HYBRID SYSTEM DESIGN AND EMBEDDED CONTROLLER SYNTHESIS FOR MULTI-MODAL CONTROL

by

Takkuen John Koo

Memorandum No. UCB/ERL M00/46

11 August 2000

HYBRID SYSTEM DESIGN AND EMBEDDED CONTROLLER SYNTHESIS FOR MUTLI-MODAL CONTROL

by

Takkuen John Koo

Memorandum No. UCB/ERL M00/46

11 August 2000

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

HYBRID SYSTEM DESIGN AND EMBEDDED CONTROLLER SYNTHESIS FOR MULTI-MODAL CONTROL

by

Takkuen John Koo

B.Eng. (The Chinese University of Hong Kong) 1992
M.Phil. (The Chinese University of Hong Kong) 1994

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Electrical Engineering and Computer Sciences

in the

GRADUATE DIVISION

of the

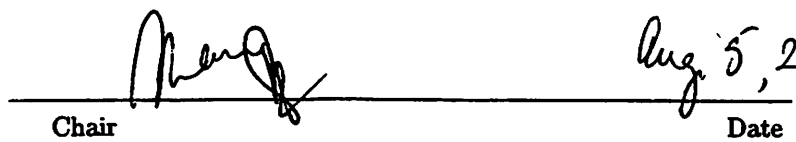
UNIVERSITY of CALIFORNIA at BERKELEY

Committee in charge:

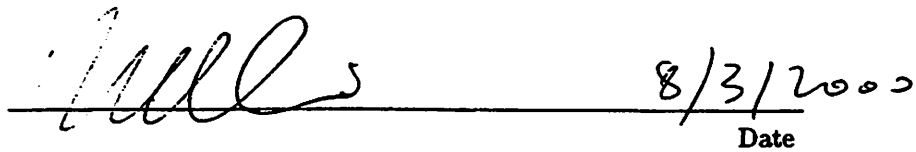
Professor S. Shankar Sastry, Chair
Professor Alberto Sangiovanni-Vincentelli
Professor Andrew Packard

Fall 2000

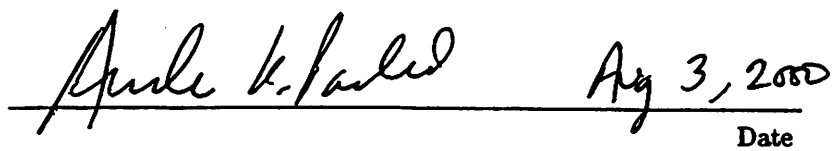
The dissertation of Takkuen John Koo is approved:



Chair Date



Date



Date

University of California at Berkeley

Fall 2000

HYBRID SYSTEM DESIGN AND EMBEDDED CONTROLLER SYNTHESIS FOR MULTI-MODAL CONTROL

by

Takkuen John Koo

Abstract

HYBRID SYSTEM DESIGN AND EMBEDDED CONTROLLER SYNTHESIS FOR MULTI-MODAL CONTROL

by

Takkuen John Koo

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Sciences

University of California at Berkeley

Professor S. Shankar Sastry, Chair

Based on operational, financial and environmental considerations, large scale systems such as automated highway systems, air traffic management systems, and unmanned aerial vehicle networks have been advocated to have higher levels of automation. By nature the systems are distributed and highly dynamic, the environments around the systems are rapidly changing, and multi-objective design specifications intensify the complexity of system design. To manage the design complexity, multi-modal control paradigm, in which control systems are designed by hierarchically nesting of composition of modes of operation such that each mode of operation is designed to cope with a designated scenario with respect to a design specification while the organization of modes of operation depends on the ordering of these specifications, is proposed.

A multi-modal control system can be modeled as a hierarchical nesting of parallel and serial composition of discrete and continuous components. A model of computation (MOC) governs the behaviors and interactions of components at each level of the hierarchy. The control system is a hybrid system which is composed of different MOCs. The formal synthesis of embedded controllers is interpreted as the generation of an architecture mapping from discrete-continuous components to hardware-software components to ensure that the implementations are correct by construction.

This dissertation is focused on a formal approach of hybrid system design and embedded controller synthesis for multi-modal control. Throughout the dissertation, a helicopter based unmanned aerial vehicle is used as a design example. First, modal controllers based on feedback linearization and differential flatness for a nonlinear non-minimum phase helicopter model are designed. Second, a general framework is developed for the derivation of control modes switching which satisfy reachability specifications. Third, embedded system synthesis based on a formal methodology is presented. Various MOCs are used and translated in different design stages according to the design properties of MOCs. Depending on the choice of architecture, the components specified by the MOCs are then mapped to hardware and software components.

Professor S. Shankar Sastry
Dissertation Committee Chair

To my family.

Contents

List of Figures vi

1 Introduction 1

1.1 Research Areas 4

1.2 Dissertation Outline 7

2 System Model and Architecture 8

2.1 Models of Computation 9

2.2 Hybrid Automata 10

2.3 System Architecture 15

3 Nonlinear Control Design 19

3.1 Introduction 20

3.2 Helicopter Model 22

3.2.1 Rigid Body Dynamics 22

3.2.2 Force and Moment Generation Processes 24

3.2.3 System equations 26

3.3 Exact Linearization by State Feedback 27

3.4 Approximate Linearization by State Feedback 34

3.5 Trajectory Generation based on Differential Flatness 38

3.6 Nonlinear Control Design Based on Outer Flatness 40

3.7 Simulation Results 47

3.8 Chapter Summary 51

4 Control Mode Switching Synthesis **53**

4.1 Introduction 53

4.2 Problem Formulation 54

4.2.1 Control Modes 55

4.3 A Mode Switching Condition 59

4.4 Mode Sequence Synthesis 61

4.4.1 Control Mode Graph 61

4.5 Reachability Computations 63

4.6 The Synthesis Procedure for Control Mode Switching 67

4.7 Chapter Summary 68

5 Embedded Controller Synthesis **70**

5.1 Introduction 70

5.2 Polis Co-design Methodology 72

5.3 System Design and Synthesis 75

5.4 Implementation and Validation 83

5.5 Chapter Summary 87

6 Conclusions **89**

A Appendix **92**

A.1 System parameters 92

A.2 Proof of Proposition 3.3.4 93

Bibliography **94**

List of Figures

1.1 A group of UAVs perform formation flying. (The animation is generated by SmartAerobots) 5

2.1 Definition of Hybrid Automaton 11

2.2 Each agent is equipped with a Flight Management System(FMS) which is reactive to mission planner and environments 16

2.3 In the directed graph, each node represents a mode of operation and each arrow connects a higher priority to a lower one. 17

3.1 Helicopter dynamics 22

3.2 Coordinate frames for specifying rigid motions and forces acting on a helicopter. 23

3.3 The free-body diagram of a helicopter in flight. (Figure courtesy of D. H. Shim) 25

3.4 Phase portrait of zero dynamics in position and heading control mode . . . 33

3.5 Partitioned inner and outer systems. 41

3.6 Block diagram of control scheme. 43

3.7 Exact input-output linearization applied on outputs $\{ p_x, p_y, p_z, \psi\}$ . . . 48

3.8 Approximate input-output linearization applied on outputs $\{p_x, p_y, p_z, \psi \}$ 49

3.9 Nonlinear control design based on outer flatness with outputs $\{ p_x, p_y, p_z, \psi\}$ Notice that the dashed lines represent the desired trajectories of ϕ, θ, T_M. . 50

3.10 An unmanned aerial vehicle of Berkeley Aerobot fleet: Ursa Minor. 52

4.1 Control mode switching for high-altitude take-off of a helicopter. The switching sequence and conditions are based on flight instruction for pilots 58

4.2 Two control modes: position mode and velocity mode 63

4.3 Control mode graph of the example 66

5.1 Polis design flow chart 72
5.2 Search and investigate scenario: to navigate through the way points and detect objects of interest along the path 76
5.3 System Architecture 78
5.4 FSM representation of Tactical Planner 79
5.5 Transitions diagram of helicopter maneuvers 80
5.6 Control mode graph of helicopter maneuvers. 81
5.7 FSM representation of Trajectory Planner. 82
5.8 System block diagram in Ptolemy 83
5.9 Simulation result using synchronous model: x, y, z, ψ from top to bottom . 84
5.10 Simulated 3D Trajectory 85
5.11 Simulation Result after HW-SW partition: x, y, z, ψ from top to bottom . . 86
5.12 Instability occurs in Z dynamics 87

Acknowledgements

I would like to express my gratitude and appreciation to my advisor, Prof. S. Shankar Sastry, for all of his advice, support, encouragement, and enthusiasm over the years. His passion for science has been a great inspiration to me. I cannot thank him enough for granting me the opportunities to visit so many different universities, for introducing me to as many interesting people as interesting research topics, and for supporting me to start such an ambitious research project in the beginning of my study. I will always grateful for everything he has taught me about science and life.

I would like to thank Professor Andrew Packard for being my *Big Brother* throughout my study at Berkeley, Professor Alberto Sangiovanni-Vincentelli for introducing me to computer-aided-design for embedded systems, and Prof. Pravin Variya for his invaluable comments about my research. Also, I would like to thank them for serving on my Qualifying and Dissertation Committees. I would like to thank Prof. Thomas Henzinger and Prof. Edward Lee for illuminating discussions on hybrid and embedded systems. I wish to express my gratitude to Prof. Lotfi Zadeh for extending my horizons on research.

My graduate experience would not have been as special without all the bright students and postdocs. I would like to thank my *Big Brothers and Sisters*, namely Lara Crawford, Datta N. Godbole, John Lygeros, John Hauser, George J. Pappas, Claire J. Tomlin, and Jeff Wendlandt for taking care of their little brother. I wish to thank Magnus Egerstedt, John-Morten Godhavn, Joao Hespanha, Frank Hoffmann, Karl H. Johansson, Jana Kosecka, Maria Prandini, Sepanta Sekhavat, and Slobodan Simic for making this international intelligent exchange possible. It is a pleasure to acknowledge the research collaborations with my colleagues at Berkeley, namely H. Jin Kim, Cedric Ma, Yi Ma, Will Morrison, Kirill Mostov, Santosh Philip, Shahid Rashid, Shawn Schaffert, Omid Shakernia, Cory Sharp, Hyunchul Shim, Bruno Sinopoli, Rene Vidal, for making the Berkeley Aerial Robot project possible. I am pleased to thank Benjamin Horowitz, Jie Liu, Xiaojun Liu, Aniruddha Pant, and Pete Seiler for working together on various exciting research projects. My special thank goes to Peter Ray for showing me the beauty of *the art of war*.

My life at Berkeley would not be complete and enjoyable without friends on the globe. I would like to thank Kostas Adam, Nick Biziouras, Cenk Cavusoglu, Tak Chan, Daniel Chan, Eddie Chan, Samson Cheung, Ken Chiang, Brian Chien, Vincent Chu, Victor Fu,

Wing Ko, Raymond Kwong, Ringo Lam, Thomus Leung, Mathew Mo, Gabriel Moy, Claudio Pinello, Luca Schenato, Marco Sgroi, Benedict Tse, Kenneth Tsui, Calvin Watt, Tommy Woo, Joseph Yan, and Jm Yu for sharing such a wonderful experience with me.

I am pleased to thank DARPA(F33615-98-C-3614), ARO(DAAH04-96-1-0341), and ONR(N00014-97-1-0946) for financial support.

My final thanks go to my family for much love, encouragement and support. I would especially like to thank my parents for providing me with full autonomy to pursue my dreams. Little did they know that their 5 years old son would be in school for 24 years and get a Ph.D. degree from Berkeley.

Chapter 1

Introduction

Large scale systems ranging from automated highway systems, air traffic management systems, unmanned aerial vehicle networks, communication networks and power distribution networks have been advocated to have higher levels of automation based on operational, financial and environmental considerations. However, each of these systems by nature is distributed as a large number of subsystems and the interactions between subsystems are complicated. Furthermore, each subsystem is highly dynamic and the environments in which the system resides in are usually rapidly evolving. All these reasons make the design and analysis for the control of large scale systems a challenging problem.

Multi-objective design requirements always result in contradictory demands on the design and this in turn has adverse effects on the complexity of system design especially for large scale systems. One natural way to reduce the complexity of system design is by compositional methods. Compositional methods attempt to solve a complex problem by decomposing the problem into a sequence of smaller problems of manageable complexity. Imposing a hierarchical structure on the system architecture has been used for solving the control problem of large scale systems. In a multi-agent system point of view, each subsystem is modeled as an agent with sensing, computation and communication capabilities. Behaviors of agents are governed by a control policy derived in an attempt to fulfill system-wide objectives. A control policy comprises tasks and each agent is assigned a task to perform.

The criteria for the selection of a control policy depend on the availability of essential

information and the complexity of computing a solution. The first criterion depends on the amount and types of information collected from sensors and exchanged via communication. The second criterion comes from the formulation of the problem and the computation capabilities of the planner and agents. Due to advances in technology, the realization of different control policies with various levels of centralization is now feasible. As a matter of fact, a completely decentralized policy could produce inefficient local solution and a completely centralized policy is usually prohibitively complex or expense to be computed. A balance needs to be sought between completely centralized and decentralized scheme.

Semi-autonomous agent control has been proposed in [66] for resolving the design conflict in providing feasible solutions to the problem. In the scheme, given a task, each agent computes its own nominal strategy to execute the task. On the other hand, conflict among agents could arise due to the presence of contingent events especially when agents are operating in a rapidly changing environment. In this case, agents would attempt to derive strategies with or without coordination to resolve the conflict. This naturally requires that agents can be operated in different discrete modes of operation for providing the balance between optimal behaviors and conflict resolution.

To execute the given strategy, each agent chooses its own course based on its dynamical properties and physical constraints. Nevertheless, in the control design for each agent, a similar balance has to be made. Due to the dynamical properties and physical constraints, it is in general difficult to derive a single controller being able to execute a given strategy while meeting multiple performance objectives such as fast response and good tracking capabilities, and protecting the operation envelope in the presence of unanticipated external disturbance. However, in many design cases, designing controller satisfying partial requirements is feasible. This naturally suggests that a realistic approach is to have the controlled system designed to be able to switch between different modes of operation by utilizing different controllers in order to resolve the design conflict.

In summary, the controlled system are designed by hierarchically nesting of composition of modes of operation. Each mode of operation suggests a discrete state of the system. In terms of formal modeling framework, the multi-modal controlled system can be modeled as a hierarchically nesting of parallel and serial composition of discrete and continuous components. Furthermore, a model of computation (MOC) [62] governs the behaviors and interactions of components at each level of the hierarchy. Hybrid systems in the general

sense of the term could be considered as formalisms used to describe a complex system as combinations of MOCs where a single one is not powerful enough or expressive enough. At each level of hierarchy, control derivation is based on a level of abstraction provided by the governing MOC. This naturally leads to generalize the design problem for the control of large scale systems as a problem of multi-modal control derivation under the modeling framework of hybrid system.

An embedded system is generally represented as a set of components which interact with each other and with the environment which is given and cannot be designed. The notion of components here is referred to a model for software and hardware that interacts with a physical environment in real time. Advances in embedded hardware and software have enabled the rapid realization of sophisticated, high-performance control systems. However, the design methodologies and design tools for the control of large scale, highly complex systems to deliver high levels of mission reliability especially in dynamic and rapidly evolving environment are inadequate. Consequently, today most of the cost in system development is spent on ad-hoc, prohibitively expensive system integration and validation techniques that rely almost exclusively on testing more or less complete versions of the entire system.

Hybrid systems refers to the distinguishing fundamental characteristics of embedded control systems, namely, the tight coupling and interaction of discrete with continuous phenomena. Hybridness is a characteristic of embedded control systems because every digital hardware/software implementation of a control design is ultimately a discrete approximation interacts through sensors and actuators with a continuous physical environment. The model of computation defines the behavior and interaction of these components. In formal embedded system design, compactness of description, fidelity to design styles, ability to verify and simulate, synthesize to an appropriate implementation and optimize its behavior are criteria to follow for the choice of an MOC to describe and manipulate a design. Its aim is to ensure the requirements and constraints are met at the system level in a formally verifiable manner in order to reduce extensively expensive down stream design errors. In the modeling framework of hybrid systems, formal synthesis of embedded controllers can be in general interpreted as automatic generation of architecture mapping from discrete/continuous components to hardware/software components to ensure the implementations that are correct by construction.

This dissertation will be primarily focused on formal approach of hybrid system design

and embedded controller synthesis for multi-modal control. Special emphasis will be put on scalability and interoperability to cover the complete design flow from architecture conception, control law derivation, and mode switching synthesis all the way to the generation of real-time embedded controller code. Helicopter based unmanned aerial vehicle (UAV) will be used as an example to demonstrate the effectiveness of the proposed design concepts for solving multi-modal control problem.

1.1 Research Areas

Advances in technology are revolutionizing the development of advanced control technologies for UAV systems and are enablers for the conduct of missions deemed impossible in the recent past. In the following, we list a few of significant issues associated with the design problem of multi-agent multi-modal control derivation for UAV systems. Research on these issues has been actively conducted in and related results has published under Berkeley Aerial Robot (BEAR) project.

Multi-agent Coordination and Conflict Resolution

UAVs are being demanded to be used more and more in a number of civilian and military operations such as search and rescue, inspection of power lines, terrain surveying, and investigation of hazardous waste sites. Successful development and deployment of UAVs eliminates the risk of human life involved in any such operation. Since the environments in which UAVs are operating usually are rapidly evolving, this requires UAVs to be capable of making autonomous decisions, coordinating their decisions with other vehicles, and being reactive to possible contingent events. In most of the UAV related missions, the control problems can be generalized as a pursuit-evasion game which is the problem of controlling a swarm of autonomous agents in the pursuit of static or moving evaders. Different approaches have been proposed in solving this problem either in deterministic [75, 84] or probabilistic framework [44] based on complete or partial information about the environment. Agents can communicate and exchange information via a dynamic mobile network which can be dynamically reconfigured based on current spatial topology of the agents. Coordination of the autonomous agents in close proximity to maintain mesh stability has been studied in

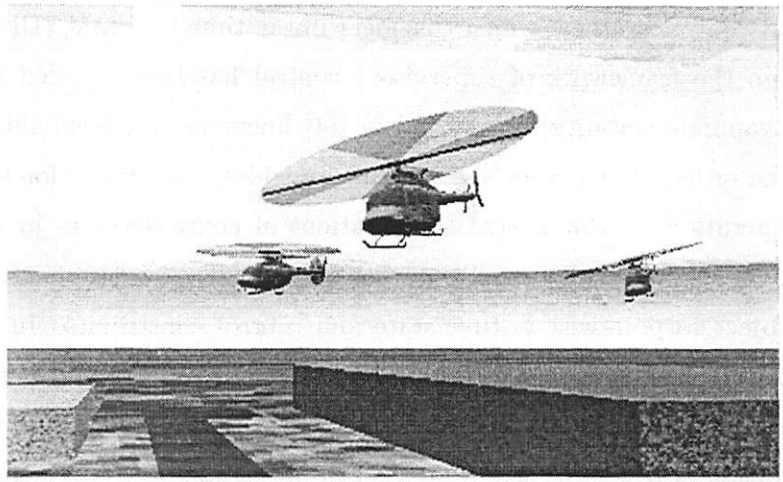


Figure 1.1: A group of UAVs perform formation flying. (The animation is generated by SmartAerobots)

[89] regarding possible information structures and topological configurations. Due to failure in establishing communication among agents, the trajectories of different agents may come into conflict. Conflict resolution using non-cooperative methods based on game theory in which each agent models the actions of other agents as disturbances and chooses its actions to be safe against the worst possible disturbance are proposed by [68]. In [75, 84, 44], the solution generated by the approaches are purely discrete. Whereas, in [89], the approach provides purely continuous solution. In [68], a hybrid solution which contains continuous trajectories and discrete strategies is derived based on the Hamilton-Jacobi equations.

Single-agent Multi-modal Control

To accomplish a mission, an UAV has to be able to perform extremely especially in a rapidly evolving environment in order to execute a given navigation task: controlling its state from one point to another one in a specified way. On the other hand, the agent must protect its states from being operated outside the operation envelope. In order to cope with different and possibly conflicting tasks, an agent should be designed to be able to switch modes of operation based on state information and use different controllers whose design are based on given design specifications for operating within the modes. This multi-modal control system can be modeled as a hybrid system [13]: a discrete transition system with closed loop dynamics embedded in each discrete locations. Stability of a subclass of the

multi-modal control systems which considers linear time-invariant (LTI) plants with LTI controllers in the framework of supervisory control has been studied by [36, 70, 5, 11] based on Lyapunov stability theory, and in [11] linear matrix inequality (LMI) technique is used. The ordering of design specification provides a guideline for the organization of modes of operation. While design specifications of controllers are in total ordering, the problem of deriving switching conditions has been studied by [45] for classes of dynamical systems subject to pointwise-in-time state and control constraints. In [68], the problem of systematically synthesizing hybrid controllers based on optimal control which satisfy multiple design specifications is considered. In both cases, the synthesis procedures involve the computation of reachable sets from a set of initial states. Hence, exact solutions to the synthesis problem can be only obtained for limited classes of dynamical systems in which the reachability problem is decidable. For general dynamical systems, numerical approximation has to be used for reachable set computation.

Vehicle Dynamics and Control

Helicopter [85] is versatile in maneuverability and this makes helicopter based UAVs indispensable for many applications where human intervention, especially in restricted areas, is considered difficult or dangerous. Helicopter control [74, 85] requires the ability to produce moments and forces on the vehicle for two purposes: first, to produce equilibrium and thereby hold the helicopter in a desired trim state; and second, to produce accelerations and thereby change the helicopter velocity, position and orientation. Like airplane control, helicopter control is accomplished primarily by producing moments about all three aircraft axes: roll, pitch, and yaw. The helicopter has in addition direct control over the vertical force on the aircraft, corresponding to its vertical take-off and landing (VTOL) capability. The engine power is controlled a rotor speed governor to automatically manage the power. In [49], a helicopter model has been shown to be nonlinear and non-minimum phase. Linearization by state feedback [39] has been successfully applied in control design for highly maneuverable aircraft such as VTOL aircraft [76] and conventional take-off and landing (CTOL) aircraft [98]. The idea of using approximate input-output linearization on helicopter control is motivated by the control design of planar VTOL in [30] and VTOL in [94]. In [30], approximate input-output linearization is applied by neglecting the coupling between rolling moment and lateral acceleration. State transformation technique is used on

constructing an output tracking control in [94]. In [93], helicopter control design based on μ-synthesis and soft computing techniques are studied. Differential flatness has been applied to approximate models of aircraft [72, 102] for trajectory generation. Since differentially flat systems are systems in which all states and inputs can be expressed as functions of the outputs and their derivatives [102, 28], trajectory generation can be then computed by considering algebraic functions of the outputs and their derivatives as constraints. Thus, the complexity of computation can be reduced and the efficiency for computation can be enhanced.

1.2 Dissertation Outline

The material of this dissertation is arranged in six chapters. In Chapter 2, a framework for modeling multi-agent multi-modal system by means of hybrid systems is presented. Some background on hybrid systems; modeling, verification, simulation, synthesis is also given, along with references where a more thorough presentation can be found.

Chapter 3 describes the dynamics and control design for helicopter based UAV. Dynamical model and properties of a helicopter model is presented. Nonlinear control designs based on feedback linearization and differential flatness are illustrated. Performance of different control design are also provided.

The goal of Chapter 4 is to synthesis control mode switching sequence along with switching conditions for navigation purpose by applying algorithms developed for hybrid systems. The controllers designed in previous chapter defines basic control modes. The controlled system can be modeled as a hybrid automaton: a finite state machine with closed loop dynamics embedded in each discrete locations.

In Chapter 5, embedded system synthesize based on formal methodology is presented. The system behavior at the top level is specified in synchronous language then translated to various MOCs in different design stages depending of the design properties of MOCs. Depending on the choice of architecture, the components specified by the MOC are then mapped to hardware and software components. The design framework allows to shorten prototyping time and to prove the correctness of the properties of the system.

Finally, Chapter 6 contains some concluding remarks and directions of future work.

Chapter 2

System Model and Architecture

In this chapter, a framework for modeling multi-agent multi-modal system by means of hybrid systems is presented. Some background material of hybrid system from modeling, verification, simulation, synthesis is also given. An architecture for implementing the proposed multi-agent multi-modal control system is presented.

The multi-disciplinary research field of hybrid systems has emerged over the last decade and lies at the interface of computer science, control engineering and applied mathematics. In [33], hybrid systems are defined as systems built from atomic discrete components and continuous components by parallel and serial composition, arbitrarily nested. Each system components consists of an interface, which determines the possible ways of using the component, and a set of executions, which define define the possible behaviors of the component in real time.

The hybrid phenomena captured by such mathematical models is manifested in a great diversity of complex engineering applications. The high-profile and safety-critical nature of such applications has fostered a large and growing body of work on formal methods for hybrid systems: mathematical logics, computational models and methods, and automated reasoning tools supporting the formal specification and verification of performance requirements for hybrid systems, and the design and synthesis of control programs for hybrid systems that are provably correct with respect to formal specifications.

2.1 Models of Computation

In component based model, a system is composed of components. In the framework of concurrent model of computation [62], the components in such a model are entities capable of performing some computation in parallel. Furthermore, the mechanism by which components communicate is defined by the model of computation. A component based design provides a clean way to integrate different models by hierarchically nesting of parallel and serial composition of heterogeneous components. This hierarchically composition allows one to manage the complexity of a design by information hiding and the reuse of components.

There are a rich variety of models of computation that deal with concurrency and time in different ways. Here, we outline some of the most useful MOCs for embedded systems presented in [61], such as continuous-time(CT), synchronous data flow (SDF), finite-state machine(FSM), synchronous/reactive (SR), discrete-event (DE). CT models represented by differential equations are excellent for modeling analog circuits and many physical systems. SDF supports modeling of multi-rate difference equations and useful in digital signal processing. In FSM, execution is a strictly ordered sequence of state transitions, not concurrent. FSM models are excellent for control logic in embedded systems, particularly safety-critical systems since FSM models are amenable to in-depth formal analysis. In the synchronous/reactive (SR) model of computation, data values are aligned with global clock ticks. Thus, they are discrete signals, as with difference equations, but unlike difference equations, a signal need not have a value at every clock tick. Because of the tight synchronization, SR models are excellent for applications with concurrent and complex control logic such as safety-critical real-time applications. Examples of languages that use the SR model of computation include Esterel [10], Signal [8], Lustre [16], and Argos [71]. In DE models of computation, an event consists of a value and time stamp. This model has been realized in a large number of simulation environments, simulation languages, and hardware description languages, including VHDL and Verilog. There is no global clock tick in DE, but there is a globally consistent notion of time.

Depending on the level of abstraction and domain-specificity of design practice, different MOCs can be chosen and mixed for design. An essential difference between concurrent models of computation is their modeling of time. One powerful modeling of time, as proposed in [62], is by taking time to be merely a constraint imposed by causality. This interpretation

results in time that is partially ordered and hence provides a mathematical framework for formally analyzing and comparing models of computation.

Mixing heterogeneous models for system design has been receiving more and more attention from both academic and industrial world. In circuit design communities, the composition of CT and DE is called the mixed-signal model; in control and computer science communities, the composition of CT with other MOCs which don't have signals continuously evolving along the time line is called the hybrid system.

2.2 Hybrid Automata

The hybrid automaton model developed by [1, 79, 12, 66, 97, 57] is one of the most popular in the hybrid systems literature. Such systems are essentially a finite state machine, or finite automaton, with a continuous dynamical system embedded in each discrete location. The continuous state in a discrete location evolves continuously according to differential equations, as long as the location's invariant remain true; then, when a transition guard becomes true, the discrete state proceeds to another discrete location, and reset some of the continuous state to new values. Hybrid automata without inputs or outputs are called autonomous, or closed, hybrid automata. Thus, non-autonomous hybrid automata are called open hybrid automata. However, the term open hybrid automata is not uniformly used for non-autonomous hybrid automata throughout the literature.

In the following, we present a definition of hybrid automata [67] which captures all of the essential features of hybrid automata will be discussed in this dissertation.

Definition 2.2.1 (Hybrid Automaton) *A hybrid automaton H is a collection $H = (Q, X, V, Y, Init, f, h, I, E, G, R, \varphi)$, where*

- *Q is a finite collection of discrete state variables;*
- *X is a finite collection of continuous state variables;*
- *V is a finite collection of continuous input variables;*
- *Y is a finite collection of continuous output variables;*
- *$Init \subseteq Q \times X$ is a set of initial states;*
- *$f : Q \times X \times V \to TX$ is an input dependent smooth vector field;*
- *$h : Q \times X \times V \to 2^Y$ is an input dependent output map;*

- $I : Q \to 2^{X \times V}$ *assigns each* $q \in Q$ *an input dependent invariant set;*
- $E \subset Q \times Q$ *is a collection of discrete transitions;*
- $G : E \to 2^{X \times V}$ *assigns to each* $e = (q, q') \in E$ *a guard;*
- $R : E \times X \times V \to 2^X$ *assigns to each* $e = (q, q') \in E$, $x \in X$ *and* $v \in V$ *a reset relation; and*
- $\varphi : Q \times X \to 2^V$ *assigns to each state a set of admissible inputs.*

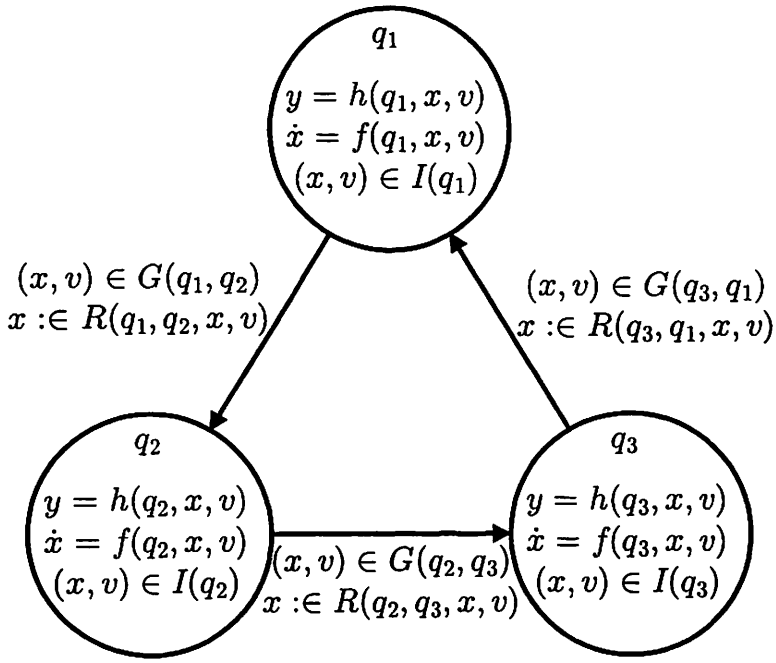


Figure 2.1: Definition of Hybrid Automaton

A hybrid automaton involves continuous evolution as well as instantaneous transitions. To distinguish the time at which discrete transitions take place we introduce the notion of a hybrid time trajectory [67].

Definition 2.2.2 (Hybrid Time Trajectory) *A hybrid time trajectory* $\tau = \{I_i\}_{i=0}^{N}$ *is a finite or infinite sequence of intervals of the real line, such that*

- *for all* $0 \leq i < N$, $I_i = [\tau_i, \tau_i']$ *with* $\tau_i \leq \tau_i' = \tau_{i+1}$;
- *if* $N < \infty$, *either* $I_N = [\tau_N, \tau_N']$ *with* $\tau_N \leq \tau_N' < \infty$, *or* $I_N = [\tau_N, \tau_N')$ *with* $\tau_N < \tau_N' \leq \infty$.

The interpretation is that τ_i are the times at which discrete transitions take place; notice that multiple transitions may take place at the same time (if $\tau_i = \tau_i' = \tau_{i+1}$). Hybrid time trajectories can extend to infinity either if τ is an infinite sequence, or if it is a finite sequence ending with an interval of the form $[\tau_N, \infty)$.

Since there are many models of hybrid automata proposed and each one has its own special properties which are useful in simulation, verification or synthesis, we will introduce them and highlight their properties according to research areas.

Simulation

Numerical simulation of continuous behavior and of discrete behavior is well understood. However, when a hybrid system is simulated, the MOCs used to describe its behavior dictate the way the components of the system interact and execute. Since MOCs differ mostly for the way their components interact, the most difficult problem to solve when simulating them is to resolve the interacting issue. Several numerical packages have recently been developed for simulating hybrid systems, for example PtolemyII [61], Dymola [25], OmSim [73], SHIFT [20], and a Simulink toolbox [38]. For a survey on these packages in regard to their performance and supports, please refer to [77]. However, in order to have efficient and to accurate simulations, some fundamental hybrid phenomena have to be first resolved by the use of theoretical methods. The numerical packages have been applied to the simualtion of AHS [26, 19, 21], ATMS [47], and a helicopter based UAV [65].

Fundamental issues such as existence and uniqueness of executions of hybrid automata are still the topic of intense research activity [101]. To derive local existence and uniqueness conditions for hybrid systems, one needs to consider issues such as blocking and nondeterminism associated with the discrete dynamics, in addition to the usual conditions associated with the existence and uniqueness of trajectories for continuous dynamical systems. Moreover, to ensure the execution can be extended over arbitrarily long time horizons, one also needs to show that an infinite number of discrete transitions cannot take place in a finite amount of time. Executions that fail to satisfy this property are referred to as Zeno executions, and hybrid automata accepting such executions are referred to as Zeno hybrid automata [41]. The Zeno phenomenon is fundamentally a hybrid phenomenon and is an important consideration when modeling, analyzing, controlling, and simulating hybrid

systems.

However, none of these packages make special provisions for the case of fast switching; as the time intervals between discrete transitions get smaller, either simulation slows down or its accuracy decreases. In some cases, the simulation may even give erroneous results or error messages. Regularization techniques have been proposed in [41] to extend the Zeno executions of Zeno hybrid automata to times beyond the Zeno time at when the Zeno phenomena occur. Different types of regularization may not be unique and may depend on the specific assumptions made about the actual model from which the hybrid automata is derived.

Verification

Hybrid systems can be used as verification model for systems which exhibits both continuous and discrete behaviors. The safety criticality of many applications requires the use of formal methods to guarantee that an unsafe region of the state space is not reachable from a set of initial conditions. A reachability problem is to answer the following question: given a set of initial condition, will a target set be reached any state of a hybrid automaton.

One standard approach to reachability called model checking attempts to compute the set of reachable states by completely exploring the whole state space, by starting from the set of initial states and repeatedly adding new reachable states, to check whether the system satisfies the desired specification. The computation can be automated and is guaranteed to converge for some hybrid automata for which the reachability problem is decidable. In general, however, this approach, may not be automated and may not terminate.

Another standard approach to verifying certain properties of a hybrid system is to find an equivalent transition system called a bisimulation with a finite number of states. Bisimulations are simply quotient systems which preserve the closed properties of the original systems and can be used to reduce the complexity of verifying properties of very large scale systems. If a hybrid automaton has a finite state bisimulation, then checking properties for the hybrid automata can be equivalently performed on the finite, discrete, quotient graph. Since the quotient graph is finite, the algorithm will terminate. However, a finite state bisimulation exists only for certain classes of hybrid automata [34, 60], again, for which the

reachability problem is decidable.

Even though the computation of reachable spaces for finite state machines is well developed, computation of the reachable spaces of a differential equation is extremely difficult. This is mainly due to the infinite cardinality of continuous state spaces. In particular, there are decidability results for timed automata [2] and rectangular automata [35]. For timed automata, the differential equations are in the form $\dot{x} = 1$, and KRONOS [23] is developed as one of the model checking tools. Rectangular automata are automata where in each discrete location the continuous dynamics are described by differential inclusions of the form $A\dot{x} \leq b$, and HYTECH [32] is used for model checking for the hybrid automata. Linear hybrid automata [3] are hybrid automata that can be analyzed symbolically in the theory of the reals with addition. Rectangular automata are linear hybrid automata. An important class of linear differential equations in the form $\dot{x} = Ax + Bu$ with a decidable reachability problem is discovered and presented in [57]. This is achieved by posing the reachability computation as a quantifier elimination problem [96] in the decidable theory of the reals based on o-minimal theories [100]. The corresponding hybrid automata are called linear hybrid systems [82]. Verification using optimal of control for hybrid automaton with continuous dynamics described by nonlinear differential equations is shown in [66]. The approach attempts to determine the worst possible execution of the automaton with respect to the given property and verify that the property is satisfied only for this one.

Synthesis

A hybrid automaton is called a controlled hybrid automaton whose underlying continuous time dynamics are modeled as inhomogeneous differential equations whose inputs are controls that can be designed. The purpose of controller synthesis is to construct not only continuous control laws but also discrete strategies in order to satisfy the given system specification. Conditions of the existence of solutions to a class of controlled hybrid automata is presented in [64].

Current emphasis on controller synthesis has been placed on solving problems with safety specifications, which are described by giving a set of good states within which the hybrid automaton should evolve. In particular, game theoretic approach is proposed by [66] for controller synthesis for continuous time nonlinear system in the presence of disturbance,

and the approach has been successfully applied to automated highway systems[69] and air traffic management systems[68]. The set of all initial states guaranteeing that the evolution of the system remains in the good set is the maximal controlled invariant set contained in the set of good hybrid states. This set is called maximal safe set and the set of all control strategies which make this set invariant is the maximal controller. Systematic procedures for solving problems with safety specifications using game theoretic approach have been proposed in [97] by computing optimal solutions of Hamilton-Jacobi-Bellman equation for continuous time nonlinear systems and in [105] by using linear programming and quantifier elimination for discrete time linear systems.

Optimal control approaches have been used in [12, 66, 31] to formulate and solve an optimal control problem for a class of hybrid systems, while providing existence of optimal and near-optimal control polices. The solutions are computed via convex optimization problem in terms of finite-dimensional linear programming.

The reachability operator, an algorithm that can determine the evolution of sets of trajectories, is key to the verification and synthesis of controllers for hybrid systems. There is extensive research in computing exactly, or approximately reachable sets [59, 56, 68, 17].

2.3 System Architecture

In designing an architecture enabling the control of multi-agent multi-modal control system, the concept of hierarchy is proposed for reducing complexity in design. In semi-autonomous agent control, depending on the mission and environment, there are several scenarios being considered for decision making. Each scenario comprise several levels arranged in a hierarchy. At any instant, the decisions conform to a particular scenario. The scenario refers to the current estimate of the system environment. Given that scenario, decisions at each level are specified according to the syntactic rules, and interpreted at each level. Behavior at each level is semantically compiled into nominal behavior at a lower level. However, in execution, changes in the environment may cause large deviations of actual from nominal behavior. This may trigger an alarm. The alarm may lead to a change in scenario and an intervention from the high level. Small changes in the environment may be handled by local decisions that don't require scenario change. It is only the large changes

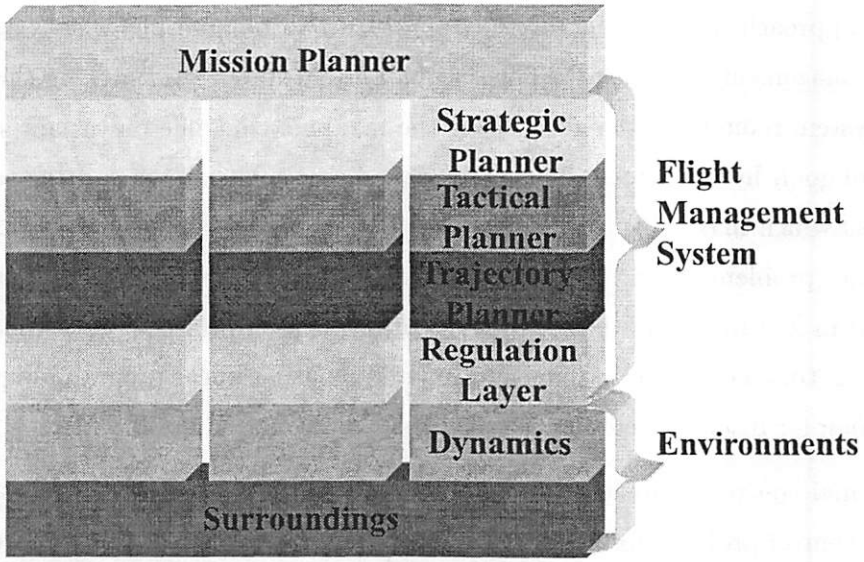


Figure 2.2: Each agent is equipped with a Flight Management System(FMS) which is reactive to mission planner and environments

for which local decisions is ineffective that trigger scenario change. This structure has been proved useful in works ranging from automated highways [104, 29] to air traffic management systems [99].

For UAV networks, we have proposed to adopt the above decision structure in constructing the architecture. For control purpose, a hierarchical structure is imposed on the system architecture by introducing a mission planner on top of the agents for generating a control policy for agents in an attempt to fulfill system-wide objectives. A control policy comprises tasks and each agent is assigned a task to perform. Each agent is equipped with a Flight Management System (FMS) which is responsible for task execution and has sensing, computation and communication capabilities. Mission planner and agents may exchange information via a communication network depending on availability. Each FMS consists of 4 layers, namely strategic planner, tactical planner, and trajectory planners, and regulation layer. They are arranged in a hierarchical manner as shown in Figure 2.2. Notice that dynamics of agents are modeled as parts of the environment since in general they are given and are not supposed to be designed at this design stage.

Based on a given task by mission planner, strategic planner computes its own nominal strategy as a coarse, self-optimal trajectory by using a highly abstracted model of vehicle

and static environment model. A task could be specified for navigation in searching scenario or coordination with other vehicles in close proximity.

The tactical planner refines the nominal strategy from strategic planner so as to cope with contingent events such as conflict resolution and collision avoidance. In order to reduce complexity, abstracted model of vehicles and estimated environment model are used in computation.

Given a strategy from tactical planner, the trajectory planner based on dynamical properties and physical constraints of the vehicle to determine its own course. The nominal course is determined by the ordering of controller design specifications related to performance. However, the course could be further refined in order to cope with situation due to the presence of unanticipated external disturbance.

The regulation layer selects a specified controller for executing the given course from trajectory planner. To simplify future discussion, we assume that the actuators and sensors related to regulation purpose are all embedded in this layer. Regulation layer directly interacts with the environment.

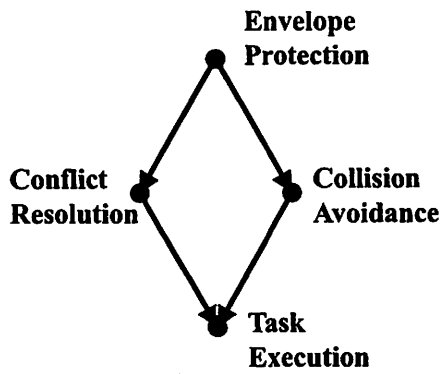


Figure 2.3: In the directed graph, each node represents a mode of operation and each arrow connects a higher priority to a lower one.

The ordering of design specifications provides a guideline for the designing an architecture which allows hierarchical nesting of composition of modes of operation. Mode switching among the contingent events is driven by the changes in environment. Envelope protection has the highest priority since this mode is used to keep the vehicle operating safely in the presence of unanticipated external disturbance. This suggests that envelope protection should be handled in trajectory planner. Whereas, collision avoidance and conflict resolu-

tion are the modes of operation related to the contingent situations at the agent level while assuming that the vehicle are operating normally. Hence, both modes have lower priority than envelope protection and should be handled in a layer upper, tactical layer. However, since the ordering of events is incomparable, there is no way to assign these two modes with different priorities. If no contingency happens, an agent execute the given task. Whereas, in the presence of events, an agent would switch modes of operation to handle the events and then resume task execution via task replaning. The ordering of events is shown in Figure 2.3 This design principle confronts with the motto of flight pilot which is "aviate, navigate, communicate" to handle different levels of contingent events.

In designing a system, there are many other aspects needed to be considered in adopting an implementation for mapping from the architecture and a detailed discussion will be provided in Chapter 5

Chapter 3

Nonlinear Control Design

In this chapter, output tracking control of a helicopter based unmanned aerial vehicle model is investigated. First, based on Newton-Euler equations, a dynamical model is derived by considering the helicopter as a rigid body upon which a set of forces and moments act. Second, we show that the model cannot be converted into a controllable linear system via exact state space linearization. In particular, for certain output functions, exact input-output linearization by state feedback results in unstable zero dynamics. Third, by neglecting weak couplings between forces and moments on the model, we apply input-output linearization to the approximate model for deriving an approximate control with positions and heading as the outputs . Given a bounded output trajectory, the tracking error of the original model by applying the approximate control is proved to be bounded. Finally, we prove by construction that the approximate model with the same outputs is differentially flat and hence the state and input can be expressed as functions of the outputs and their derivatives. This property is very useful for real-time trajectory generation. Based on geometric control theory, we decompose the dynamics into two subsystems: inner and outer systems. A nonlinear controller is proposed based on differential flatness of the outer system.

3.1 Introduction

Helicopter [85] is versatile in maneuverability and this makes helicopter based unmanned aerial vehicles (UAVs) indispensable for both civilian and military applications where human intervention, especially in restricted areas, is considered difficult or dangerous. Given a multi-agent, multi-objective UAV mission, control system design for a single UAV is a very complicated and challenging task. One natural way to reduce the complexity of system design is by compositional methods. Compositional methods attempt to solve a complex problem by decomposing the problem into a sequence of smaller problems of manageable complexity. In sophisticated flight management systems [46, 54], a single UAV flies from origin to destination while satisfying a large number of aerodynamic, scheduling, and environmental constraints by switching among a finite set of *control modes*, where each control mode essentially corresponds to a different output tracking controller. For example, for regulating at a fix location, *position and heading control mode* is used. In the case of sensor failure such as in the absence of position information from the global positioning system (GPS), *altitude and attitude control mode* is more desired to be used for stabilizing the vehicle since the on-board inertial navigation system (INS) would still be able to provide altitude and attitude information for control. The resulting hierarchical control strategy which involves the interaction of continuous and discrete dynamics can be modeled as a hybrid system [4] for system analysis and controller synthesis[68, 48]. In this chapter, output tracking control of a helicopter model is investigated. The helicopter model is based on a UAV [53] being developed by the Berkeley Aerial Robot (BEAR) team at UC Berkeley.

Helicopter control [74, 85] requires the ability to produce moments and forces on the vehicle for two purposes: first, to produce equilibrium and thereby hold the helicopter in a desired trim state; and second, to produce accelerations and thereby change the helicopter's velocity, position and orientation. Like airplane control, helicopter control is accomplished primarily by producing moments about all three aircraft axes: roll, pitch, and yaw. The helicopter has in addition direct control over the vertical force on the aircraft, corresponding to its vertical take-off and landing (VTOL) capability. The engine power is controlled by a rotor speed governor to automatically manage the power. Helicopter flight dynamics are inherently unstable, particularly in hover.

Linearization by state feedback [39] has been successfully applied in control design for

highly maneuverable aircraft such as VTOL aircraft [76] and conventional take-off and landing aircraft [98]. In this chapter, we design an output tracking controller for a helicopter model based on input-output linearization. Our control design is constructed by first neglecting the coupling effect, then showing that the approximate control results in bounded tracking on the exact model. The idea of using approximate input-output linearization in helicopter control is motivated by the control design of planar VTOL in [30] and VTOL in [94]. In [30], approximate input-output linearization is applied by neglecting the coupling between rolling moment and lateral acceleration. A state transformation technique is used on constructing an output tracking control in [94]. In [93, 37], helicopter control design based on μ-synthesis and fuzzy logic are studied.

Differentially flat systems are systems in which all states and inputs can be expressed as functions of the outputs and their derivatives [102, 28]. They have the useful property that there is a one-to-one mapping between trajectories in output space and trajectories in state and input space. Instead of incorporating the dynamical equations as constraints, trajectory generation for differentially flat system can be computed by considering algebraic functions of the outputs and their derivatives as constraints. Thus, the complexity of computation can be reduced and the efficiency for computation can be enhanced. Differential flatness has been applied to approximate models of aircraft [72, 102] for trajectory generation. Trajectory plays a significant role in determining the performance of a closed-loop system especially under saturation. For details related to trajectory generation under the effect of control saturation, please refer to [42, 81].

In this chapter, we first derive a helicopter dynamical model which is derived by considering the helicopter as a rigid body upon which a set of forces and moments act. In section 3, we prove that the model cannot be converted into a controllable linear system via exact state space linearization. In particular, for certain output functions, exact input-output linearization results in unstable zero dynamics. In section 4, by neglecting weak couplings between forces and moments on the model, we apply input-output linearization to the approximate model for deriving an approximate control with positions and heading as the outputs . Given a bounded output trajectory, the tracking error of the original model by applying the approximate control is proved to be bounded. Then, in section 5, we prove by construction that the approximate model with the same outputs is differentially flat and hence the state and input can be expressed as functions of the outputs and their derivatives.

Based on geometric control theory, we decompose the dynamics into two subsystems: an inner and outer system. A nonlinear controller is proposed based on differential flatness of the outer system. Finally, in section 6, simulation results using both output tracking controllers based on exact and approximate input-output linearization are presented for comparison. We conclude our work in section 7.

3.2 Helicopter Model

A model of a helicopter can be divided into four different subsystems, which are actuator dynamics, rotary wing dynamics, force and moment generation processes, and rigid body dynamics. The connections between subsystems, and state and control variables are defined in Figure 3.1. In this chapter, due to the fact that the complete dynamics of a helicopter, taking into account flexibility of the rotors and fuselage, the dynamics of the engine and actuators is quite complex and somewhat unmanageable for the purpose of control, we consider a helicopter model as a rigid body incorporating with a force and moment generation process. For illustration, we use model data obtained from a model helicopter on which we will apply the proposed control law in real flight. However, the result is also applicable to other helicopters with similar force and moment generation processes.

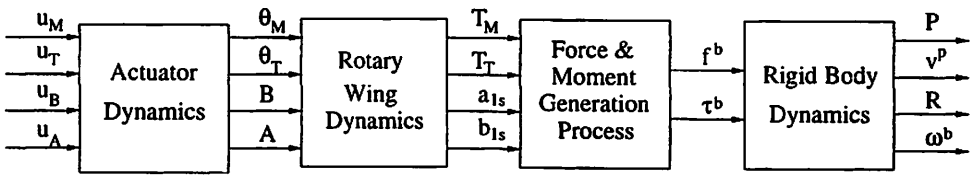


Figure 3.1: Helicopter dynamics

By regarding the helicopter as a rigid body as in [85], the equations of motion of a model helicopter can be derived by applying Newton-Euler equation.

3.2.1 Rigid Body Dynamics

Consider the helicopter depicted in Figure 3.2. The equations of motion for a model helicopter can be written with respect to the body coordinate frame, which is attached

towards the center of mass of the model helicopter. The x axis is pointed to the body head and y axis goes to the right of the body. As shown in [78], the equations of motion for a rigid body subject to body force $f^b \in \mathbb{R}^3$ and torque $\tau^b \in \mathbb{R}^3$ applied at the center of mass and specified with respect to the body coordinate frame is given by the *Newton-Euler* equation in body coordinate, which can be written as

$$\begin{bmatrix} mI & 0 \\ 0 & \mathcal{I} \end{bmatrix} \begin{bmatrix} \dot{v}^b \\ \dot{\omega}^b \end{bmatrix} + \begin{bmatrix} \omega^b \times mv^b \\ \omega^b \times \mathcal{I}\omega^b \end{bmatrix} = \begin{bmatrix} f^b \\ \tau^b \end{bmatrix} \tag{3.1}$$

where $v^b \in \mathbb{R}^3$ is the body velocity vector, $\omega^b \in \mathbb{R}^3$ is the body angular velocity vector, $m \in \mathbb{R}$ specifies the mass, $I \in \mathbb{R}^{3\times3}$ is an identity matrix, and $\mathcal{I} \in \mathbb{R}^{3\times3}$ is an inertial matrix.

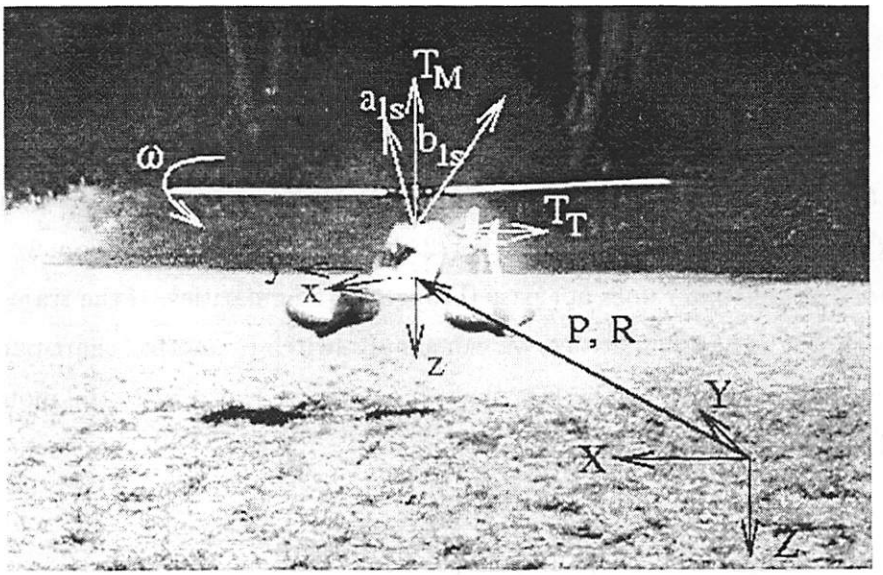


Figure 3.2: Coordinate frames for specifying rigid motions and forces acting on a helicopter.

The position and velocity of the helicopter center of gravity are given by $P \in \mathbb{R}^3$ and $v^p = \dot{P} \in \mathbb{R}^3$, respectively, expressed to the spatial frame in North-East-Down orientation. Let $R \in SO(3)$ be the rotation matrix of the body axes relative to the spatial axes and $\omega^b \in \mathbb{R}^3$ be the body angular velocity vector. Given $e = [e_1\ e_2\ e_3]^T \in \mathbb{R}^3$, we define $\hat{e} \in so(3)$, the space of skew-symmetric matrices in $\mathbb{R}^{3\times3}$, by

$$\hat{e} = \begin{bmatrix} 0 & -e_3 & e_2 \\ e_3 & 0 & -e_1 \\ -e_2 & e_1 & 0 \end{bmatrix}$$

Differentiating the orthogonality constraint $R^TR = I$, $\hat{\omega}^b = R^T\dot{R}$ as shown in [78]. We parameterize R by ZYX Euler angles with ϕ, θ and ψ about the x, y, z axes respectively.

$$\begin{aligned} R &= e^{\hat{z}\psi}e^{\hat{y}\theta}e^{\hat{x}\phi} \\ &= \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\theta s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix}, \end{aligned} \tag{3.2}$$

where $x = [1\ 0\ 0]^T$, $y = [0\ 1\ 0]^T$, $z = [0\ 0\ 1]^T$ and $c\theta$, $s\theta$ are abbreviations for $\cos\theta$ and $\sin\theta$ respectively, and similarly for the other terms. By differentiating R with respect to time, we have the state equations of the Euler angles, $\Theta = [\phi\ \theta\ \psi]^T$, which are

$$\begin{aligned} \dot{\Theta} &= \Psi\omega^b \\ &= \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi/c\theta & c\phi/c\theta \end{bmatrix} \omega^b \end{aligned}$$

where $t\theta$ is an abbreviation for $\tan\theta$. In the ZYX Euler angle parameterization of rotation matrix, there are singularities at $\theta = \pm\pi/2$. For the following discussion, we assume that the trajectory of helicopter does not pass through the singularities. If the trajectory is required to pass through the singularities, we can simply switch to another chart parameterizing the rotation matrix. By using the fact that $v^p = Rv^b$, we can rewrite the motion equations of a rigid body as

$$\begin{bmatrix} \dot{P} \\ \dot{v}^p \\ \dot{\Theta} \\ \dot{\omega}^b \end{bmatrix} = \begin{bmatrix} v^p \\ \frac{1}{m}Rf^b \\ \Psi\omega^b \\ \mathcal{I}^{-1}(\tau^b - \omega^b \times \mathcal{I}\omega^b) \end{bmatrix} \tag{3.3}$$

3.2.2 Force and Moment Generation Processes

The helicopter system can be considered as a lumped model consisting of a main rotor, a tail rotor, a horizontal stabilizer, a vertical stabilizer and a fuselage, which are denoted with subscripts M, T, H, V, F, respectively.

In the following, we express the external wrench, a force/moment pair, exerted on the helicopter. The force experienced by the helicopter is the resultant force of the thrust generated by the main and tail rotors, damping forces from the horizontal and vertical stabilizer,

aerodynamic force due to fuselage, and gravitational force. The torque is composed of the torques generated by the main rotor, tail rotor and fuselage, and moment generated by the forces as defined in Figure 3.3.

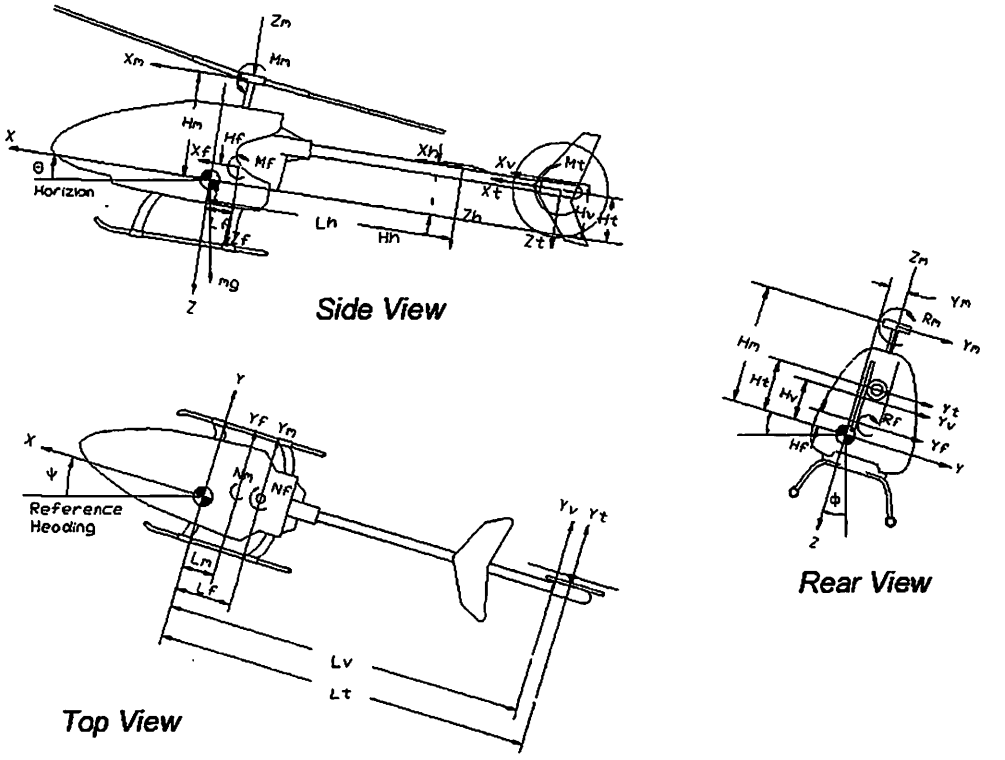


Figure 3.3: The free-body diagram of a helicopter in flight. (Figure courtesy of D. H. Shim)

Assume that the helicopter is operated at low speeds and hence the drag contributed from the horizontal, vertical stabilizers, and the fuselage can be ignored. Therefore, the terms with subscripts H, V and F are discarded. As shown in [63], the external wrench can be written as:

$$f^b = \begin{bmatrix} X_M \\ Y_M + Y_T \\ Z_M \end{bmatrix} + R^T \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}$$

$$\tau^b = \begin{bmatrix} R_M \\ M_M + M_T \\ N_M \end{bmatrix} + \begin{bmatrix} Y_M h_M + Z_M y_M + Y_T h_T \\ -X_M h_M + Z_M l_M \\ -Y_M l_M - Y_T l_T \end{bmatrix}$$

The forces and torques generated by the main rotor are controlled by T_M, a_{1s} and b_{1s}, in which T_M is the main rotor thrust, a_{1s} and b_{1s} are the longitudinal and lateral tilts of the tip path plane of the main rotor with respect to the shaft, respectively. The tail rotor is considered as a source of pure lateral force Y_T and anti-torque Q_T, which are controlled by T_T, tail rotor thrust. The forces and torques can be expressed as

$$X_M = -T_M \sin a_{1s} \tag{3.4}$$

$$Y_M = T_M \sin b_{1s} \tag{3.5}$$

$$Z_M = -T_M \cos a_{1s} \cos b_{1s} \tag{3.6}$$

$$Y_T = -T_T \tag{3.7}$$

$$R_M \simeq \frac{\partial R_M}{\partial b_{1s}} b_{1s} - Q_M \sin a_{1s} \tag{3.8}$$

$$M_M \simeq \frac{\partial M_M}{\partial a_{1s}} a_{1s} + Q_M \sin b_{1s} \tag{3.9}$$

$$N_M \simeq -Q_M \cos a_{1s} \cos b_{1s} \tag{3.10}$$

$$M_T = -Q_T \tag{3.11}$$

The moments generated by the main and tail rotor can be calculated by using the constants, $\{l_M, y_M, h_M, h_T, l_T\}$, as defined in Figure 3.3. In above, we approximate the rotor torque equations by $Q_i \simeq C_i^Q T_i^{1.5} + D_i^Q$ for $i = M, T$. The details of generation of the rotor torques, Q_M, Q_T, one can obtain by applying the equations as shown in [63]. The system parameters are given in Appendix A.1.

3.2.3 System equations

We assume that all the states can be measured accurately. In order to present the system in an input-affine form, we assume that the inputs of the about nonlinear system are the derivatives of T_M, T_T, a_{1s} and b_{1s}. Define $P = [p_x \; p_y \; p_z]^T$, $v^p = [v_x^p \; v_y^p \; v_z^p]^T$, and $\omega^b = [\omega_x^b \; \omega_y^b \; \omega_z^b]^T$. The system equations can be rewritten as

$$\dot{x} = f(x) + \sum_{i=1}^{4} g_i w_i \tag{3.12}$$

where

$$f(x) = \begin{bmatrix} v^p \\ \frac{1}{m}Rf^b \\ \Psi\omega^b \\ \mathcal{I}^{-1}(\tau^b - \omega^b \times \mathcal{I}\omega^b) \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \; g_1 = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \; g_2 = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \; g_3 = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \; g_4 = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

with $x = [p_x \; p_y \; p_z \; v_x^p \; v_y^p \; v_z^p \; \phi \; \theta \; \psi \; \omega_x^b \; \omega_y^b \; \omega_z^b \; T_M \; T_T \; a_{1s} \; b_{1s}]^T \in \mathbb{R}^n (= X)$ with $n = 16$ and $w = [w_1, w_2, w_3, w_4] \in \mathbb{R}^m$ with $m = 4$ is defined as the state vector and auxiliary input vector of the system, respectively. It can be easily seen that $f(x)$, g_i are smooth vector fields.

As defined in [85], a helicopter is said to be in trim if all the forces, aerodynamic and gravitational, and aerodynamic moments acting on the helicopter about the center of gravity are in balance. Hence, by solving the nonlinear equations of body wrench, which are $f^b = 0$ and $\tau^b = 0$, one can solve for the system trim condition. The trim condition is irrespective of the values of P, v^p, ψ. In the following, we define x_0 in trim condition, in which $p_x = 0$, $p_y = 0$, $p_z = 0$, $v_x^p = 0$, $v_y^p = 0$, $v_z^p = 0$, $\psi = 0$, $T_M = 47.97$, $T_T = 2.42$, $a_{1s} = -0.018$, $b_{1s} = 0.0061$, $\phi = 0.044$, $\theta = 0.018$, and hence $\omega^b = 0$ and $w_i = 0$ for $i = 1, \ldots, 4$.

3.3 Exact Linearization by State Feedback

In this section, we show that the model cannot be converted into a controllable linear system via exact state space linearization. In addition, for certain output functions, exact input-output linearization results in unstable zero dynamics.

Consider the square system (*i.e.* system with as many inputs as outputs) multi-input multi-output (MIMO) nonlinear control system described by

$$\sum : \begin{cases} \dot{x} &= f(x) + gw \\ y &= h(x) \end{cases} \tag{3.13}$$

with

$$g = [g_1\ g_2\ g_3\ g_4] \quad \text{and} \quad y_j = h_j(x) \text{ for } j = 1, \ldots, m$$

where $h_1(x), \cdots, h_m(x)$ are smooth functions on $\mathbb{R}^n$ and $m = 4$. Define strict relative degree γ_j at $x_0 \in X$ with respect to output y_j as an integer such that

$$\begin{cases} L_{g_i} L_f^l h_j(x) \equiv 0 & \forall x \in X, 0 \le l \le \gamma_j - 2,\ \forall i \in [1, m] \\ L_{g_i} L_f^{\gamma_j - 1} h_j(x_0) \neq 0 \end{cases}$$

Collecting these calculation, we have

$$\begin{aligned} \begin{bmatrix} y_1^{(\gamma_1)} \\ \vdots \\ y_m^{(\gamma_m)} \end{bmatrix} &= \begin{bmatrix} L_f^{\gamma_1} h_1 \\ \vdots \\ L_f^{\gamma_m} h_m \end{bmatrix} + \begin{bmatrix} L_{g_1} L_f^{\gamma_1 - 1} h_1 & \cdots & L_{g_m} L_f^{\gamma_1 - 1} h_1 \\ \vdots & \ddots & \vdots \\ L_{g_1} L_f^{\gamma_m - 1} h_m & \cdots & L_{g_m} L_f^{\gamma_m - 1} h_m \end{bmatrix} w \\ &:= b(x) + A(x) w \end{aligned} \tag{3.14}$$

where $A(x)$ is called the decoupling matrix. If $A(x)$ is invertible at every point in X, then the static-state feedback given by $w = (A(x))^{-1}[-b(x) + v]$ will result in a closed-loop system that is decoupled from input v to output y. This decoupled and input-output linearized system is given by

$$\begin{bmatrix} y_1^{(\gamma_1)} \\ \vdots \\ y_m^{(\gamma_m)} \end{bmatrix} = \begin{bmatrix} v_1 \\ \vdots \\ v_m \end{bmatrix} \tag{3.15}$$

If the matrix $A(x)$ is singular, we cannot use a static state feedback to linearized the nonlinear system, and we have to look for a dynamic state feedback to achieve linearization by state feedback.

Definition 3.3.1 (Vector Relative Degree [39].) *The system (3.13) is said to have vector relative degree* $\{\gamma_1, \ldots, \gamma_m\}$ *at* $x_0 \in X$ *if*

$$\begin{cases} L_{g_i} L_f^l h_j(x) \equiv 0 & \forall x \in X, 0 \le l \le \gamma_j - 2,\ \forall i \in [1, m] \\ L_{g_i} L_f^{\gamma_j - 1} h_j(x_0) \neq 0 \end{cases}$$

for $j = 1, \ldots, m$ *and the matrix* $A(x_0)$ *is nonsingular.*

Define the distributions,

$$\begin{aligned} G_0 &= \text{span}\{g_1, \ldots, g_m\} \\ G_1 &= \text{span}\{g_1, \ldots, g_m, ad_f g_1, \ldots, ad_f g_m\} \\ &\vdots \\ G_i &= \text{span}\{ad_j^k g_j : 0 \leq k \leq i, 1 \leq j \leq 4\} \end{aligned}$$

for $i = 1, \ldots, n-1$. The conditions under which there exist outputs $h_1, \ldots, h_m$ such that the MIMO system has vector relative degree and furthermore is such that $\gamma_1 + \cdots + \gamma_m = n$ are given as the following lemma:

Lemma 3.3.2 (Full State MIMO Linearization [39].) *Suppose the matrix $g(x_0)$ has rank m. Then, there exist m functions $\lambda_1(x), \lambda_2(x), \ldots, \lambda_m(x)$ such that the system*

$$\begin{aligned} \dot{x} &= f(x) + g(x)w, \\ y &= \lambda(x), \end{aligned}$$

has vector relative degree $\{\gamma_1, \ldots, \gamma_m\}$ with

$$\gamma_1 + \cdots + \gamma_m = n$$

iff

1. *For each $0 \leq i \leq n-1$ the distribution G_i has constant dimension in a neighborhood X of x_0.*

2. *The distribution G_{n-1} has dimension n.*

3. *For each $0 \leq i \leq n-2$ the distribution G_i is involutive.*

By applying the above lemma, we have the following result regarding full state MIMO linearization of the system equations (3.12).

Theorem 3.3.3 *Consider the system equations (3.12). There do not exist any m functions* $\lambda_1(x), \lambda_2(x), \ldots, \lambda_m(x)$ *defined on* X*, such that the system has vector relative degree* $\{\gamma_1, \ldots, \gamma_m\}$ *at* x_0 *with* $\sum_{k=1}^{m} \gamma_k = n$

Proof: It can be computed that the distributions have constant dimensions near x_0*, which are* $\{4, 8, 12, 14, 16, \cdots, 16\}$*. The distribution* G_{15} *has dimension 16. However, the distribution* G_2 *and* G_3 *fail to be involutive. By applying Lemma 3.3.2, we complete the proof.*

Thus, the above theorem suggests that it is impossible to find a set of outputs such that the model can be converted into a controllable linear system via exact state space linearization.

However, if the MIMO system has relative degree $\gamma_1 + \cdots + \gamma_m < n$, then we can write a normal form [87] for the equations. In particular, consider the following set of outputs

$$\{p_x,\ p_y,\ p_z,\ \phi,\ \theta,\ \psi\}, \tag{3.16}$$

which can be used to form various control mode and obtained from sensors such as GPS and INS. For the square system, there are $C_4^6 = 15$ possible input-output pairs. We define $k_1, \cdots, k_4 \in \{1, \ldots, 6\}$ as the indices of the output functions selected from the combinations. To perform exact input-output linearization, we pick the jth output y_j of the system equation and differentiate it with respect to time. For all $j = 1, \ldots, 6$, one can check that one has to differentiate every of the outputs 3 times before encountering one of the inputs, *i.e.*,

$$y_j^{(3)} = L_j^3 h_j + \sum_{i=1}^{4} L_{g_i} L_f^2 h_j \ w_i. \tag{3.17}$$

Given $k_1, \cdots, k_4$, the input-output system can be written as

$$\begin{bmatrix} y_{k_1}^{(3)} \\ \vdots \\ y_{k_4}^{(3)} \end{bmatrix} = \begin{bmatrix} L_f^3 h_{k_1} \\ \vdots \\ L_f^3 h_{k_4} \end{bmatrix} + A_{k_1 k_2 k_3 k_4}(x) \begin{bmatrix} w_1 \\ \vdots \\ w_4 \end{bmatrix} \tag{3.18}$$

where the decoupling matrix is defined by

$$A_{k_1 k_2 k_3 k_4}(x) := \begin{bmatrix} L_{g_1} L_f^2 h_{k_1} & \cdots & L_{g_4} L_f^2 h_{k_1} \\ \vdots & \ddots & \vdots \\ L_{g_1} L_f^2 h_{k_4} & \cdots & L_{g_4} L_f^2 h_{k_4} \end{bmatrix} \tag{3.19}$$

Then, we have the following proposition to show that the decoupling matrix for each possible combination is nonsingular for all $x \in X$.

Proposition 3.3.4 *Given system equations (3.12) and any m output functions of (3.16), the decoupling matrix $A_{k_1k_2k_3k_4}(x)$ of the input-output system has full rank for all $x \in X$ for all possible combinations of $k_1, \cdots, k_4 \in \{1, \ldots, 6\}$.*

Proof: See Appendix A.2.

Since $A_{k_1k_2k_3k_4}(x)$ has full rank and $L_{g_i}h_j \equiv 0$ and $L_{g_i}L_fh_j \equiv 0$ for all $x \in X$, by the definition, the system has vector relative degree $\{3,3,3,3\}$ for all $x \in X$. It follows that $A_{k_1k_2k_3k_4}(x)$ is nonsingular and $A^{-1}_{k_1k_2k_3k_4}(x)$ exists for all $x \in X$. Thus, the state feedback control law

$$\begin{bmatrix} w_1 \\ \vdots \\ w_4 \end{bmatrix} = A^{-1}_{k_1k_2k_3k_4}(x) \begin{bmatrix} -L_f^3h_{k_1} + v_1 \\ \vdots \\ -L_f^3h_{k_4} + v_4 \end{bmatrix} \tag{3.20}$$

yields the linear closed loop system

$$\begin{bmatrix} y^{(3)}_{k_1} \\ \vdots \\ y^{(3)}_{k_4} \end{bmatrix} = \begin{bmatrix} v_1 \\ \vdots \\ v_4 \end{bmatrix} \tag{3.21}$$

This feedback law makes the input-output map linear, but has the unfortunate side-effect of making some dynamics unobservable. In order to guarantee the internal stability of the system, it is not sufficient to look at input-output stability, we must also show that all internal (unobservable) modes of the system are stable as well.

If a system has relative degree $\gamma = \gamma_1 + \cdots + \gamma_p < n$, then we can write a normal form for the equations (3.13) by choosing as coordinates

$$\begin{aligned} &\xi^1_1 = h_1(x), \quad \xi^1_2 = L_fh_1(x), \quad \ldots, \quad \xi^1_{\gamma_1} = L_f^{\gamma_1-1}h_1(x), \\ &\xi^2_1 = h_2(x), \quad \xi^2_2 = L_fh_2(x), \quad \ldots, \quad \xi^2_{\gamma_2} = L_f^{\gamma_2-1}h_2(x), \\ &\vdots \\ &\xi^m_1 = h_m(x), \quad \xi^m_2 = L_fh_m(x), \quad \ldots, \quad \xi^m_{\gamma_m} = L_f^{\gamma_m-1}h_m(x). \end{aligned} \tag{3.22}$$

As shown in [87], these ξ_i^j qualify as a partial set of coordinates. Furthermore, one can complete the basis by choosing $n-\gamma$ more functions $\eta_1(x), \eta_2(x), \ldots, \eta_{n-\gamma}(x)$. Define $\xi = [\xi_1^1, \ldots, \xi_{\gamma_1}^1, \xi_1^2, \ldots, \xi_{\gamma_2}^2, \ldots, \xi_1^m, \ldots, \xi_{\gamma_m}^m]^T$ and $\eta = [\eta_1, \eta_2, \ldots, \eta_{n-\gamma}]^T$. In these ξ, η coordinates the system equations (3.13) have the following normal form

$$
\begin{aligned}
& y_1 = \xi_1^1, \quad y_2 = \xi_1^2, \quad \ldots, \quad y_m = \xi_1^m \\
& \dot{\xi}_1^1 = \xi_2^1, \qquad \dot{\xi}_1^2 = \xi_2^2, \qquad \ldots, \quad \dot{\xi}_1^m = \xi_2^m \\
& \vdots \qquad\qquad \vdots \qquad\qquad \vdots \quad \vdots \\
& \dot{\xi}_{\gamma_1 - 1}^1 = \xi_{\gamma_1}^1, \quad \dot{\xi}_{\gamma_2 - 1}^2 = \xi_{\gamma_2}^2, \quad \ldots, \quad \dot{\xi}_{\gamma_m - 1}^m = \xi_{\gamma_m}^m \\
& \begin{bmatrix} \dot{\xi}_{\gamma_1}^1 \\ \dot{\xi}_{\gamma_2}^2 \\ \vdots \\ \dot{\xi}_{\gamma_m}^m \end{bmatrix} = b(\Phi(\xi,\eta)) + A(\Phi(\xi,\eta))w \\
& \dot{\eta} = q(\xi,\eta) + p(\xi,\eta)w
\end{aligned}
\tag{3.23}
$$

where $\Phi : x \mapsto (\xi, \eta)$ is the diffeomorphism mapping from x into the normal form coordinates. Note that $p \in \mathbb{R}^{n-\gamma \times m}$, $q \in \mathbb{R}^{n-\gamma}$.

In order to analyze the internal stability of the system, zero dynamics of the system should be examined. The zero dynamics of a nonlinear system are the internal dynamics of the system subject to the constraint that the outputs and all derivatives of the outputs are set to zero for all time, *i.e.* $\xi \equiv 0$. This can be done by using the control law with $v_1 = \cdots = v_4 = 0$ and initializing the system with the trim condition $(\xi_0, \eta_0) = \Phi^{-1}(x_0)$. Hence, the zero dynamics can be written as

$$\dot{\eta} = q(0,\eta) + p(0,\eta)A(\Phi(0,\eta))^{-1}b(\Phi(0,\eta)).$$

For illustration, two modes of operation will be explained.

In **position and heading control mode**, $\{p_x, p_y, p_z, \psi\}$ are chosen as outputs, one can easily verify that the zero dynamics may be parameterized by $\eta = [\phi, \omega_x^b, \theta, \omega_y^b]^T$, and the linearized zero dynamics at equilibrium point has eigenvalues $\pm 16.4528i$ and $\pm 12.1123i$. The linearization is inconclusive.

We define a nonsingular matrix T which transforms the state transition matrix of the linearized zero dynamics into block diagonal form. By applying $\eta^* = T^{-1}\eta$, the nonlinear dynamics are locally decoupled. From the phase portrait as shown in Figure 3.4, one can

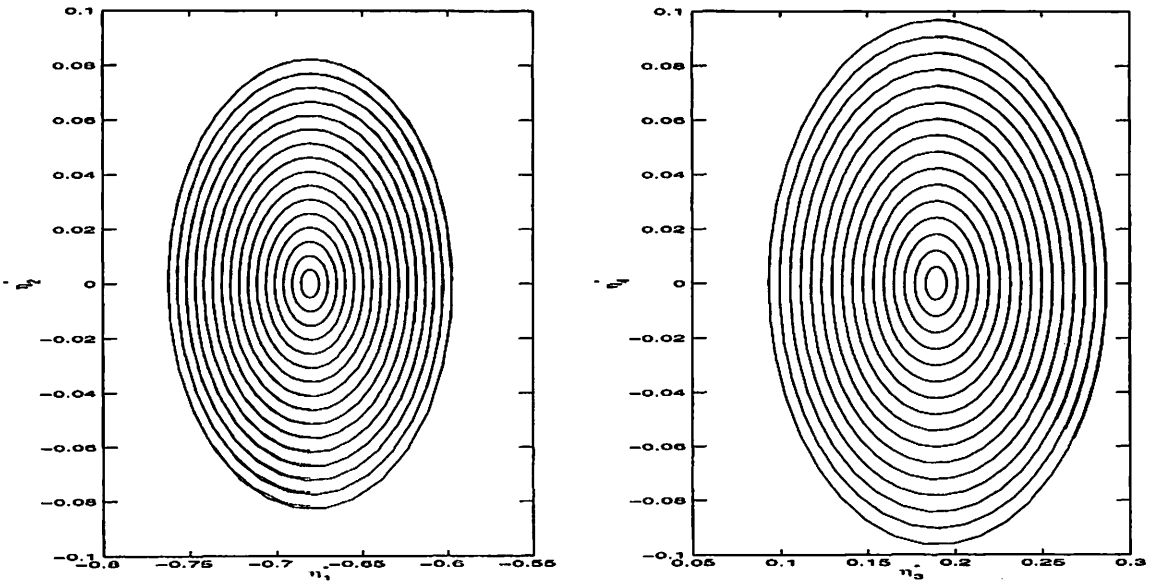


Figure 3.4: Phase portrait of zero dynamics in position and heading control mode

conclude that the nonlinear system is not asymptotically stable, since the equilibrium point is surrounded by a family of periodic orbits. The zero dynamics is non-minimum phase.

In **attitude and altitude control mode**, $\{\phi, \theta, \psi, p_z\}$ are chosen as outputs. It can be easily verified that the zero dynamics becomes

$$\begin{aligned} \ddot{p}_x &= 0.1757 \\ \ddot{p}_y &= -0.4335, \end{aligned}$$

where the states p_x,p_y and their first derivatives become unobservable. The zero dynamics is unstable and hence the system is non-minimum phase. In this mode, the helicopter is constantly drifting in the $X - Y$ plane while attitude and altitude are held constantly.

3.4 Approximate Linearization by State Feedback

In this section, approximate linearization technique is applied to the helicopter system. First, the system equations (3.3) is rewritten as

$$\ddot{P} = \frac{1}{m}R\begin{bmatrix} -T_M \sin a_{1s} \\ T_M \sin b_{1s} - T_T \\ -T_M \cos a_{1s} \cos b_{1s} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{3.24}$$

$$\dot{R} = R\hat{\omega} \tag{3.25}$$

$$\dot{\omega} = \mathcal{I}^{-1}(\tau^b - \omega^b \times \mathcal{I}\omega^b). \tag{3.26}$$

The reason of the failure of exact linearization is due to the existence of couplings between rolling (pitching) moments and lateral (longitudinal) acceleration. Those couplings are introduced due to the presence of a_{1s}, b_{1s} and T_T. As shown before, if position and heading control mode is chosen the internal dynamics are not regulated and exhibit an unstable behavior.

Here, we propose to approximately linearize the system by neglecting the coupling terms. This can be done by assuming that $a_{1s}, b_{1s}, T_T/T_M$ are small. Therefore, equation (3.24) can be modeled as follows

$$\ddot{P}_m = \frac{1}{m}R\begin{bmatrix} 0 \\ 0 \\ -T_M \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{3.27}$$

while keeping equations (3.25) and (3.26) the same. In what follows, we demonstrate the idea of approximate linearization in position and heading mode.

We differentiate outputs $P_m = [p_{xm}\ p_{ym}\ p_{zm}]^T$ on (3.27) and ψ on (3.25) until at least one input appears in each output equation, and we get the final equations in the form:

$$\begin{bmatrix} p_{xm}^{(3)} \\ p_{ym}^{(3)} \\ p_{zm}^{(3)} \\ \psi^{(3)} \end{bmatrix} = \begin{bmatrix} * \\ * \\ * \\ * \end{bmatrix} + \begin{bmatrix} * & 0 & 0 & 0 \\ * & 0 & 0 & 0 \\ * & 0 & 0 & 0 \\ * & * & * & * \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}$$

Since the decoupling matrix has rank 2, decoupling of the system cannot be achieved by static state feedback. Here, we propose to use dynamic decoupling[18] . Following the

algorithm, we have to add an integrator to w_1, since column one in the decoupling matrix has more than one nonzero elements. Hence, the first three rows in the decoupling matrix are all zeros, and we have to differentiate the first three outputs again. Then, we can see that the decoupling matrix has the same form as in previous iteration. By following the algorithm, it requires to add another integrator to w_1, and continue differentiating the first three outputs. And, the iteration ends, since the decoupling matrix finally has full rank. The extended system is in the following form:

$$\begin{bmatrix} p_{xm}^{(5)} \\ p_{ym}^{(5)} \\ p_{zm}^{(5)} \\ \psi^{(3)} \end{bmatrix} = \underbrace{\begin{bmatrix} b^P \\ \\ b^\psi \end{bmatrix}}_{b_e} + \underbrace{\begin{bmatrix} A^P \\ \\ A^\psi \end{bmatrix}}_{A_e} \underbrace{\begin{bmatrix} \ddot{w}_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}}_{u}$$

of which the vector relative degree is $\{5,5,5,3\}$.

We can rewrite the true system in normal form (ξ,η) of modeled system. Define $\xi_1^1 = p_{xm}$, $\xi_1^2 = p_{ym}$, $\xi_1^3 = p_{zm}$, $\xi_1^4 = \psi$, $\xi_1^P = [\xi_1^1\ \xi_1^3\ \xi_1^3]^T$, $\xi_1^\psi = \xi_1^4$ and we have

$$\left.\begin{aligned} \dot{\xi}_1^P &= \xi_2^P \\ \dot{\xi}_2^P &= \xi_3^P + \rho(x) \\ \dot{\xi}_3^P &= \xi_4^P \\ \dot{\xi}_4^P &= \xi_5^P \\ \dot{\xi}_5^P &= b^P + A^P u \end{aligned}\right\} \qquad \left.\begin{aligned} \dot{\xi}_1^\psi &= \xi_2^\psi \\ \dot{\xi}_2^\psi &= \xi_3^\psi \\ \dot{\xi}_3^\psi &= b^\psi + A^\psi u \end{aligned}\right\} \tag{3.28}$$

in which

$$\rho(x) = \frac{1}{m} R T_M \begin{bmatrix} -\sin a_{1s} \\ \sin b_{1s} - T_T/T_M \\ \cos a_{1s} \cos b_{1s} - 1 \end{bmatrix}. \tag{3.29}$$

In above, the linearized model system does not contain any unobservable (zero) dynamics and hence is minimum phase, since the sum of Kronecker indices $\{\gamma_1,\gamma_2,\gamma_3,\gamma_4\}$ and $\sum_{j=1}^4 \gamma_j = 18$ is equal to the order of the extended system $n_e = 16+2$. As defined in [30],

the system is said to be slightly non-minimum phase, since the true system is non-minimum phase but the approximate system is minimum phase.

We can then apply the tracking control law designed for the model system to the true system, which is

$$\begin{aligned} u &= -A_e^{-1}b_e \\ &\quad + A_e^{-1}\begin{bmatrix} y_{d1}^{(\gamma_1+1)} - \alpha_1^1 e_1^{(\gamma_1)} - \cdots - \alpha_{\gamma_1+1}^1 e_1 \\ \vdots \\ y_{d4}^{(\gamma_4+1)} - \alpha_1^4 e_4^{(\gamma_4)} - \cdots - \alpha_{\gamma_4+1}^4 e_4 \end{bmatrix} \end{aligned}$$

where $e_i^{(j)} = \xi_{j+1}^i - y_{di}^{(j)}$ for $j = 0, \ldots, \gamma_i$ and $i = 1, \ldots, 4$ and the polynomials

$$s^{\gamma_i+1} + \alpha_1^i s^{\gamma_i} + \cdots + \alpha_{\gamma_i+1}^i \tag{3.31}$$

chosen Hurwitz. The following theorem provides a bound for the performance of this control when applied to the true system.

Theorem 3.4.1 *Given that the desired trajectory and its first $\gamma_i - 1$ derivatives are bounded, the states of the system (3.28) are bounded and the tracking errors satisfy*

$$\begin{aligned} |e^1| &= |\xi_1^1 - y_{d1}| \le k\epsilon \\ &\vdots \\ |e^4| &= |\xi_1^4 - y_{d4}| \le k\epsilon \end{aligned} \tag{3.32}$$

where $\epsilon = \max(|a_{1s}|, |b_{1s}|, |T_T/T_M|)$ and k is bounded.

Proof. For the i^{th} component of ρ, it can be easily shown that

$$\|\rho_i\| \le \frac{T_M}{m}|r_i| \left\| \begin{bmatrix} -\sin a_{1s} \\ \sin b_{1s} - T_T/T_M \\ \cos a_{1s}\cos b_{1s} - 1 \end{bmatrix} \right\| \le \epsilon \underbrace{\sqrt{14}|T_M/m|}_{K}$$

where r_i is the i^{th} row of R and the following facts are used: $|\sin x| \le |x|, \cos x \le |1 - x|$ for $x \in (-\pi/2, \pi/2]$. For illustration of bounded tracking, we consider ξ_1^1 only in the following

proof, and the result applies to $\xi_1^2, \xi_1^3, \xi_1^4$ as well. Define an error vector as

$$e^1 = \begin{bmatrix} \xi_1^1 \\ \vdots \\ \xi_5^1 \end{bmatrix} - \begin{bmatrix} y_{d1} \\ \vdots \\ y_{d1}^4 \end{bmatrix} \tag{3.33}$$

Then the closed loop system can be expressed as

$$\begin{bmatrix} \dot{e}_1^1 \\ \dot{e}_2^1 \\ \dot{e}_3^1 \\ \dot{e}_4^1 \\ \dot{e}_5^1 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & & & 1 \\ -\alpha_0^1 & -\alpha_1^1 & \cdots & -\alpha_4^1 \end{bmatrix}}_{A_1} \begin{bmatrix} e_1^1 \\ e_2^1 \\ e_3^1 \\ e_4^1 \\ e_5^1 \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ \epsilon K \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{\zeta}$$

We first show that e^1 is bounded. To this end, consider as Lyapunov function for the above error system $V = {e^1}^T P e^1$ where $P > 0$ is chosen so that $A_1^T P + P A_1 = -I$. This can be done since $\dot{e}^1 = A_1 e^1$ is stable. Taking the derivative of V along the trajectory, we find

$$\begin{aligned} \dot{V} &= -|e^1|^2 + 2{e^1}^T P \zeta \\ &\leq -|e^1|^2 + \epsilon 2 |e^1| \bar{\sigma}(P) K \\ &\leq -|e^1|^2 + \epsilon |e^1| K \end{aligned}$$

Thus, $\dot{V} < 0$ whenever $|e^1| \geq \epsilon K$ which implies $|e^1|$ is bounded and, hence, $|\xi^1|$ is bounded. Furthermore, we can conclude that the tracking error will be $O(\epsilon)$. □

In above, we have demonstrated the idea of applying approximate linearization in position and heading mode. However, one can apply the same principle to show that there exists another set of outputs, such as $\{p_x, p_y, p_z, \beta\}$, called **position and side slip angle mode**, approximately linearize the system. In particular, if the controller tries to keep slide slip angle, β, to zero, the system is said to be operated in **coordinated flight mode**. Since the desired side-ward movement is zero, and the desired heading is aligned with the tangent of the trajectory projected on X-Y plane, *i.e.* $\psi = \text{atan2}(v_y^p, v_x^p)$. Thus, to minimize the drag force introduced by the fuselage, coordinated flight mode is the most desirable mode of operation.

3.5 Trajectory Generation based on Differential Flatness

In the previous section, we have shown the idea of using approximate linearization to derive a tracking controller. However, one can use the same framework to design a trajectory generator. In this section, we prove by construction that the approximate model with the same outputs is differentially flat and hence the state and input can be expressed as functions of the outputs and their derivatives. This property is very useful for real-time trajectory generation in hierarchical control system as described in [46].

A system is said to be differentially flat, if there exist output functions, called flat outputs, such that all states and inputs can be expressed in terms of the flat outputs and their derivatives.

Definition 3.5.1 (Differentially Flat System[28, 102]) *Given a system* $\dot{x} = f(x,u)$ *has states* $x \in \mathbb{R}^n$, *and inputs* $u \in \mathbb{R}^m$, *the system is said to be differentially flat if there exist outputs* $y \in \mathbb{R}^m$ *of the form* $y = y(x,u,\dot{u},\ldots,u^{(p)})$ *such that,* $x = x(y,\dot{y},\ldots,y^{(q)})$, $u = u(y,\dot{y},\ldots,y^{(q)})$. □

There doesn't exist any systematic characterization of flat systems and the search for flat outputs is usually difficult. However, as stated in [28], dynamically feedback linearizable systems [39] by a special class of dynamic feedbacks, called endogenous, are differentially flat.

As shown above, the helicopter model is approximately linearized. Hence, it suggests that one could generate an approximate state-input trajectory which is close to the true state-input trajectory. Furthermore, it can also be used for the generation of output trajectory while taking state and input constraints into consideration. In following, we present a constructive proof for showing that the model system with positions and heading as outputs is differentially flat.

Theorem 3.5.2 *Consider the system equations(3.27)(3.25)(3.26) with output chosen to be* $\{p_{xm}, p_{ym}, p_{zm}, \psi\}$. *The resulting system is differentially flat on sets where* $\phi \neq \pm\pi/2$ *and*

$\theta \neq \pm\pi/2$.

Proof: To show that all the states can be derived by the outputs and their derivatives, first we rewrite (3.27) as

$$\underbrace{\begin{bmatrix} \ddot{p}_{xm} \\ \ddot{p}_{ym} \\ \ddot{p}_{zm} \end{bmatrix}}_{\ddot{P}_m} = e^{\hat{z}\psi}e^{\hat{y}\theta}e^{\hat{x}\phi}\begin{bmatrix} 0 \\ 0 \\ -T_M/m \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{3.34}$$

and the equation becomes

$$e^{-\hat{z}\psi}\begin{bmatrix} \ddot{p}_{xm} \\ \ddot{p}_{ym} \\ \ddot{p}_{zm} - g \end{bmatrix} = e^{\hat{y}\theta}e^{\hat{x}\phi}\begin{bmatrix} 0 \\ 0 \\ -T_M/m \end{bmatrix} \tag{3.35}$$

By taking norm on both sides, T_M can be obtained as

$$T_M = m\sqrt{(\ddot{p}_{xm})^2 + (\ddot{p}_{ym})^2 + (\ddot{p}_{zm} - g)^2} \tag{3.36}$$

and hence $\dot{T}_M, \ddot{T}_M, T_M^{(3)} = \ddot{w}_1$. From (3.35), we can simplify as

$$e^{-\hat{z}\psi}\begin{bmatrix} \ddot{p}_{xm} \\ \ddot{p}_{ym} \\ \ddot{p}_{zm} - g \end{bmatrix}\frac{-m}{T_M} = \begin{bmatrix} \sin\theta\cos\phi \\ -\sin\phi \\ \cos\theta\cos\phi \end{bmatrix} \tag{3.37}$$

Thus, one can easily verify that

$$\phi = \sin^{-1}\left(\frac{-\ddot{p}_{xm}\sin\psi + \ddot{p}_{ym}\cos\psi}{T_M/m}\right) \tag{3.38}$$

and hence

$$\theta = \text{atan2}\left(\frac{\ddot{p}_{xm}\cos\psi + \ddot{p}_{ym}\sin\psi}{-\cos\phi\ T_M/m}, \frac{\ddot{p}_{zm} - g}{-\cos\phi\ T_M/m}\right) \tag{3.39}$$

By applying Ψ^{-1} to the derivatives of the Euler angles, we obtain vector ω. Then, one can solve for a_{1s}, b_{1s} and T_T from the Euler equations. Hence, taking the derivatives, we have $w_2 = \dot{T}_T$, $w_3 = \dot{a}_{1s}$, $w_4 = \dot{b}_{1s}$. Hence, we have set up that

$$(P, v^P, \phi, \theta, \psi, \omega^b, T_M, T_T, a_{1s}, b_{1s}, w_1, \dot{w}_1, \ddot{w}_1, w_2, w_3, w_4)$$
$$\mapsto (P, \dot{P}, \ddot{P}, P^{(3)}, P^{(4)}, P^{(5)}, \psi, \dot{\psi}, \ddot{\psi}, \psi^{(3)})$$

is a diffeomorphism on sets where $\phi \neq \pm\pi/2$ and $\theta \neq \pm\pi/2$ showing that the model system is indeed differentially flat. □

Similarly, one can show that in coordinated flight mode, $\{p_{xm}, p_{ym}, p_{zm}, \beta\}|_{\beta \equiv 0}$ are flat outputs of the model system.

Proposition 3.5.3 *Consider the system equations(3.27)(3.25)(3.26) with output chosen to be $\{p_{xm}, p_{ym}, p_{zm}, \beta\}|_{\beta \equiv 0}$. The resulting system is differentially flat on sets where $\phi \neq \pm\pi/2$ and $\theta \neq \pm\pi/2$.*

Proof: By using the fact that $\psi = \text{atan2}(v_y^p, v_x^p)$ when $\beta \equiv 0$, and applying Theorem 3.5.2. It is straightforward to show that

$$(P, v^P, R, \omega^b, T_M, T_T, a_{1s}, b_{1s}, w_1, \dot{w}_1, \ddot{w}_1, w_2, w_3, w_4)$$
$$\mapsto (P, \dot{P}, \ddot{P}, P^{(3)}, P^{(4)}, P^{(5)}, \beta, \dot{\beta}, \ddot{\beta}, \beta^{(3)})$$

is a diffeomorphism. Hence the result.

Conditions for switching between different flat outputs for trajectory generation has been studied in [51].

3.6 Nonlinear Control Design Based on Outer Flatness

A system $\dot{x} = f(x, t, u)$ is called *differentially flat* if there exist output functions, called *flat outputs*, such that all states and inputs can be expressed in terms of the flat outputs

and their derivatives [27]. Differential flatness has been applied to approximate models of aircraft [72] and helicopter [49] for trajectory generation. The full helicopter dynamics are not flat in general, however it can be shown that the dynamics can be partitioned into an "inner system" (*e.g.*the attitude dynamics) and an "outer system" (*e.g.*the position dynamics) where the outer system is flat. This scheme has been successfully used for generating a two stage control synthesis for many systems which are not completely flat [103]. Such a scheme which utilizes the flatness of the outer system is roughly illustrated in Figure 3.5. In the figure, P_O is the outer system which is flat, and P_I is the inner system which is not necessarily flat. Given a desired output trajectory, say $y_d^O(\cdot)$, the mapping F in Figure 3.5 utilizes the flatness property of the outer system to generate an desired output trajectory $y_d^I(\cdot)$ for the inner system. The control synthesis for the overall system then reduces to the design of an inner system controller, C, which drives the inner system output $y^I(t) \to y_d^I(t)$ (exponentially) as $t \to \infty$. As the inner system output converges, one can show that the outer system output converges to the desired one, $y^O(t) \to y_d^O(t)$ as $t \to \infty$. That is, the overall system asymptotically tracks the desired trajectory.

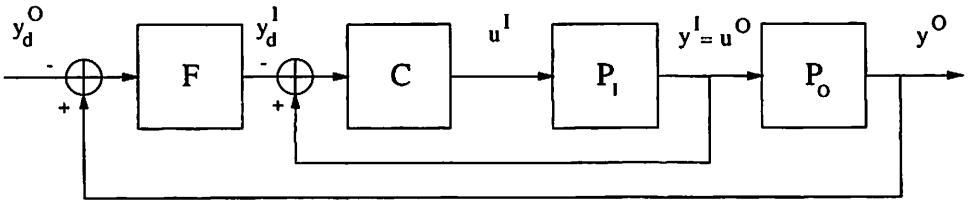


Figure 3.5: Partitioned inner and outer systems.

It has been shown in [49] that the helicopter dynamics are *approximately* differentially flat with the position and heading $\{P, \psi\}$ as the flat outputs. The approximation is based on the assumption that the coupling terms a_{1s}, b_{1s}, T_T are small and can be neglected in the model. So if $a_{1s}, b_{1s}, T_T \approx 0$, the outer system dynamics (3.24) can be rewritten as:

$$\ddot{P} = \frac{1}{m} R(\Theta) \begin{bmatrix} 0 \\ 0 \\ -T_M \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + h \tag{3.40}$$

with

$$h = \frac{1}{m} R(\Theta) \begin{bmatrix} -T_M \sin a_{1s} \\ T_M \sin b_{1s} - T_T \\ -T_M(\cos a_{1s} \cos b_{1s} - 1) \end{bmatrix}.$$

where the inputs are $u^O = y^I = (\Theta, T_M)$, and the outputs are $y^O = (P, \dot{P}, \ddot{P}, \psi)$. The inner system dynamics are described by (3.25), (3.26) and in addition the *rotary wing dynamics* which can be approximated by the following equations(for details see [85]):

$$T_M = c_{M1}\theta_M + c_{M3}\theta_M^3, \quad T_T = c_{T1}\theta_T + c_{T3}\theta_T^3, \quad a_{1s} = -B, \quad b_{1s} = A$$

for near hovering operation, where θ_M, θ_T are the main and tail rotor collective pitch, and B, A are the longitudinal and lateral cyclic pitch. One must notice that this approximation introduces a small non-vanishing modeling error h which depends on $\Theta, T_M, a_{1s}, b_{1s}, T_T$. We will soon show its effect on the stability of the closed-loop system.

The control design for the overall system is be based on an assumption that there exists a controller C such that $e^I = 0$ is an exponentially stable equilibrium point for the inner error system:

$$\dot{e}^I = f(e^I, e^O, t)|_{e^O=0}, \quad f(0,0,t) = 0$$

where $e^O = y^O - y_d^O$ and $e^I = y^I - y_d^I$. There have been various design methodologies proposed for the controller of the inner system, *e.g.* [63]. The details of the design of inner controller deployed in here, please refer to [50]. In this section, we are only interested in the performance of the overall system assuming such a controller C is already available. As shown in [49], for the approximated outer system (3.27), there exists a smooth mapping from the outer system output to the inner system output:

$$\begin{aligned} \Phi: \ \mathbb{R}^4 &\to \mathbb{R}^4 \\ (\ddot{p}, \psi) &\mapsto (\Theta, T_M) \end{aligned}$$

which is defined by the equations:

$$\begin{aligned} T_M &= m\sqrt{(\ddot{p}_x)^2 + (\ddot{p}_y)^2 + (\ddot{p}_z - g)^2} \\ \phi &= \sin^{-1}\left(\frac{-\ddot{p}_x \sin\psi + \ddot{p}_y \cos\psi}{T_M/m}\right) \\ \theta &= \text{atan2}\left(\frac{\ddot{p}_x \cos\psi + \ddot{p}_y \sin\psi}{-T_M \cos\phi/m}, \frac{\ddot{p}_z - 1}{-T_M \cos\phi/m}\right) \\ \psi &= \psi \end{aligned}$$

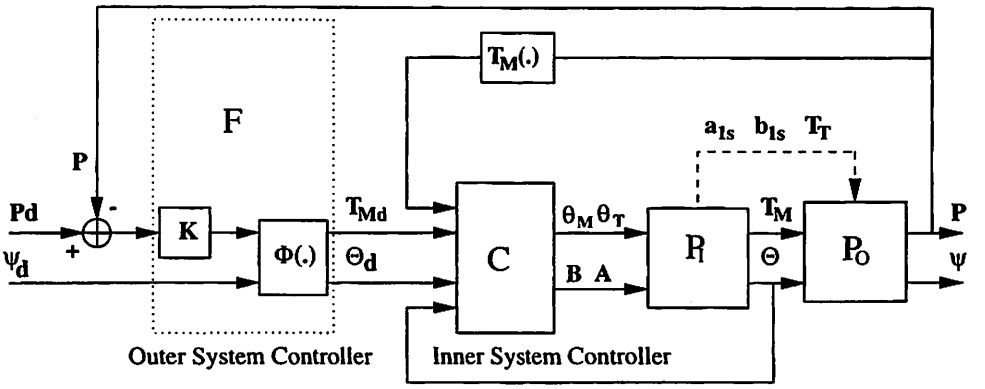


Figure 3.6: Block diagram of control scheme.

where $\phi, \theta \neq \pm\pi/2$. Suppose that the desired output trajectory of the outer system is $y_d^O = (p_d, \dot{p}_d, \ddot{p}_d, \psi_d)$. To obtain the desired trajectory of the inner system, we define a pseudo-input vector:

$$v_p = \ddot{p}_d + K_v(\dot{p} - \dot{p}_d) + K_p(p - p_d) \tag{3.41}$$

where K_p, $K_v \in \mathbb{R}^{3\times 3}$ are control parameters. With the above pseudo-input, the desired output of the inner system y_d^I is given by:

$$(\Theta_d, T_{Md}) = \Phi(v_p, \psi_d). \tag{3.42}$$

A more detailed schematic of the controller for this system is illustrated in Figure 3.6. Clearly, if the inner system *exactly* tracks the desired trajectory (Θ_d, T_{Md}), that is, $y_d^I = y^I$ in Figure 3.5, then the behavior of the overall closed-loop system is specified by the outer system only, which, due to chosen the control law (3.41), is *approximately* a linear system with poles assigned by the parameters K_v, K_p.

Now if we summarize all conditions so far and rewrite the dynamics of the overall closed-loop system in terms of the tracking errors e^I and e^O of the inner and outer systems respectively, they have the form:

$$\begin{cases} \dot{e}^I &= f(e^I, e^O, t) \\ \dot{e}^O &= Ae^O + g(e^I, t) + h(e^I, e^O, t) \end{cases} \tag{3.43}$$

where

$$g(e^I,t) = \frac{1}{m}R(\Theta)\begin{bmatrix}0\\0\\-T_M\end{bmatrix} - \frac{1}{m}R(\Theta_d)\begin{bmatrix}0\\0\\-T_{Md}\end{bmatrix}.$$

In the above equations, $f(e^I,e^O,t)$ is in general a function of both e^I and e^O since the input of the inner system is a function of e^O.

The function $h(e^I,e^O,t)$ from (3.27) is a small non-vanishing approximation error, and $g(e^I,t)$ vanishes when the inner system exactly tracks the desired trajectory, *i.e.*, $g(0,t)=0$. Since the helicopter model is smooth and many of the parameters are physically bounded, $g(e^I,t)$ is in fact (globally) bounded as $\|g(e^I,t)\| \le L_1\|e^I\|$ for some constant $L_1 > 0$[1] and $f(e^I,e^O,t)$ is Lipschitz, *i.e.*$\|f(e_1^I,e_1^O,t) - f(e_2^I,e_2^O,t)\| \le L_2(\|e_1^I - e_2^I\| + \|e_1^O - e_2^O\|)$.

Stability Analysis

We now analyze the performance of the overall closed-loop system. As we have argued before, the function f in (3.43) is in general a function of both e^I and e^O. However, in practice, the inner system is usually designed to have a much faster convergence rate than the outer system. To simplify the analysis, for now we assume that the inputs $T_{Md}(\cdot)$ and $\Theta_d(\cdot)$ of the inner system are approximately constant, and thus f is only a function of e^I (the more general case will be presented afterwards).

Recall that given an general system $\dot{x} = f(x,t)$, by the Lyapunov theorem and its converse [87], the system is exponentially stable if and only if there exists a *Lyapunov function* $V(x,t)$ satisfying:

$$\alpha_1\|x\|^2 \le V(x,t) \le \alpha_2\|x\|^2 \tag{3.44}$$

$$\frac{\partial V}{\partial t} + \frac{\partial V}{\partial x}f(x,t) \le -\alpha_3\|x\|^2 \tag{3.45}$$

$$\left\|\frac{\partial V}{\partial x}\right\| \le \alpha_4\|x\| \tag{3.46}$$

for some positive constants $\alpha_1,\alpha_2,\alpha_3,\alpha_4 > 0$.

However, we may write:

$$f(e^I,e^O,t) = f(e^I,0,t) + d(e^I,e^O,t)$$

[1]Such a L can be estimated from the system equation (3.24).

where $d(e^I, e^O, t) = f(e^I, e^O, t) - f(e^I, 0, t)$. The nominal system $\dot{e}^I = f(e^I, 0, t)$ is exponentially stable as designed. We can apply this theorem to both the nominal outer system $\dot{e}^O = Ae^O$ and the nominal inner system $\dot{e}^I = f(e^I, 0, t)$ and denote the corresponding Lyapunov functions as V^O and V^I and the Lyapunov constants as $\alpha_1, \alpha_2, \alpha_3, \alpha_4 > 0$ and $\beta_1, \beta_2, \beta_3, \beta_4 > 0$ respectively.

Then for the overall system, we have the result:

Theorem 3.6.1 *Consider the following perturbed system:*

$$\begin{cases} \dot{e}^I &= f(e^I, e^O, t) &= f(e^I, 0, t) + d(e^I, e^O, t) \\ \dot{e}^O &= Ae^O + g(e^I, t) \end{cases} \tag{3.47}$$

where $g(e^I, t)$ is a perturbation term that satisfies $\|g(e^I, t)\| \leq L_1\|e^I\|$ for some $L_1 > 0$. If, for $f(e^I, e^O, t)$, there exists $L_2 > 0$ such that $\|f(e_1^I, e_1^O, t) - f(e_2^I, e_2^O, t)\| \leq L_2(\|e_1^I - e_2^I\| - \|e_1^O - e_2^O\|)$, then the overall system is exponentially stable if the product of the two Lipschitz constants satisfies the inequality:

$$L_1 \cdot L_2 < \frac{\alpha_3}{\alpha_4} \cdot \frac{\beta_3}{\beta_4}. \tag{3.48}$$

Proof: Since $f(e^I, e^O, t)$ is Lipschitz, we have $\|d(e^I, e^O, t)\| \leq L_2\|e^O\|$. We consider the candidate Lyapunov function $V = V^I + \mu V^O$ for the overall system. Then we have:

$$\begin{aligned} \dot{V} = \dot{V}^I + \mu\dot{V}^O &\leq -\beta_3\|e^I\|^2 + \beta_4 L_2\|e^I\|\|e^O\| - \mu\alpha_3\|e^O\|^2 + \mu\alpha_4 L_1\|e^O\|\|e^I\| \\ &= -(\|e^I\|, \|e^O\|)Q(\|e^I\|, \|e^O\|)^T \end{aligned}$$

where the matrix $Q \in \mathbb{R}^{2\times 2}$ is:

$$Q = \begin{bmatrix} \beta_3 & -\frac{1}{2}(\beta_4 L_2 + \mu\alpha_4 L_1) \\ -\frac{1}{2}(\beta_4 L_2 + \mu\alpha_4 L_1) & \mu\alpha_3 \end{bmatrix}.$$

Q is positive definite if and only if $\det(Q) > 0$. That is, there exists $\mu > 0$ such that:

$$-\alpha_4^2 L_1^2 \mu^2 + (4\beta_3\alpha_3 - 2\beta_4 L_2 \alpha_4 L_1)\mu - \beta_4^2 L_2^2 > 0.$$

This is true if and only if the discriminant of the quadratic function of μ on the left hand side is positive which yields: $L_1 \cdot L_2 < \frac{\alpha_3}{\alpha_4} \cdot \frac{\beta_3}{\beta_4}$. ∎

This theorem states a very interesting fact about the system (3.47): heuristically, α_3 and β_3 are proportional to the convergence rates of the outer and inner systems respectively,[2] hence the stability of the perturbed systems requires *only* that the *product* of the Lipschitz constants of the perturbation terms is less than the *product* of the two convergence rates, regardless of the rate of each individual system.

The stability of a similar model of the overall closed-loop system has been studied before in [103], however, no explicit conditions are provided under which a μ exists such that the overall system is stable. Here, Theorem 3.6.1 give more detailed and useful results in characterizing the properties of the closed-loop system.

Although we have established the conditions for the system (3.47) to be exponentially stable, estimates of its Lyapunov constants indeed depend on L_1, L_2 and all the Lyapunov constants of the inner and outer systems. These constants can be optimized by maximizing the smaller eigenvalue of Q with respect to μ. We here omit the detail and carry on the analysis by assuming that the system (3.47) is exponentially stable and its Lyapunov constants are denoted by $\gamma_1, \gamma_2, \gamma_3, \gamma_4 > 0$. We now want to estimate the effect of the non-vanishing error term h on the performance of the closed-loop system (3.43). In general, we can no longer expect asymptotic stability when a non-vanishing perturbation is introduced. However, according to [43], we can still have good estimates of a bound on the tracking error and the rate of convergence outside this bound.

Proposition 3.6.2 *Assume that the system (3.47) has the Lyapunov constants* $\{\gamma_i\}_{i=1}^4$. *Then, for the closed-loop system (3.43), if* $\|h(e^I, e^O, t)\| \leq \delta < \frac{\gamma_3}{2\gamma_4}\sqrt{\frac{\gamma_1}{\gamma_2}}$, *then the tracking error of the overall system is bounded by* $b = \frac{2\gamma_4}{\gamma_3}\sqrt{\frac{\gamma_2}{\gamma_1}}\delta$, *and, outside this bound, the error exponentially decreases with a rate larger than* $\lambda = \frac{\gamma_3}{4\gamma_2}$.

The control parameters K_v and K_p can be adjusted so as to minimize the error bound b. For the helicopter model, the error term $h(e^I, e^O, t)$ is usually extremely small, as is δ. We

[2] A more precise estimates of the convergences rates are given by $\frac{\alpha_3}{2\alpha_2}$ and $\frac{\beta_3}{2\beta_2}$.

can also choose the control parameters such that the inner and outer systems have very fast rates of convergence, hence a large γ_3. Consequently, the error bound b is very small, and usually barely noticeable in simulations and experiments.

The nonlinear controller based on outer flatness has been successfully applied in landing an UAV based on computer vision and formation flight of UAV cluster with mesh stability as shown in [90, 91] and [89] respectively.

3.7 Simulation Results

We apply our approximate method and compare with the exact method on the true system. The initial conditions are $p_x = 0.1g$, $p_y = 0.05g$, $p_x = 0.2g$, $v_x^p = v_y^p = v_z^p = 0$, $\psi = 0.01$, $g = 9.8$ and other states are in trim conditions. For both control designs, the dominant conjugate poles are $-1.4 \pm 1.4283i$ and other poles are placed at -5. To illustrate the idea, we set y_d and all their derivatives equal to zero, and the controllers are required to hover the helicopter back to origin while turning the heading to zero.

The result of using exact linearization is shown in Figure 3.7, and that of using approximate linearization is shown in Figure 3.8. Both controllers are successful in stabilizing the outputs. However, in the exact method the internal dynamics, roll and pitch angles, are excited and continue to oscillate. Furthermore, it exhibits large control effort. While applying the approximate control law, the internal dynamics are stabilized. The control inputs are kept relatively small throughout the simulation, and it validates the assumption made on a_{1s}, b_{1s} and T_T.

We present the simulation result of the propsed nonlinear controller based on outer flatness. The inner controller has poles placed at -5 for controlling the main rotor thrust and at -10 and $-7 \pm 7.1414i$ for controlling the attitude dynamics; the outer controller has both poles placed at -2. With the same initial conditions and desired trajectory, the controller is applied to the exact model and the simulation result is shown in Figure 3.9. The nonlinear controller stabilizes the both inner and outer system and results in bouned errors. These phenomena have been predicted in Theorem 3.6.1 and Proposition 3.6.2.

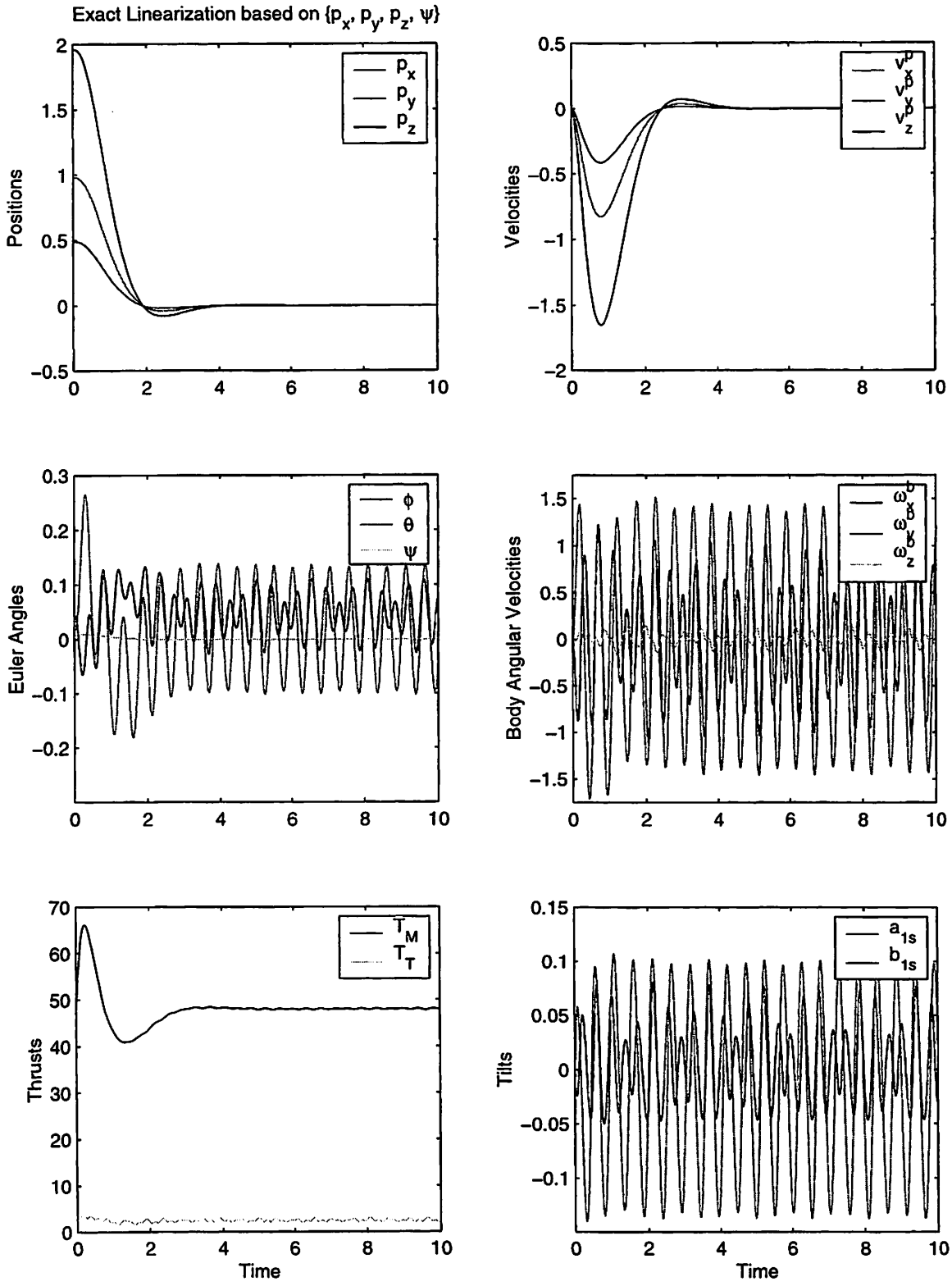


Figure 3.7: Exact input-output linearization applied on outputs $\{ p_x,\ p_y,\ p_z,\ \psi\}$

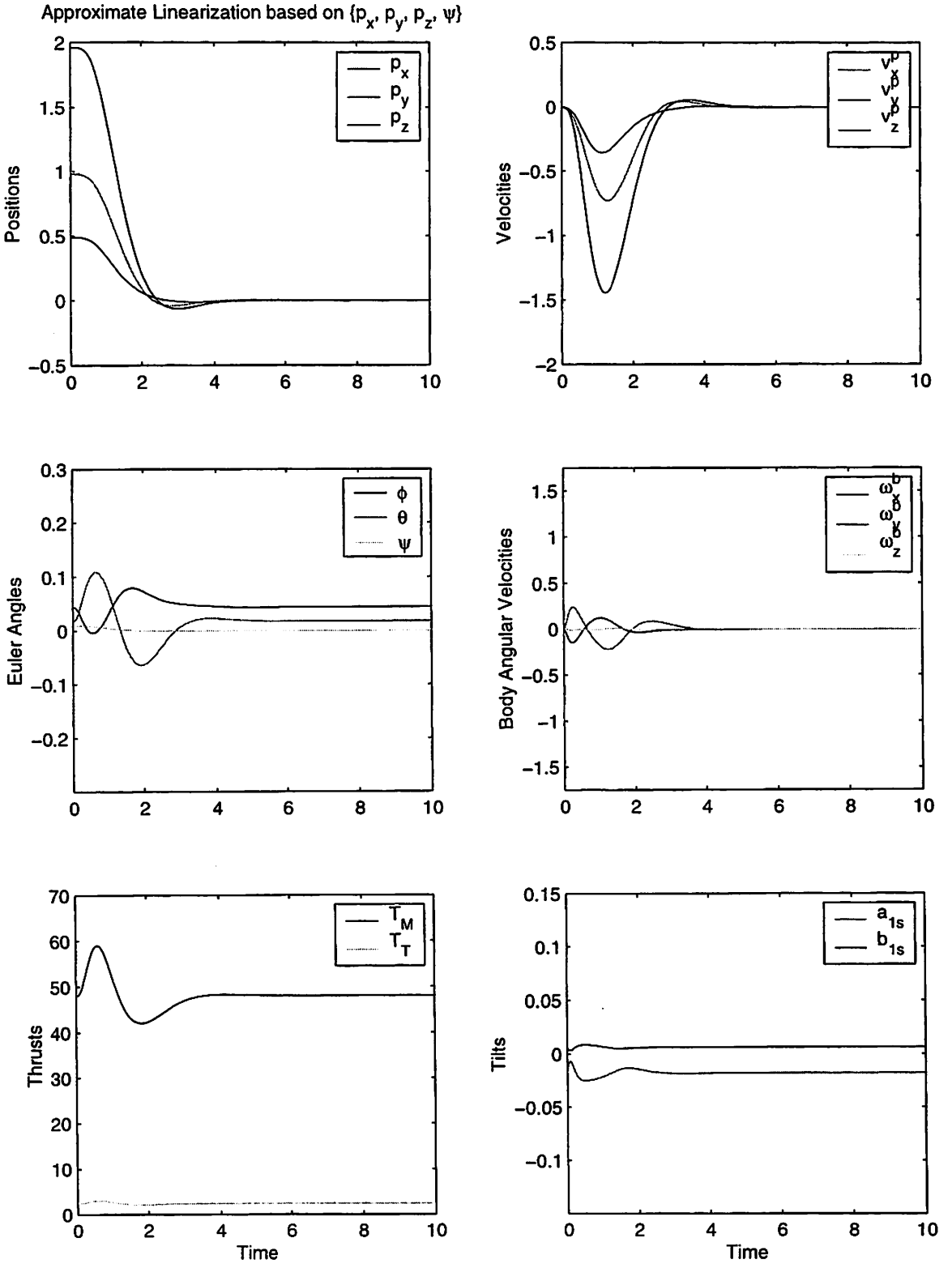


Figure 3.8: Approximate input-output linearization applied on outputs $\{p_x,\ p_y,\ p_z,\ \psi\ \}$

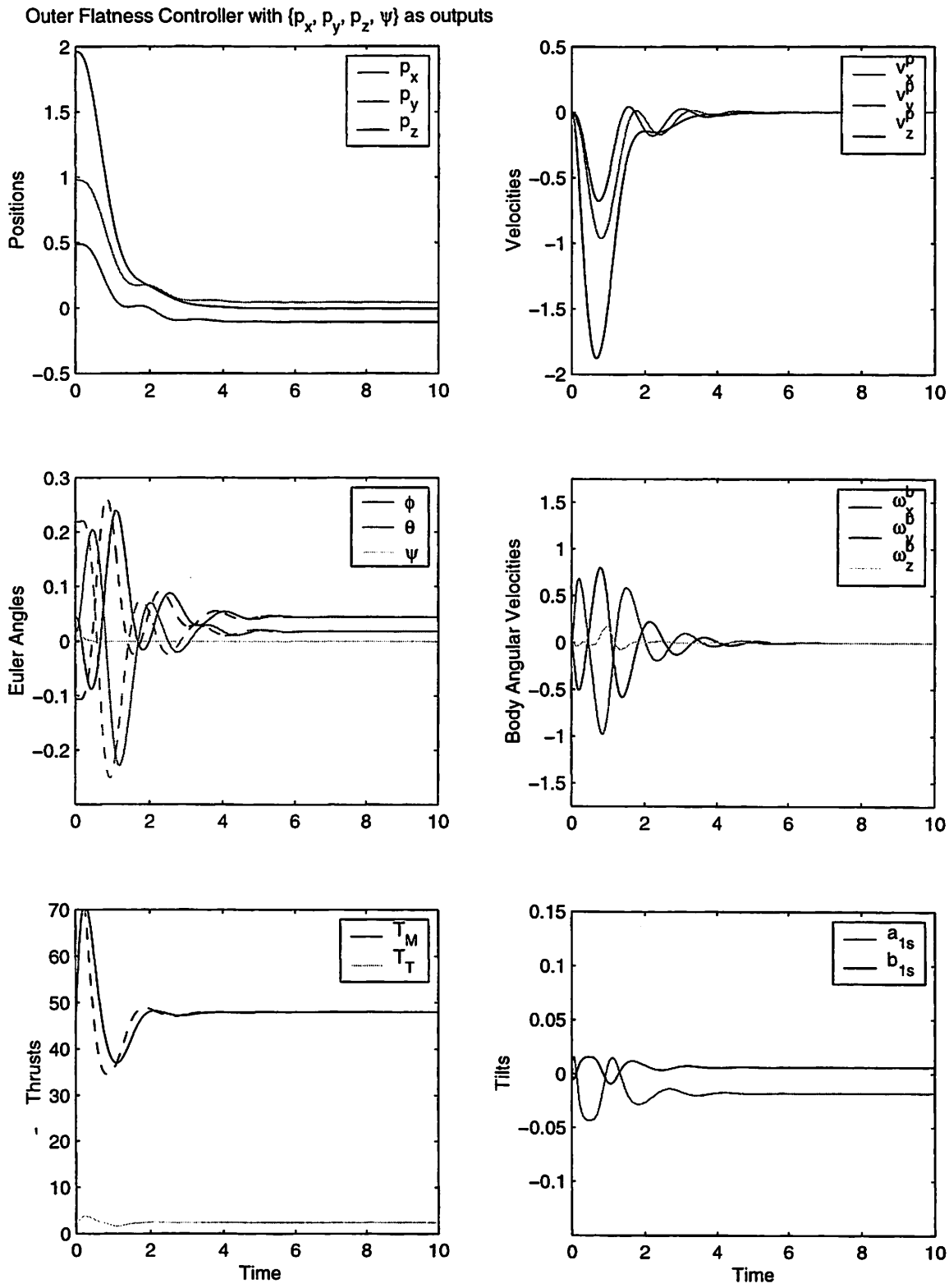


Figure 3.9: Nonlinear control design based on outer flatness with outputs $\{ p_x, p_y, p_z, \psi\}$ Notice that the dashed lines represent the desired trajectories of ϕ, θ, T_M.

3.8 Chapter Summary

In this chapter, the output tracking control design of a helicopter based unmanned aerial vehicle model based on approximate input-output linearization is illustrated. We show that the model cannot be converted into a controllable linear system via exact state space linearization. By neglecting the couplings between rolling/pitching moments and lateral/longitudinal forces, we show that the dynamically extended approximated system with positions and heading as outputs is linearizable without zero dynamics. We have proved that bounded tracking is achievable by applying the approximate control with a given bounded output trajectory. Next, we derive a diffeomorphism showing that the approximate system with the same outputs is differentially flat. By that, state and input can be expressed as functions of the outputs and their derivatives. Hence, output trajectory generation can take state and input constraints into consideration. Application of path generation for helicopter based on differential flatness can be found in [24]. Based on geometric control theory, we decompose the dynamics into two subsystems: inner and outer systems. A nonlinear controller is proposed based on differential flatness of the outer system. This control design only assumes that the outer system is differentially flat and the inner system is exponentially stable. Also, the assumptions made on the outer system can be applied to many different helicopter models. Hence, it is possible to design a nonlinear controller based on the propsed scheme and implement it for controlling actual helicopters.

Simulation results show that the approximate control law produces desired performance without excite the internal states into oscillation. The performance of the control design based on outer flatness is simulated. The nonlinear controller stabilizes both the inner and outer systems and it results in bouned errors.

To reduce the bounded error, one can simply design robust controllers to augment the presented controllers for compensating the effects due to the existence of non-vanishing terms. For distubance satisfying matching condition, sliding mode controller can be applied. For mismatched disturbance, controllers can be derived using the Backstepping method [55] and Dynamic Surface Control [95]. However, linear techniques such as μ-synthesis [80] can also be used since the systems are linearized by state feedback.

Computer Aided Control System Design(CACSD) has enabled the analysis and design of control system. The proofs of Theorem 3.3.4 and 3.3.3 are performed symbolically. The

description of the system (3.24) and controller derivation using Lie derivatives are computed symbolically.

In future, we will extend our control design to include the fuselage drag force, rotary wing dynamics and actuator dynamics. Robustness issue will be addressed to accommodate the presence of external disturbance and uncertainty of the dynamical model. The final tracking controller will be implemented and tested on a UAV called Ursa Minor as shown in Figure 3.10, on which we have mounted embedded controller, GPS, INS and wireless Ethernet.

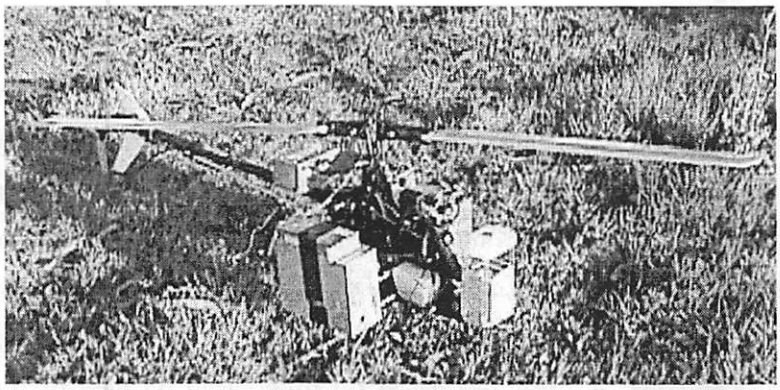


Figure 3.10: An unmanned aerial vehicle of Berkeley Aerobot fleet: Ursa Minor.

Chapter 4

Control Mode Switching Synthesis

In many control applications, a specific set of output tracking controllers of satisfactory performance have already been designed and must be used. When such a collection of *control modes* are available, an important problem is to be able to accomplish a variety of high level tasks by appropriately switching between the low-level control modes. In this chapter, we define the concept of control mode, and propose an algorithm for determining the sequence of control modes that will satisfy reachability tasks.

4.1 Introduction

Large scale systems like automated highway systems, air traffic management systems, UAV networks are multi-agent, multi-objective systems that operate in many modes of operation. This results in systems of very high complexity which may dramatically limit the applicability of current analysis and design methods.

One natural way to reduce the complexity of system design is by compositional methods. Compositional methods attempt to solve a complex problem by decomposing the problem into a sequence of smaller problems of manageable complexity. For example, in sophisticated flight management systems [54], modern aircraft fly from origin to destination while satisfying a large number of aerodynamic, scheduling, and air traffic constraints by switch-

ing among a finite set of *flight modes*, where each flight mode essentially corresponds to a different output tracking controller.

More generally, given a continuous control system, a *control mode* is defined as the operation of the system under a controller that is *guaranteed* to track a certain class of output trajectories. Different outputs of interest correspond to different control modes. Given a set of control modes, the mode switching problem attempts to find a switching *sequence* of the control modes as well as *switching conditions* in order to satisfy various tasks. In this chapter, we focus on reachability tasks.

Problem 4.1.1 *Given a control system, a set of control modes for the system, determine whether there exists a sequence of modes that will steer the system from an initial control mode to a desired final control mode. If such a sequence exists, then determine the switching conditions.*

Clearly, in this setup, many more interesting questions can be asked. For example one can ask what are the optimal switching conditions, where optimality can mean minimum time, or minimum number of switchings. Furthermore, one can ask whether a set of modes is sufficient for performing a reachability, or more general, task. In this paper, we do not consider these more difficult questions. We focus on Problem 4.1.1, while setting up the framework for considering these more general questions in the future.

4.2 Problem Formulation

In this section, the definition of control modes is given and a planner helicopter controller design is used as an example to illustrate the ideas of control mode. Finally, the problem of mode switching is posed.

4.2.1 Control Modes

Throughout this chapter, we consider a nonlinear system modeled by differential equations of the form

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t), \quad x(t_0) = x_0, \quad t \geq t_0 \tag{4.1}$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in R^m$. The system is assumed to be as smooth as needed. We now define a concept of control mode.

Definition 4.2.1 (Control Modes) *A control mode, labeled by q_i, is the operation of the nonlinear system (4.1) under a closed-loop feedback controller described by*

$$\begin{cases} y^i(t) &= h^i(x(t)) \to r^i \\ u(t) &= k^i(x(t), r^i) \end{cases} \tag{4.2}$$

where $i \in \{1, \ldots, N\}$, $y^i(t) \in \mathbb{R}^m$, $r^i \in \mathbb{R}^m$, $h^i : \mathbb{R}^n \to \mathbb{R}^m$, $k^i : \mathbb{R}^n \times \mathbb{R}^{s_i} \to \mathbb{R}^m$.

The *constant* trajectory r^i is the desired output trajectory, and $y^i(t)$ is the output vector which shall track r^i. Hence, in each mode the closed-loop dynamics is parameterized by the desired output trajectory, r^i. Clearly, more general definition of mode can be easily defined. For example, one can define a mode as the operation of the system under an open loop control, or one can consider more general classes of trajectories, or may require the input to satisfy certain input constraints. Such generalizations will be considered in future research.

In this chapter we are interested in switching between controllers, rather than the design of output tracking controllers. We therefore make the following assumption.

Assumption 1 *For each control mode q_i, $i \in \{1, \ldots, N\}$, we assume that a controller (4.2) has been designed to achieve asymptotic output tracking, while the state satisfies a set of state constraints $x(t) \in X^i \subset \mathbb{R}^n$, when the initial condition of the system (4.1) start in the set $X_0^i \subseteq X^i \subseteq \mathbb{R}^n$.*

The above assumption is justified given the maturity of output tracking controllers for large classes of linear and nonlinear systems [39, 40, 22]. Of course, much remains to be discovered for the performance of these controllers in the presence of state constrains. For pointwise-in-time input and/or state hard constraints in nonlinear control systems, a reference governor [7] is proposed to augment the system to fulfill the constraints as well as stability and tracking requirements.

Because of incompatible constraints, one cannot simply switch from one mode to another. A natural question is then whether this task can be achieved by a *finite sequence* of modes.

In general, controller derivation follows a typical design pattern. First, an **objective** of control design which is defined by the *output*, *desired output*, *purpose* and *constraints* is given. Then, a controller is derived and implemented based on a specific control design methodology. Depending on **implementation**, the **performance** of the closed-loop system is specified by *initial set*, *flow*[1], *stability type*, and *desired output range*. A complete **control mode specifications** comprises objective, implementation and performance. On the other hand, a **control mode** is defined by *output*, *desired output*, and *purpose*; while in Assumption 1, *constraints*, *initial set*, and *stability type* are specified. Whereas, *flow* and *desired output range* are useful for describing the dynamical behaviors of control modes and hence will be used to determine possible sequence of modes.

Example 4.2.2 2-D Helicopter Model and Control Modes *In this example, a helicopter model [65] described in longitudinal and vertical axes with simplified force and moment generation process is considered. The x, z-axes of the spatial frame are pointing to north and down directions. The body x-axis is defined from the center of gravity to the nose of the helicopter, and body z-axis is pointing down from the center of gravity. The motion of the helicopter is controlled by main rotor thrust, T_M and longitudinal tilt path angle, a.*

[1]The solution of (4.2) that starts from a point of the initial set at time $t = 0$.

The pitch angle is defined by θ. The equations of motion can be expressed as:

$$\begin{bmatrix} \ddot{p}_x \\ \ddot{p}_z \end{bmatrix} = \frac{1}{m} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} -T_M \sin a \\ -T_M \cos a \end{bmatrix} + \begin{bmatrix} 0 \\ g \end{bmatrix} \tag{4.3}$$

$$\ddot{\theta} = \frac{1}{I_y}(M_M a + h_M T_M \sin a) \tag{4.4}$$

The state vector is defined as $x = [p_x, \dot{p}_x, p_z, \dot{p}_z, \theta, \dot{\theta}]^T \in \mathbb{R}^6$. *The inputs vector is defined as* $u = [T_M, a]^T \in \mathbb{R}^2$.

In this example, each controller is designed as regulator hence the output is asked to track the desired output which is specified as a constant trajectory. In generally, a controller can be specified to have the output to track a desired output with respect to certain classes of trajectories. Four control modes are designed for different control objective.

Mode Label	*Class*		*Output*	*Desired Output*	*Purpose*	*Constraint Set*
q_1	*Hover*	y^1	$= h^1(x)$	$r^1 \in R^1$	$y^1 \to r^1$	X^1
			$= [p_x, p_z]^T$	$R^1 \subseteq \mathbb{R} \times [-h_{\max}, 0]$		
q_2	*Cruise*	y^2	$= h^2(x)$	$r^2 \in R^2$	$y^2 \to r^2$	X^2
			$= [\dot{p}_x, p_z]^T$	$R^2 \subseteq \mathbb{R}_+ \times [-h_{\max}, 0]$		
q_3	*Acc/ALH*	y^3	$= h^3(x)$	$r^3 \in R^3$	$y^3 \to r^3$	X^3
			$= [\ddot{p}_x, p_z]^T$	$R^3 \subseteq \mathbb{R}_+ \times [-h_{\max}, 0]$		
q_4	*Dec/ALH*	y^4	$= h^4(x)$	$r^4 \in R^4$	$y^4 \to r^4$	X^4
			$= [\ddot{p}_x, p_z]^T$	$R^4 \subseteq \mathbb{R}_- \times [-h_{\max}, 0]$		
q_5	*Climb*	y^5	$= h^5(x)$	$r^5 \in R^5$	$y^5 \to r^5$	X^5
			$= [V, \gamma]^T$	$R^5 \subseteq \mathbb{R}_+ \times (0, \pi/2]$		
q_6	*Descent*	y^6	$= h^6(x)$	$r^6 \in R^6$	$y^6 \to r^6$	X^6
			$= [V, \gamma]^T$	$R^6 \subseteq \mathbb{R}_+ \times [-\pi/2, 0)$		

where total velocity $V = \sqrt{\dot{p}_x^2 + \dot{p}_z^2}$, *flight path angle* $\gamma = \arctan 2(\dot{p}_z, \dot{p}_x)$, *and ALH stands for "Altitude Hold". Define* $X^i = \mathbb{R} \times [V_{\min}, V_{\max}] \times [-h_{\max}, 0] \times [V_{\min}, V_{\max}] \times [-\theta_{\max}, \theta_{\max}] \times [-\dot{\theta}_{\max}, \dot{\theta}_{\max}]$ *for* $i \in \{1, 2\}$ *and* $X^i = \mathbb{R} \times [V_{\min}, V_{\max}] \times \mathbb{R} \times [V_{\min}, V_{\max}] \times [-\theta_{\max}, \theta_{\max}] \times [-\dot{\theta}_{\max}, \dot{\theta}_{\max}]$ *for* $i \in \{3, \cdots, 6\}$. *Depending on a given implementation for each mode,*

the performance of the corresponding closed-loop system can be further specified by flow, stability type, initial set, and desired output range.

Given a set of control modes, a controlled system can perform complex task via the composition of the control modes. In the following, we use high-altitude takeoff of a helicopter as an example to illustrate the idea.

Example 4.2.3 High-altitude takeoff of a helicopter *In flight instruction for pilots, high-altitude takeoff can be performed by the following steps: (1) Set power to just below hover power and ease cyclic forward to start slow acceleration. Maintain heading with pedals. (2) After passing translational life airspeed and before reaching the landing gear groundspeed limitation, ease back on the cyclic to become airborne. (3) Maintain a level attitude over the surface to accelerate to normal climb speed. (4) Raise the nose to maintain normal climb speed.*

A controlled helicopter can perform high-altitude takeoff by utilizing a set of control modes. An example is shown in Figure 4.1 with the control modes defined in Example 4.2.2 and switching sequence, switching conditions and regulating values are translated from the flight instruction.

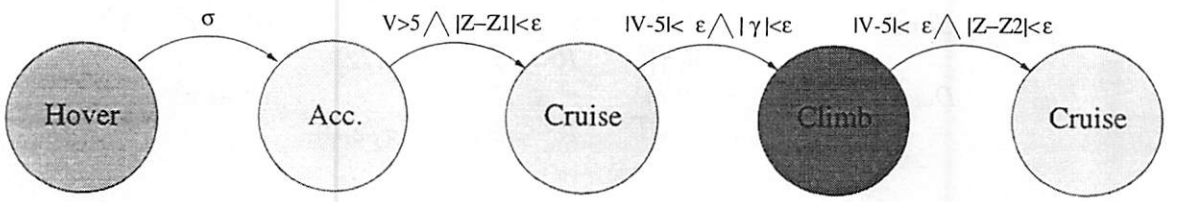


Figure 4.1: Control mode switching for high-altitude take-off of a helicopter. The switching sequence and conditions are based on flight instruction for pilots

Based on the above example, we can now define the mode switching problem that we will address in this chapter.

Problem 4.2.4 (Mode Switching Problem) *Given a initial control mode q_0 with desired set point r_0, does there exist a sequence of control modes such that for any state starting from mode q_0, the system can reach a desired mode q_F with set point r_F? If so, then determine a control mode sequence $q_0 \rightarrow q_1 \cdots \rightarrow q_F$ along with regulating values r^i for each control mode q_i, as well as state conditions for switching between control mode q_i and q_j.*

In Figure 4.1, one can consider that a mode switching problem is specified with hover mode given as initial mode with initial positions defined, cruise mode given as final mode with specified velocity and altitude, and the set of control modes defined in Example 4.2.2.

4.3 A Mode Switching Condition

In its full generality, Problem 4.2.4 can be posed as a controller synthesis problem for general hybrid systems [68]. Such synthesis methods involve *nested*, and possibly *cyclic* reachability computations, where each reachability computation involves computing the capture set of a differential game. Furthermore, recent decidability results for controller synthesis methods are rather restrictive [92].

In our mode switching problem, however, there is enough structure to take advantage of in order to simplify the complexity of the synthesis task. First of all, the continuous controllers are assumed to have been designed, and therefore we do not have to design the continuous part of the system, but simply determine the mode switching conditions. Furthermore, by imposing certain conditions on the allowable mode switches, *we reduce the complexity of the synthesis problem*, by essentially decoupling the discrete and continuous aspects of the synthesis.

To address the problem, we have to characterize the reachable set of each mode and switching condition among them. Let $\phi^i(t, r^i, x_0)$ denote the flow of system (4.1) operating in mode q_i with the controller defined by (4.2) for initial condition x_0, and desired output trajectory r^i.

Definition 4.3.1 (Predecessor set) *Given a set $P \subseteq X^i$, the predecessor set $Pre^i(P)$ in mode q_i is defined by*

$$Pre^i(P) = \{\ x_0 \in X^i \ \mid\ \exists r^i\ \exists t \geq 0\ \exists x \in P\ \textit{such that}\ x = \phi^i(t, r^i, x_0)\ \} \tag{4.5}$$

Therefore $Pre^i(P)$ consists of all states that can be reached from the set P in mode q_i, at *some* future time, for *some* output tracking parameter r_i. Furthermore, because of Assumption 1, we have a guarantee that throughout the whole trajectory, with initial condition starting in the set X_0^i, the state constraints are satisfied, that is $\phi^i(t, r^i, x_0) \in X^i$.

At this point, we assume that the Pre^i operators are available to us. There is extensive research in computing exactly, or approximately such reachable sets [59, 56, 68, 17]. However, in Section 4.5, we take advantage of the fact that in each control mode the closed loop dynamics for part of the state are stable in order to obtain easier approximations of reachable sets.

Next, we need to define the condition under which mode switching is allowed. Given control modes q_i, and q_j, one would typically allow a switch from mode q_i to q_j if during the operation of the system under mode q_i, the state reaches the allowable set of initial conditions X_0^j. If one allows this type of mode switching, then reachability critically depends on the initial conditions since some initial conditions in X_0^i may reach the set X_0^j of mode q_j while others may not. If this is the case, then nested reachability computations seem necessary. However, such nested computations can be avoided if one places the following condition on mode switching.

Definition 4.3.2 (Mode switching condition) *A transition from mode q_i to mode q_j is allowed only if*

$$X_0^i \subseteq Pre^i(X_0^j) \tag{4.6}$$

If the system starts at any $x_0 \in X_0^i$, then switching from mode q_i to q_j can occur at any time t such that $\phi^i(t, r^i, x_0) \in X_0^j$.

The condition expressed in Definition 4.3.2 guarantees that our ability to get from mode q_i to

mode q_j is independent of the choice of initial condition in X_0^i. The only thing that depends on the initial condition is *when* we will reach X_0^j, and *how* (choice of r^i) we will reach X_0^j, but not *if* we will reach X_0^j. This is reminiscent of the time-abstract bisimulation property from formal verification [59], or the consistency property for hierarchical systems [15, 83]. In this case, however, Definition 4.3.2 is quite different since no partitioning of the state space is involved.

4.4 Mode Sequence Synthesis

The mode switching condition 4.6, even though not necessary, makes the mode switching problem tractable since one can first determine the sequence of modes using discrete graph reachability, and then determine the continuous parameters r^i for each mode. This decouples the discrete from the continuous aspects of the problem.

4.4.1 Control Mode Graph

Given a collection of control modes $q_1, \ldots, q_N$, we define a *control mode graph* as a finite graph $(Q, \rightarrow)$, where the set of vertices of the graph is $Q = \{q_1, \ldots, q_N\}$, and the transition relation $\rightarrow \subseteq Q \times Q$ is defined by

$$(q_i, q_j) \in \rightarrow \quad \Longleftrightarrow \quad X_0^i \subseteq Pre^i(X_0^j) \tag{4.7}$$

We denote $(q_i, q_j) \in \rightarrow$ by $q_i \rightarrow q_j$. Given an initial control mode q_0, the problem of whether we can reach control mode q_F, can be efficiently solved using, for example, the following reachability algorithm, which computes the set of all reachable states from q_0 until we reach the target mode q_F. For a set $R \subseteq Q$, let $Post(R) = \{q_j \in Q \mid \exists q_i \in R \text{ with } q_i \rightarrow q_j\}$ be the set of successors of R in the graph.

Algorithm 1 (Graph Reachability)

Set $R := \{q_0\}$

while *true* do

Sequence Found if $q_F \in R$; STOP

Task Infeasible if $Post(R) \subseteq R$; STOP

Compute Reach Set $R := R \cup Post(R)$

end while

Of course, standard algorithms can determine the shortest path (minimum number of mode switches) between mode q_i and q_j, in the control mode graph. The structure that we have imposed on our control mode graph, immediately result in the following theorem which solves the mode switching problem. The proof is straightforward given the the mode switching condition above.

Theorem 4.4.1 (Mode Switching Solution) *Given a collection of control modes, consider the mode switching Problem 4.2.4. Construct the control mode graph as described above. If there exists a path in the control mode graph, then Problem 4.2.4 is solvable.*

Having determined the sequence of modes that can steer our system from q_0 to q_F, we are left with the problem of determining the parameters r_i for each mode of the sequence. By construction (Definition 4.5), such parameters exist. However, depending on which control modes are connected to, the range of r^i could be different. Thus, there should be a cost function for staying in one mode but associated with a permissible transition. Therefore it is reasonable to pose the problem of choosing r_i within mode i as an optimization of optimal control problem over control mode graph.

A key issue for this approach (as well as for *all* controller synthesis approaches for hybrid systems), is to be able to compute $Pre^i(X_0^j)$ in order to check condition 4.3.2. This is the focus of the next section of this chapter.

4.5 Reachability Computations

There has been a rapidly growing interest in computing reachable sets for various classes of systems [59, 56, 68, 17]. In particular, the approach of [59] has been extended to classes of parametric linear control systems [58], which is highly relevant for computing the operator 4.5. However, these computations are based on quantifier elimination methods whose complexity is prohibitive for large scale systems.

In our case, however, the continuous dynamics are those of output-tracking, closed-loop systems. Therefore part of the state is forced to converge to a trajectory that we get to design, and part of the trajectory is guaranteed to satisfy state constraints. This gives us the opportunity to obtain very reasonable approximations of the reachable sets, and even *design reachable sets* by appropriately designing output trajectories. The following example illustrates the main idea.

Example 4.5.1 Reachability Computation *To illustrate the idea of reachability computation, point mass example is used. The dynamics of a point mass can be described by*

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= u \end{aligned} \tag{4.8}$$

with $x = [x_1 \ x_2]^T$. *Consider there are two control modes defined for the above dynamics, and they are:*

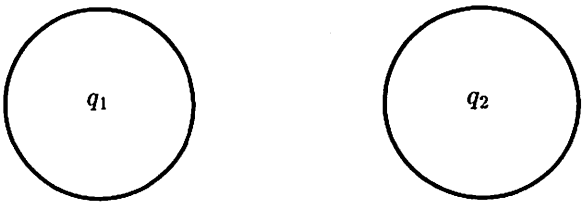


Figure 4.2: Two control modes: position mode and velocity mode

Control Mode Specification

Mode label: q_1
class: *position regulation*
Objective
output: $y^1(t) = h^1(x(t)) = x_1(t)$
trajectory: $r^1 \in R^1$
purpose: $y^1(t) \to r^1$
constraint: $\forall t\ x(t) \in X^1,\ X^1 = \mathbb{R} \times (-V^1_{\max}, V^1_{\max}),\ V^1_{\max} > 0$
Implementation
input: $u = k^1(x(t), r^1) = K^1(x(t) - x^1_d),\ x^1_d = [r^1\ 0]^T$
Performance
flow: $x(t) = \phi^1(t, r^1, x(0))$
stability: *if* $x(0) \in X^1_0\ \exists M_1, \alpha_1 > 0 \Rightarrow$
$\forall t \geq 0\ |x(t) - x^1_d| \leq M_1 e^{-\alpha_1 t}|x(0) - x^1_d|,$
$\lim_{t\to 0} y^1(t) - r^1 = 0$
initial: $X^1_0 = B(x^1_d, V^1_{\max}/M_1)$
range: $R^1 = \mathbb{R}$

Mode label: q_2
class: *velocity regulation*
Objective
output: $y^2(t) = h^2(x(t)) = x_2(t)$
trajectory: $r^2 \in R^2$
purpose: $y^2(t) \to r^2$
constraint: $\forall t\ x(t) \in X^2,\ X^2 = \mathbb{R} \times (-V^2_{\max}, V^2_{\max}),\ V^2_{\max} > 0$
Implementation
input: $u = k^2(x(t), r^2) = K^2_1(x_2(t) - x^2_d),\ x^2_d = r^2$
Performance
flow: $x(t) = \phi^2(t, r^2, x(0))$
stability: *if* $x(0) \in X^2_0\ \exists M_2, \alpha_2 > 0 \Rightarrow$
$\forall t \geq 0\ |x_2(t) - x^2_d| \leq M_2 e^{-\alpha_2 t}|x_2(0) - x^2_d|,$
$\lim_{t\to 0} y^2(t) - r^2 = 0$
initial: $X^2_0 = \mathbb{R} \times (-V^2_{\max}, V^2_{\max})$
range: $R^2 = (-V^2_{\max}, V^2_{\max})$

Consider the case when $V^1_{\max}/M_1 > V^2_{\max}$, *we are going to examine the mode switching conditions between the modes. First, according to Definition 4.5, a transition from mode* q_1 *to* q_2 *is allowed only if*

$$X^1_0 \subseteq Pre^1(X^2_0).$$

To show this is true, we rewrite $Pre^1(X_0^2)$ *as*

$$\begin{aligned} Pre^1(X_0^2) &= \left(\bigcup_{t\geq 0} \bigcup_{r^1 \in R^1} \phi^1(t, r^1, X_0^2)^{-1} \right) \bigcap X^1 \\ &= \bigcup_{r^1 \in R^1} \left(\bigcup_{t\geq 0} \phi^1(t, r^1, X_0^2)^{-1} \bigcap \{r^1\} \times (-V^1_{\max}, V^1_{\max}) \right) \end{aligned}$$

Now, we would like to show that $\bigcup_{t\geq 0} \bigcup_{x\in X_0^2} \phi^1(t, r^1, x)^{-1} \supseteq \{r^1\} \times (-V^1_{\max}, V^1_{\max})$ *since if this is true,* $Pre^1(X_0^2) = X^1$ *and hence we show that* $X_0^1 \subseteq Pre^1(X_0^2)$. *However, even for such a simple dynamical system, the reachability problem is decidable only for certain classes of linear control systems [59]. Here, we demonstrate how to perform the reachability computation for a class of linear control systems as a quantifier elimination problem in the decidable theory of the reals based on o-minimal theories.*

Consider a linear system with diagonalizable matrix A with real rational eigenvalues, it has been shown that in [59] the reachability problem is decidable. Let

$$K^1 = \begin{bmatrix} 0 & 1 \\ -k_1 & -k_2 \end{bmatrix}$$

with $k_1 = 2$ *and* $k_2 = 3$. *Hence,* $M_1 = 2.6180$ *and* 0.3820. *Choose* $V^1_{\max} = 5$ *and* $V^2_{\max} = 1$ *such that* $V^1_{\max}/M_1 > V^2_{\max}$. *With loss of generality, we set* $r^1 = 0$ *to simplify the computation. Since* $X_0^2 = \mathbb{R} \times (-V^2_{\max}, V^2_{\max})$, *we only have to show that there exists* $t \geq 0$ *such that* $\phi^1(t, r^1, x)^{-1} > V^1_{\max}$ *and* $\phi^1(t, r^1, x)^{-1} < -V^1_{\max}$ *with initial condition* X_0^2. *There are several quantifier elimination packages such as* REDLOG, QEPCAD, Mathematica *can be used for performing such symbolic computations. For example, in* Mathematica *version 4.0, via loading the "Experimental" package, one can use the following instruction to perform the computation. Notice that the change of variable* $s = \exp(t)$ *is used in the*

expression, as suggested in [59].

```
Resolve[∃s,s≥1,s∈Reals(∃x1,x1∈Reals(∃x2,x2>-1&&x2<1&&x2∈Reals(2s^2-2s)x1+(2s^2-s)x2>5))]∧
    Resolve[∃s,s≥1,s∈Reals(∃x1,x1∈Reals(∃x2,x2>-1&&x2<1&&x2∈Reals(2s^2-2s)x1+(2s^2-s)x2<-5))]
```

After few seconds for computation, Mathematica then returns True *as the result.*

Now, we would like to check that a transition from mode q_2 to q_1 is allowed. Therefore, we need to show that

$$X_0^2 \subseteq Pre^2(X_0^1)$$

is true. However, we cannot use the same technique as before, since this closed-loop system does not belong to any known decidable class of systems. In this case, we are lucky enough to show the result via the use of simple arguments. First, in mode q_1, we know that $X_1^0 = B(x_d^1, V_{\max}^1/M_1)$ where $x_d^1 = [r^1\ 0]^T$ and $r^1 \in R^1 = \mathbb{R}$. Since the parameters r^1 is not specified, by the definition of the predecessor, one can interpret $Pre^2(X_0^1)$ as $Pre^2(\bigcup_{r^1 \in \mathbb{R}} X_0^1)$. Therefore, we have $Pre^2(X_0^1) \supseteq \bigcup_{r^1 \in \mathbb{R}} X_0^1 = \mathbb{R} \times (-V_{\max}^1/M_1, V_{\max}^1/M_1)$ since the reachable set is computed for $t \geq 0$. Due to the fact that $V_{\max}^1/M_1 > V_{\max}^2$, it can be easily seen that $X_0^2 = X^2 \subseteq Pre^2(X_0^1)$. Hence the result.

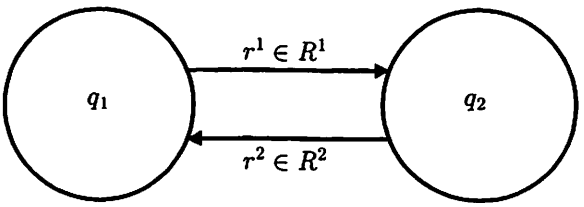


Figure 4.3: Control mode graph of the example

4.6 The Synthesis Procedure for Control Mode Switching

In the following, we provide a procedure which summarizes the previous discussion for solving the mode switching problem posed as Problem 4.2.4. If solvable, a hybrid controller associated with an optimal hybrid automaton is synthesized as the outcome of the algorithm.

The Synthesis Procedure

Step 1: Control Mode Graph Synthesis Given a set of control modes $Q = \{q_1, \ldots, q_N\}$, construct a control mode graph $(Q, \rightarrow)$ according to the mode switching condition defined in Definition 4.3.2.

Step 2: Control Mode Sequence Synthesis Given initial and final control modes with desired set points as (q_0, r_0) and (q_F, r_F) respectively, deploy Algorithm 1 for checking the existence of solution; if Problem 4.2.4 is solvable, search for all permissible solutions, and according to given optimal criteria select a solution which contains both discrete and continuous variables.

Step 3: Hybrid Controller Synthesis Given an optimal solution, construct a hybrid controller by using the discrete sequence in the solution to define the discrete state of a FSM. In a discrete location deploying control mode q_i and a control mode q_j being deployed in the next discrete location, the guard is defined as $X^i \cap X_0^j$, the outputs are both y^i and u which are defined by 4.2, and the inputs are the state variables x.

Step 4: Hybrid Automaton Synthesis Construct a hybrid automaton as defined in Definition 2.2.1 by combining the hybrid controller defined in the previous step. The continuous state of (4.1) defines the continuous state of the automaton. In a discrete location deploying control mode q_i and a control mode q_j being deployed in the next discrete location, outputs are defined by y^i, no input either discrete or continuous is defined for the automaton since close-loop dynamics are considered, guard is defined as $X^i \cap X_0^j$, reset relation is defined simply as an identity map, and invariant set $I(q_i)$ is the maximum invariant set within the constraint set X^i.

4.7 Chapter Summary

The proposed approach has generalized the idea of multi-modal controller presented in [45, 68] which assumes that all the given controllers are totally ordered in terms of performance. Therefore, the switching sequence is implicitly designed and only the switching conditions based on state information are needed to be constructed. However, in our proposed synthesis procedure, no presumption is made on the ordering of performance and a directed transition between any two modes is possible as long as the mode switching condition is satisfied. Hence, our procedure provides a constructive method for the derivation of multi-modal control. Since the hybrid controller is constructed by correctness, formal verification becomes unnecessary for verifying the system behaviors meeting the specifications.

In hierarchical control design perspective, in order to reduce design complexity in design time and computation time in run time, a subset of states which contain important or critical information should be considered and used for high-level planning. To apply the proposed multi-modal control procedure on dynamical system like helicopters as mentioned in previous example, not only a set of controllers which can cover the whole operational envelope for extreme performance is needed but also abstracted models, *i.e.*reduced order models which preserve specific close-loop system properties, are important since this can greatly reduce the complexity on solving the control mode switching problem. In [15, 83], systematic methods are proposed for obtaining an abstracted model which is consistent with the original model on certain properties such as controllability and stability. Therefore, if such an abstracted model can be obtained, one can perform the reachability computations on a reduced dimension state space for solving the mode switching problem while there is a guarantee that the derived solution can be implementable on the original model.

There are several major issues related to the applicability of the procedure. First, if the reachability problem for classes of dynamical systems are decidable and there exist efficient algorithms for all necessary computations, the procedure return exact results. But, if the given dynamical systems are not belong to any decidable classes, a conservative way of solving the problem is to use under-approximation methods for computing reachable set in checking mode switching condition since it provides a guarantee to the existence of a solution. Second, allowing controllers with parameters varying continuous over compact sets could introduce difficulties in applying the proposed procedure to certain classes of

dynamical systems, mainly computation of parametric reachable sets and parameterization of cost functions associated to closed-loop behaviors.

However, to solve this problem, one possible way is to design a reasonable number of controllers with fixed parameters to cover the spectrum of parameter space as much as possible. Then, one can apply the same procedure again since both reachability computation and obtaining cost function for closed-loop systems with fix parameters can be performed with standard methods. Since the search on a graph can be done in polynomial time, therefore, the problem is still tractable. However, the drawbacks are that optimal solutions could not necessary be obtained and extensive amount of computing resource would be used for computation because of excessive number of control modes introduced.

In its full generality, the mode switching problem can be posed as a *Model Matching* problem which is a problem of finding a controller C for a give plant P such that the composition of C and P conform to a given specification M. The plant P is a continuous system but the controller C is a hybrid controller which is a finite automata with a continuous control law at each discrete location. The specification is given in the from $(q_0, r_0) \rightarrow (q_F, r_F)$. However, similar problems have been considered in various communities under labels such as "supervisory control", "scheduler synthesis", "equation solving" and "interacting FSM synthesis". The common notion of conformance for finite automata is language containment. However, since we have a hybrid specification, a new framework is needed in order to address the needs and it will be fleshed out in future research.

Chapter 5

Embedded Controller Synthesis

This chapter presents a formal methodology for the design, implementation and validation of reactive systems. The methodology has been applied to the design of a Flight Management Systems (FMS) for a model helicopter in the BEAR project[53]. POLIS[6], a compute-aided-design (CAD) tool developed at the University of California at Berkeley, is extensively used. The automation of the design problem and the validation techniques provided by this tool allow to shorten prototyping time and to prove the correctness of the properties of the system. Automatic code generation guarantees error free implementation, which is fundamental in safety critical applications. Simulation of the entire design is performed using Ptolemy, a hierarchical heterogeneous simulation environment.

5.1 Introduction

Reactive systems [88] react continuously to their environment at the speed of environment. Reactive systems are prominent in industrial process control, airplane or automobile control, embedded systems, man-machine interfaces, etc. They can be contrasted with interactive systems, which react with the environment at their own speed. This class covers operating systems, data bases, networking, distributed algorithms, etc. Interactive and reactive systems deeply differ on the key issue of behavioral determinism. Interactive sys-

tems are naturally viewed as being non-deterministic, while behavioral determinism is a highly desirable and often mandatory of reactive systems. A real-time system is defined as a reactive system that is subject to externally defined timing constraints.

Flight Management Systems (FMS) were first proposed for smart aircrafts in future Air Traffic Management Systems (ATMS) [86, 97] for decentralized air traffic control. FMS are responsible for

Navigation planning of the route for task execution, calculating a feasible course, and regulating an UAV along the course;

Emergency handling switching among different modes of operation to handle contingent events such as conflict resolution among UAVs, obstacle avoidance, and envelope protection.

FMS for UAV[46] are inherently reactive, since they react to the environment by changing mode of operation, as described above.

An FMS operates in a mission critical environment, where reliability and safety are more important criteria than performance. Formal verification and automatic synthesis of implementations are the surest ways to guarantee safety. Managing the design complexity and heterogeneity is the key problem. The design approach should be based on the use of one or more formal models of computation (MOC) [62] to describe the behavior of the system at a high level of abstraction. The implementation of the system should be made using automatic synthesis as much as possible from this high level of abstraction, to ensure implementation that are "correct by construction". Validation should be done at the highest possible levels of abstraction.

The POLIS [6] system is intended for control-dominated systems whose implementation is based on micro-controller for tasks to be implemented in software and Application Specific Integrated Circuits (ASICs) for tasks to be implemented in hardware. The input to POLIS is a combination of graphics and text describing the behavior of each single finite state machine in the formal language ESTEREL. The analysis at the behavioral level can be carried out with formal tools. Performance evaluation can be carried out by simulating the behavior of the architecture selected with an abstract timing model of processor in the heterogeneous simulation environment offered by PTOLEMY[14]. In our design example we

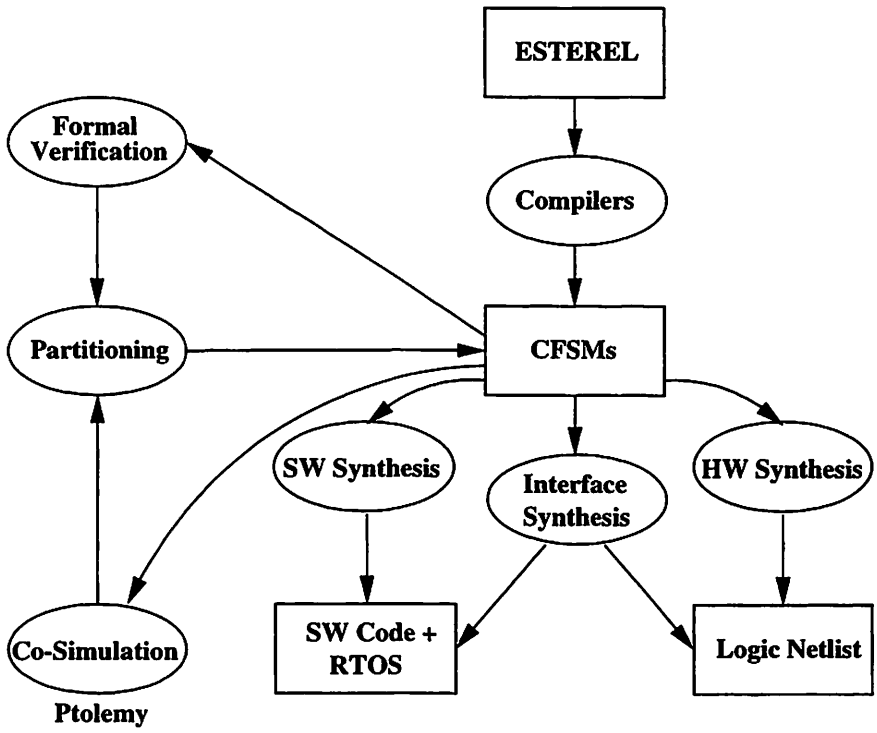


Figure 5.1: Polis design flow chart

carried several simulations with different types of Hardware-Software (HW-SW) partition and choices of microprocessors in order to validate the design.

Section 2 presents formal synthesis of reactive system design. In Section 3, the example design and its representation in the POLIS/Ptolemy domain is discussed in details. Concluding remarks are offered in Section 4.

5.2 Polis Co-design Methodology

In the formal approach to the design of a control system, the choice of the Model of Computation (MOC) to represent the behavior of the system is often crucial. In order to do it we need to know what are the main characteristics of the system that we are going to implement and then find the MOC that can capture most of the properties [62]. In the case of a reactive system, we need a specification language that includes constructs for hierarchy, pre-emption (input events arrive at irregular and unpredictable times), concurrency

and sequencing. A widely used MOC for description of such system is the Finite State Machine(FSM).

Definition 5.2.1 *A Finite State Machine (FSM) is a quintuple* $\mathcal{F} = (I, O, X, R, F)$ *where* I *is a finite set of input symbols,* O *is a finite set of output symbols,* X *is a finite set of states,* $R \subseteq X$ *is the set of initial states,* $F \subseteq I \times X \times X \times O$ *is the transition relation.*

A FSM contains all the desired properties and also, under mild conditions, it is deterministic and completely specified. The drawback of such a representation is the difficulty posed on the possibility of data computation. The FSM models purely reactive systems, while in almost all control application data computation is also needed.

A more suitable model is the **Extended Finite State Machine** (EFSM), which is a FSM where the transition relation may depend on a set of internal variables. It operates on a set of finite-valued variables by using arithmetic, relational, boolean operators and user defined functions. The EFSM model has a fundamental limitation: communication between EFSMs is totally synchronous, therefore it is not implementable on a distributed environment where a combination of hardware and software modules is used. Hardware and software implementations are characterized by different behavior in execution and communication. The desired MOC should then reflect this situation. The necessity of extending the FSM semantics to include an asynchronous communication mechanism brings us the choice of the so called **Co-Design Finite State Machine** (CFSM). A CFSM can be defined as a FSM which has also a data computation part. CFSM exploits a locally synchronous behavior. It produces an output in reaction to an input assignment in zero time. Globally the CFSM has an asynchronous behavior; each CFSM reads inputs, and produces outputs in an unbounded but finite amount of time. Several specification languages can be used to model CFSMs. POLIS uses synchronous language to model each individual CFSM. The synchronous approach is very attractive for several reasons. Computation and internal communication take no time. The behavior is totally predictable. The problem of synchronization doesn't exist. Determinacy allows formal verification and the synchronous approach allows translation in EFSM in a fully abstract way, so that behaviorally equivalent specifications are mapped into syntactically equivalent EFSMs. Communication between

CFSMs happens by means of events which are control signals that may or may not contain also data information.

The high level specification language used by POLIS is **Esterel**[9]. This language is very simple and contains the necessary constructs for the description of our system. Hierarchy is handled via procedure calls, preemption consists of two basic constructs, one which allows the module to terminate its computation for the current instant and one which does not, concurrency is specified by using a parallel composition construct. Data manipulation cannot be done naturally in Esterel. The user needs to define functions outside the environment and then link them in the program. Also the synchronous hypothesis makes difficult to model the communication among subsystems that operate at different rates. The timing constraints need to be specified outside Esterel. This will be assessed during the HW-SW synthesis phase. Starting from the behavioral specification, CFSMs are generated using the Software Hardware Intermediate Formal (SHIFT) language. The next step is to connect the various CFSMs so generated. This can be done in the simulation environment Ptolemy, which we will discuss later. At this stage formal verification can take place. This technique is very powerful but at the same time computationally expensive. Performing formal verification at the behavioral level allows detection of errors at an early stage and keeps the complexity of the formal verification scheme low.

The next step involves the selection of an implementation architecture. The advantage of using formal methods consists in the use of automatic synthesis techniques. There are three fundamental decisions to be taken: *Partitioning, Architecture Selection and Scheduling.* These three steps are based on experience and therefore the designer is allowed to choose. POLIS provides libraries for several types of micro-controllers for software implementation and ASICS for hardware synthesis. POLIS provides a choice of schedulers, which regulate the communication among CFSMs. After the selection of the architecture is complete the system is simulated within Ptolemy for performance validation.

Co-simulation is used in POLIS both for functional debugging and for performance analysis during the architecture selection process. Hardware-Software (**HW-SW**) co-simulation is generally performed with separate simulation models. POLIS allows for HW-SW co-simulation within the same environment. The basic concept is to use synthesized C code to model all the components of a system, regardless of their future implementation. For the software partition the simulation code is the same that will run on the target processor.

Depending on the selected architecture, each task will take a specified number of clock to be executed (one clock cycle for hardware, a number of cycles for software depending on the selected target processor for software) and will result in different execution constraints (concurrency for hardware, mutual exclusion for software). The Ptolemy system provides the simulation engine and the graphical interface. Among the various computation models offered by Ptolemy, the *discrete event* (DE) model has been used, since it matches the CFSM execution semantics. Co-simulation provides a truthful estimate of the performance of the system, i.e. of the capacity of the software architecture to meet the timing constraint imposed by the discretized dynamic system. In case the designer doesn't find the result satisfactory, the aforementioned design process needs to be iterated. If, on the other hand, the simulation results meet the specifications, synthesis can take place. POLIS provides automatic code generation which is specific to the selected micro-processor.

5.3 System Design and Synthesis

This section presents an application of this design methodology to the modeling and synthesis of a FMS for a single UAV performing a particular task. However, the same methodology can be applied for multiple UAVs with different mission specifications. The FMS can be modeled by a hierarchical finite state machine. Consider a search and investigate scenario, UAV is requested to search for objects of interest in a confined area by navigating via a series of way points and perform an investigation when an object is found. After investigation, UAV should resume its nominal route. The system to be modeled can be decomposed into three parts: the Flight Management System which is responsible for planning and controlling the operation of the UAV, a Detector for the detection and investigation of objects of interest and an aerial vehicle (the plant to be controlled). Figure 5.2 shows the given mission. In order to illustrate the reactiveness of the system we put an object that the UAV will be able to sense along path.

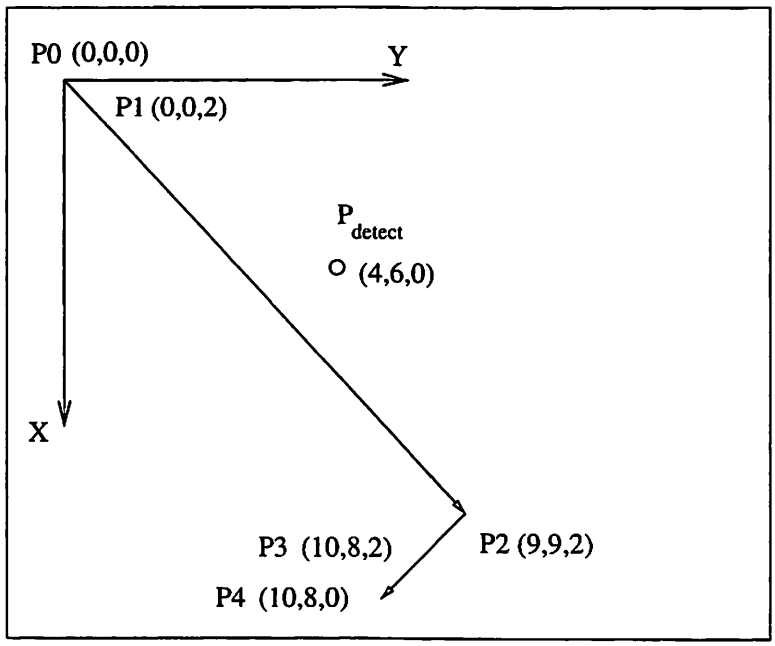


Figure 5.2: Search and investigate scenario: to navigate through the way points and detect objects of interest along the path

Aerial vehicle dynamics

An aerial vehicle can be represented by a rigid body moving in a 3-dimensional space in response to gravity, aerodynamics, and propulsion. Due to the characteristics of the aerial vehicle designed, the force and moment are generated by different actuators which corresponding to different control inputs. For aircraft, the control inputs are generated through engine, aileron, elevator and rudder which produce thrust and 3-axes moments. For an helicopter, the control inputs are generated through engine, main rotor collective pitch, tail rotor collective pitch, longitudinal cyclic pitch, lateral cyclic pitch, which produce thrust and 3-axes moments.

In the search and investigate scenario, a helicopter based UAV is chosen to execute the mission mainly because of its vertical take-off and landing (VTOL) and hovering capabilities. As shown in [49, 52], a helicopter model has been shown to be nonlinear and non-minimum phase. Although the idea of using approximate input-output linearization on helicopter control has been successfully applied, it is not desirable to use the nonlinear controller because the dimension of the system and the order of desired outputs trajectory are rather

high. However, it has be shown that the dynamics can be partitioned into an "inner system" (*e.g.*the attitude dynamics) and an "outer system" (*e.g.*the position dynamics) where the outer system is differentially flat. In Chapter 3, a nonlinear controller based on outer flatness is designed. The beauty of this control design is that the planning can be done on a reduced order system, the outer system, by generating desired accelerations according to a given output trajectory while the controller is guaranteed to have the system output tracking the given output trajectory. The outer system can be regarded as the abstraction of the whole system, and a formal treatment of the problem is shown in [83].

Therefore, we have a double integrator to represent the dynamics each output variable of the outer system. Although the output, heading, can track a desired heading trajectory and is not controlled in the outer system. To consider the dynamic behavior, we also use a double integrator to represent the dynamics. Hence, the **Dynamics** can be greatly simplified and modeled as:

$$\begin{aligned} \ddot{x} &= u_x \\ \ddot{y} &= u_y \\ \ddot{z} &= u_z \\ \ddot{\psi} &= u_\psi. \end{aligned}$$

Detector

The **Detector** can be considered as a computer vision system with limited range of detection capability The computing time for vision algorithm together with the view angle set an upper bound on the maximum possible cruise velocity. The detector communicates with the FMS. When detection takes place, a preemptive signal is sent to the FMS to request the FMS to perform investigation by navigating through a series of intermediate way points as $\{P_1', P_2'\}$ where P_1' is a point above the object and P_2' is has the same coordinates as the previous point but with lower altitude. Once the investigation is accomplished, the detector will assert a signal to the FMS to indicate the end of investigation phase.

Flight Management System

The FMS consists of four layers, the strategic, tactical, and trajectory planners, and the regulation layer, as described in Figure 5.3. Hierarchy is handled very naturally in Esterel, as described in the previous section.

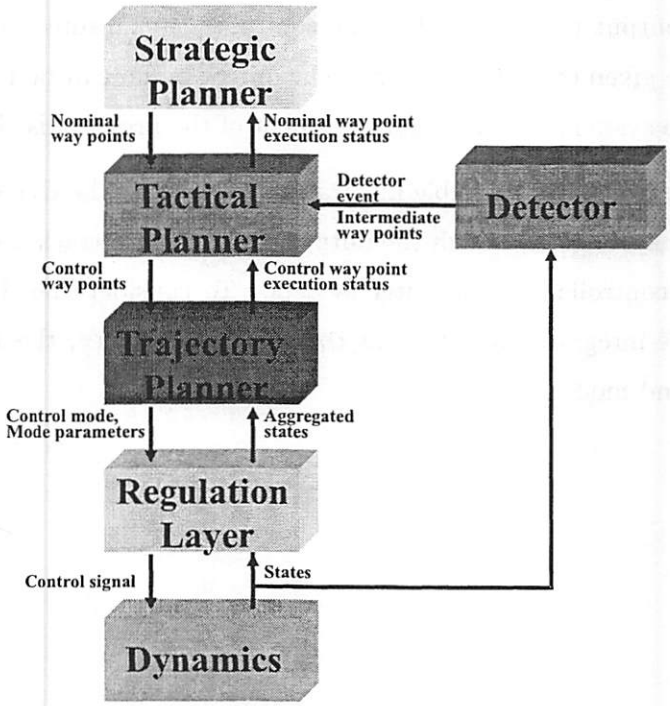


Figure 5.3: System Architecture

The **Strategic Planner** is concerned with the planning and execution of the central UAV mission. It designs a coarse, self-optimal route, which is stored in form of a sequence way-points. This layer also takes care of the transition between the points, by acknowledging the completion of a subtask and scheduling the next one. In our example the strategic planner contains a set of way points $P = \{P_0, P_1, P_2, P_3, P_4\}$, where $P_i \in R^3$ for $i = 0, \ldots, 4$.

The **Tactical Planner** is responsible for handling contingent events among UAVs and reacts to the detector by generating corresponding way points to trajectory planner. It is capable of overruling the route proposed by the strategic planner, in case of safety critical situations such as collision avoidance. Given the scenario, when an object of interest is detected, the tactical planner react to the event generated by the detector and force a

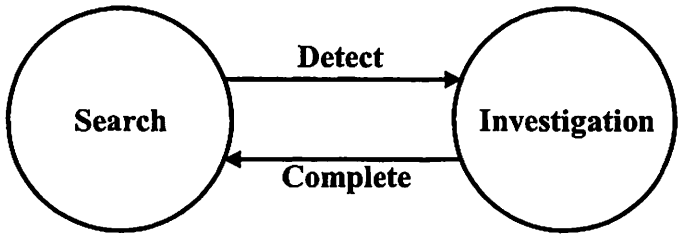


Figure 5.4: FSM representation of Tactical Planner

discrete transition to have it operated in the investigation mode as shown in Figure 5.4.

In investigation mode, given the way points $\{P_1', P_2'\}$ from the detector, tactical planner introduces a sequence of way points that will be tracked in order to complete the new task. At the time instant when the object is detected, the current position is saved as P_0' and hence a subsequence is inserted as $\{P_0', P_1', P_2', P_1', P_0'\}$. Once the investigation has taken place by passing through all the way points specified in the subsequence, the search resumes from P_0'.

Given the current and future way points, the **Trajectory Planner** generates a course as a sequence of control mode to accomplish the reachability specifications. As defined in Chapter 4, a *control mode* is defined as the operation of a given continuous control system under a controller that is *guaranteed* to track a certain class of output trajectories. Different outputs of interest correspond to different control modes. Since we are considered a reduced dimension system defined in **Dynamics**, we can consider there are two controllers, position controller and velocity controller as shown in Chapter 4, being designed for each direction of motion. Hence, there are two modes for each direction of motion, and we have totally eight control modes, namely:

In x-direction : $(q_1^x, r_1^x), (q_2^x, r_2^x)$

In y-direction : $(q_1^y, r_1^y), (q_2^y, r_2^y)$

In z-direction : $(q_1^z, r_1^z), (q_2^z, r_2^z)$

In ψ-direction : $(q_1^\psi, r_1^\psi), (q_2^\psi, r_2^\psi)$

It has been shown that it is feasible that in each direction of motion two control modes can be switched to and fro and form a control mode graph. Since the dynamics considered in

Dynamics are decoupled, there are 4 disjoint control mode graph and hence 2^4 possible combinations of control modes.

In order to simplify control mode sequence generation, it is more desirable to the reduce number of control modes. In helicopter flight instruction, the transition of maneuver is restricted in proper sequence which is shown in Figure 5.5.

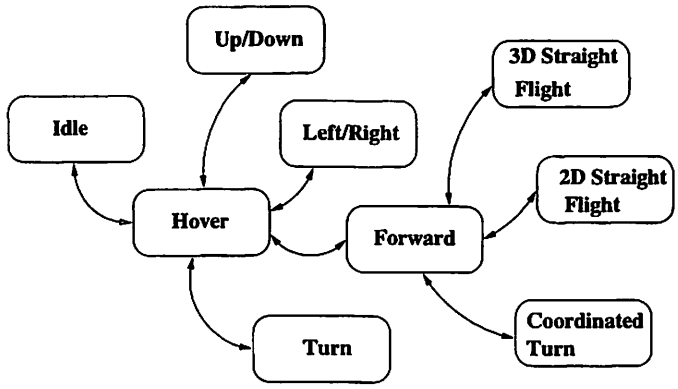


Figure 5.5: Transitions diagram of helicopter maneuvers

Motivated by the flight instruction, a subset of the maneuvers are selected and each maneuver can be translated into a set of control modes possibly with constraints on the parameters. The maneuvers composed of control modes are *Hover*, *Turn*, *Up/Down*, *Forward* and *Left/Right*:

$$\textbf{Hover: } \left\{ \ (\ (q_1^x, r_1^x), (q_1^y, r_1^y), (q_1^z, r_1^z), (q_1^\psi, r_1^\psi) \ ) \ \right\}$$

$$\textbf{Turn: } \left\{ \ (\ (q_1^x, r_1^x), (q_1^y, r_1^y), (q_1^z, r_1^z), (q_2^\psi, r_2^\psi) \ ) \ \right\}$$

$$\textbf{Up/Down: } \left\{ \ (\ (q_1^x, r_1^x), (q_1^y, r_1^y), (q_2^z, r_2^z), (q_1^\psi, r_1^\psi) \ ) \ \right\}$$

$$\textbf{Forward: } \left\{ \begin{array}{l} (\ (q_2^x, r_2^x), (q_1^y, r_1^y), (q_1^z, r_1^z), (q_1^\psi, r_1^\psi) \ ) \text{ with } r_1^\psi = \text{atan2}(0, r_2^x), \\ (\ (q_1^x, r_1^x), (q_2^y, r_2^y), (q_1^z, r_1^z), (q_1^\psi, r_1^\psi) \ ) \text{ with } r_1^\psi = \text{atan2}(r_2^y, 0), \\ (\ (q_2^x, r_2^x), (q_2^y, r_2^y), (q_1^z, r_1^z), (q_1^\psi, r_1^\psi) \ ) \text{ with } r_1^\psi = \text{atan2}(r_2^y, r_2^x) \end{array} \right\}$$

$$\textbf{Left/Right: } \left\{ \begin{array}{l} (\ (q_2^x, r_2^x), (q_1^y, r_1^y), (q_1^z, r_1^z), (q_1^\psi, r_1^\psi) \ ) \text{ with } r_1^\psi = \text{atan2}(0, r_2^x) \pm \frac{\pi}{2}, \\ (\ (q_1^x, r_1^x), (q_2^y, r_2^y), (q_1^z, r_1^z), (q_1^\psi, r_1^\psi) \ ) \text{ with } r_1^\psi = \text{atan2}(r_2^y, 0) \pm \frac{\pi}{2}, \\ (\ (q_2^x, r_2^x), (q_2^y, r_2^y), (q_1^z, r_1^z), (q_1^\psi, r_1^\psi) \ ) \text{ with } r_1^\psi = \text{atan2}(r_2^y, r_2^x) \pm \frac{\pi}{2} \end{array} \right\}$$

In *Hover*, position modes are used in all four directions. Velocity mode is used in ψ-direction, but position modes are applied to other directions in *Turn*. Similarly, in *Up/Down*, only velocity mode is used in z-direction but others are operated in position mode. In *Forward*, position mode are used in z-direction and ψ-direction but neither in x-direction nor in y-direction are operated in position mode. Furthermore, the desired motion direction in x, y-plane has to be aligned with the desired heading. In *Left/Right*, control modes are defined as in *Forward* but with an offset $\pm\frac{\pi}{2}$ between the desired motion direction in x, y-plane and the desired heading where the sign of the offset depends on the direction of motion. Hence, by combining the decoupled control graphs, the corresponding control mode graph is obtained as shown in Figure 5.6.

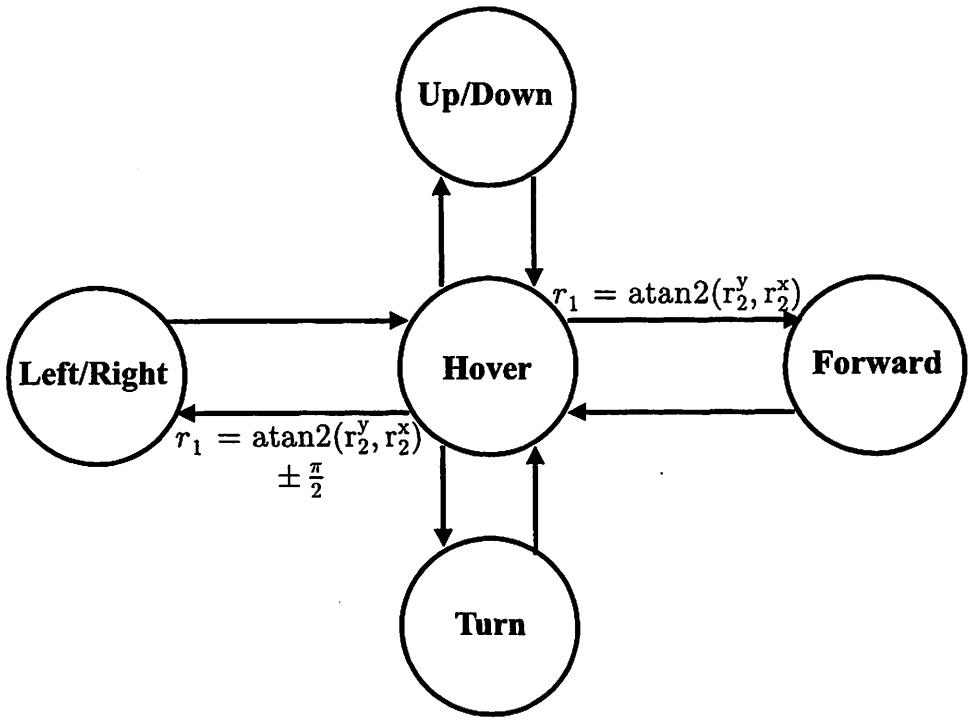


Figure 5.6: Control mode graph of helicopter maneuvers.

Given (p_x^i, p_y^i, p_z^i) as the i^{th} point, and (p_x^j, p_y^j, p_z^j) as the j^{th} point, we would like to find a sequence of maneuvers for reaching the j^{th} point from the i^{th} point. With loss of generality, we assume that the helicopter is operated in *Hover*, *i.e.*($(q_1^x, p_x^i), (q_1^y, p_y^i), (q_1^z, p_z^i), (q_1^\psi, \psi^i)$) where ψ^i defines desired heading at the i^{th} point. Here, we present a solution of the

reachability problem by having

$$
\begin{array}{ll}
 & \textit{Hover} \\
 & \text{at } ((q_1^x, p_x^i), (q_1^y, p_y^i), (q_1^z, p_z^i), (q_1^\psi, \psi^i)) \\
\rightarrow & \textit{Hover} \\
 & \text{at } ((q_1^x, p_x^i), (q_1^y, p_y^i), (q_1^z, p_z^j), (q_1^\psi, \psi^i)) \\
\rightarrow & \textit{Hover} \\
 & \text{at } ((q_1^x, p_x^i), (q_1^y, p_y^i), (q_1^z, p_z^i), (q_1^\psi, \psi^{ij})) \\
\rightarrow & \textit{Hover} \\
 & \text{at } ((q_1^x, p_x^j), (q_1^y, p_y^j), (q_1^z, p_z^j), (q_1^\psi, \psi^{ij}))
\end{array}
$$

$$
\begin{array}{ll}
\rightarrow & \textit{Up/Down} \\
 & \text{at } ((q_1^x, p_x^i), (q_1^y, p_y^i), (q_2^z, r_2^z), (q_1^\psi \psi^i)) \\
\rightarrow & \textit{Turn} \\
 & \text{at } ((q_1^x, p_x^i), (q_1^y, p_y^i), (q_1^z, p_z^j), (q_2^\psi, r_2^\psi)) \\
\rightarrow & \textit{Forward} \\
 & \text{at } \left\{ \begin{array}{l} ((q_2^x, r_2^x), (q_1^y, p_y^j), (q_1^z, p_z^j), (q_1^\psi, \psi^{ij})) \\ \quad \text{if } p_y^i = p_y^j \\ ((q_1^x, p_x^j), (q_2^y, r_2^y), (q_1^z, p_z^j), (q_1^\psi, \psi^{ij})) \\ \quad \text{if } p_x^i = p_x^j \\ ((q_2^x, r_2^x), (q_2^y, r_2^y), (q_1^z, p_z^j), (q_1^\psi, \psi^{ij})) \\ \quad \text{otherwise} \end{array} \right\}
\end{array}
$$

where $\psi^{ij} = \text{atan2}(p_y^j - p_y^i, p_x^j - p_x^i)$ and the intermediate parameters r_2^z, r_2^ψ, r_2^x and r_2^y can be chosen within the constraint sets to optimize a given cost function such as traveling time and energy consumption. This solution , as shown in Figure 5.7, completely decouples the maneuvers and performs them independently. Tactical Planner sends the acknowledgment signal to the Strategic Planner which sends the next way point when the current way point is reached.

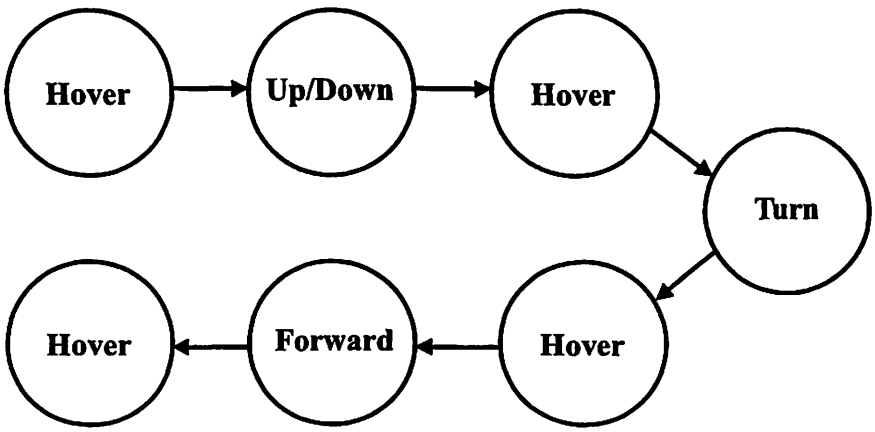


Figure 5.7: FSM representation of Trajectory Planner.

The **Regulation Layer**, together with the plant, represents the continuous part of the system. The regulation layer has access to sensory information about the actual state of the

UAV to generate control signal to regulate the vehicle according to the control modes and parameters given by trajectory planner. For each subsystem, there are two control modes, namely position mode and velocity mode. Therefore, there are 8 controllers in total and at any time there are four of them being used.

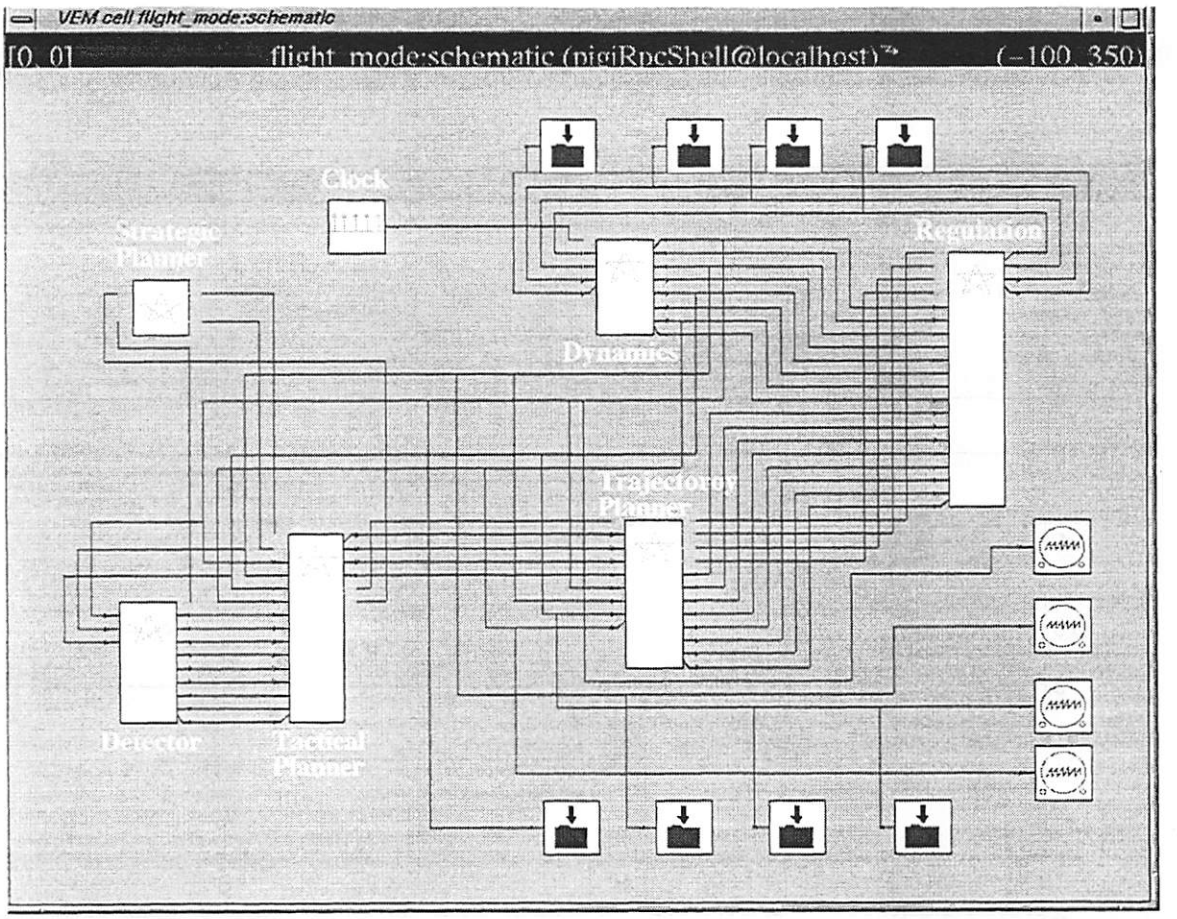


Figure 5.8: System block diagram in Ptolemy

5.4 Implementation and Validation

The system is divided into blocks and each block is modeled as a FSM by specifying the desired behavior. The individual modules are synthesized by POLIS and then composed in Ptolemy to yield to complete system. The system architecture in the Ptolemy domain is

shown in Figure 5.8. The stars represent the functional blocks of different modules. In this section we give a description of the role of each of the functional blocks.

To validate the design, we carried out several simulations under different assumptions. The behavior simulated was a search-and-investigate mission. The UAV should deviate from the nominal path if the detector finds an object of interest. The initial states are set to zero, i.e. idle position. The mission terminates when the UAV lands at the point p_4 (10,8,0).

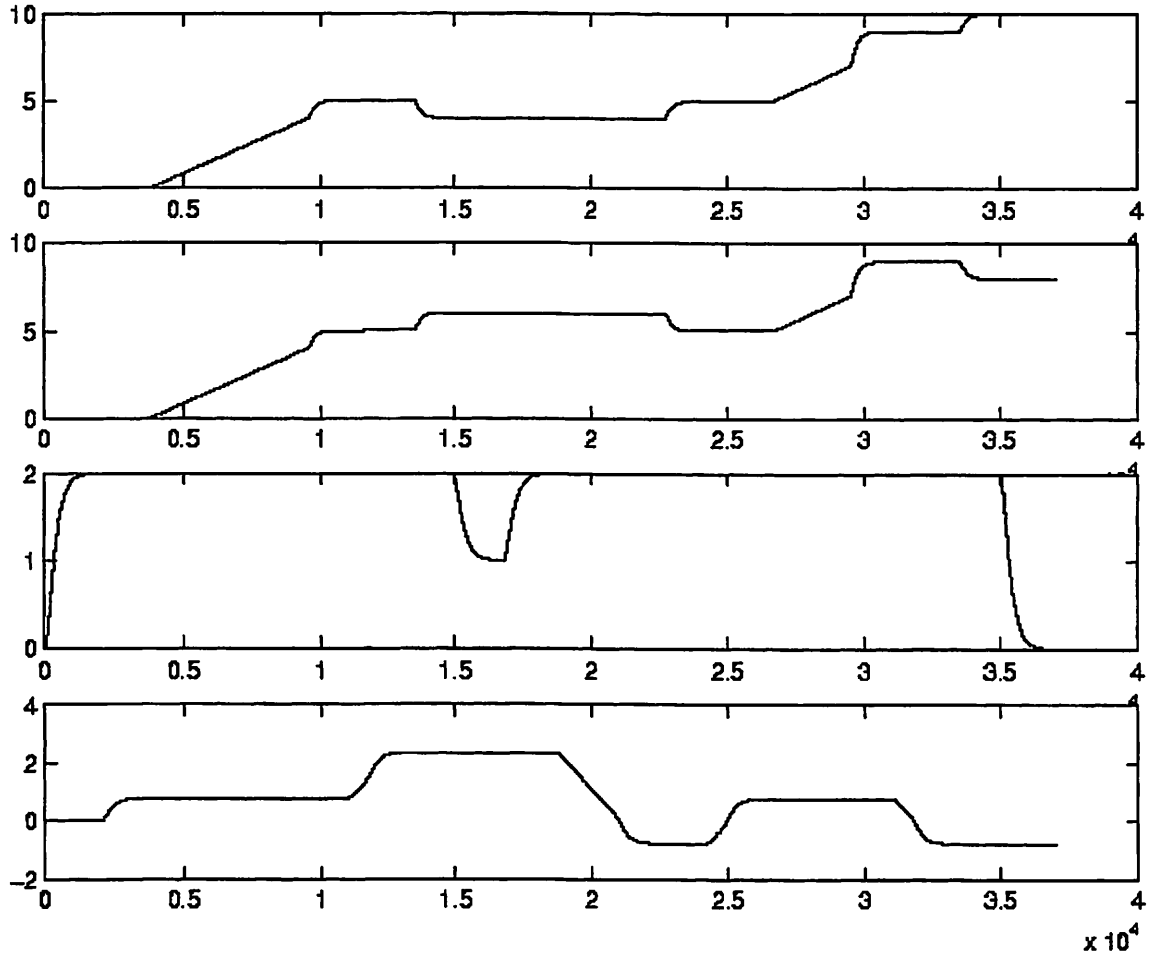


Figure 5.9: Simulation result using synchronous model: x, y, z, ψ from top to bottom

First, as depicted in Figure 5.9, we simulated the behavior of the system under the synchronous hypothesis, *i.e.*in the absence of delay due to communication and computation

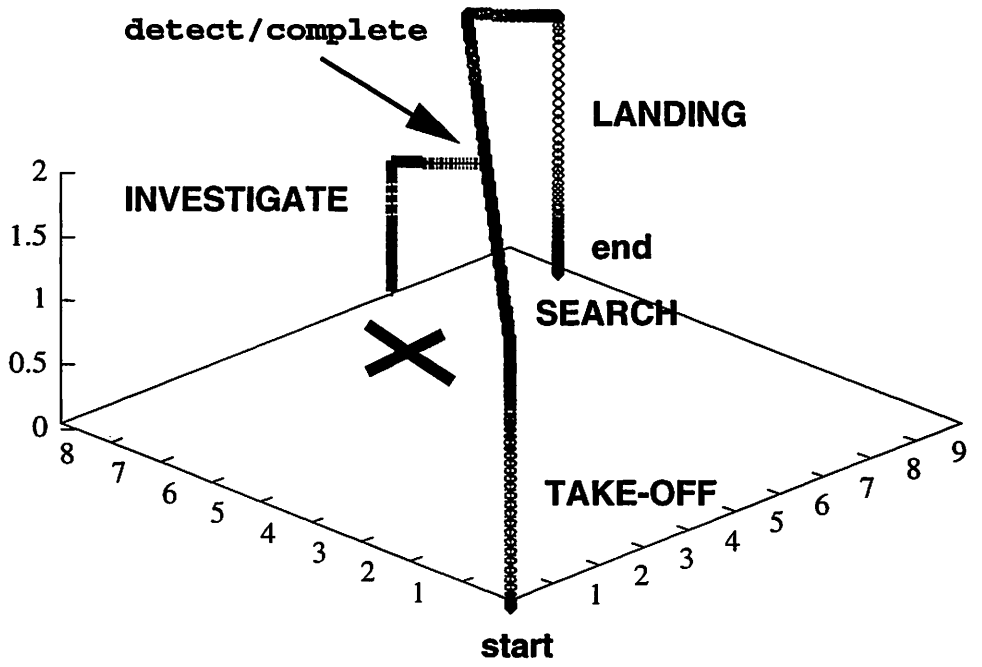


Figure 5.10: Simulated 3D Trajectory

time. This corresponds to representing the system entirely in Esterel. This assumption is often used to simulate the *behavior* of the system but yields poor implementations since it requires implementing the system either in synchronous hardware or as a single task in software. This results in overheads in space for hardware and in memory and sometimes in running time for software.

After the behavior has been simulated and verified, a specific implementation has to be selected. Our first architectural choice was to implement the algorithm entirely in software on a RISC MIPS R3000 micro-controller with a simple round-robin scheme to schedule the tasks. The synchronous assumption was maintained for the plant and the detector, thus implying that their operation was considered as a single task for the operating system. The simulation results in Figure 5.11 show that the timing constraint is met by the chosen architecture. Simulation of 8 minutes of actual running time of the system was obtained with 5 minutes of simulation time. At first sight, this result seems surprising since simulation required less time than the actual time that would have been required by the embedded system. The explanation of this result is simple: the software is run on a workstation whose processor is more powerful than the micro-controller core.

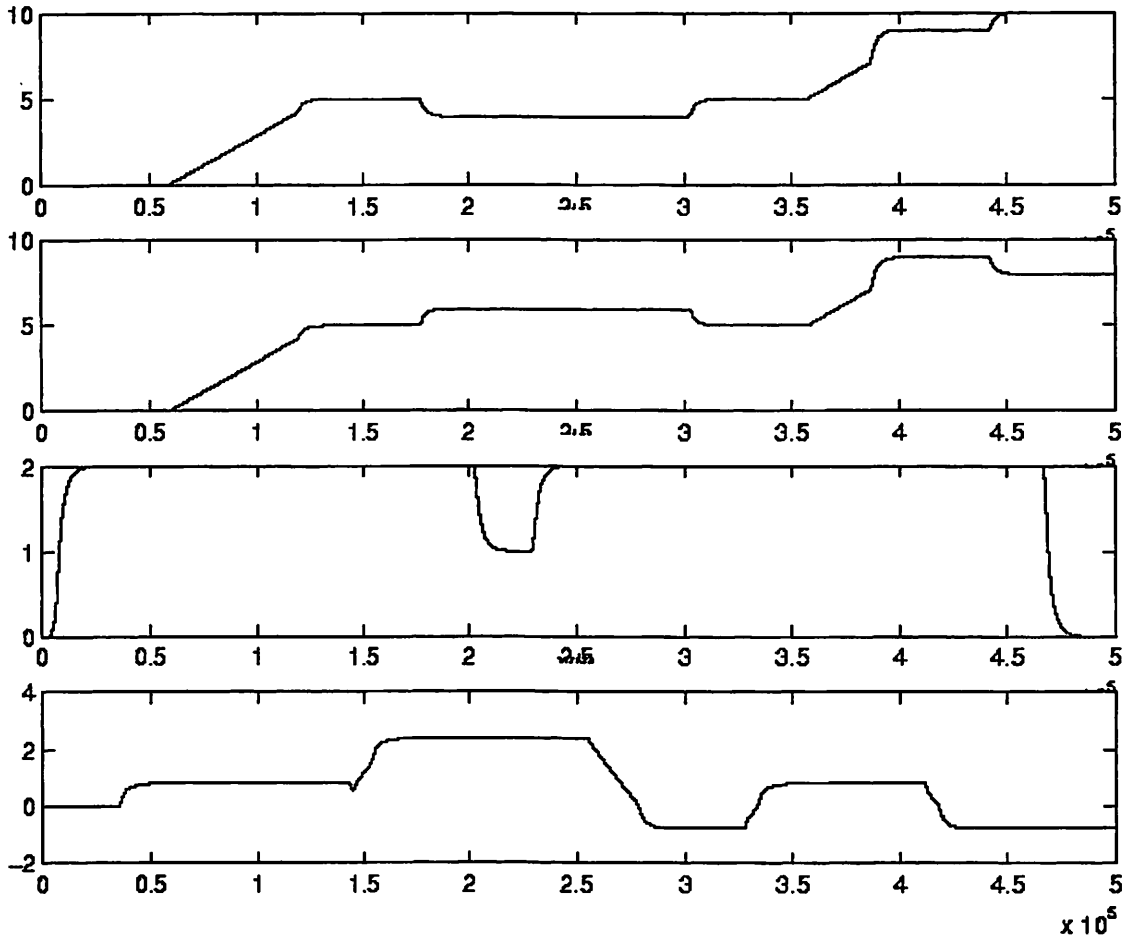


Figure 5.11: Simulation Result after HW-SW partition: x, y, z, ψ from top to bottom

The timing results needed to verify the correctness of the implementation with respect to the real time constraints are obtained using the performance model used for the controller. In particular, the number of cycles required by each instruction in the instruction set of the processor have to be provided. Because of the complex architecture of modern microprocessors, the POLIS group has developed a method to obtain the parameters of the performance model by running on an evaluation board a set of tests.

The actual running time estimates provided by the simulation are not 100% accurate but it has been possible to demonstrate that the accuracy of the estimation is acceptable (the actual running time on the implemented system was within 20% of the estimation).

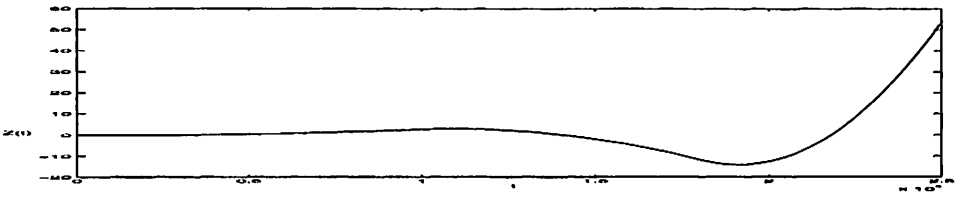


Figure 5.12: Instability occurs in Z dynamics

Because of the speed of the simulation with the performance estimation models (which include a model for the operating system and the scheduler as well), several different implementation choices can be tried. In particular, different micro-controllers and different scheduling algorithms can be evaluated. Once several architectures have been validated, the designer can choose the most suitable architecture, depending on cost efficiency trade off consideration.

To exemplify this point, we implemented the system using a Motorola 68hc11, 8 bit micro-controller and a round robin scheduler. The implementation yields an unstable system, as shown in figure 5.12. Analysis on the dynamical system shows that bandwidth of the system is too high to achieve acceptable performance with a slow processor. In particular, part of the design has to be implemented in hardware to meet the design specification with such architecture. Further simulations show that the design specification is met if the regulation layer is implemented with an ASIC. The switching condition between flight modes needs to be relaxed, since the state information is not available to the tactical planner at all times.

5.5 Chapter Summary

A formal approach to reactive system synthesis using POLIS has been presented. A design example of FMS of UAV has been exploited. For such a hierarchical system with complex behavior, the behavior is specified in Esterel and compiled into a network of CFSMs chosen as the MOC for describing a locally synchronous and globally asynchronous system. Formal verification can be carried out with the CFSM model. After verification of the desired design properties, partition of hardware and software can take place under the

choice provided by the designer. Automatic synthesis of hardware, software and interface, including the Real Time Operating System (RTOS), is then performed. In this experiment, we found that a great value of the POLIS system is in the quick system-level analysis that allows an architectural optimization that would not be possible otherwise especially for complex systems such as the ones analyzed in this chapter.

The design is validated through simulation in Ptolemy environment. However, it is clear that final validation has to be carried out in a prototype implementation of the entire system since the high-level simulation carried out in the Ptolemy environment with POLIS models and software synthesis methods is based on approximate performance models. It is our intention to build such a system with the micro-processor selected in the evaluation phase with the scheduling algorithm tested in the simulation.

Chapter 6

Conclusions

Based on operational, financial and environmental considerations, large scale systems ranging from automated highway systems, air traffic management systems, unmanned aerial vehicle networks, communication networks and power distribution networks have been advocated to have higher levels of automation. The control of large scale systems is extremely challenging since by nature the systems are distributed and highly dynamic, the environments in which the systems reside in are usually rapidly evolving, and multi-objective design specifications intensify the complexity of system design. One natural way to manage the complexity of system design is by compositional methods. Of particular interest is multi-modal control paradigm in which control systems are designed by hierarchically nesting of composition of modes of operation such that each mode of operation is designed to cope with a designated scenario with respect to a design specification while the organization of modes of operation depends on the ordering of these specifications.

A multi-modal control system can be modeled as a hierarchical nesting of parallel and serial composition of discrete and continuous components. Furthermore, a model of computation (MOC) governs the behaviors and interactions of components at each level of the hierarchy. Hybrid systems are considered as formalisms used to describe a complex system as combinations of MOCs. This naturally leads to the generalization of the design problem for the control of large scale systems as a problem of multi-modal control synthesis/design

in the modeling framework of hybrid systems.

Recent advances of embedded system technologies in sensing, computation and communication have enabled the rapid realization of sophisticated, high-performance embedded controllers. Hybridness is a characteristic of embedded control systems because every digital hardware/software implementation of a control design is ultimately a discrete approximation interacting through sensors and actuators with a continuous physical environment. The model of computation defines the behavior and interaction of these components. In the modeling framework of hybrid systems, the formal synthesis of embedded controllers can be, in general, interpreted as generation of an architecture mapping from discrete/continuous components to hardware/software components to ensure the implementations that are correct by construction.

This dissertation is primarily focused on formal approach of hybrid system design and embedded controller synthesis for multi-modal control. Special emphasis will be put on scalability and interoperability to cover the complete design flow from architecture conception, control law derivation, and mode switching synthesis all the way to the generation of real-time embedded controller code. Throughout, a helicopter based unmanned aerial vehicle (UAV) is used as a design example to demonstrate the effectiveness of the proposed design paradigm for solving multi-modal control problem.

First, modal controllers for a nonlinear non-minimum phase helicopter model are designed. The vehicle model includes rigid body dynamics and a force & moment generation process. Nonlinear control designs based on feedback linearization and differential flatness are illustrated. Performance of different control designs are provided.

Second, a general framework is developed for the derivation of control modes switching which satisfy reachability specifications. Control mode switching sequences and switching conditions are derived based on existing efficient methods for reachability computation developed for hybrid systems. Furthermore, the controlled system can be modeled as a hybrid automaton: a finite state machine with closed loop dynamics embedded in each discrete locations.

Third, embedded system synthesis based on formal methodology is presented. The system behavior at the top level is specified in synchronous language for describing reactive system behaviors then translated to various MOCs in different design stages depending of

the design properties of MOCs. Depending on the choice of architecture, the components specified by the MOC are then mapped to hardware and software components. The design framework allows for shorter prototyping time and to proof of correctness of the system properties. An implementation of the complete system architecture is synthesized and the performance is validated via simulation.

There are many areas for future research and development in the areas of nonlinear control design, control mode switching synthesis, and embedded controller synthesis. Some directions for future include the following.

Nonlinear Control Design. The controllers presented in Chapter 3 address the issue in the control of nonlinear non-minimum phase system via approximation, but there are more areas for further study. New control designs need to be developed to handle modeling errors introduced by approximation, measurement noise, and external disturbance from the environment. Controllers for the control of vehicle models including not only the rigid body dynamics and force & moment generation processes but also the rotary wing dynamics and actuator dynamics need to be created and designed.

Control Mode Switching Synthesis. The synthesis procedure presented in Chapter 4 produced hybrid controllers which satisfied reachability specifications. The procedure should be improved to allow jumps in continuous states between control modes. The applicability of the proposed procedure to classes of dynamical systems depends on the availability of algorithms for reachability computation. New algorithms need to be developed to cover wider classes of dynamical systems. The structure of the hybrid controllers varied with given specifications. New control design should be created so that the controller structure remains invariant with respect to specifications.

Embedded Controller Synthesis. Embedded controller based on the proposed flight management system was synthesized based on a given architecture and the simulation of the system was performed using discrete event models. Hierarchical hybrid system simulation needs to be used to simulate the interaction between discrete and continuous components of the multi-modal system. Component-based design technique should be used to construct the multi-modal system to enhance the scalability and modularity on system design.

Appendix A

Appendix

A.1 System parameters

All variables except for the state variables and inputs are numeric constants, which can be obtained by measurements and experiments. The followings are the values of the constants.

I_x	=	0.142413	I_y	=	0.271256	I_z	=	0.271492
l_M	=	-0.015	y_M	=	0	h_M	=	0.2943
h_T	=	0.1154	l_T	=	0.8715	m	=	4.9
C_M^Q	=	0.004452	D_M^Q	=	0.6304	$\frac{\partial R_M}{\partial b_{1s}}$	=	25.23
C_T^Q	=	0.005066	D_T^Q	=	0.008488	$\frac{\partial M_M}{\partial a_{1s}}$	=	25.23
c_{M1}	=	6.4578	c_{M3}	=	100.3752	c_{T1}	=	0.1837
c_{T3}	=	0.1545						

The operation regions in radian for a_{1s}, b_{1s} and newton for T_M, T_T are:

$$|a_{1s}| \leq 0.4363 \qquad -20.86 \leq T_M \leq 69.48$$

$$|b_{1s}| \leq 0.3491 \qquad -5.26 \leq T_T \leq 5.26$$

A.2 Proof of Proposition 3.3.4

Define a 6×4 matrix

$$L(x) = \begin{bmatrix} L_{g_1} L_f^2 h_1 & \cdots & L_{g_4} L_f^2 h_1 \\ \vdots & \ddots & \vdots \\ L_{g_1} L_f^2 h_6 & \cdots & L_{g_4} L_f^2 h_6 \end{bmatrix}. \tag{A.1}$$

It can be easily shown that $L(x)$ can be decomposed as the product of two matrices, that is,

$$L(x) = \begin{bmatrix} \frac{1}{m} R & 0 \\ 0 & \Psi \mathcal{I}^{-1} \end{bmatrix}_{6\times 6} \begin{bmatrix} \frac{\partial f^b}{\partial w} \\ \frac{\partial \tau^b}{\partial w} \end{bmatrix}_{6\times 4}. \tag{A.2}$$

One can observe the left matrix are nonsingular for all $x \in X$ The right matrix has full column rank for all $x \in X$, i.e. the column vectors are linearly independent, and it can be shown by applying Gaussian elimination on columns.

Since $A_{k_1k_2k_3k_4}(x)$ is constructed by extracting the rows of the matrix $L(x)$ according to the indexes $k_1, \cdots, k_4$, we can define $A_{k_1k_2k_3k_4}(x) = E_{k_1k_2k_3k_4} L(x)$, where the extraction matrix $E_{k_1k_2k_3k_4}$ is defined as, in each row, it has a 1 in the position specified by the index and zeros elsewhere. By substitution, we have

$$A_{k_1k_2k_3k_4}(x) = \underbrace{E_{k_1k_2k_3k_4} \begin{bmatrix} \frac{1}{m} R & 0 \\ 0 & \Psi \mathcal{I}^{-1} \end{bmatrix}}_{A_L} \underbrace{\begin{bmatrix} \frac{\partial f^b}{\partial w} \\ \frac{\partial \tau^b}{\partial w} \end{bmatrix}}_{A_R}. \tag{A.3}$$

It is clear that

$$\text{rank}(A_{k_1k_2k_3k_4}(x)) \leq \min\{\text{rank}(A_L), \text{rank}(A_R)\} = 4.$$

In order to prove that $A_{k_1k_2k_3k_4}(x)$ has rank equal to 4, we have to show $\text{rank}(A_{k_1k_2k_3k_4}(x)) \geq 4$. The lower bound of Sylvester's inequality theorem is achieved if the rows of A_L lie in the left null space of A_R. By checking the rank of the matrix constructed by appending the basis of left null space of A_R with the transpose of the row vectors of A_L, one can verify symbolically that the vectors of row space of A_L are linearly independent with the vectors in the left null space of A_R. Hence, we have $4 \leq rank(A_{k_1k_2k_3k_4}(x))$ and the result is applied to all 15 possible combinations. Since the upper and lower bound are equal, we prove that $rank(A_{k_1k_2k_3k_4}(x)) = 4$ for all $x \in X$. □

Bibliography

[1] R. Alur, C. Courcoubetis, T.A. Henzinger, and P.-H. Ho. Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems I*, Lecture Notes in Computer Science 736, pages 209–229. Springer-Verlag, 1993.

[2] R. Alur and D. Dill. A theory of time automata. *Theoretical Computer Science*, 126:183–235, 1994.

[3] R. Alur, T. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22(3):181–201, 1996.

[4] Panos Antsaklis, Wolf Kohn, Anil Nerode, and Shankar Sastry, editors. *Hybrid Systems II.* Springer-Verlag, 1995.

[5] P.J. Antsaklis, K.M. Passino, and S.J. Wang. An introduction to autonomous control systems. *IEEE Control Systems Magazine*, 11(4):5–13, 1991.

[6] F. Balarin, P. Giusto, A. Jurecska, C. Passerone, E. Sentovich, B. Tabbara, M. Chiodo, H. Hsieh, L. Lavagno, A. Sangiovanni-Vincentelli, and K. Suzuki. *Hardware-Software Co-Design of Embedded Systems: The POLIS Approach.* Kluwer Academic Publishers, 1997.

[7] A. Bemporad. Reference governor for constrained nonlinear systems. *IEEE Transactions on Automatic Control*, 43(3):415–419, March 1998.

[8] A. Benveniste and P. Le Guernic. Hybrid dynamical systems theory and the signal language. *IEEE Transactions on Automatic Control*, 35(5):525–546, May 1990.

[9] G. Berry. The foundations of esterel. In C. Stirling G. Plotkin and M. Tofte, editors, *Proof, Language and Interaction: Essays in Honor of Robin Milner*. MIT Press, 1998.

[10] G. Berry and G. Gonthier. The esterel synchronous programming language: Design, semantics, implementation. *Science of Computer Programming*, 19(2):87–152, 1992.

[11] M. Branicky. Multiple lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):475–482, March 1998.

[12] M. Branicky, V. Borkar, and S. Mitter. A unified framework for hybrid control:background, model and theory. Technical Report LIDS-P-2239, Massachusetts Institute of Technology, 1994.

[13] R. W. Brockett. Hybrid models for motion control systems. In H. Trentelman and J. Willems, editors, *Essays in Control: Perspectives in the Theory and Its Applications*, pages 29–53. Birkhäuser, Boston, 1993.

[14] J. T. Buck, S. Ha, E. A. Lee, and D. G. Messerschmitt. Ptolemy: A framework for simulating and prototyping heterogeneous systems. *Int. Journal of Computer Simulation, special issue on Simulation Software Development*, 4:155–182, 1994.

[15] P. Caines and Y. J. Wei. Hierarchical hybrid control systems : A lattice theoretic formulation. *IEEE Transactions on Automatic Control*, 43(4), March 1998.

[16] P. Caspi, D. Pilaud, N. Halbwachs, and J. A. Plaice. Lustre: A declarative language for programming synchronous systems. In *Conference Record of the 14th Annual ACM Symp. on Principles of Programming Languages*, Munich, Germany, January 1987.

[17] A. Chutinan and B.H. Krogh. Verification of polyhedral-invariant hybrid systems using polygonal flow pipe approximations. In J. H. van Schuppen F. W. Vaandrager, editor, *Lecture Notes in Computer Science 1569*, pages 76–90. Springer-Verlag, Germany, 1999.

[18] J. Descusse and C. Moog. Dynamic decoupling for right invertible nonlinear systems. *System and Control Letter*, 8:345–9, 1987.

[19] A. Deshpande, D. Godbole, A. Gollu, and P. Varaiya. Design and evaluation tools for automated highway systems. In R. Alur, T.A. Henzinger, and E.D. Sontag, editors, *Hybrid Systems III*, Lecture Notes in Computer Science 1066. Springer Verlag, New York, 1995.

[20] A. Deshpande, A. Gollu, and L. Semenzato. The SHIFT programming language for dynamic networks of hybrid automata. *IEEE Transactions on Automatic Control*, 43(4):4283–4288, April 1998.

[21] Akash Deshpande, Aleks Gollu, and Luigi Semenzato. The SHIFT Programming Language and Run-time System for Dynamic Networks of Hybrid Automata. Technical report, University of California at Berkeley, 1996.

[22] S. Devasia, D. Chen, and B. Paden. Nonlinear inversion-based output tracking. *IEEE Transactions on Automatic Control*, 41(7):930–942, July 1996.

[23] C. Dows, A. Olivero, S. Tripakis, and S. Yovine. The tool KRONOS. In *Hybrid Systems III*, pages 208–219. Springer Verlag, LCNS 1066, New York, 1996.

[24] M. Egerstedt, T. J. Koo, F. Hoffmann, and S. Sastry. Path planning and flight controller scheduling for an autonomous helicopter. In J. H. van Schuppen F. W. Vaandrager, editor, *Lecture Notes in Computer Science 1569*, pages 91–102. Springer-Verlag, Germany, 1999.

[25] H. Elmqvist. Dymola-dynamic modeling language, user's manual. Technical report, Dynasim AB, Sweden, 1994.

[26] Farokh Eskafi, Delnaz Khorramabadi, and Pravin Varaiya. Smartpath: An automated highway system simulator. Technical Report PATH Memorandum 92-3, Institute of Transportation Studies, University of California, Berkeley, 1992.

[27] M. Fliess, J. Lèvine, Ph. Martin, and P. Rouchon. Flatness and defect of nonlinear systems: introductory theory and applications. *Int. J. of Control*, 61(6), 1995.

[28] M. Fliess, J. Lévine, Ph. Martin, and P. Rouchon. A lie-backlund approach to equivalence and flatness of nonlinear systems. *IEEE Transactions on Automatic Control*, 1999.

[29] Datta N. Godbole, John Lygeros, and Shankar S. Sastry. Hierarchical hybrid control: an IVHS case study. In *Proceedings of the 33th IEEE Conference on Decision and Control*, pages 1592–1597, 1994.

[30] J. Hauser, S. Sastry, and G. Meyer. Nonlinear control design for slightly nonminimum phase system: Application to V/STOL aircraft . *Automatica*, 28(4):665–679, 1992.

[31] S. Hedlund and A. Rantzer. Optimal control of hybrid systems. In *Proceedings of the 38th IEEE Conference on Decision and Control*, volume 4, pages 3972–3977, Phoenix, AZ, December 1999.

[32] T. Henzinger, P.-H. Ho, and H. wong Toi. A user guide to HYTECH. In *TACAS 95: Tools and Algorithms for the Construction and Analysis of Systems*, pages 41–71. Springer Verlag, LCNS 1019, New York, 1995.

[33] T. A. Henzinger. Masaccio: a formal model for embedded components. In *Proceedings of the First IFIP International Conference on Theoretical Computer Science, LNCS.* Springer Verlag, 2000.

[34] T.A. Henzinger. Hybrid automata with finite bisimulations. In Z. Fülöp and F. Gécseg, editors, *ICALP 95: Automata, Languages, and Programming*, Lecture Notes in Computer Science 944, pages 324–335. Springer-Verlag, 1995.

[35] T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *Proceedings of the 27th Annual Symposium on Theory of Computing*, pages 373–382. ACM Press, 1995.

[36] J. Hespanha and A. S. Morse. Switching between stabilizing controllers. Submitted to Automatica, May 2000.

[37] F. Hoffmann, T. J. Koo, and O. Shakernia. Evolutionary design of a helicopter autopilot. In P.K. Chawdhry R. Roy, T. Furuhasi, editor, *Advances in Soft Computing - Engineering Design and Manufacturing, Part 3: Intelligent Control*, pages 201–214. Springer-Verlag, Great Britain, 1999.

[38] The Mathworks Inc. Simulink: dynamic system simulation for Matlab. Technical report, Natick, MA, 1997.

[39] A. Isidori. *Nonlinear Control Systems.* Springer-Verlag, 1995.

[40] A. Isidori and C. I. Byrnes. Output regulation of nonlinear systems. *IEEE Transactions on Automatic Control*, 35(2):131–140, February 1990.

[41] K H Johansson, M Egerstedt, J Lygeros, and S Sastry. On the regularization of Zeno hybrid automata. *System & Control Letters*, 38:141–150, 1999.

[42] P. Kapasouris, M. Athans, and G. Stein. Design of feedback control systems for stable plants with saturating actuators. In *Proceedings of the 27th IEEE Conference in Decision and Control*, pages 469–79, Austin, Texas, December 1988.

[43] H. K. Khalil. *Nonlinear Systems.* Macmillan, 1992.

[44] J. P. Hespanha H. J. Kim and S. Sastry. Multiple-agent probabilistic pursuit-evasion games. In *IEEE Conference on Decision and Control, Phoenix*, pages 2432–2437, Phoenix,Arizona, December 1999.

[45] I. Kolmanovsky and E. G. Gilbert. Multimode regulators for systems with state & control constraints and disturbance inputs. In A. Stephen Morse, editor, *Lecture Notes in Control and Information Sciences 222, Control Using Logic-Based Switching*, pages 104–117. Springer-Verlag, London, 1997.

[46] T. J. Koo, F. Hoffmann, H. Shim, B. Sinopoli, and S. S. Sastry. Hybrid control of model helicopter. In *Proceedings of IFAC Workshop on Motion Control*, pages 285–290, Grenoble, France, October 1998.

[47] T. J. Koo, Y. Ma, G. Pappas, and C. Tomlin. SmartATMS : A simulator for air traffic management systems. In *Winter Simulation Conference*, pages 1199–1205, Atlanta, Georgia, December 1997.

[48] T. J. Koo, G. J. Pappas, and S. S. Sastry. Mode switching synthesis for reachability specifications. Submitted to Hybrid System: Computation and Control, Rome, Italy, 2001.

[49] T. J. Koo and S. Sastry. Output tracking control design of a helicopter model based on approximate linearization. In *Proceedings of the 37th Conference on Decision and Control*, pages 3635–40, Tampa, Florida, December 1998.

[50] T. J. Koo and S. Sastry. Differential flatness based full authority helicopter control design. In *IEEE Conference on Decision and Control, Phoenix*, pages 1982–87, Phoenix,Arizona, December 1999.

[51] T. J. Koo and S. Sastry. Uav trajectory generation with switching on flat outputs. In *IEEE Hong Kong Symposium on Robotics and Control*, volume I, pages 91–96, Hong Kong, July 1999.

[52] T. J. Koo and S. S. Sastry. Output tracking control design of a helicopter model based on approximate line arization. Submitted to IEEE Transactions on Control Systems Technology, 2000.

[53] T. J. Koo, D. H. Shim, O. Shakernia, B. Sinopoli, Y. Ma, F. Hoffmann, and S. Sastry. Hierarchical hybrid system design on berkeley uav. In *The International Aerial Robotics Competition*, Richland, Washington, USA, August 1998.

[54] T. J. Koo, B. Sinopoli, A. Sangiovanni-Vincentelli, and S. Sastry. A formal approach to reactive system design: A uav flight management system design example. In *IEEE International symposium on Computer-Aided Control System Design*, pages 522–7, Kohala Coast, Hawaii, September 1999.

[55] M. Kristic, I. Kanellakopouslos, and P. Kokotovic. *Nonlinear and Adaptive Control Design*. John Wiley and Sons, 1995.

[56] A.B. Kurzhanski and P.Varaiya. Ellipsoidal techniques for reachability analysis. In N. Lynch and B. H. Krogh, editors, *Hybrid Systems: Computation and Control, LNCS 1790*, pages 202–214. Springer Verlag, Berlin, 2000.

[57] G. Lafferriere, G. J. Pappas, and S. Yovine. A new class of decidable hybrid systems. In F. W. Vaandrager and J.H. van Schuppen, editors, *Hybrid Systems: Computation and Control, LNCS 1569*, pages 137–151. Springer Verlag, Berlin, 1999.

[58] G. Lafferriere, G.J. Pappas, G. Schneider, and S. Yovine. Symbolic reachability computations for families of linear vector fields. Accepted by Journal of Symbolic Computation, 2000.

[59] G. Lafferriere, G.J. Pappas, and S. Yovine. Reachability computation for linear hybrid systems. In *Proceedings of 14th World Congress of IFAC*, volume E, pages 7–12, Beijing, China, July 1999.

[60] Gerardo Lafferriere, George J. Pappas, and Shankar Sastry. Hybrid systems with finite bisimulations. In P. Antsaklis, W. Kohn, M. Lemmon, A. Nerode, and S. Sastry, editors, *Hybrid Systems V*, volume 1567 of *Lecture Notes in Computer Science*, pages 186–203. Springer Verlag, New York, 1998.

[61] E. Lee. Overview of the ptolemy project. Technical Report Memorandum UCB/ERL M98/71, EECS, University of California, Berkeley, CA, USA 94720, November 26, 1998.

[62] E. A. Lee and A.Sangiovanni-Vincentelli. A framework for comparing models of computation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17(12):1217–1229, December 1998.

[63] E. H. Lee, H. Shim, H. Park, and K. I. Lee. Design of hovering attitude controller for a model helicopter. In *Proceedings of Society of Instrument and Control Engineers*, pages 1385–1389, August 1993.

[64] M. Lemmon. On the existence of solutions to controlled hybrid automata. In N. Lynch and B. Krogh, editors, *Hybrid Systems: Computation and Control, LNCS 1790*, pages 229–242. Springer Verlag, New York, 2000.

[65] J. Liu, X. Liu, T. J. Koo, B. Sinopoli, S. S. Sastry, and E. A. Lee. Hierarchical hybrid system simulation. In *IEEE Conference on Decision and Control*, pages 3508–13, Phoenix,Arizona, December 1999.

[66] J. Lygeros. *Hierarchical, Hybrid Control of Large Scale Systems.* PhD thesis, Department of Electrical Engineering, University of California, Berkeley, California, 1996.

[67] J. Lygeros. Hybrid systems: Modeling, analysis and control. Technical Report Memorandum UCB/ERL M99/34, EECS, University of California, Berkeley, CA, USA 94720, Spring 1999.

[68] J. Lygeros, C. Tomlin, and S. Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 35(3), March 1999.

[69] John Lygeros, Datta N. Godbole, and Shankar Sastry. A verified hybrid controller for automated vehicles. Submitted to IEEE Transactions on Automatic Control, Special Issue on Hybrid Systems, 1996.

[70] J. Malmborg, B. Bernhardsson, and K. J. Astrom. A stabilizing switching scheme for multi controller systems. In *Proceedings of 13th World Congress of IFAC*, volume F, pages 229–234, San Francisco, U.S.A., July 1996.

[71] F. Maraninchi. The argos language: Graphical representation of description of reactive systems. In *Proceedings of the IEEE Workshop on Visual Languages*, Kobe, Japan, October 1991.

[72] P. Martin. Aircraft control using flatness. In *Proceedings of Symposium on Control, Optimization and Supervision*, volume 1.2 vol. 1322, Lille, France, July 1996.

[73] S. E. Mattsson, M. Andersson, and K. J. Astrom. Object-oriented modeling and simulation. In *CAD for Control Systems*, chapter 2, pages 31–69. Marcel Dekker Inc., New York, 1993. To appear.

[74] Barnes W. McCormick. *Aerodynamics Aeronautics and Flight Mechanics.* John Wiley and Sons, Inc., 1995.

[75] N. Megiddo, S. L. Hakimi, M. R. Garey, D.S. Johnson, and C. H. Papadimitriou. The complexity of searching a graph. *Journal of the ACM*, 35(1):18–44, January 1988.

[76] G. Meyer, R. Su, and L.R. Hunt. Application of nonlinear transformations to automatic flight control. *Automatica*, 20(1):103–7, 1984.

[77] P. J. Mosterman. An overview of hybrid simulation phenomena and their support by simulation packages. In F. W. Vaandrager and J.H. van Schuppen, editors, *Hybrid Systems: Computation and Control, LNCS 1569*, pages 165–177. Springer Verlag, Berlin, 1999.

[78] R.M. Murray, Z. Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation.* CRC Press, 1994.

[79] A. Nerode and W. Kohn. Models for hybrid systems: Automata, topologies, controllability, observability. In Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel, editors, *Hybrid System*, pages 317–356. Springer Verlag, New York, 1993.

[80] A. Packard and J. Doyle. The complex structured singular value. *Automatica*, 29(1):71–109, 1993.

[81] George J. Pappas. Avoiding saturation by trajectory reparameterization. In *Proceedings of the 35th IEEE Conference in Decision and Control*, pages 76–81, Kobe, Japan, December 1996.

[82] George J. Pappas. *Hybrid Systems : Computation and Abstraction.* PhD thesis, University of California at Berkeley, Berkeley, CA, 1998.

[83] G.J. Pappas, G. Lafferriere, and S. Sastry. Hierarchically consistent control systems. Accepted by IEEE Transactions on Automatic Control, 2000.

[84] T. D. Parsons. Pursuit-evasion in a graph. In Y. Alani and D. R. Lick, editors, *Theory and Application of Graphs*, pages 426–441. Springer-Verlag, 1976.

[85] Raymond W. Prouty. *Helicopter Performance, Stability, and Control.* Krieger Publishing Co., Inc., 1995.

[86] S. Sastry, G. Meyer, C. Tomlin, J. Lygeros, D. Godbole, and G. Pappas. Hybrid control in air traffic management systems. In *Proceedings of the 1995 IEEE Conference in Decision and Control*, pages 1478–1483, New Orleans, LA, December 1995.

[87] S. S. Sastry. *Nonlinear Systems: Analysis, Stability, and Control.* Springer-Verlag, New York, 1999.

[88] S.Edwards,L.Lavagno,E.A.Lee and A.Sangiovanni-Vincentelli. Design of embedded systems: Formal models, validation, and synthesis. *Proceedings of the IEEE*, 85(3), March, 1997.

[89] P. Seiler, A. Pant, and J. K. Hedrick. Preliminary investigation of mesh stability for linear systems. In *IMECS99/DSC-7B-1*, 1999.

[90] O. Shakernia, Y. Ma, T. J. Koo, , J. Hespanha, and S. Sastry. Vision guided landing of an unmanned air vehicle. In *IEEE Conference on Decision and Control, Phoenix*, pages 4143–48, Phoenix,Arizona, December 1999.

[91] O. Shakernia, Y. Ma, T. J. Koo, , and S. Sastry. Landing an unmanned air vehicle: Vision based motion estimation and nonlinear control. *Asian Journal of Control*, 1(3):128–145, September 1999.

[92] Omid Shakernia, George J. Pappas, and Shankar Sastry. Decidable controller synthesis for classes of linear systems. In N. Lynch and B. H. Krogh, editors, *Hybrid Systems: Computation and Control, LNCS 1790*, pages 407–420. Springer Verlag, Berlin, 2000.

[93] H. Shim, T. J. Koo, F. Hoffmann, and S. Sastry. A comprehensive study of control design for an autonomous helicopter. In *Proceedings of the 37th Conference on Decision and Control*, pages 3653–3658, Tampa, Florida, December 1998.

[94] R. Su, L.R. Hunt, and G. Meyer. Theory of design using nonlinear transformations. In *Proceedings of the 1982 American Control Conference*, volume 1, pages 247–51, 1982.

[95] D. Swaroop, J.K. Hedrick, P.P. Yip, and J.C. Gerdes. Dynamic surface control for a class of nonlinear systems. In *Proceedings of the 1997 American Control Conference*, volume 5, pages 3028–34, Albuquerque, NM, June 1997.

[96] A. Tarski. *A decision method for elementary algebra and geometry.* University of California Press, second edition, 1951.

[97] C. Tomlin. *Hybrid Systems with Application to Air Traffic Management.* PhD thesis, Department of Electrical Engineering, University of California, Berkeley, California, 1998.

[98] Claire Tomlin, John Lygeros, Luca Benvenuti, and Shankar Sastry. Output tracking for a non-minimum phase dynamic CTOL aircraft model. In *Proceedings of the 1995 IEEE Conference in Decision and Control*, pages 1867–1872, Kobe, Japan, December 1996.

[99] Claire Tomlin, George J. Pappas, and Shankar Sastry. Conflict resolution for air traffic management : A study in muti-agent hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):509–521, April 1998.

[100] L. van den Dries. Remarks on tarski's problem concerning $(\mathbb{R}, +\cdot, \exp)$. In G. Lolli, G. Longo, and A. Marcja, editors, *Logic Colloquium'82*, pages 97–121. Springer Verlag, 1992.

[101] A. J. van der Schaft and J.M. Schumacher. Complementarity modeling of hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):483–490, April 1998.

[102] M. J. van Nieuwstadt and R. M. Murray. Real time trajectory generation for differentially flat systems. In *Proceedings of 13th World Congress of IFAC*, San Francisco, CA, July 1996.

[103] M. J. van Nieuwstadt and R. M. Murray. Outer flatness: Trajectory generation for a model helicopter. In *Proceedings of European Control Conference*, 1997.

[104] Pravin Varaiya. Smart cars on smart roads: problems of control. *IEEE Transactions on Automatic Control*, AC-38(2):195–207, 1993.

[105] R. Vidal, S.Schaffert, J. Lygeros, and S. Sastry. Controlled invariance of discrete time systems. In N. Lynch and B. Krogh, editors, *Hybrid Systems: Computation and Control, LNCS 1790*, pages 437–450. Springer Verlag, New York, 2000.