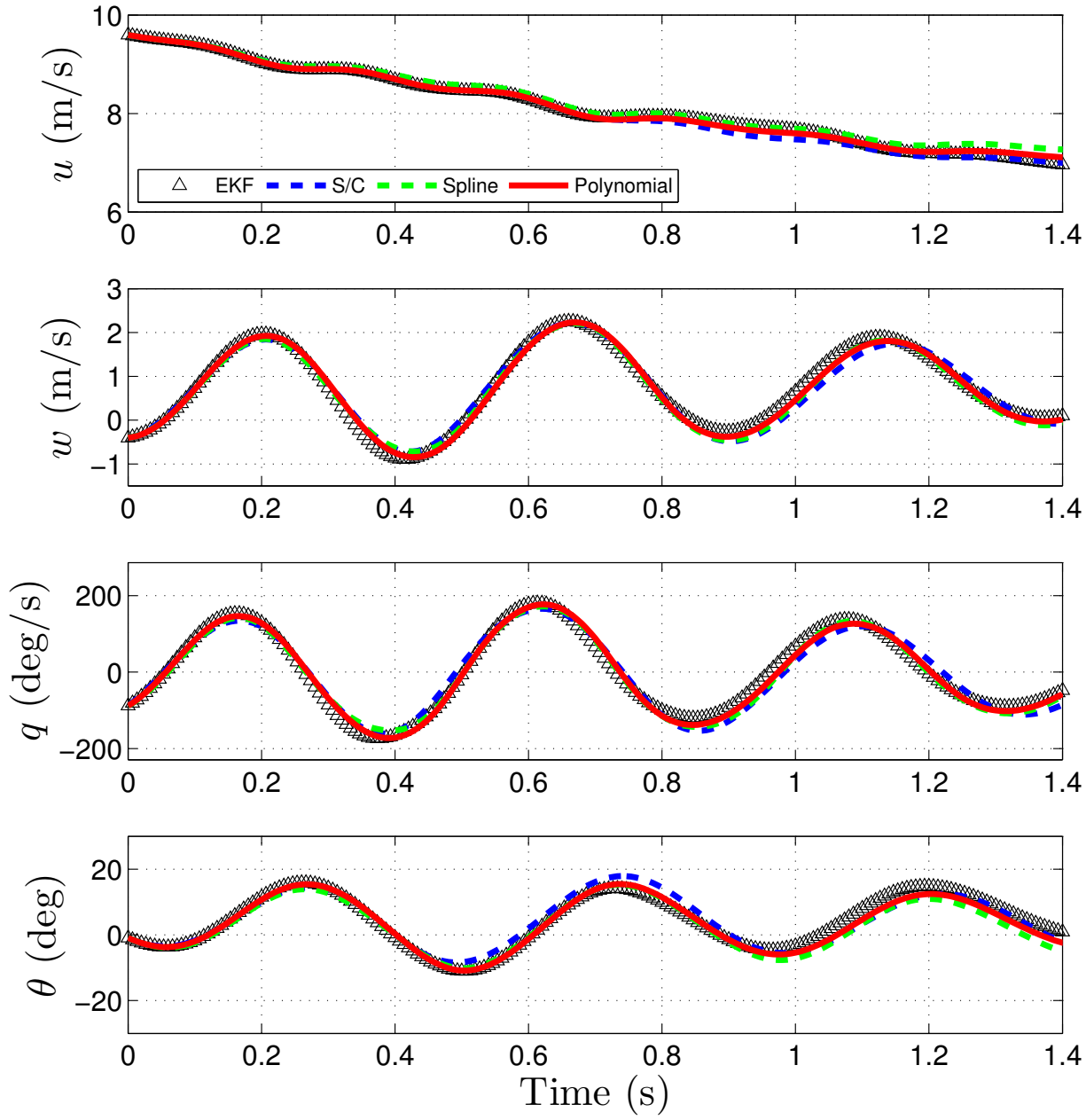


Figure 5.14: Forward simulation of modeling test 8.

tests are included in the average values. From the table, it is clear that the polynomial model structure is superior to the other models. The maximum TIC values for the longitudinal variables u, w, and q all fall below or within the acceptable range of 0.2-0.3 for both the spline and polynomial model; this is not the case for the stability and control derivative model. The worst case simulation prediction of the w velocity and q angular rate histories are rather poor for both the stability and control derivative model as well as the spline model in comparison to the polynomial model. From Fig. 5.13 and Fig. 5.14, in addition to Table 5.10, it is clear that the polynomial model is able to capture the behavior of the aircraft best out of the three model structures proposed.

For all three model structures, the match of θ time histories is poor in comparison to the other variables, however this is somewhat expected. The aerodynamic coefficient models reproduce the forces and moments acting on the aircraft, so any discrepancy, particularly in the pitching moment coefficient prediction, will result in deviation from the reference elevation angle. Note that substituting the EKF estimated elevation angle, θ_E, in the same manner that lateral-directional variables were substituted into the equations of motion, will improve the Theil inequality coefficient values for all models. This illustrates that although a particular model structure may be acceptable when examining the aerodynamic coefficient predictions individually, deficiencies in capturing the behavior of the aerodynamic coefficients may degrade the overall system performance in simulation.

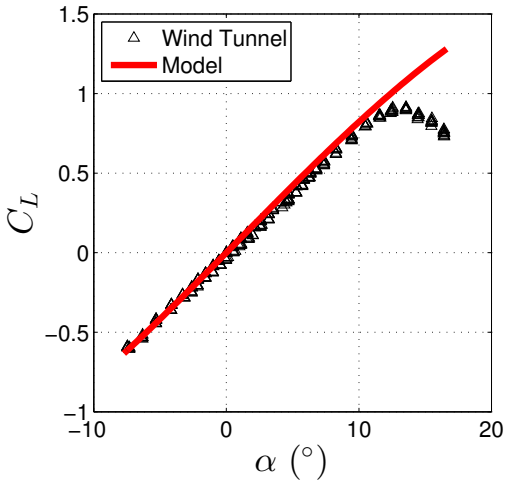
Table 5.10: Theil's Inequality Coefficient - Longitudinal variables from forward simulation.

Variable	TIC	S/C	Spline	Polynomial
u	average	0.1386	0.1226	0.1191
	maximum	0.3306	0.2789	0.2351
w	average	0.1320	0.0910	0.0831
	maximum	0.2467	0.2159	0.1354
q	average	0.1452	0.0986	0.0887
	maximum	0.3012	0.2627	0.1683
θ	average	0.2843	0.2263	0.1976
	maximum	0.4579	0.4825	0.3482

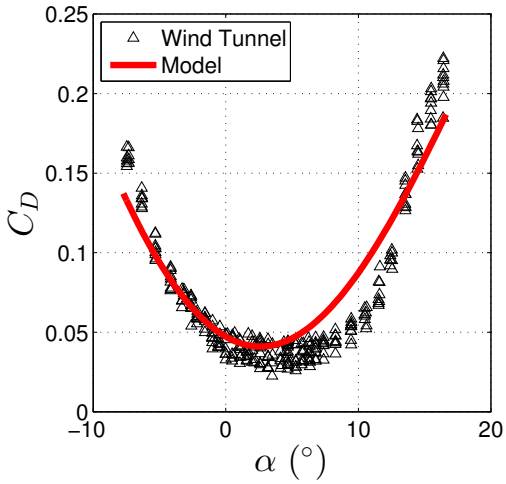
5.4.1 Comparison with Wind Tunnel Data

A comparison between wind tunnel data and the polynomial model identified from flight test data is given in Fig. 5.15. The wind tunnel data was obtained as part of a separate modeling effort. The model predicted coefficient values correspond to $\hat{q} = \delta_E = \dot{\alpha} = 0$. Based on Fig. 5.15(b) and 5.15(c), the drag and pitching moment coefficient models capture the general static aerodynamic behavior of the aircraft based on the wind tunnel data. The R^2 values between the wind tunnel prediction and model fit are 0.8146 and 0.2793 for C_D and C_m, respectively. The static lift coefficent predicted by the model, shown in Fig. 5.15(a), matches the wind tunnel data better, with $R^2 = 0.9708$. The general trends predicted by the model, i.e. the variation of static coefficients with respect to angle of attack, follow the trends of the wind tunnel data. The model, however, does not capture the stall behavior seen at approximately 12° angle of attack. Although the static C_L and C_D predictions from the model match the wind tunnel data reasonably well, it is not unexpected that the static predicted pitching moment does not match the wind tunnel measurements well. For the

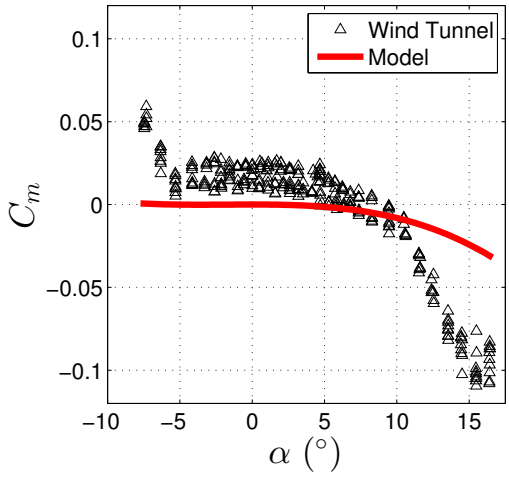
majority of flight tests, the mean angle of attack is $\approx 4°$; flight tests with a different mean angle of attack and amplitude may provide data to better capture the static pitch moment behavior, however this may be difficult to practically accomplish given the restrictions of the test facility.



(a) C_X vs. α



(b) C_Z vs. α



(c) C_m vs. α

Figure 5.15: Model predicted aerodynamic coefficients compared to wind tunnel data at $\hat{q} = \dot{\hat{\alpha}} = \delta_E = 0$.

Chapter 6

Conclusions & Future Work

In Chapter 3, graph-based search techniques are applied for motion planning using a finite library of dynamically feasible trajectories, referred to as motion primitives. By using a library of pre-specified motion primitives, the task of finding a dynamically feasible trajectory through an obstacle field is reduced to a search over a graph, in which the nodes correspond to the vehicle states and the edges represent the motion primitives. Developing the motion primitives according to a state lattice sampling benefits the search process with respect to computation time. For motion primitives that do not form a state lattice, a tree representation is instead used in order to maintain continuity of states between successive primitives. Consequently, this adversely affects the solution time for the search algorithms since a duplicate node with a higher cost-to-go, which in general will not match exactly the current node, is placed along with its descendants in the closed set to avoid a trajectory with gaps

or discontinuities, thus wasting previous computations. As an alternative, a locally greedy algorithm is presented. This algorithm exhibits improved performance with respect to run time over weighted A*, particularly when considering motion primitive libraries with multiple velocities. The improved performance in solution time can be attributed to the size of the search trees created by each algorithm. Specifically, the check to see whether a location within the configuration space has already been visited is far more computationally demanding for larger search trees, and the weighted A* algorithm constructs much larger search trees than the locally greedy algorithm. Ultimately, the success of both of these approaches is dependent on the heuristic used. Obstacle fields can be created where the Euclidean distance, for instance, is a poor estimate of the true cost-to-goal, resulting in computation times that are unacceptable. In these scenarios, a sampling-based planner may be preferable.

A control approach is also presented, which uses the ℓ_2-induced norm as the performance measure, providing a set of discrete-time controllers that accompany the motion primitive library. The solution returned by the motion planner consists not only of a sequence of motion primitives leading from the initial state to the goal, but also a set of controllers with stability and performance guarantees. An example of the hierarchical process applied to a hovercraft details the development of motion primitives, the design of subcontrollers using convex optimization, the motion planning task, and the execution of the motion plan in simulation subject to exogenous disturbances, measurement noise, and various uncertainties. The resulting hierarchical motion planning and feedback control strategy is able to guide the

hovercraft through an obstacle field while avoiding collisions. The benefit of this hierarchical approach is that the feedback control strategy is developed offline in conjunction with the motion primitive library, and thus does not require any additional computation when determining a dynamically feasible trajectory in real-time. Future work includes investigation of other motion planning algorithms, such as sampling-based methods, in order to determine their applicability to the problem posed herein.

In Chapter 4, time-domain system identification is applied to a small radio-controlled aircraft. Specifically, the two step method, or estimation before modeling, approach is utilized to identify an appropriate aerodynamic model structure and associated parameter values. The resulting model structure uses spline functions of the aircraft states and control inputs to capture the aerodynamic behavior throughout the flight test envelope. Model validation is performed through comparison of predicted aerodynamic coefficient time histories as well as simulation using the nonlinear equations of motion. Additionally, the application of pseudospectral optimal control software to generate desired dynamically feasible aircraft motions is illustrated. Specifically, an aggressive heading angle change that both starts and ends at a steady straight and level flight condition, subject to both input as well as nonlinear flight envelope constraints, is derived using the aforementioned aerodynamic model. Discrete-time feedback controllers are then developed to regulate the aircraft about the reference trajectory despite exogeneous disturbances and measurement errors. Controller performance is analyzed through both simulation in a realistic test environment as well as flight testing

of the actual aircraft system. The simulation and flight test results show that the aircraft is capable of tracking the pre-specified trajectory in calm atmospheric conditions. Future work includes developing a library of motion primitives in order to apply the control and motion planning approach of Chapter 3 to fixed-wing aircraft systems.

The time-domain aircraft system identification approach in Chapter 4 is applied to flight test data obtained from motion capture system in Chapter 5. Specifically, application of the estimation-before-modeling approach for determining time-varying aerodynamic forces and moments is demonstrated for flight tests where only position and attitude information is available. This approach utilizes an extended Kalman filter to estimate forces and moments as augmented system states. Additionally, aside from providing initial condition estimates for body-axis velocities and angular rates, this approach does not require any smoothing or numerical differentiation of individual measurements. Though flight tests are short in duration and exhibit large and rapid pitching motions, it is shown that a quasi-steady model utilizing polynomials of the indepedent variables is able to capture the aerodynamic behavior of the vehicle. Three candidate model structures are posed and then compared. The first utilizes a conventional stability and control derivative formulation for perturbations about a specific flight condition. The second model structure is an extension of the first, however it allows for variation of stability and control derivatives with respect to angle of attack as well as elevon deflection. Though these models are able to predict the variation in the aerodynamic coefficient time histories reasonably well, the quality of fit in simulation with

the models is sometimes lacking. The third model structure utilizes polynomials of the independent variables; while this model is nonlinear with respect to the independent variables, it remains linear in parameters. Based on the predicted aerodynamic force and moment coefficient time histories as well as model simulation, the polynomial model is clearly superior to the prior two models.

References

[1] Nilsson, N., *Problem-Solving Methods in Artificial Intelligence*, McGraw-Hill, New York, 1971. 1, 3, 15, 47

[2] LaValle, S., *Planning Algorithms*, Cambridge University Press, 2006. 1

[3] Frazzoli, E., Dahlah, M., and Feron, E., "Maneuver-based Motion Planning for Nonlinear Systems with Symmetries," *IEEE Transactions on Robotics*, Vol. 21, No. 6, 2005, pp. 1077–1091. 2, 7, 44, 61

[4] Goerzen, C., Kong, Z., and Mettler, B., "A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance," *Journal of Intelligent and Robotic Systems*, Vol. 57, No. 1-4, 2009, pp. 65–100. 2, 44

[5] Pearl, J., *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley, Reading, MA, 1984. 3, 15, 59

[6] Hansen, E. and Zhou, R., "Anytime Heuristic Search," *Journal of Artificial Intelligence Research*, Vol. 28, No. 1, 2007, pp. 267–297. 3

[7] Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., and Thrun, S., "Anytime Search in Dynamic Graphs," *Artificial Intelligence*, Vol. 172, No. 14, 2008, pp. 1613–1643. 3

[8] Kavraki, L., Svestka, P., Latombe, J., and Overmars, M., "Probabilistic Roadmaps for Path-Planning in High-Dimensional Configuration Spaces," *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 4, 1996, pp. 556–580. 3

[9] LaValle, S. and Kuffner, J., "Randomized Kinodynamic Planning," *International Journal of Robotics Research*, Vol. 20, No. 5, 2004, pp. 378–400. 3

[10] Karaman, S. and Frazzoli, E., "Incremental Sampling-Based Algorithms for Optimal Motion Planning," *Proceedings of Robotics: Science and Systems*, 2010. 3

[11] Karaman, S., Walter, M., Perez, A., Frazzoli, E., and Teller, S., "Anytime Motion Planning using the RRT*," *Proceeding of the 2011 IEEE Conference on Robotics and Automation*, Shanghai, China, 2011, pp. 1478–1483. 4

[12] Barraquand, J. and Latombe, J., "Nonholonomic Multibody Mobile Robots: Controllability and Motion Planning in the Presence of Obstacles," *Algorithmica*, Vol. 10, 1993, pp. 121–155. 4, 49, 57, 58

[13] Go, J., Vu, T., and Kuffner, J., "Autonomous Behaviors for Interactive Vehicle Animations," *Graphical Models*, Vol. 68, No. 2, 2006, pp. 90–112. 4

[14] Pancanti, S., Pallottino, L., Salvadorini, S., and Bichi, A., "Motion Planning through Symbols and Lattices," *Proceedings of the 2004 IEEE ICRA*, New Orleans, LA, 2004, pp. 3914–3919. 4

[15] Pivtoraiko, M., Knepper, R., and Kelly, A., "Differentially Constrained Mobile Robot Motion Planning in State Lattices," *Journal of Field Robotics*, Vol. 26, No. 3, 2009, pp. 308–333. 4, 46, 49

[16] Burridge, R., Rizzi, A., and Koditschek, D., "Sequential Composition of Dynamically Dextrous Robot Behaviors," *International Journal of Robotics Research*, Vol. 18, 2009, pp. 534–555. 5

[17] Conner, D., Choset, H., and Rizzi, A., "Flow-Through Policies for Hybrid Controller Synthesis Applied to Fully Actuated Systems," *IEEE Transactions on Robotics*, Vol. 25, No. 1, 2009, pp. 136–146. 5

[18] Lindemann, S. and LaValle, S., "Simple and Efficient Algorithms for Computing Smooth, Collision-Free Feedback Laws over Given Cell Decompositions," *International Journal of Robotics Research*, Vol. 28, 2009, pp. 600–621. 5

[19] Neas, C. and Farhood, M., "A Hybrid Architecture for Maneuver-Based Motion Planning and Control of Agile Vehicles," *Proceedings of the 18th IFAC World Congress*, Milano, Itay, 2011, pp. 3521–3526. 5, 38, 44, 53

[20] Grymin, D., Neas, C., and Farhood, M., "A Hierarchical Approach for Primitive-Based Motion Planning and Control of Autonomous Vehicles," *Robotics and Autonomous Systems*, 2013. 5, 12, 38, 39, 40, 41

[21] Hoffer, N., Coopmans, C., Jensen, A., and Chen, Y., "A Survey and Categorization of Small Low-Cost Unmanned Aerial Vehicle System Identification," *Journal of Intelligent Robot Systems*, Vol. -, No. -, 2013, pp. –. 5, 7

[22] Jategaonkar, R., *Flight Vehicle System Identification - a Time-Domain Methodology*, AIAA, Reston, VA, 2006. 6, 7, 22, 23, 24, 30, 31, 32, 81, 90, 97, 144, 150

[23] Tischler, M., *Aircraft and Rotorcraft System Identification: Engineering Methods with Flight Test Examples*, AIAA, Reston, VA, 2006. 6, 7, 22

[24] Klein, V. and Morelli, E., *Aircraft System Identification - Theory and Practice*, AIAA, Reston, VA, 2006. 6, 7, 8, 22, 23, 24, 26, 27, 30, 79, 81, 90, 94, 144, 150, 156, 160

[25] Mulder, J., Chu, Q., and Sridhar, J., "Decomposition of Aircraft State and Parameter Estimation Problem," *Proceedings of the 10th IFAC Symposium on System Identification*, 1994, pp. 61–66. 6, 22

[26] Mulder, J., Chu, Q., Sridhar, J., Breeman, J., and Laban, M., "Non-linear Aircraft Flight Path Reconstruction Review and New Advances," *Progess in Aerospace Sciences*, Vol. 35, 1999, pp. 673–726. 6, 23, 24, 26, 79, 81

[27] Raol, J., Girija, G., and Singh, J., *Modeling and Parameter Estimation of Dynamical Systems*, The Institution of Electrical Engineers, London, UK, 2004. 7, 22

[28] Stalford, H., "High-Alpha Aerodynamic Model Identification of T-2C Aircraft using the EBM Method," *Journal of Aircraft*, Vol. 18, No. 10, 1981, pp. 801–809. 7, 24, 26

[29] Sri-Jayantha, M. and Stengel, R., "Determination of Nonlinear Aerodynamic Coefficients Using the Estimation-Before-Modeling Method," *Journal of Aircraft*, Vol. 25, No. 9, 1988, pp. 796–804. 7, 22, 24, 26, 27, 79

[30] Oliveira, J., Chu, Q., Mulder, J., Balini, H., and Vos, W., "Output Error Method and Two Step Method for Aerodynamic Model Identification," *AIAA Guidance, Navigation, and Control Conference*, 2005. 7, 22, 23, 97

[31] Salman, S., Sreenatha, A., and Choi, J., "Attitude Dynamics Identification of Unmanned Aerial Vehicle," *International Journal of Control, Automation, and Systems*, Vol. 4, No. 6, 2006, pp. 782–787. 7, 22

[32] Jung, D., Levy, E. J., Zhou, D., Fink, R., Moshe, J., Earl, A., and Tsiotras, P., "Design and Development of a Low-Cost Test-Bed for Undergraduate Education in UAVs," *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, 2005, pp. 2739–2744. 7, 22

[33] Dorobantu, A., Murch, A., Mettler, B., and Balas, G., "System Identification for Small, Low-Cost, Fixed-Wing, Unmanned Aircraft," *Journal of Aircraft*, Vol. 50, No. 4, 2013, pp. 1117–1130. 7, 30

[34] Hall, J. and McLain, T., "Aerobatic Maneuvering of Miniature Air Vehicles Using Attitude Trajectories," *AIAA Guidance, Navigation, and Control Conference*, 2008, AIAA 2008-7257. 7

[35] Drury, R. and Whidborne, J., "Quaternion-Based Inverse Dynamics Model for Expressing Aerobatic Aircraft Trajectories," *Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 4, 2009, pp. 1388–1392. 7

[36] Park, S., "Autonomous Aerobatic Flight by Three-Dimensional Path-Following with Relaxed Roll Constraint," *AIAA Guidance, Navigation, and Control Conference*, 2011, AIAA 2011-6593. 7

[37] Hall, J., Beard, R., and McLain, T., "Quaternion Control for Autonomous Path Following Maneuvers," *AIAA Infotech Aerospace*, 2012, AIAA 2012-2483. 7

[38] Dever, C., Mettler, B., Feron, E., Popović, J., and McConley, M., "Nonlinear Trajectory Generation for Autonomous Vehicles via Parameterized Maneuver Classes," *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 2, 2006, pp. 289–302. 7

[39] Schouwennaars, T., Mettler, B., Feron, E., and How, J., "Hybrid Model for Trajectory Planning of Agile Autonomous Vehicles," *Journal of Aerospace Computing, Information, and Communication*, Vol. 1, 2004, pp. 629–651. 7

[40] Fisch, F., Lenz, J., Holzapfel, F., and Sachs, G., "Trajectory Optimization Applied to Air Races," *AIAA Atmospheric Flight Mechanics Conference*, 2009, AIAA 2009-5930. 7, 8

[41] Bittner, M., Fisch, F., and Holzapfel, F., "A Multi-Model Gauss Pseudospectral Optimization Method for Aircraft Trajectories," *AIAA Atmospheric Flight Mechanics Coneference*, 2012, AIAA 2012-4728. 7, 8

[42] Bittner, M., Bruhs, P., Richter, M., and Holzapfel, F., "An Automatic Mesh Refinement Method for Aircraft Trajectory Optimization Problems," *AIAA Guidance, Navigation, and Control Conference*, 2013, AIAA 2013-4555. 7, 8

[43] Bryson, A. and Ho, Y., *Applied optimal Control: Optimization, Estimation, and Control*, Taylor & Francis, 1975. 7, 62

[44] Stengel, R., *Optimal Control and Estimation*, Dover Publications, New York, 1994. 7

[45] Farhood, M. and Dullerud, G., "LMI Tools for Eventually Periodic Systems," *Systems & Control Letters*, Vol. 47, 2002, pp. 417–432. 8, 35, 36, 37, 79, 112

[46] Farhood, M. and Dullerud, G., "Duality and Eventually Periodic Systems," *International Journal of Robust and Nonlinear Control*, Vol. 15, 2005, pp. 575–599. 8, 35, 36, 79, 112

[47] Draper, N. and Smith, H., *Applied Regression Analysis*, Wiley, Hoboken, NJ, 1998. 8, 28, 29, 30

[48] Rhinehart, M. and Mettler, B., "Extracting Aerodynamic Coefficients using Direct Trajectory Sampling," *AIAA Atmospheric Flight Mechanics Conference*, 2008, AIAA Paper 2008-6899. 10

[49] Cory, R. and Tedrake, R., "Experiments in Fixed-Wing UAV Perching," *AIAA Guidance, Navigation and Control Conference*, 2008, AIAA Paper 2008-7256. 10, 133

[50] Mettler, B., "Extracting Micro Air Vehicles Aerodynamic Forces and Coefficients in Free Flight using Visual Motion Tracking Techniques," *Experiments in Fluids*, Vol. 49, No. 3, 2010, pp. 557–569. 10, 133

[51] Hoburg, W. and Tedrake, R., "System Identification of Post Stall Aerodynamics for UAV Perching," *AIAA Infotech Aerospace Conference*, 2009, AIAA Paper 2009-1930. 10

[52] Uhlig, D. and Selig, M., "Stability Characteristics of Micro Air Vehicles from Experimental Measurements," *29th AIAA Applied Aerodynamics Conference*, 2011, AIAA Paper 2011-3659. 10

[53] Uhlig, D. and Selig, M., "Determining Aerodynamic Characteristics of a Micro Air Vehicle Using Motion Tracking," *Journal of Aircraft*, Vol. 50, No. 5, 2013, pp. 1481–1490. 10, 133

[54] Meckstroth, C. and Reich, G., "Aerodynamic Modeling of Small UAV for Perching Experiments," *31st AIAA Applied Aerodynamics Conference*, 2013, AIAA Paper 2013-3191. 10, 11, 131, 133

[55] Savitzky, A. and Golay, M., "Smoothing and Differentiation of Data by Simplified Least Squares," *Analytical Chemistry*, Vol. 36, No. 8, 1964, pp. 1627–1639. 11, 134

[56] Lin, M. and Gottschalk, S., "Collision Detection Between Geometric Models: A Survey," *Proceedings of IMA Conference on Mathematics of Surfaces*, Vol. 1, 1998, pp. 602–608. 15

[57] Kockara, S., Halic, T., Iqbal, K., Bayrak, C., and Rowe, R., "Collision detection: A survey," *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*, 2007, pp. 4046–4051. 15

[58] Foley, J., *Computer Graphics: Principles and Practice*, Addison-Wesley systems programming series, Addison-Wesley, 1996. 15

[59] Cyrus, M. and Beck, J., "Generalized Two- and Three-Dimensional Clipping," *Computers & Graphics*, Vol. 3, No. 1, 1978, pp. 23 – 28. 15

[60] Phillips, W., *Mechanics of Flight*, Wiley, Hoboken, NJ, 2010. 16, 80

[61] Nelson, R., *Flight Stability and Automatic Control (Second Edition)*, McGraw Hill, Boston, MA, 1998. 16

[62] Etkin, B. and Reid, L., *Dynamics of Flight - Stability and Control (Third Edition)*, Wiley, Hoboken, NJ, 1996. 16

[63] Wang, K. and Iliff, K., "Retrospective and Recent Examples of Parameter Identification at NASA Dryden Flight Research Center," *Journal of Aircraft*, Vol. 41, No. 4, 2004, pp. 752–764. 22

[64] Hoffer, N., Coopmans, C., Jensen, A., and Chen, Y., "Small low-cost unmanned aerial vehicle system identification: A survey and categorization," *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, 2013, pp. 897–904. 22

[65] Morelli, E., "Global Nonlinear Aerodynamic Modeling Using Multivariate Orthogonal Functions," *Journal of Aircraft*, Vol. 32, No. 2, 1995, pp. 270–277. 22, 23

[66] de Visser, C., *Global Nonlinear Model Identification with Multivariate Splines*, Ph.D. thesis, T.U. Delft, 2011. 22, 23, 28, 79

[67] Morelli, E., "Efficient Global Aerodynamic Modeling from Flight Data," *50th AIAA Aerospace Sciences Meeting*, 2012, AIAA Paper 2012-1050. 22, 23, 144

[68] Paris, A. and Bonner, M., "Nonlinear Model Development from Flight-Test Data for F/A-18E Super Hornet," *Journal of Aircraft*, Vol. 41, No. 4, 2004, pp. 692–702. 23, 30

[69] Stalford, H., "Application of the Estimation-Before-Modeling (EBM) System Identification Method to the High Angle of Attack/Sideslip Flight of the T-2C Jet Trainer Aircraft. Volume III. Identification of T-2C Aerodynamics Stability and Control Characteristics from Actual Flight Test Data." Tech. rep., United States Navy, 1979, AD-A079 924/7. 24

[70] Gibbs, B., *Advanced Kalman Filtering, Least-Squares and Modeling: A Practical Approach*, Wiley, Hoboken, NJ, 2011. 25, 28

[71] Rauch, T., Tung, F., and Striebel, C., "Maximum Likelihood Estimates of Linear Dynamic Systems," *AIAA Journal*, Vol. 3, 1965, pp. 1445–1450. 27

[72] Crassidis, J. and Junkins, J., *Optimal Estimation of Dynamic Systems*, Chapman & Hall/CRC Press, Boca Raton, FL, 2004. 28

[73] Theil, H., *Economic Forecasts and Policy*, North-Holland Publishing Company, Amsterdam, Netherlands, 1961. 31, 32

[74] Dullerud, G. and Lall, S., "A New Approach to Analysis and Synthesis of Time-Varying Systems," *IEEE Transactions on Automatic Control*, Vol. 44, No. 8, 1999, pp. 1486–1497. 35, 40, 116

[75] Sturm, J., "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optimization Methods and Software*, Vol. 11–12, 1999, pp. 625–653, Version 1.05 available from `http://fewcal.kub.nl/sturm`. 40, 116

[76] Toh, K., Todd, M., and Tutuncu, R., "SDTP3, - A Matlab Software Package for Semidefinite Programming," *Optimization Methods and Software*, Vol. 11, 1999, pp. 545–581. 40, 62, 75

[77] Dullerud, G. and Lall, S., "A New Approach to Analysis and Synthesis of Time-Varying Systems," *IEEE Transactions on Automatic Control*, Vol. 44, No. 8, 1999, pp. 1486–1497. 44

[78] Farhood, M. and Dullerud, G., "LMI Tools for Eventually Periodic Systems," *Systems and Control Letters*, Vol. 47, No. 5, 2002, pp. 417–432. 44

[79] Gillula, J., Huang, H., Vitus, M., and Tomlin, C., "Design of Guaranteed Safe Maneuvers Using Reachable Sets: Autonomous Quadrotor Aerobatics in Theory and Practice," *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 1649–1654. 44

[80] Sanfelice, R. and Frazzoli, E., "A Hybrid Control Framework for Robust Maneuver-Based Motion Planning," *Proceedings of the 2008 American Control Conference*, 2008, pp. 2254–2259. 44

[81] Hart, P., Nilsson, N., and Raphael, B., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, Vol. 4, No. 2, 1968, pp. 100–107. 47, 59

[82] Pohl, I., "Heuristic Search Viewed as Path Finding in a Graph," *Artificial Intelligence*, Vol. 1, No. 3-4, 1970, pp. 193–204. 47

[83] Ebendt, R. and Drechsler, R., "Weighted A* Search - Unifying View and Application," *Artificial Intelligence*, Vol. 173, No. 14, 2009, pp. 1310–1342. 47

[84] Neas, C., *A Greedy Search Algorithm for Maneuver-Based Motion Planning of Agile Vehicles*, Master's thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, 2010. 53

[85] Russell, S. and Norvig, P., *Artificial Intelligence: A Modern Approach*, Prentice Hall, New York, 2010. 59

[86] Cormen, T., Leiserson, C., and Rivest, R.L. Stein, C., *Introduction to Algorithms*, Vol. 3, MIT Press, Cambridge, MA, 2009. 59

[87] Ross, I. and Fahroo, F., *Legendre Pseudospectral Approximations of Optimal Control Problems*, Vol. 295, Springer-Verlag, 2003. 62

[88] Rao, A., Benson, D., Darby, C., Patterson, M., Fancolin, C., and Huntington, G., "Algorithm 902: GPOPS, A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using the Gauss Pseudospectral Method," *ACM Transactions on Mathematical Software*, Vol. 37, No. 2, 2010, pp. 1–39. 62

[89] Gong, Q., Kang, W., Bedrossian, N., Fahroo, F., Sekhavat, P., and Bollino, K., "Pseudospectral Optimal Control for Military and Industrial Applications," *Proceedings of the 46th IEEE Conference on Decision and Control*, 2007, pp. 4128–4142. 62

[90] Grant, M. and Boyd, S., "CVX: Matlab Software for Disciplined Convex Programming, Version 1.21," http://cvxr.com/cvx, September 2010. 62

[91] Minka, T., "The Lightspeed Matlab Toolbox, Efficient Operations for Matlab Programming, Version 2.2," December 2007, Available from http://research.microsoft.com/en-us/um/people/minka/software/lightspeed/. 65

[92] Devore, J., *Probability and Statistics for Engineering and the Sciences, 6th ed.*, Duxbury Press, 2003. 66

[93] Lfberg, J., "YALMIP : A Toolbox for Modeling and Optimization in MATLAB," *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004. 75, 116

[94] Hepperle, M., "JavaProp - Design and Analysis of Propellers," Tech. rep., MH Aerotools, 2010, available from http://www.mh-aerotools.de. 89

[95] Rao, A. V., Benson, D., Darby, C., Patterson, M., Francolin, C., Sanders, I., and Huntington, G., "Algorithm 902: GPOPS, A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using the Gauss Pseudospectral Method," *ACM Transactions on Mathematical Software*, Vol. 37, No. 2, 2010, pp. 22:1–39. 102

[96] Garg, D., Patterson, M., Hager, W., Rao, A., Benson, D., and Huntington, G., "A Unified Framework for the Numerical Solution of Optimal Control Problems Using Pseudospectral Methods," *Automatica*, Vol. 46, No. 11, 2010, pp. 1843–1851. 102

[97] Garg, D., Hager, W., and Rao, A., "Pseudospectral Methods for Solving Infinite-Horizon Optimal Control Problems," *Automatica*, Vol. 47, No. 4, 2011, pp. 829–837. 102

[98] Garg, D., Patterson, M., Darby, C., Francolin, C., Huntington, G., Hager, W., and Rao, A., "Direct Trajectory Optimization and Costate Estimation of Finite-Horizon and Infinite-Horizon Optimal Control Problems via a Radau Pseudospectral Method," *Computational Optimization and Applications*, Vol. 49, No. 2, 2011, pp. 335–339. 102

[99] Wächter, A. and Biegler, L., "On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming," *Mathematical Programming*, Vol. 106, No. 1, 2006, pp. 25–57. 102

[100] Stevens, B. and Lewis, F., *Aircraft Control and Simulation - Second Edition*, Wiley, Hoboken, NJ, 2003. 112

[101] Packard, A., "Gain Scheduling via Linear Fractional Transformations," *Systems and Control Letters*, Vol. 22, 1994, pp. 79–92. 116

[102] Gahinet, P. and Apkarian, P., "A Linear Matrix Inequality Approach to H_∞ Control," *International Journal of Robust and Nonlinear Control*, Vol. 4, 1991, pp. 421–448. 116

[103] Moorhouse, D. and Woodcock, R., "Background Information and User Guide for MIL-F-8785C, Military Specification - Flying Qualities of Piloted Airplanes," Tech. rep., Air Force Wright Aeronautical Labs, Wright-Patterson AFB, OH, 1982. 117

[104] Beal, T., "Digital Simulation of Atmospheric Turbulence for Dryden and von Kármán Models," *Journal of Guidance, Control, and Dynamics*, Vol. 16, No. 1, 1993, pp. 132–138. 117

[105] Arifianto, O. and Farhood, M., "A Low-Cost Unmanned Aerial Vehicle Research Platform: Development and Modeling," *Submitted to AIAA Journal of Aircraft*, 2013. 120

[106] Robertson, D. and Reich, G., "Design and Perching Experiments of Bird-like Remote Controlled Planes," *54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2013, AIAA Paper 2013-1788. 131

[107] Regisford, S. and VanderMey, J., "Perching a Minimally-Actuated Micro Air Vehicle," *51st AIAA Aerospace Sciences Meeting*, 2013, AIAA Paper 2013-0359. 131

[108] DreamFlight, "Alula Glider," Oct. 2013. 130

[109] "Micro Air Vehicle Integration & Application Research Institute Fact Sheet," Oct. 2013. 129

[110] "Vicon Motion Systems, Ltd." Oct. 2013. 129

[111] Morelli, E., "System Identification Programs for AirCraft (SIDPAC)," *AIAA Atmospheric Flight Mechanics Conference*, 2002, AIAA Paper 2002-4704. 144, 155, 156