The Pennsylvania State University
The Graduate School

HYBRID SUPERVISORY CONTROL OF COMPLEX DYNAMICAL SYSTEMS

A Thesis in
Mechanical Engineering
by
Murat Yasar

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2007

The thesis of Murat Yasar was reviewed and approved* by the following:

Asok Ray
Distinguished Professor of Mechanical Engineering
Thesis Advisor, Chair of Committee

Joseph F. Horn
Associate Professor of Aerospace Engineering

Alok Sinha
Professor of Mechanical Engineering

Christopher D. Rahn
Professor of Mechanical Engineering

Jeffrey S. Mayer
Associate Professor of Electrical Engineering

Karen A. Thole
Professor of Mechanical Engineering
Head of the Department of Mechanical Engineering

*Signatures are on file in the Graduate School.

Abstract

Supervisory control of Discrete Event Systems is a relatively new research area motivated by the pragmatic need to solve the problem of modeling and control of large-scale complex systems in discrete-event setting. The pertinent literature on Discrete Event Supervisory (DES) control offers the innovative idea of language measure to quantitatively evaluate the performance of a DES controller. This dissertation addresses fundamental issues about language measure and optimal DES control to initiate a benchmark control strategy which is not available in existing literature. The objective of this dissertation is to enhance the theory and application aspects of the DES control in a hybrid perspective. The new optimal control strategies along with the enhanced event generation algorithms are developed and used as benchmark to fully elucidate and improve the optimal control and event generation concepts in the literature of hybrid supervisory control.

The integrated experimental and analytical studies presented in this dissertation envelop the implementation of the new theories of DES control in a hierarchical and hybrid fashion that will incorporate continuously-varying and discrete-event dynamics. Emerging applications of hybrid supervisory control to complex dynamical systems such as gas turbine engines, rotorcrafts and mobile robotic platforms are investigated, and validation of the theoretical work by simulation and experimentation is performed. Continuous and DES controllers are implemented in a hybrid setting to regulate both system dynamics and operational behaviors. Optimal DES controllers are designed for a twin-engine propulsion system which is integrated with an aircraft model and flight control, and a Segway robot which autonomously performs coordinated missions together with an unmanned aerial vehicle.

It is envisioned that this dissertation will make a positive contribution toward successful implementation of the hierarchically structured DES decision and control theory to many complex systems.

Table of Contents

List of Figures

List of Tables

List of Symbols

$\mathbb{R}$	Real numbers, p. 4
x	States of a continuous system, p. 4
y	Outputs of a continuous system, p. 4
u	Inputs of a continuous system, p. 4
t	Time, *sec*, p. 4
G	Deterministic finite state automaton, p. 7
Q	Discrete state space, p. 7
Σ	Finite set of discrete event alphabet, p. 7
δ	State transition function, p. 7
q_0	Initial state of the automaton, p. 7
Q_m	Set of marked (accepted) states, p. 7
ε	Empty string, p. 7
$L(G)$	Language generated by the plant automaton, p. 7
$L(S/G)$	Language generated by supervised plant automaton, p. 25
$L_m(G)$	Marked (accepted) sublanguage of the plant automaton, p. 26
$\bar{\chi}$	Characteristic weight vector of the states, p. 26

$\tilde{\mathbf{\Pi}}$ Event cost matrix, p. 27

$\emptyset$ Empty set, p. 27

$\mathbf{\Pi}$ State transition cost matrix, p. 27

μ Signed real measure of a regular language, p. 27

θ Residue of state transition probability, p. 28

$<,>$ Inner product, p. 31

$\mathbf{\Pi}^{\diamondsuit}$ Optimal state transition cost matrix for *Absolute Disabling*, p. 33

$\mathbf{\Pi}^{\blacklozenge}$ Optimal state transition cost matrix for *Relative Disabling*, p. 35

h Altitude, ft, p. 73

$\mathbf{M}$ Mach number, p. 73

T_a ambient temperature, K, p. 73

g Acceleration due to gravity, ft/sec^2, p. 99

s Complex frequency, p. 99

α Angle of attack, rad, p. 100

β Angle of side slip, rad, p. 100

q Pitch rate, rad/sec, p. 103

Z Total force along the z-body axis, lb, p. 103

M Total body axis aerodynamic pitching moment, $lb \cdot ft$, p. 103

δ_e Elevator deflection, p. 103

δ_t Thrust command, p. 103

ν Pseudo-control, p. 103

ω_n Natural frequency of the command filter, p. 104

ξ Damping ratio of the command filter, p. 104

K_P Proportional control gain, p. 104

K_D Differential control gain, p. 104

p Roll rate, rad/sec, p. 105

r Yaw rate, rad/sec, p. 105

L Total body axis aerodynamic rolling moment, $lb \cdot ft$, p. 105

Y Total side force along the y-body axis, lb, p. 105

N Total body axis aerodynamic yawing moment, $lb \cdot ft$, p. 105

δ_a Aileron deflection, p. 105

δ_r Rudder deflection, p. 105

δ_{dt} Differential thrust, p. 105

V Total velocity, ft/sec, p. 105

θ Pitch angle, rad, p. 105

ϕ Roll angle, rad, p. 105

p_s Stability axis roll rate, ras/sec, p. 105

K_I Integral control gain, p. 106

ψ Heading angle, rad, p. 113

Superscript

T Transpose, p. 29

$\dot{}$ First derivative with respect to time, p. 103

$\ddot{}$ Second derivative with respect to time, p. 103

Subscript

0 Steady state value, p. 104

cmd Commanded value, p. 104

d Desired value, p. 104

Acknowledgments

This work would not have been accomplished without the guidance, inspiration and dedication of Prof. Asok Ray. He was a tutor and mentor to me through out my graduate studies.

I would like to thank Dr. Joseph Horn for his guidance on aircraft control problems and providing the most practical suggestions and solutions. I would also like to thank Dr. Alok Sinha, Dr. Christopher Rahn, and Dr. Jeffrey Mayer for their invaluable inputs and being in my dissertation committee. Dr. Mayer was also my advisor for MS EE degree and provided significant insights in the formulation of this thesis.

There is a handful of people that I am honored to work together. Among them, Dr. Devendra Tolani, Dr. Jinbo Fu and Dr. Ishanu Chattopadhyay have influential and supportive work in my research. I also thank Goutham, Derek and Soumik for being an integral part of the simulation laboratory.

This work has been supported in part by the U.S. Army Research Laboratory and the United States Army Research Office under Grant No. DAAD19-01-1-0646 and NASA Glenn Research Center under Grant Nos. NAG3-2448 and NNC04GA49G.

Last, but certainly not the least, the support and encouragement of my close friends in State College, especially Burcu Unal, was crucial for the last five years.

Dedication

to my mother Ayşegül and my father Osman; they gave me everything without expecting anything in return ...

Chapter 1

Introduction

From the beginning of the history, mankind aspired to observe and master all kinds of systems, which are defined as collections of inter-related elements comprising a unified whole. The term "system" comes from the Latin word "systema" [1] that means to combine, to set up or to place together. A system typically consists of components (or elements) which are connected together in order to facilitate the flow of information, matter or energy.

The foremost aim of this dissertation is to formulate and implement a comprehensive hybrid control and decision-making strategy for complex dynamical systems to achieve desired performance in continuous sense and enhanced behavior in discrete logic. This goal could be achieved by a hierarchically structured control architecture that utilizes continuously-varying control laws in lower levels, whereas the intelligence of the system is increased in higher levels by using discrete event supervision. Therefore, the overall architecture has a hybrid characteristic in the sense that the supervisory control in the higher levels not only drives the discrete event dynamics of the system but also has a direct impact on the continuous dynamics.

1.1 Background and Literature Survey

1.1.1 Complex Systems

A complex system can be defined in terms of mutually interacting and interwoven parts, entities or agents, and whose properties are not fully explained by an understanding of its components which exhibit nonlinear couplings. Although, a special edition of Science [2] highlighted several definitions of complex systems, it is more elegant to identify the complex systems with their features that distinguish them from being a merely complicated system:

Nonlinearity - In general, nonlinearity signifies a complicated situation that results in an unpredictable cause and effect relation between parts of a complex system. In practical terms, this means even a small perturbation may cause a large effect or no effect at all. In mathematics, this is also known as the *butterfly effect*, which is a phrase that encapsulates the more technical notion of sensitive dependence on initial conditions [3]. Small variations of the initial condition of a dynamical system may produce large variations in the long term behavior of the system. Therefore, the behavior of nonlinear systems is not subject to the principle of superposition, as linear systems are. It is common to encounter complex phenomena such as chaos effects, strange attractors, and bifurcation in nonlinear systems [4].

Nested Structure - It is possible that the individual components of a complex system may be complex systems themselves. For example, an economy is made up of organizations, which are made up of people, which are made up of cells, in turn all of which are complex systems. Another example, which is detailed as part of a control application in this thesis, is the propulsion system of an aircraft, which is made up of individual gas turbine engines, which themselves are complex systems.

Feedback Relations - In complex systems, feedback is a key aspect, which may be negative (e.g. damping), tending to reduce output (but may be used to stabilize and linearize the process), or positive (e.g. amplifying), tending to increase output. The effects of a component's behavior are fed back to all or

some of the components of the system such that the system's future behavior is altered. Systems which include feedback are susceptible to oscillations of output resulting from improperly tuned inputs of alternating positive and negative feedback.

Emerging Behavior - Another key aspect of a complex system that makes it different from a sheer complicated one is that some behaviors and patterns emerge in complex systems as a result of the patterns of relationship between the elements. This is often stated as *the whole not being sum of the parts*. This means that the components are not able to reveal what is happening in the system as a whole. If they could, all the complexity would have to be present in that component. A corollary to this argument is that controlling an individual element in a complex system is not sufficient for controlling the whole system.

Open Systems - Complex systems are usually open systems i.e., they exist in a thermodynamic gradient where energy and information (or entropy) [5] are constantly being exchanged through system boundaries. In other words, complex systems are usually not in an energetic equilibrium and subject to fluctuations. Despite this flux, the system might be in a quasi-static equilibrium that results in pattern stability.

Imprecise Boundaries - It is often difficult to determine the boundaries of a complex system. The decision is usually made by the observer based on not only any intrinsic property of the system itself, but also perception of the observer.

System Memory - Unlike memoryless systems, the history of a complex system may have a direct effect on its future behavior, because of time-varying relations between the components. That is, the dynamics of the system change over time, and prior states may have an influence on present and future states. The most frequently seen example is complex systems that exhibit *hysteresis*, which loosely means that the system does not instantly follow the forces applied to it, but reacts slowly, or does not return completely to its original state [6].

Dynamic Networking - The dynamic network of a complex system plays a key role in understanding the system. Complex systems often exhibit such topologies of networks which have many local interactions and a smaller number of inter-area connections are often employed. That is, information is normally received from the nearest neighbors. Richness of connections implies communications that pass across the system but are probably modified on the way.

A direct quote in Sync by Steven Strogatz [7] states: "Every decade or so, a grandiose theory comes along, bearing similar aspirations and often brandishing an ominous-sounding C-name. In the 1960 it was cybernetics. In the '70s it was catastrophe theory. Then came chaos theory in the '80s and complexity theory in the '90s."

1.1.2 Continuously-varying Systems

In engineering and mathematics, a dynamical system is defined as a process which has components or flows or both, that change over time according to a continuously-varying rule that is defined in terms of the system dynamics. There are two major types of dynamical systems which are observed in engineering and mathematics. Discrete-time systems, for which the time is measured in discrete steps, are modeled as recursive relations; whereas continuous-time dynamical systems, for which the time is measured continuously, are expressed as ordinary or partial differential equations. Consequently, ordinary and partial differential equations which are represented in finite- and infinite-dimensional state spaces respectively, have been often used to model dynamical processes (often referred as the plant) encountered in nature.

A continuously-varying dynamical system (CVDS) is defined as $P = (X, U, f)$ where $X \subseteq \mathbb{R}^n$ is the state-space of the system with n being the dimension of the state-space; $U \subseteq \mathbb{R}^m$ is the space of the system inputs with m being the dimension of the input space; and $f : X \times U \to \mathbb{R}^n$ is a continuous vector field

The behavior of the system is governed by the differential equations:

$$\dot{x} \;=\; f(x, u, t), \qquad x(t_0) = x_0 \tag{1.1a}$$

$$y = g(x, u, t) \tag{1.1b}$$

where $t \in \mathbb{R}$, $t \geq t_0$ is the time index. At any time instant t, $x \in X$ is the n-dimensional state of the system, $u \in U$ is the m-dimensional input of the system, x_0 is the initial condition of the state vector x at t_0, $y \in \mathbb{R}^l$ is the l-dimensional output or measurement of the system.

As time evolves, the dynamic behavior of the system P starting from $x_0 \in X$ is captured by the state vector $x(t)$, which is the solution to the initial value problem of the differential equation 1.1, and steered by the control input $u(t)$, as shown in Figure 1.1.

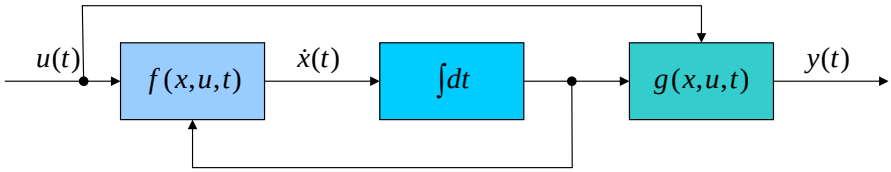


Figure 1.1. Continuously-varying dynamical systems

The continuous-time dynamical system controllers process the sensor data to generate actuator commands that drive the plant through continuously manipulated actuators. For both continuous and discrete-time dynamical systems, a number of metrics such as H_2, H_∞ and ℓ_1 norms have been developed to measure the robust performance of the controlled plant in the presence of bounded uncertainties and disturbances in the plant dynamics [8]. The above mentioned systems have also been investigated through a performance index in optimal control theory [9] and through Lyapunov stability analysis in linear and nonlinear systems theory [10][6].

1.1.3 Discrete Event Systems

Complex systems are the essential elements of science and engineering. In science, we would like to understand complex systems, and in engineering, we would like to master them. Since complex systems are difficult to understand, a central problem in science and engineering is the ability to predict and control the behavior of these systems. In order to overcome this difficulty, human reasoning is often attempted to be emulated for modeling and control of these systems [11]. The models of these

dynamical systems are usually driven by discrete events that occur asynchronously in time and are represented by logical variables instead of real values. The resulting systems exhibit discrete logical states; consequently, these systems are also called Discrete Event Systems that cannot be adequately modeled by ordinary or partial differential equations. Especially, human-engineered complex systems (e.g., aircraft, power plants, networked robotic systems) may require Discrete Event Supervisory (DES) decision and control for enhanced reliability, performance, and operating range [12]. For example, military Command, Control, Communications, Computer, Intelligence, Surveillance and Reconnaissance (C^4ISR) systems belong to the Discrete Event Systems class where the evolution of system dynamics in time depends on the complex interactions of various discrete events, such as initiation or completion of battlefield operations and success or failure of certain reconnaissance or surveillance missions [13].

A finite state machine is a model of behavior composed of states and transitions, and used as one of several mathematical representations of discrete distributed systems and discrete event systems. In the automata theory, deterministic finite state automata (DFSA) and nondeterministic finite state automata (NFSA) are equivalent in the sense that both generate and recognize only the set of regular languages [14]. Consequently, discrete event dynamical behavior of physical plants (e.g., mobile robot and gas turbine engine) is often modeled as regular languages that can be realized by finite state automata [15][16] $G = (Q, \Sigma, \delta, q_0, Q_m)$, where Q is the finite set of states with $|Q| = n$ and $q_0 \in Q$ is the initial state; Σ is the (finite) alphabet of events with $|\Sigma| = m$; Σ^* is the set of all finite-length strings of events including the empty string ε. The (possibly partial) function $\delta : Q \times \Sigma \to Q$ represents state transitions and $\hat{\delta} : Q \times \Sigma^* \to Q$ is an extension of δ and $Q_m \subseteq Q$ is the set of marked (also known as accepted) states which have some importance (positive or negative) on the DFSA model.

Given an initial state $q_0 \in Q$, the behavior of the deterministic finite state automaton (DFSA) G is a sequence of states such that events trigger the state transitions as shown in Figure 1.2. The generating language of G is

$$L(G) = \{s \in \Sigma^* | \hat{\delta}(q_0, s) \in Q\} \tag{1.2}$$

and the marked language of G is

$$L_m(G) = \{s \in \Sigma^* | \hat{\delta}(q_0, s) \in Q_m\} \tag{1.3}$$

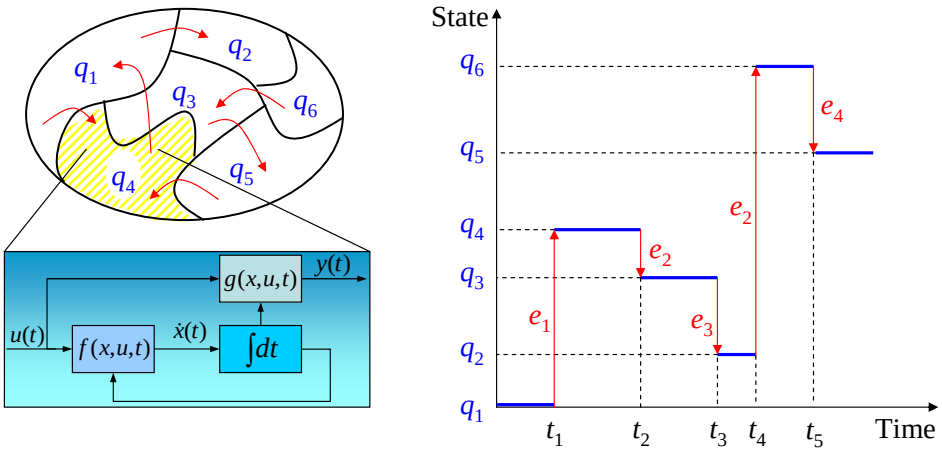


Figure 1.2. Discrete event dynamical systems

1.1.4 Hybrid Systems

Hybrid systems involve both continuously varying and discrete-event dynamics. Their evolution is governed by equations that generally depend on both dynamics in an interacting manner. Hybrid control systems are control systems that involve both continuous and discrete dynamics, and continuous and discrete controls. The continuous dynamics of such a system is usually modeled by a controlled vector field or differential equations. Its hybrid nature is expressed by a dependence on some discrete phenomena, corresponding to discrete states, events, and controls. Examples of hybrid systems emerges in many areas including computer disk drives [17], automatic control systems [18, 19], robotic systems [20, 21], intelligent vehicle and highway systems [22, 23] etc.

A hybrid system, denoted by H, is modeled as an hybrid automaton $H = (Q, X, \Sigma, \delta, \mathcal{X}_0, I, f)$, where Q is the discrete state space representing a set of operation modes of the system; $X \in \mathbb{R}^n$ is a bounded continuous state space. The state space of H is thus $\mathcal{X} = (Q \times X)$ and states of H are of the form $(q, x) \in \mathcal{X}$ with $q \in Q$ and $x \in \mathbb{R}^n$; Σ is the set of discrete events; $\delta : (Q \times X) \times \Sigma \times (Q \times X) \to 2^X$ is the transition function capturing mode switching by transition guards $g(\sigma)$, where

$\sigma \in \Sigma$, and reset mapping $r(\sigma, x)$; $\mathcal{X}_0 \subseteq \mathcal{X}$ is the set of initial states; $I : Q \to 2^X$ is the staying/invariant conditions $I(q)$ that constrain the values of the continuous variables x for being at a discrete state $q \in Q$; $f : Q \to (X \times \mathbb{R}^m \to \mathbb{R}^n)$ assigns a continuous vector field $f_q(x, u)$ on the continuous state $x \in X$, given an input $u \in \mathbb{R}^m$, to every discrete state $q \in Q$.

The transition guard $g(\sigma)$ is defined to characterize the behavior of a hybrid system

$$g(\sigma) = \{x \in I(q) | \delta((q, x), \sigma) = (\acute{q}, \acute{x}), \text{ for some } \acute{x} \in I(\acute{q})\} \tag{1.4}$$

and the set-valued reset map $r(\sigma, x)$ is given by

$$r(\sigma, x) = \{\acute{x} \in I(\acute{q}) | \delta((q, x)\sigma) = (\acute{q}, \acute{x})\} \tag{1.5}$$

Therefore, when the hybrid automaton is in a discrete state q and continuos state $x \in g(\sigma)$, it makes a transition from q to $\acute{q}$ by generation of the discrete event σ, and resets the continuous variables to $\acute{x} \in r(\sigma, x)$.

In other words, a hybrid system has continuous dynamics modeled by a differential equation $\dot{x}(t) = \xi(t)$, $t \geq 0$ that depends on some discrete phenomena. $x(t)$ is the continuous component of the state taking values in some subset of a Euclidean space. $\xi(t)$ is a controlled vector field that generally depends on $x(t)$, the continuous component, $u(t)$ of the control policy, and the aforementioned discrete phenomena. There exist four identified phenomena for hybrid systems [24]:

- Autonomous switching: The vector field $\xi(\bullet)$ changes discontinuously, or "switches", when the state $x(\bullet)$ hits certain boundaries. Example: Hysteresis

- Autonomous impulses: The continuous state $x(\bullet)$ jumps discontinuously on hitting prescribed regions of the state space. Example: Collisions

- Controlled switching: $\xi(\bullet)$ switches in response to a control command with an associated cost. This can be interpreted as switching between different vector fields. Controlled switching arises when one is allowed to pick among a number of vector fields, $\dot{x} = f_i(x)$ $i \in \{0, 1, \ldots N\}$. Example: Manual transmission in automobile

- Controlled impulses: $x(\bullet)$ jumps in response to a control command with an

associated cost. Example: Inventory management in manufacturing production systems.

Figure 1.3 depicts the structure of hybrid systems. It is clear that a hybrid system can be reduced to a CVDS captured by a set of differential equations if there is no discrete state transition (*Continuation paradigm*), or to a discrete event system captured by a finite state automaton if its continuous dynamics are not considered (*Aggregation paradigm*). In the aggregation paradigm, one endeavors to treat the entire system as a finite automaton or discrete-event dynamic system. This is usually accomplished by partitioning the continuous state space and considering only the aggregated dynamics from cell to cell in the partition. In the continuation paradigm, one endeavors to treat the whole system as a differential equation. This is accomplished by "simulating"or "embedding" the discrete actions in nonlinear ordinary differential equations and treating the discrete actions as "disturbances" of some differential equation.

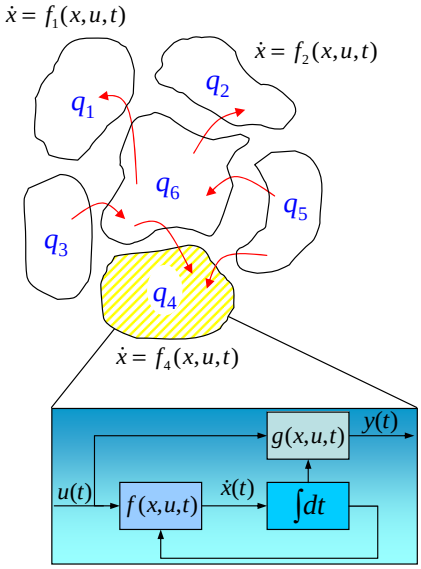


Figure 1.3. Hybrid systems

1.1.5 Supervisory Control Theory (SCT)

The supervisory control theory (SCT) provides a framework for achieving behavioral specifications of physical plants. The objective of the SCT is to design su-

pervisors in such a way that the controlled plant behaves according to given specifications [25]. Supervisory control of Discrete Event Systems is a new research area pioneered by Ramadge and Wonham [15][26] and subsequently extended by several researchers. Because of the above motives, Discrete Event Supervisory (DES) control of complex dynamical systems has been investigated and put in a mathematical setting by these research efforts including those of Ray et al. [27] Instead of processing continuous numerical data, DES controllers disable or enable certain controllable events in the physical plant based on the control specification and observed event strings. The algorithms for DES control synthesis have evolved based on the automata theory and formal languages in the discipline of Computer Science.

The legal behavior of a controlled plant under different controllers could be different as they are designed according to different control specifications. Therefore, the respective controlled sublanguages of the plant may not be totally ordered. The technical literature on DES control by Ray and Phoha [28] and Wang and Ray [29] offers the innovative idea of language measure to quantitatively evaluate the performance of a DES controller. This measure allows total ordering for individual sublanguages of the controlled plant behavior under different supervisors and thus helps quantitatively evaluate efficacy of different DES controllers. The measure of regular languages eventually led to the development of optimal DES controllers for control of automata (Fu et al. [30]).

This thesis focuses on development of intelligent decision and control algorithms based on DES control of complex dynamical systems like rotorcraft and aircraft propulsion systems. A good feature of the DES control approach is that the control policy can be adaptively updated on-line and that the system is tolerant to small anomalies and component faults. Although the theory of DES control has been developed for almost two decades [15], only a very few applications have been reported in literature [31]. A possible explanation is that, until recently, no quantitative analytical tool was available for design and evaluation of DES controllers. The work reported here makes use of a quantitative measure of regular languages [12], and is a novel application of hierarchical DES control synthesis for complex dynamical systems. The real-time implementation of the DES scheme is challenging because it requires integration of several disciplines such as systems theory,

computer hardware and software, and domain knowledge of the dynamical system being controlled.

1.1.6 Optimal Supervisory Control

In continuous varying dynamical systems, optimal control has been developed for decades, and usual practice is to try a number of optimality criteria before a satisfactory system response is obtained. In optimal control theory, different performance index structures can be utilized, resulting in different optimal solutions for the control input. Given the structure of the performance index, several solution procedures have been postulated such as "Riccati equation", "Controllability Grammian", and "Maximum principle" [9]. Similarly in DES control theory, different performance criteria can be used to obtain different supervisory control laws that disable or enable different sets of controllable events. This immediately follows from the postulation of how the disabled events affect the dynamic behavior of the controlled plant.

In the Discrete Event Systems literature, optimal control problem have been addressed by some researchers after the supervisory control theory was introduced by Ramadge and Wonham [15] two decades ago. Some of these works follow the path of Ramadge and Wonham [26] in the sense that the fundamental control philosophy requires the events to be disabled and not forced where the plant generates events spontaneously, instantaneously and asynchronously [32][33]. The concept of minimally restrictive behavior has been considered optimal, in regard to the partially ordered spaces encountered in conjunction with computation of the supremal sublanguage [34]. Ray et al. [12] reported the way of total ordering of the sublanguages using a language measure. The concept of optimal supervisory control arises naturally based on the quantitative evaluation of the sublanguages, and the supervisor with the maximum language measure is regarded as optimal. They have also shown in the language-theoretical framework that this optimal supervisor is the maximally permissive one within the set of acceptable supervisors.

On the other hand, some researchers have introduced the concepts of the event costs and control costs that have to be optimized in a performance index structure [35][36]. These kinds of control strategies are rather contradictory with the

optimality sense of Ramadge and Wonham since they search for a solution of minimum path problem which does not generate the least restrictive supervisor. Thus, their approach is to construct a controller which may enforce some events unlike a supervisor. An even more restrictive approach is introduced in [37] that construct a controller which selects a unique controllable event to be executed per state. All of these control policies attempt to find and force the shortest path on the graph representing the uncontrolled plant. The optimal attractor problem introduced in [38] assumes the same costs but events are disabled instead of being forced to occur. This approach is more consistent with the DES control philosophy. However, in none of the above mentioned solutions to the optimal control problem, the incurred cost is a function of the dynamic behavior of the system.

1.2 Motivation

1.2.1 Limitation of Continuous Models

The traditional formalism for mathematical modeling of dynamic systems has exhibited great success. There is a variety of approaches for control synthesis of continuous dynamic systems. Some of them are widely used in practice, e.g., PID control, and some are of a purely theoretic interest. In a rich mathematical domain such as the Euclidean space or more generally a differentiable manifold over the real number field, the dynamics of the system under consideration can be described using functions constructed from arithmetic, algebraic, and trigonometric relationships without logical elements. Together with the physical laws such as those of mechanics, electromagnetics, or thermodynamics, system and control theory play a dominant role in those systems that can be specified by differential or difference equations. However, there are several reasons that the continuous modeling formalism is insufficient for describing real systems.

1. The dynamics of many physical components of plants and controllers can not be modeled solely by continuous models, for example, the saturation phenomena of sensors and actuators. The behaviors of valves and switches are best modeled as discrete transitions.

2. High level intelligence might require sequential behaviors that can not be described in continuous trajectories, for example, a moving robot is designed to search for objects of interest while avoiding obstacles.

3. Linear approximation of highly non-linear systems is valid only in some region of the state space. When the system leaves this region, a different linear model should be sought if linearization is possible.

4. Many control systems are part of larger systems and interact with entities such as human operators or computers. The controller thus may receive discrete commands that will activate or suspend its execution or force it to switch to another mode of operation.

The Discrete Event System modeling formalism is motivated by the fact that it is often natural to hierarchically decompose complex dynamic systems into lower-level components representing the time-driven continuous behavior and higher-level components representing the instantaneous event-driven abstraction of the physical systems.

1.2.2 Applicability to Life Extending Control (LEC)

The other motivating fact that brought Discrete Event Systems into our focus is the possibility of it to provide a very effective tool in health management and Damage Mitigating Control (DMC) [39] also called Life Extending Control (LEC) [40] for large, complex systems such as power plants [41], spacecraft [42], and aircraft [43]. For example, the current market-driven environment of energy deregulation has made generating units to experience uncertain and rapidly fluctuating load demands. Consequently, some of these units are subjected to rapid load maneuvering and thus undergo excessive structural damage in critical plant components [41]. This would excessively increase the probability of unscheduled shutdown and loss of availability unless the equipment health of these generating units is appropriately maintained. Therefore, researchers have put more interest towards the area of LEC. In LEC, the information collecting and decision making processes are more convenient to be characterized in discrete settings, which requires the concept of discrete event modeling of the system. The reason is the necessity

to model the damage progress and gather the damage information, after which the decision can be made to reduce the damage and extend the life of the device without significantly reducing the performance.

Consequently, the LEC concept also opens a new window to the need for detection and modeling of changes in the system dynamics over the life span of a complex system and its subsystems. These changes happen in the slow-time scale rather than being in the time scale of system's dynamical equations and are often addressed in the fault detection research and have been referred to as anomalies rather than faults [44], since these slow-time scale changes do not necessarily comes from an error in the system, but might be the result of a characteristic inherent to the system such as aging, deterioration or wear. Therefore, it is necessary to formulate a measure for the anomalies that evolve in time [45]. This measure should also capture the essential changes in the system's dynamical behavior by only using the available time series data obtained through the sensors.

1.2.3 Requirement for Enhancement in SCT

From the perspectives of optimal control theory, which suggests that different performance indices lead to different optimal controllers, the optimal control policy proposed by Fu et al. [30] is a frame that can be used to develop new optimal control strategies by changing the optimality criteria. Therefore, it is essential to explore new optimal control approaches that could increase the performance of the previously developed algorithms.

In the discrete-event optimal control procedure, the dynamic response of plants to the disabling and enabling of discrete events under the direct supervision of a DES controller is postulated as the update of state transition cost matrix. The disabled events are erased from the plant automata whereas the enabled events are inserted again. This is reflected mathematically to the transition cost matrix by deleting and adding the event cost of that particular event which is disabled or enabled. The choice for the dynamic response of the plants to the supervisory decisions is analogous to the selection for the structure of the cost function in the optimal control of continuous dynamical systems. Consequently, there is more than one way of updating the matrix as there is more than one structure of the

cost function in continuous case. In the view of this argument, it is possible to enhance the optimal control strategy further.

The optimal control strategies that are based on the signed real measure of regular languages [46] require two sets of parameters of the controlled plant. While the relative event probabilities are determined using the system identification technique developed by Ray and Wang [47], the characteristic values associated with each state is yet to be decided by the designer. This procedure is very much similar to the one used in the optimal control of continuous systems where the designer decides the state and control input weights in a performance index structure. The performance of the resulting controller on the actual plant depends on the selected weights; therefore, the weights cannot be assigned arbitrarily. Likewise, for the Discrete Event Systems, the performance of the supervisors obtained using the optimal control algorithms will reflect the selection of the state weights during the design stage. Therefore, a poor decision at this point will adversely affect the performance of the controlled plant, although the optimality is achieved in terms of the chosen weights. The criteria to decide the weight of each state can be put into the context of resolving the importance of the states which is not totally subjective.

The last, but not the least important issue about the developed control strategies is the way to create the events that drive the Discrete Event Systems through state transitions by using the available sensory and non-sensory information. This difficulty should be addressed with the same conceptual frame of anomaly detection [44], since both need the same principal signal processing issues. Although the event generation is not restricted to the fusion of sensory information, the basic idea is development of a metric to decide when and how the events are created. Therefore, the proposed metric development in the anomaly detection problem can be brought into picture with necessary modifications to envelop the fusion of non-sensory information with the sensory ones.

Event generation, especially from noise corrupted sensor data, is the key link between the DES control model and real complex system which will ultimately be in continuous time setting. Event generation has paramount importance since it is only possible to make correct supervisory decisions after identifying what really happens underneath the sensor data. Therefore, signal processing and anomaly measure [44] play important roles in event generation. Some techniques that have

been discussed under anomaly measure concept such as wavelet analysis [48] and symbolic dynamics [49][50] together with other mathematical structures like fuzzy logic [51] can be very useful for event generation. Some events are very easy to be generated, such as a number of man-made events or easily observable events, or on-off events. However, some events may not be straightforward to generate, especially those hidden in large ensemble of noise contaminated sensor data. Failure to generate those events will deteriorate the performance of the supervisor and make the system very difficult to meet the control specifications. On the other hand, falsely generation of those events will certainly jeopardize the DES control and system stability and performance.

1.3 Research Objectives and Contributions

The research reported in this dissertation addresses fundamental issues about language measure and optimal DES Control, which are not available in the current literature. The applications of this fundamental theoretical research are examined on fixed wing aircraft, propulsion and flight management systems, as well as rotary wing Unmanned Aerial Vehicle (R-UAV) operations. Discrete-event modeling and parameter identification are performed for a Segway Robotic Mobile Platform (RMP). Event generation for Discrete Event Systems, based on information fusion, and the interactions between continuous dynamical systems and Discrete Event Systems in hybrid and hierarchical control architectures are also investigated. The research efforts are summarized as follows:

1. In the optimal control context, theorems and algorithms that guarantee different global optimality criteria in terms of performance are proposed and compared. This is analogous to using different performance indices in optimal control of continuous dynamical systems. The resulting supervisory control laws of different approaches may or may not be the same with each other or with the previous approach introduced by Fu et al.[30] Although the proofs of these strategies may appear to be deducted from the previous strategy, they indeed require much more consideration, and have challenging features. With these new tools, quantitative evaluation of DES control will have a wider prospect to offer to the real engineering problems.

2. Implementation of discrete-event control on a complex system requires modeling of the behavioral patterns of the actual system. Discrete event modeling and identification has utmost importance as the incorrect modeling may ultimately yield to a failure in control synthesis. Even worse, the control algorithms, even the optimal one, designed based on an incorrect or incomplete model may degrade the system performance. Therefore, modeling and model parameter identification techniques for the DES control design are proposed and tested specifically on a Segway RMP to show that the theoretical developments in this thesis provide implementable solutions to real-world applications.

3. The distance functions and metrics for anomaly measure are proposed and applied to synthesize event generation methods in DES control. In the case of failure to generate the events, the performance of DES controllers under partial observability should also be explored. In order to make sure the events are properly generated, we must have a good understanding of the physical dynamics associated with the events. For example, in order to properly generate the damage related events, it is necessary to understand the dynamics of the damage progress. The factors that are essential for event generation are:

   - Knowledge about the dynamics of the process associated with the events, and a good model, either a deterministic or stochastic model, to represent the progress.
   - Good sensor data that can capture the signal features which is the expression of the event in the physical progress.
   - Efficient signal processing tools to analyze the data.

   This research attempts to provide a seamless link between the continuously-varying dynamical systems and the discrete-event dynamics. Only after this problem is properly solved, the DES control could be implemented in the real systems.

4. There are only a few papers in technical literature reporting applications of DES control in complex dynamical processes (for example, see [52]). This

is clearly a major shortcoming of DES control. This research attempts to overcome this shortcoming by augmenting the theoretical work conducted in DES control with both simulation and laboratory experimentation. The proposed DES control system is tested within a hybrid and hierarchical architecture on computer simulations of integrated aircraft engine and flight models and experimented for autonomous robotic behaviors.

One of the major applications for simulation experimentation is dedicated to mission and operational management for integrated flight and propulsion control systems. Development of intelligent decision and control algorithms, based on the theory of Discrete Event Supervisory (DES) control has paramount importance in many military and commercial applications. Especially, since interactions among complex systems, such as propulsion and flight systems may not be adequately described solely through an understanding of individual component's dynamics, hierarchical hybrid architecture is employed for development of future generation control systems that will take advantage of these interactions.

The mission management and path planning of UAVs and the coordination of aerial and ground vehicles are also investigated. Although there is a certain trend for the autonomous control applications among the control community, the current practices do not embrace the DES control in the language theoretical framework. Moreover, our approach, which introduces possible intriguing virtues, tackles the problem of mission management in a different way from the established methods. Real-time control of multiple robotic devices and their coordination with the UAV system is also demonstrated in a networked robotics laboratory. These two systems forms a basic structure for a C^4ISR system composed of different unmanned vehicles to demonstrate the capabilities and advantages of our approach on a large scale dynamical system that consists of heterogeneous automata. In this research, the work is concentrated on different experimental setups of the DES control systems in the Complex Systems Simulation Laboratory and the Networked Robotic Systems Laboratory at The Pennsylvania State University [53].

Integration and real-time implementation of complex systems under the um-

brella of hybrid supervisory control scheme is established to demonstrate the practical initiative for overcoming the challenge of incorporating several disciplines such as systems theory, computer hardware and software, and domain knowledge of dynamical systems.

Based on the above details, the major contributions of this dissertation are outlined below:

1. *Development of optimal control algorithms for different global optimality criteria and their comparison.*

2. *Development of discrete-event modeling and model parameter identification techniques for complex dynamical systems.*

3. *Derivation of advanced analytical methods for event generation from sensory and non-sensory information.*

4. *Real-time implementation of hybrid supervisory control laws to complex dynamical systems such as coordinated unmanned vehicles and integrated aircraft flight and propulsion systems.*

1.4 Organization of the Dissertation

This thesis is organized in eight chapters including the present one and two appendices. After the introduction to the concepts and contributions of the thesis in the first chapter, Chapter 2 reviews the previous work on supervisory control, language measure and optimal control policy that is based on this measure. It provides the necessary background information for the topics that will be discussed in later chapters of the dissertation.

Modeling of hybrid supervisory systems is the first step towards providing a seamless link between the continuously-varying dynamical systems and the discrete event systems. Chapter 3 addresses the general modeling and identification of model parameters issues regarding the hybrid supervisory control in a robotic behavior application. Both discrete-event and continuous parameter identification is performed on a Segway Robotic Mobile Platform (RMP) to demonstrate the underlying concepts.

In order to achieve the desired performance of a Discrete Event Supervisory (DES) controller, it is essential to have an effective event generation algorithm to ensure fusion of the heterogeneous information. Chapter 4 presents the application of advanced pattern recognition techniques for event generation in complex dynamical systems.

Chapter 5 presents an application of the recently developed theory of language-measure-based discrete event supervisory (DES) control to aircraft propulsion systems. A two-layer hierarchical architecture is proposed to coordinate the operations of a twin-engine propulsion system. Integration of flight and propulsion control systems in aircraft is established in Chapter 6 for enhancement of performance and reliability. Since the dynamic couplings and nonlinear interactions of flight and propulsion systems are too complex for real-time computation, hierarchical hybrid (i.e., combined continuously-varying and discrete-event) architecture is employed for development of future generation control systems that will take advantage of these interactions for mission enhancement. Chapter 7 presents a hierarchical and hybrid architecture for control and coordination of future generation unmanned vehicles. Specifically, the coordination of unmanned aerial vehicles (UAV) and unmanned ground vehicles (UGV) to develop a command, control, computer, communication, intelligence, surveillance and reconnaissance (C^4ISR) system is demonstrated.

The dissertation is concluded in Chapter 8 and a road map for future research is drawn. For completeness of the dissertation, the mathematical models of simulated dynamics are presented in the appendices. The mathematical model describing the two-spool, low bypass turbofan engine is provided in detail in Appendix A. The engine model is simulated as a FORTRAN program and utilized in the simulation studies of Chapters 4, 5 and 6. Appendix B describes a generic, high-performance aircraft model, including detailed, full-envelope, nonlinear aerodynamics. The model, which is simulated as a FORTRAN program, is a collection of modules each performing a specific function and employed in the simulation studies of Chapter 6.

Chapter 2

Discrete Event Supervisory (DES) Control

2.1 Introduction

Discrete event systems belong to a special class of dynamical systems. The states of a discrete event system may take discrete (or symbolic) values and change only at (possibly asynchronous) discrete instants of time, in contrast to the familiar continuously varying dynamical systems of the physical world, which can be modeled by differential or difference equations. The dynamics of many human-engineered systems evolve asynchronously in time via complex interactions of various discrete-valued events with continuously varying physical processes. The relatively young discipline of discrete event systems has undergone rapid growth over the last two decades with the evolution of human-engineered complex systems, such as integrated control and communication systems, distributed sensing and monitoring of large-scale engineering systems, manufacturing and production systems, software fault management, and military Command, Control, Computer, Communication, Intelligence, Surveillance, and Reconnaissance (C^4ISR) systems.

The discipline of discrete event systems was initiated with simulation of human-engineered processes about four decades ago in the middle of nineteen sixties. Theoretical concepts of formal languages and automata theory was introduced by computer scientists and control theorists for modeling of discrete event systems. In

the late nineteen sixties, Arbib et. al [54] showed how algebraic methods could be used to explore the structure of finite automata to model dynamical systems. In the meantime, computer scientists focused on formal languages, automata theory, and computational complexity for application of language-theoretic concepts (e.g., regular expressions and context-free grammars) in software development including design of compilers and text processors [14]. In nineteen eighties, Ho et. al introduced the concept of finite perturbation in discrete event systems for modeling and analysis of human-engineered systems [55].

This chapter reviews the previous work on supervisory control, language measure and optimal control policy that is based on this measure with no event disabling penalty [56]. It provides the background information for the topics that will be discussed in this dissertation. It is also necessary to provide the fundamentals of developing a performance index and the optimal DES control law, because the theoretical performance of DES controllers will be associated with language measure through out this thesis.

2.2 Basics of Supervisory Control

The concept of discrete event supervisory (DES) control was first introduced in the seminal paper of Ramadge and Wonham [15] and this important paradigm has been subsequently extended by other researchers (for example, see citations in Chapter 1). These efforts have led to the evolution of a new discipline in decision and control, called Supervisory Control Theory (SCT), that requires partitioning the discrete event behavior of a physical process, called the plant, into legal and illegal sub-behaviors. The legal behavior of the plant dynamics is modeled by a deterministic finite-state automaton, abbreviated as DFSA in the sequel. The DFSA model is equivalent to a regular language that is built upon an alphabet of finitely many events; the event alphabet is partitioned into subsets of controllable events (that can be disabled) and uncontrollable events (that cannot be disabled). Based on the regular language of an unsupervised plant, SCT synthesizes a DES controller as another regular language, having the common alphabet with the plant language, that guarantees a (restricted) legal behavior of the controlled plant satisfying desired specifications. Instead of continuously handling numerical data, DES

controllers are designed to process event strings to disable certain controllable events in the physical plant. A number of algorithms for DES control synthesis have evolved based on the automata theory and formal languages relying on the disciplines of Computer Science and Control Science.

For a discrete-event system that is modeled by a finite state automata G with $\Sigma = \Sigma_c \cup \Sigma_u$ is the partition into two mutually exclusive sets, the controllable event set Σ_c and uncontrollable event set Σ_u, a supervisory control is any map $S : L(G) \rightarrow \Gamma$, where Γ is the set of all control patterns $\Gamma = \{\gamma \in 2^{\Sigma} | \gamma \supseteq \Sigma_u\}$. The pair (G, S) is denoted as S/G to suggest '*G under the supervision of S*'. The closed behavior of S/G is defined to be the language $L(S/G) \subseteq L(G)$ described as follows.

1. $\epsilon \in L(S/G)$
2. If $s \in L(S/G)$, $\sigma \in S(s)$, and $s\sigma \in L(G)$ then $s\sigma \in L(S/G)$

The marked behavior of S/G is

$$L_m(S/G) = L(S/G) \cap L_m(G)$$

A supervisor S is nonblocking if $pr(S/G) = L(S/G)$, where the prefix closure of a language K, denoted $pr(K) \subseteq \Sigma^*$ is the language

$$pr(K) = \{s \in \Sigma^* | \exists u \in K, t \in \Sigma^*, \text{such that } u = st\}$$

A language $K \subseteq \Sigma^*$ is controllable with respect to G and Σ_u if

$$\forall s \in pr(K), \forall \sigma \in \Sigma_u, s\sigma \in L(G) \Rightarrow s\sigma \in pr(K)$$

Then K is controllable if and only if

$$pr(K)\Sigma_u \cap L(G) \subseteq pr(K)$$

Theorem 2.2.1: Let $K \subseteq L_m(G)$ be nonempty. There exists a nonblocking supervisory control S for G such that $L_m(S/G) = K$ if and only if K is controllable.
Proof: The proof is given in pages 66-67 of [25].

Theorem 2.2.2: Let $K \subseteq L(G)$ be nonempty and prefix closed. There exists a nonblocking supervisory control S for G such that $L(S/G) = K$ if and only if K is controllable.

Proof: The proof is given in page 68 of [25].

In general, a supervised plant DFSA is synthesized as a parallel composition of the unsupervised plant DFSA G and a supervisor DFSA S. The supervised plant DFSA S/G yields a sublanguage of the unsupervised plant language, which enables restricted legal behavior of the supervised plant [15][16]. However, there have been no quantitative methods for evaluating the performance of supervisory controllers or the supervisor(s) that satisfies the design requirements.

2.3 Total Ordering of Regular Languages

The concept of permissiveness has been used in DES control literature [34][25] to facilitate qualitative comparison of DES controllers under the language controllability condition. Design of maximally permissive DES controllers has been proposed by many researchers based on different conditions. However, maximal permissiveness does not necessarily imply best performance of the supervised plant from the perspectives of plant operational objectives, because no quantitative measure of performance is addressed in this type of supervisor design. Hence, permissiveness is not an adequate measure of performance.

The controlled behavior of a given plant DFSA under different supervisors could vary if they are designed to meet different control specifications. As such, the respective controlled sublanguages of the plant language form a partially ordered set that is not necessarily totally ordered. Since the literature on DES control does not apparently provide a language measure, it may not be possible to quantitatively evaluate the performance of a DES controller. Therefore, it is necessary to formulate a mathematically rigorous concept of language measure(s) to quantify performance of individual supervisors such that the measures of controlled plant behavior, described by a partially ordered set of controlled sublanguages, can be structured to form a totally ordered set. From this perspective, it is necessary to formulate a signed real measure μ [44] that can be assigned to sublanguages of the regular language of the unsupervised plant (see Equation 1.2) to achieve the

following objective:

> *Given that the relation $\subseteq$ induces a partial ordering on a set of controlled sublanguages $\{L(S^j/G), j = 1, \ldots, N\}$ of the plant language $L(G)$ under supervisors whose languages are $\{L(S^j), j = 1, \ldots, N\}$, the language measure μ induces a total ordering $\leq$ on $\{\mu(L(S^j/G))\}$.*

The major motivation here is to present a signed real measure of regular languages, which can be used for quantitative analysis and synthesis of DES control systems for physical plants, instead of relying on permissiveness as the (qualitative) performance index. Construction of the proposed language measure follows Myhill-Nerode Theorem [14] and Hahn Decomposition [57], where a state-based partitioning of the (unsupervised) plant language yields equivalence classes of finite-length event strings. Each marked state is characterized by a signed real value that is chosen based on the designer's perception of the state's impact on the system performance. Conceptually similar to the conditional probability, each event is assigned a cost based on the state at which it is generated. This procedure permits a string of events, terminating on a good (bad) marked state, to have a positive (negative) measure; each of the remaining strings of events in the language, terminating on an unmarked state, is assigned zero measure. In this setting, a supervisor can be designed with the goal of eliminating bad strings and retaining good strings by disabling controllable event(s) at selected states. Different supervisors may attempt to achieve this goal in different ways and generate a partially ordered set of controlled languages. The language measure then creates a total ordering on the performance of the controlled languages, which provides a precise quantitative comparison of the controlled plant behavior under different supervisors.

The language measure has been used as a performance index for analysis and synthesis of DES control systems. For example, Fu et al. have proposed optimal control [30] and robust optimal control [58] of regular languages where a state-based optimal control policy is obtained by selectively disabling controllable events to maximize the measure of the controlled plant language. The concept of DES control has been validated by laboratory experimentation on mobile robots [59] as well as on simulation test beds for applications to gas turbine engines [60], software fault management and control of malicious executables in computers [61].

2.4 Brief Review of Language Measure Concept

Let the dynamical behavior of a physical plant be modeled as a deterministic finite state automaton (DFSA) $G_i = (Q, \Sigma, \delta, q_i, Q_m)$, where Q is the finite set of states q_j with $|Q| = n$ and $q_i \in Q$ is the initial state; Σ is the (finite) alphabet of events with $|\Sigma| = m$; Σ^* is the set of all finite-length strings of events including the empty string ε. The (possibly partial) function $\delta : Q \times \Sigma \to Q$ represents state transitions and $\hat{\delta} : Q \times \Sigma^* \to Q$ is an extension of δ and $Q_m \subseteq Q$ is the set of marked (also known as accepted) states which have some importance (positive or negative) for the DFSA model.

Definition 2.4.1: A DFSA G_i, initialized at $q_i \in Q$, generates the language $L(G_i) \equiv \{s \in \Sigma^* : \hat{\delta}(q_i, s) \in Q\}$ and its marked (or accepted) sublanguage $L_m(G_i) \equiv \{s \in \Sigma^* : \hat{\delta}(q_i, s) \in Q_m\}$.

Following [27][28], it is possible to construct a signed real measure $\mu : 2^{\Sigma^*} \to \mathbb{R} \equiv (-\infty, \infty)$ that allows quantitative evaluation of every event string $s \in \Sigma^*$ based on state-based decomposition of $L(G_i)$ into null (i.e., $L^0(G_i)$), positive (i.e., $L^+(G_i)$), and negative (i.e., $L^-(G_i)$) sublanguages of $L(G_i)$.

Definition 2.4.2: The language of all strings that, starting at $q_i \in Q$, terminates at $q_j \in Q$, is denoted as: $L(q_i, q_j)$. That is, $L(q_i, q_j) \equiv \{s \in L(G_i) : \hat{\delta}(q_i, s) = q_j\}$.

Definition 2.4.3: The characteristic function that assigns a signed real weight to each state q_j is defined as: $\chi : Q \to [-1, 1]$ such that

$$\chi_j \equiv \chi(q_j) \in \begin{cases} [-1, 0), & q_j \in Q_m^- \\ 0, & q_j \notin Q_m \\ (0, 1], & q_j \in Q_m^+ \end{cases} \tag{2.1}$$

The $(n \times 1)$ characteristic vector is denoted as:

$$\bar{\chi} \equiv [\ \chi_1 \quad \chi_2 \quad \cdots \quad \chi_n\ ]^T$$

Definition 2.4.4: The event cost is conditioned on a DFSA state at which the event is generated, and is defined as: $\tilde{\pi} : \Sigma^* \times Q \to [0, 1]$ such that $\forall q_j \in Q, \forall \sigma_k \in \Sigma, \forall s \in \Sigma^*$,

- $\tilde{\pi}[\sigma_k|q_j] \equiv \tilde{\pi}_{jk} \in [0,1); \qquad \sum_k \tilde{\pi}_{jk} < 1;$
- $\tilde{\pi}[\sigma_k|q_j] = 0$ if $\delta(q_j,\sigma_k)$ is undefined; $\qquad \tilde{\pi}[\varepsilon|q_j] = 1;$
- $\tilde{\pi}[\sigma_k s|q_j] = \tilde{\pi}[\sigma_k|q_j]\tilde{\pi}[s|\delta(q_j,\sigma_k)]$

The $(n \times m)$ event cost matrix is denoted as: $\tilde{\mathbf{\Pi}} \equiv [\tilde{\pi}_{ij}]$.

Definition 2.4.5: The state transition cost of the DFSA is defined as a function $\pi : Q \times Q \to [0,1)$ such that $\forall q_j, q_k \in Q$

$$\pi(q_k|q_j) = \sum_{\sigma\in\Sigma:\delta(q_j,\sigma)=q_k} \tilde{\pi}(\sigma|q_j) \equiv \pi_{jk} \tag{2.2}$$

and $\pi_{jk} = 0$ if $\{\sigma \in \Sigma : \delta(q_j,\sigma)\} = \varnothing$. The $(n \times n)$ state transition cost matrix, denoted as $\mathbf{\Pi}$-matrix, is defined as:

$$\mathbf{\Pi} = \begin{bmatrix} \pi_{11} & \pi_{12} & \cdots & \pi_{1n} \\ \pi_{21} & \pi_{22} & \cdots & \pi_{2n} \\ \vdots & & \ddots & \vdots \\ \pi_{n1} & \pi_{n2} & \cdots & \pi_{nn} \end{bmatrix}.$$

Now we define the measure of any sublanguage of the plant language $L(G_i)$ in terms of the signed characteristic function χ and the non-negative event cost $\tilde{\pi}$.

Definition 2.4.6: The signed real measure μ of a singleton string set $\{s\}$ is defined as: $\mu(\{s\}) \equiv \chi(q_j)\tilde{\pi}(s|q_i)\ \forall s \in L(q_i,q_j) \subseteq L(G_i)$.

The signed real measure of is defined as:

$$\mu(L(q_i,q_j)) \equiv \sum_{s\in L(q_i,q_j)} \mu(\{s\}) \tag{2.3}$$

The signed real measure of a DFSA G_i, initialized at the state $q_i \in Q$, is defined as:

$$\mu_i \equiv \mu(L(G_i)) = \sum_j \mu(L(q_i,q_j)) \tag{2.4}$$

The $(n \times 1)$ real signed measure vector is denoted as:

$$\bar{\mu} \equiv \begin{bmatrix} \mu_1 & \mu_2 & \cdots & \mu_n \end{bmatrix}^T$$

Theorem 2.4.1: The set of symbolic equation for a regular language can be written as:

$$L(G_i) \equiv L_i = \sum_j \sigma_i^j L_j + \varepsilon_i \tag{2.5}$$

where $\sigma_i^j \equiv \sigma \in \Sigma : \delta(q_i, \sigma) = q_j$. The measure of the language $L(G_i)$, where $G_i = (Q, \Sigma, \delta, q_i, Q_m)$ can be expressed as: $\mu_i = \sum_i \pi_{ij}\mu_j + \chi_i$. Equivalently, in vector notation: $\bar{\mu} = \mathbf{\Pi}\bar{\mu} + \bar{\chi}$. Since $\mathbf{\Pi}$ is a contraction operator, the measure vector $\bar{\mu}$ is uniquely determined as:

$$\bar{\mu} = [I - \mathbf{\Pi}]^{-1}\bar{\chi} \tag{2.6}$$

Proof: The proof is given in [27] and [28].

In the formulation of the optimal control policy for a plant automaton G, the performance index vector is defined as the measure vector $\bar{\mu}^S \equiv [I - \mathbf{\Pi}^S]^{-1}\bar{\chi}$ of the controlled plant sublanguage $L(S/G)$ controlled under the supervisor S without incorporating the cost of event disabling.

2.5 Renormalization of Language Measure

The previously introduced concept of language measure provides a state based evaluation for DFSA model of physical plant. The elements of the measure vector are associated with the performance of individual states. It can be concluded that the overall efficacy of the DFSA is directly related to that of the starting state of the DFSA. However, it is imperative to determine the average effectiveness of the DFSA and to estimate the long term behavioral performance of the supervisor.

In the calculation of the language measure, $\mathbf{\Pi}$-matrix is similar to the state transition probability matrix of a Markov Chain with the exception that $\sum_k \tilde{\pi}_{jk} = 1$ condition is replaced by $\sum_k \tilde{\pi}_{jk} < 1$. The residue $\theta_j = 1 - \sum_k \tilde{\pi}_{jk}$ denotes the probability of transition from q_j to q_{n+1}, i.e. the probability of unmodeled dynamics of the physical plant. Since the plant model is an inexact representation of the physical plant, there exist unmodeled dynamics to be accounted for. This phenomenon manifests itself either as unmodeled events that may occur at each state or as unaccounted state(s) in the model. A new unmarked state q_{n+1}, called

the dump state [15], is created and the transition function δ is extended.

Before proceeding to the renormalization of the language measure and the imminent results of this procedure, it is essential to introduce several properties of stochastic matrices. Let $\mathbf{P}$ be the state transition probability matrix of a stationary Markov chain with finitely many states. Then, $\mathbf{P}$ is a stochastic (i.e., non-negative with each row sum being identically equal to unity) matrix [62].

Definition 2.5.1: A stochastic matrix $\mathbf{P}$ is irreducible and aperiodic, also called *primitive*, if $\exists\ k \in \mathbb{N}$ such that $\mathbf{P}^k > 0$ [62].

Proposition 2.5.1: For any stochastic matrix $\mathbf{P}$, the following limit exists:

$$\lim_{k\to\infty} \frac{1}{k} \sum_{j=0}^{k-1} P^j \to \mathcal{P} \tag{2.7}$$

where $\mathcal{P}$ is a stochastic matrix. Furthermore, $\mathcal{P}$ commutes with $\mathbf{P}$ and idempotency property holds. That is,

$$\mathcal{P}\mathbf{P} = \mathbf{P}\mathcal{P} = \mathcal{P} = \mathcal{P}^2. \tag{2.8}$$

Proof: The proof is given in pages 50-51 of [62].

Proposition 2.5.2: For any irreducible and aperiodic stochastic matrix $\mathbf{P}$, the following limit exists:

$$\lim_{k\to\infty} \mathbf{P}^k \to \mathcal{P} \tag{2.9}$$

where P is primitive. In this case, each row of $\mathcal{P}$ is identically equal to the unique probability vector $\bar{p}^T$ that is a fixed point of the operator $\mathbf{P}$, i.e., $\bar{p}^T\mathbf{P} = \bar{p}^T$.
Proof: The proof is given in pages 49-50 of [62].

Let $\mathbf{P}$ be a stochastic matrix where each row sum is equal to unity and each element of $\mathbf{P}$ is non-negative, implying that the induced sup norm $\|\mathbf{P}\|_\infty = 1$. The process dynamics can be defined in terms of θ and $\mathbf{P}$ as:

$$\mathbf{\Pi}(\theta) \equiv (1-\theta)\mathbf{P}. \tag{2.10}$$

The induced sup norm $\|\mathbf{\Pi}(\theta)\|_\infty = (1-\theta)$. The language measure can be parameterized in θ as:

$$\bar{\mu}(\theta) = [I - \mathbf{\Pi}(\theta)]^{-1}\bar{\chi} \tag{2.11}$$

where $\bar{\chi} \in [-1, 1]^n$ is the normalized state weight vector for the DFSA. Since $\mathbf{\Pi}(\theta)$

is a linear continuous operator in space $\mathbb{R}^n$, the following inequality holds:

$$\|[I - \mathbf{\Pi}(\theta)]^{-1}\|_\infty \leq \theta^{-1} \tag{2.12}$$

Furthermore, since $\|\bar{\chi}\|_\infty \leq 1$, it follows from equations (2.10) and (2.12) that

$$\|\bar{\mu}(\theta)\|_\infty \leq \theta^{-1} \ \forall \ \theta \in (0,1). \tag{2.13}$$

The universal set of the regular languages is Σ^* which can be realized by the single state automaton. In this case, the state transition matrix degenerates to scalar $[1-\theta]$ and the state weight vector to scalar $\chi = 1$. This implies that $\mathbf{P}$ reduces to unity and the measure of Σ^* becomes θ^{-1}.

Hence, the measure vector, given in Equation 2.6, can be renormalized with respect to the measure of Σ^* and the normalized measure of regular languages is defined as:

$$\bar{\mu}^{norm}(\theta) = \frac{\bar{\mu}(\theta)}{\mu(\Sigma^*)} = \theta[I - \mathbf{\Pi}(\theta)]^{-1}\bar{\chi} \tag{2.14}$$

Proposition 2.5.3: The limit of renormalized measure vector at $\theta \to 0^+$ exists and the infinity norm is bounded by unity, i.e.

$$\bar{\mu}^{norm}(0) \equiv \lim_{\theta \to 0^+} \bar{\mu}^{norm}(\theta) \tag{2.15}$$

$$\|\bar{\mu}^{norm}(0)\|_\infty \leq 1 \tag{2.16}$$

Proof: The proof is given in [63].

Proposition 2.5.4: Let $\mathbf{P}$ be the transition matrix of a finite Markov chain (or equivalently a regular language). Then, as the parameter $\theta \to 0^+$, the limiting renormalized measure vector is obtained as:

$$\bar{\mu}^{norm}(0) = \mathcal{P}\bar{\chi} \tag{2.17}$$

where the matrix operator $\mathcal{P} \equiv \lim_{k\to\infty} \frac{1}{k} \sum_{j=0}^{k-1} P^j$ is defined in equation (2.7). The expression $\mathcal{P}\bar{\mu}^{norm}(\theta)$ is independent of θ and the following equality holds $\forall \ \theta \in (0,1)$:

$$\mathcal{P}\bar{\mu}^{norm}(\theta) = \mathcal{P}\bar{\chi} \tag{2.18}$$

Proof: The proof is given in [63].

The following results hold specifically for primitive matrices. In this case, $\mathbf{P}$ has exactly one unity eigenvalue and the remaining eigenvalues are located within the unit circle with its center at the origin.

Proposition 2.5.5: Let $\bar{p}^T$ be the ℓ_1-normalized left eigenvector of $\mathbf{P}$ corresponding to the unique unity eigenvalue and let $\bar{e} \equiv [\ 1 \quad 1 \quad \cdots \quad 1\ ]^T \in \mathbb{R}^n$. The renormalized measure vector is given by the following expression:

$$\bar{\mu}^{norm}(0) = \bar{e}\mu^{ave} \tag{2.19}$$

where the scalar μ^{ave} is the expected average measure of the language and computed as:

$$\mu^{ave} = \bar{p}^T\bar{\chi}. \tag{2.20}$$

Similarly, the expression $\bar{p}^T\bar{\mu}^{norm}(\theta)$ is independent of θ and the following equality holds $\forall\ \theta \in (0,1)$:

$$\bar{p}^T\bar{\mu}^{norm}(\theta) = \bar{p}^T\bar{\chi}. \tag{2.21}$$

Proof: The proof is given in [63].

Scalar-valued expected average measure μ^{ave} of the language represents the expected long-term behavior of plant dynamics in terms of assigned state weights $\bar{\chi}$ and probability distribution of DFSA states p.

The computation of average measure μ^{ave} of the language can be extended to infinite dimensional Banach spaces as:

$$\mu^{ave} = \langle \bar{p}, \bar{\chi} \rangle. \tag{2.22}$$

2.6 Review of Optimal DES control

The performance index of the optimal control policy is the signed language measure of the supervised sublanguage, which is expressed in terms of a state transition cost matrix $\mathbf{\Pi}$ and a characteristic vector $\bar{\chi}$ that are parameters for the language measure introduced in Section 2.4.

The state-based optimal control policy is obtained by selectively disabling con-

trollable events to maximize the measure of the controlled plant language without any constraints. In each iteration, the optimal control algorithm attempts to disable all controllable events leading to "bad" states and enable all controllable events leading to "good" states. It has been also shown in [30][12] that computational complexity of the control synthesis is polynomial in the number of plant states.

Let $\mathcal{S} \equiv \{S_0, S_1, \ldots, S_n\}$ be a finite set of all supervisory control policies for the unsupervised plant automaton G where S_0 is the null controller (i.e., no event being disabled) implying that $L(S_0/G) = L(G)$. Therefore, $\mathbf{\Pi}(S_0) \equiv \mathbf{\Pi}^0 = \mathbf{\Pi}^{plant}$, i.e., the state transition cost matrix $\mathbf{\Pi}^0$ is equal to the $\mathbf{\Pi}$ matrix of unsupervised plant automaton G. For a supervisor S_k, $k \in 1, 2, \ldots, n$, the control policy selectively disables certain controllable events, and therefore the following elementwise inequality holds: $\mathbf{\Pi}^k \equiv \mathbf{\Pi}(S_k) \leq \mathbf{\Pi}^0$ and $L(S_k/G) \subseteq L(G)\ \forall S_k \in \mathcal{S}$.

Definition 2.6.1: For any supervisor $S_k \in \mathcal{S}$ and any language measure vector $\bar{\nu} \in \mathbb{R}^n$, the affine operator $T(S_k) : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is defined as: $T(S_k)\bar{\nu} = \mathbf{\Pi}(S_k)\bar{\nu} + \bar{\chi}$.

2.6.1 Absolute Disabling

The following propositions that are proved in [30] are restated for completeness of the work reported in this thesis.

Proposition 2.6.1: $\forall S_k \in \mathcal{S}$, $T(S_k)$ is a contraction operator and there exists a unique measure vector $\bar{\mu}(S_k)$ such that $\bar{\mu}(S_k) = T(S_k)\bar{\mu}(S_k)$. The fixed point of the contraction operator $T(S_k)$ is: $\bar{\mu}^k = [I - \mathbf{\Pi}^k]^{-1}\bar{\chi}$ where $\bar{\mu}^k \equiv \bar{\mu}(S_k)$ and $\mathbf{\Pi}^k \equiv \mathbf{\Pi}(S_k)$. Furthermore, the operator $[I - \mathbf{\Pi}^k]^{-1}$ has a real positive determinant.
Proof: The proof is given in [30].

Proposition 2.6.2: If $\mu_j^k = \min_{\ell \in 1,2,\cdots,n} \mu_\ell^k$. If $\mu_j^k \leq 0$ then $\chi_j \leq 0$; and if $\mu_j^k < 0$ then $\chi_j < 0$. Likewise, if $\mu_j^k = \max_{\ell \in 1,2,\cdots,n} \mu_\ell^k$. If $\mu_j^k \geq 0$ then $\chi_j \geq 0$; and if $\mu_j^k > 0$ then $\chi_j > 0$.
Proof: The proof is given in [30].

Proposition 2.6.3: Given $\mathbf{\Pi}(S_0) \equiv \mathbf{\Pi}^0 = \mathbf{\Pi}^{plant}$ and $\bar{\mu}^k = [I - \mathbf{\Pi}^k]^{-1}\bar{\chi}$, let $\mathbf{\Pi}^{k+1}$ be generated from $\mathbf{\Pi}^k$ for as follows: $\forall i, j \in 1, 2, \cdots, n$, the ij^{th} element of

$\mathbf{\Pi}^{k+1}$ is modified as:

$$\pi_{ij}^{k+1} \begin{cases} \geq \pi_{ij}^{k}, & \mu_j^k > 0 \\ = \pi_{ij}^{k}, & \mu_j^k = 0 \\ \leq \pi_{ij}^{k}, & \mu_j^k < 0 \end{cases} \tag{2.23}$$

and $\mathbf{\Pi}^k \leq \mathbf{\Pi}^0 \ \forall k$.

Then, $\mu^{k+1} \geq \mu^k$ elementwise and equality holds if and only if $\mathbf{\Pi}^{k+1} = \mathbf{\Pi}^k$. Moreover, if $\mu_j^k \leq 0$ and $\mathbf{\Pi}^{k+1}$ is generated from $\mathbf{\Pi}^k$ by disabling controllable events that lead to state q_j, then $\mu_j^{k+1} < 0$.

Proof: The proof is given in [30].

Proposition 2.6.4: Iteration of the algorithm in Proposition 2.6.3 leads to an optimal state transition cost matrix $\mathbf{\Pi}^{\Diamond}$ that maximizes the performance vector $\bar{\mu}^{\Diamond} = [I - \mathbf{\Pi}^{\Diamond}]^{-1}\bar{\chi}$ elementwise.

Proof: The proof is given in [30].

Proposition 2.6.5: The control policy induced by $\mathbf{\Pi}^{\Diamond}$ the matrix is unique in the sense that the controlled language is most permissive among all controller(s) having the best performance.

Proof: The proof is given in [30].

The algorithm for synthesis of the optimal control policy, which is developed based on the above propositions, is summarized as follows: Let G be the DFSA plant model without any constraint of operational specifications. Let the state transition cost matrix of the unsupervised plant be: $\mathbf{\Pi}^{plant} \in \mathbb{R}^{n \times n}$ and the characteristic vector be: $\chi_j \in [-1, 1] \ \forall j \in 1, 2, \cdots, n$. Starting with $k = 0$ and $\mathbf{\Pi}^0 \equiv \mathbf{\Pi}^{plant}$, the control policy is constructed by Algorithm 1.

At this stage, the optimal value of the performance cost, which is the language measure, is

$$\bar{\mu}^{\Diamond} = [I - \mathbf{\Pi}^{\Diamond}]^{-1}\bar{\chi} \tag{2.24}$$

This control strategy is named as *Absolute Disabling* due to the nature of the disabling criteria that leads to the optimal control policy. The controllable events that lead to states with $\mu_j^k < 0$ are to be disabled at iteration k to maximize the targeted optimization index, signed measure of the unsupervised plant language. The strategy is described as follows:

Let q_i and q_j be two states of a DFSA and σ_ℓ denotes a controllable

Algorithm 1 Computation of $\mathbf{\Pi}^{\Diamond}$

Input: $\mathbf{\Pi}^0$, $\tilde{\mathbf{\Pi}}^0$ and $\bar{\chi}$
Output: $\mathbf{\Pi}^{\Diamond}$ such that $\bar{\mu}^{\Diamond} = [I - \mathbf{\Pi}^{\Diamond}]^{-1}\bar{\chi}$

$n = sizeof(\mathbf{\Pi}^0)$ // number of states
$\bar{\mu}^0 = [I - \mathbf{\Pi}^0]^{-1}\bar{\chi}$
begin Step 1
$k = 0$
for $j = 0$ to n **do**
  if $\mu_j^0 < 0$ **then** // criterion for disabling
    for all $\sigma_\ell : \delta(q_i, \sigma_\ell) = q_j$ controllable **do**
      Disable σ_ℓ
      $\Delta_{ij}^0 = \tilde{\pi}_{\ell j}^0$
    end for
  end if
end for
$\mathbf{\Pi}^1 = \mathbf{\Pi}^0 - \mathbf{\Delta}^0$ where $\mathbf{\Delta}^0 \geq 0$
$\bar{\mu}^1 = [I - \mathbf{\Pi}^1]^{-1}\bar{\chi}$
begin Step 2
$k = 1$
while $\bar{\mu}^k > \bar{\mu}^{k-1}$ **do**
  for $j = 0$ to n **do**
    if $\mu_j^k \geq 0$ **then** // criterion for enabling
      for all $\sigma_\ell : \delta(q_i, \sigma_\ell) = q_j$ controllable **do**
        Enable σ_ℓ
        $\Delta_{ij}^k = \tilde{\pi}_{\ell j}^k$
      end for
    end if
  end for
  $\mathbf{\Pi}^{k+1} = \mathbf{\Pi}^k + \mathbf{\Delta}^k$ where $\mathbf{\Delta}^k \geq 0$
  $\bar{\mu}^{k+1} = [I - \mathbf{\Pi}^{k+1}]^{-1}\bar{\chi}$
  $k = k + 1$
end while

event connecting state q_i to q_j with the event cost $\tilde{\pi}_{i\ell}$. Let $\tilde{\pi}_{i\ell}$ be subtracted from the state transition cost π_{ij} for disabling and added to the state transition cost π_{ij} for enabling. Remaining transition costs doesn't change. *The measure of the language initiated at state q_i increases at iteration k (i.e. $\mu_i^{k+1} > \mu_i^k$) by disabling a controllable event σ_ℓ if and only if $\mu_j^k < 0$, and increases or remains same at iteration k (i.e. $\mu_i^{k+1} \geq \mu_i^k$) by enabling a controllable event σ_ℓ if and only if $\mu_j^k \geq 0$.*

2.6.2 Relative Disabling

One important shortcoming of the previous strategy is regarding the row sums of the state transition cost matrix $\mathbf{\Pi}$. The strategy assumes that the cost of a disabled event simply disappears from $\mathbf{\Pi}$-matrix. Although this assumption is mathematically valid, since $\sum_k \tilde{\pi}_{jk} < 1$ condition and properties of the $\mathbf{\Pi}$-matrix as a contraction operator still holds, the probabilistic interpretation of the process as a Markov chain fails. Therefore, it is possible to enhance *Absolute Disabling* if we can preserve the row sums of the $\mathbf{\Pi}$-matrix. The rationale is to treat the supervisor as an inhibitor on plant dynamics. When the plant wants to execute an event that will cause a transition from state q_i to state q_j, if the supervisor disables that event, the plant cannot go to state q_j, but stays in state q_i. Hence, the cost of the disabled event is transferred to a self loop.

The following propositions are slightly modified versions of previously stated propositions 2.6.3-2.6.5.

Proposition 2.6.6: Given $\mathbf{\Pi}(S_0) \equiv \mathbf{\Pi}^0 = \mathbf{\Pi}^{plant}$ and $\bar{\mu}^k = [I - \mathbf{\Pi}^k]^{-1}\bar{\chi}$, let $\mathbf{\Pi}^{k+1}$ be generated from $\mathbf{\Pi}^k$ for as follows: $\forall i, j \in 1, 2, \cdots, n$, the ij^{th} element of $\mathbf{\Pi}^{k+1}$ is modified as:

$$\pi_{ij}^{k+1} \begin{cases} \geq \pi_{ij}^k, & \mu_j^k > \mu_i^k \\ = \pi_{ij}^k, & \mu_j^k = \mu_i^k \\ \leq \pi_{ij}^k, & \mu_j^k < \mu_i^k \end{cases} \tag{2.25}$$

and

$$\pi_{ii}^{k+1} \begin{cases} \leq \pi_{ii}^k, & \mu_j^k > \mu_i^k \\ = \pi_{ii}^k, & \mu_j^k = \mu_i^k \\ \geq \pi_{ii}^k, & \mu_j^k < \mu_i^k \end{cases} \tag{2.26}$$

Then, $\mu^{k+1} \geq \mu^k$ elementwise and equality holds if and only if $\mathbf{\Pi}^{k+1} = \mathbf{\Pi}^k$. Moreover, if $\mu_j^k \leq \mu_i^k$ and $\mathbf{\Pi}^{k+1}$ is generated from $\mathbf{\Pi}^k$ by disabling controllable events that lead to state q_j, then $\mu_j^{k+1} < \mu_i^k$.

Proof: The proof exactly follows the construct of [30] except that both i^{th} and j^{th} columns of the $[I - \mathbf{\Pi}^{k+1}]$ matrix are different from those of $[I - \mathbf{\Pi}^k]$.

Proposition 2.6.7: Iteration of the algorithm in Proposition 2.6.6 leads to an optimal state transition cost matrix $\mathbf{\Pi}^{\blacklozenge}$ that maximizes the performance vector $\bar{\mu}^{\blacklozenge} = [I - \mathbf{\Pi}^{\blacklozenge}]^{-1}\bar{\chi}$ elementwise.

Proof: The proof exactly follows the proof in [30] except that $(-\sum_i \mathrm{Col}_i \cdot \mu_i^{\blacklozenge}) \leq 0$ term is added to $\sum_j \mathrm{Col}_j \cdot \mu_j^{\blacklozenge}$ term, therefore the inequalities are preserved.

Proposition 2.6.8: The control policy induced by $\mathbf{\Pi}^{\blacklozenge}$ the matrix is unique in the sense that the controlled language is most permissive among all controller(s) having the best performance.

Proof: The proof follows the proof in [30] except that relative performances of the states are compared instead of comparing the performances of the states with respect to 0.

The algorithm for synthesis of the optimal control policy, which is developed based on the above propositions, is summarized as follows: Let G be the DFSA plant model without any constraint of operational specifications. Let the state transition cost matrix of the unsupervised plant be: $\mathbf{\Pi}^{plant} \in \mathbb{R}^{n \times n}$ and the characteristic vector be: $\chi_j \in [-1, 1]\ \forall j \in 1, 2, \cdots, n$. Starting with $k = 0$ and $\mathbf{\Pi}^0 \equiv \mathbf{\Pi}^{plant}$, the control policy is constructed by Algorithm 2.

At this stage, the optimal value of the performance cost, which is the language measure, is

$$\bar{\mu}^{\blacklozenge} = [I - \mathbf{\Pi}^{\blacklozenge}]^{-1} \bar{\chi} \tag{2.27}$$

This control strategy is named as *Relative Disabling* due to the nature of the disabling criteria that leads to the optimal control policy. The controllable events that satisfy the condition $\mu_j^k < \mu_i^k$ are to be disabled at iteration k to maximize the targeted optimization index, signed measure of the unsupervised plant language. The strategy is described as follows:

> Let q_i and q_j be two states of a DFSA and σ_ℓ denotes a controllable event connecting state q_i to q_j with the event cost $\tilde{\pi}_{i\ell}$. Let $\tilde{\pi}_{i\ell}$ be subtracted from the state transition cost π_{ij} and added to π_{ii} for disabling and subtracted from the state transition cost π_{ii} and added to π_{ij} for enabling. Remaining transition costs doesn't change. *The measure of the language initiated at state q_i increases at iteration k (i.e. $\mu_i^{k+1} > \mu_i^k$) by disabling a controllable event σ_ℓ if and only if $\mu_j^k < \mu_i^k$, and increases or remains same at iteration k (i.e. $\mu_i^{k+1} \geq \mu_i^k$) by enabling a controllable event σ_ℓ if and only if $\mu_j^k \geq \mu_i^k$.*

Algorithm 2 Computation of $\mathbf{\Pi}^{\blacklozenge}$

Input: $\mathbf{\Pi}^0$, $\tilde{\mathbf{\Pi}}^0$ and $\bar{\chi}$
Output: $\mathbf{\Pi}^{\blacklozenge}$ such that $\bar{\mu}^{\blacklozenge} = [I - \mathbf{\Pi}^{\blacklozenge}]^{-1}\bar{\chi}$

$n = sizeof(\mathbf{\Pi}^0)$ // number of states
$\bar{\mu}^0 = [I - \mathbf{\Pi}^0]^{-1}\bar{\chi}$
begin Step 1
$k = 0$
for $j = 0$ to n **do**
  for $i = 0$ to n **do**
    if $\mu_j^0 < \mu_i^0$ **then** // criterion for disabling
      for all $\sigma_\ell : \delta(q_i, \sigma_\ell) = q_j$ controllable **do**
        Disable σ_ℓ
        $\Delta_{ij}^0 = \tilde{\pi}_{\ell j}^0$; $\Delta_{ii}^0 = -\tilde{\pi}_{\ell j}^0$
      end for
    end if
  end for
end for
$\mathbf{\Pi}^1 = \mathbf{\Pi}^0 - \mathbf{\Delta}^0$ where $\mathbf{\Delta}^0 \geq 0$
$\bar{\mu}^1 = [I - \mathbf{\Pi}^1]^{-1}\bar{\chi}$
begin Step 2
$k = 1$
while $\bar{\mu}^k > \bar{\mu}^{k-1}$ **do**
  for $j = 0$ to n **do**
    for $i = 0$ to n **do**
      if $\mu_j^k < \mu_i^k$ **then** // criterion for disabling
        for all $\sigma_\ell : \delta(q_i, \sigma_\ell) = q_j$ controllable **do**
          Disable σ_ℓ
          $\Delta_{ij}^0 = \tilde{\pi}_{\ell j}^0$; $\Delta_{ii}^0 = -\tilde{\pi}_{\ell j}^0$
        end for
      end if
    end for
  end for
  $\mathbf{\Pi}^{k+1} = \mathbf{\Pi}^k - \mathbf{\Delta}^k$ where $\mathbf{\Delta}^k \geq 0$
  $\bar{\mu}^{k+1} = [I - \mathbf{\Pi}^{k+1}]^{-1}\bar{\chi}$
  $k = k + 1$
end while
$k = k - 1$
while $\bar{\mu}^k > \bar{\mu}^{k-1}$ **do**
  for $j = 0$ to n **do**
    for $i = 0$ to n **do**
      if $\mu_j^k \geq \mu_i^k$ **then** // criterion for enabling
        for all $\sigma_\ell : \delta(q_i, \sigma_\ell) = q_j$ controllable **do**
          Enable σ_ℓ
          $\Delta_{ij}^0 = \tilde{\pi}_{\ell j}^0$; $\Delta_{ii}^0 = -\tilde{\pi}_{\ell j}^0$
        end for
      end if
    end for
  end for
  $\mathbf{\Pi}^{k+1} = \mathbf{\Pi}^k + \mathbf{\Delta}^k$ where $\mathbf{\Delta}^k \geq 0$
  $\bar{\mu}^{k+1} = [I - \mathbf{\Pi}^{k+1}]^{-1}\bar{\chi}$
  $k = k + 1$
end while

Chapter 3

Modeling and Identification of Robot Behavior

Modeling of hybrid supervisory systems is the first step towards providing a seamless link between the continuously-varying dynamical systems and the discrete event systems. Only after proper modeling is done and the identification of model parameters correctly, the DES control shall be implemented in the real systems.

This chapter demonstrates the modeling and model parameter identification stages for developing a supervisory control scheme for mobile robotic behavior. Since the rest of the complex systems that will be presented in this dissertation are simulation based models, it is imperative to show that the theoretical developments in this thesis provide implementable solutions to real-world applications.

Designing a robotic system interacting with a possibly dynamically changing environment is truly a multi-disciplinary task, which includes the knowledge of control theory, robotics, computer vision, and artificial intelligence. The design and modeling of the discrete-event behavior based mobile robotic system is presented in this chapter. The plant model of the robotic system is identified based on the available sensing and action capability. Later in Chapter 7, synthesizing a supervisor for an experimental scenario will be described.

3.1 Architecture of Robotic System

Figure 3.1 depicts the blocks that all supervisory control systems have in common. Continuous plant constitutes the complex dynamical system that is possibly under continuous control action. The Event Generator is the continuous-to-discrete block that generates the events from the available sensory and non-sensory information. This events, in turn, are inputs to unsupervised Deterministic Finite State Automata (DFSA) model of the system under discrete-event supervision. The DFSA model serves as a state estimator of the actual process. The DES controller represents the control policy applied to the DFSA model of the system, and it could be a conventional DES controller based on the control specifications [15]; alternatively, an optimal discrete-event supervisor can be designed by the quantitative method introduced in Chapter 2. The primary task of the Action Generator, which is the discrete-to-continuous (D/C) block, is to convert control commands from the supervisor (i.e. DES controller) into necessary input functions for the continuously varying plant, eg., new set points for the continuous controllers.

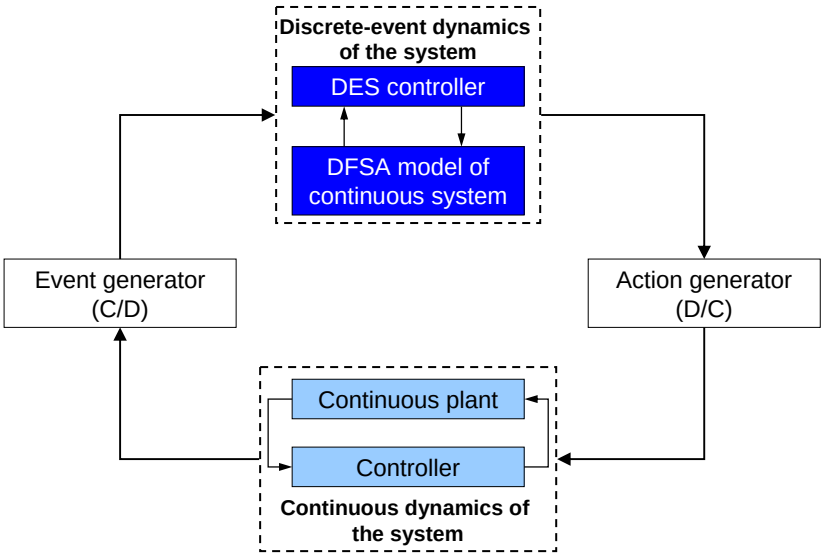


Figure 3.1. Architecture of hybrid supervisory control system

The features of the behavior based robotic system are

- A robot needs to perform autonomously to execute local tasks. The local DES control runs on the robot to interact with the lower layers of the continuous system to perform its own task.

- For different mission objectives, it is sufficient to redesign DES control law without changing the robot's component behaviors.
- Only high level symbolic events are communicated for status monitoring of the robot, thus saving the network bandwidth.
- The robot can participate in different coordination schemes among a set of robots without making any change of the available robot behaviors.

The system integrates the event driven discrete event dynamics modeled by finite state automaton and the time driven continuous dynamics modeled by ordinary differential equations through the continuous/discrete interface. The DES control module communicates with the robot's continuous time-varying control module by sending and receiving of a set of discrete events. The discrete module is designed to be independent of the underlying physical process and it provides a mechanism to interact with the continuous module.

A DES controller is allowed to plug and play in discrete module. The other modules consist of three blocks: *Action Generator*, *Event Generator*, and continuous controller. The continuous module connects to robot server via a standard TCP socket for sending and receiving formatted messages that encode up-to-date sensor readings and continuously varying commands of reference signals, respectively, at 10 Hz. The control strategy is event-driven, in which *Event Generator* block generates discrete events based on the continuous sensor data received from the server. The discrete events are transmitted to discrete module in symbolic form. Discrete module reacts immediately by sending out a discrete command back to continuous module according to the currently loaded supervisor. After receiving a particular event from the DES controller, *Action Generator* sends out a set of continuous reference signals to robot server for execution of a specific behavior. The robot continues executing this behavior until the sensor readings trigger the occurrence of a new event. Under this architecture, DES control strategy for both real robot and a simulator are designed and exercised. The hardware integration of the self-customized real robot is given below.

Four Segway robotic mobile platforms (RMPs) of the Networked Robotic and Sensor Intelligence Lab at Penn State have been used for the research on robotic system behaviors. Figure 3.2 shows the commercial Segway RMP on the left and

the customized version at Penn State on the right. Each of the Segway RMPs is equipped with a SICK LMS200 laser range finder, six Devantech SRF05 sonars and six Sharp GP2D12 infrared sensors for obstacle avoidance. The laser range finder with a range of 80 m and resolution in the order of millimeters is used for both long range and short range obstacle avoidance. Three sonars in the front and three in the back are used to provide short range obstacle avoidance information with a range of about 3 meters. The IRs are placed on the sides to enable the robot to see if it is safe to make turns. The sonars and IRs are controlled by a dedicated network of Brainstem General Purpose (GP) modules based on the PIC18C252 micro-controller. Each robot also uses a SONY EVI-D30 pan-tilt camera for object recognition and tracking. The brain of the entire system is a Dell Latitude D410 laptop which is connected to the Brainstem Network and Sick Laser and the RMP's on board controller through universal serial bus (USB) port. The robot itself is connected to the rest of the network through the use of IEEE 802.11b/g wireless network.

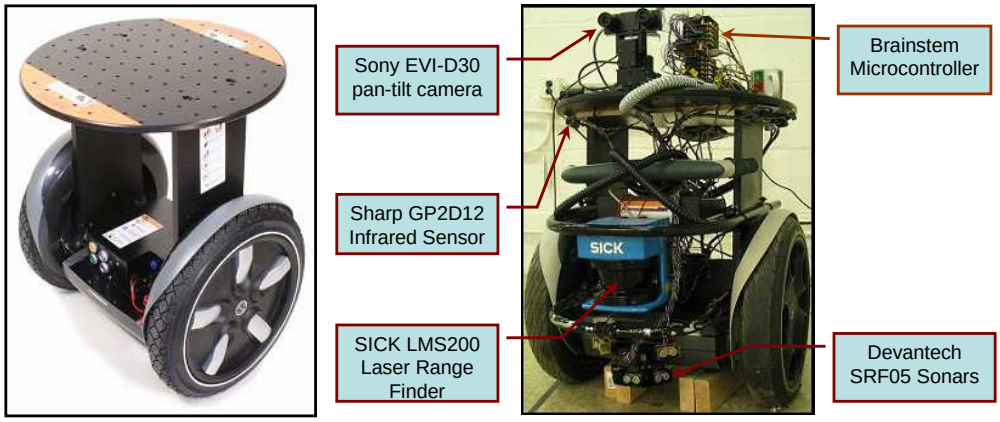


Figure 3.2. Segway RMPs that are customized at Penn State

The Player [64] is the lowest level functional block in the hierarchy which accesses to the robot's physical hardware such as laser, sonar and the actuators, and is usually run on the onboard computer. The *Action Generator* is the block that consists a set of fundamental behaviors that the robot can perform. More sophisticated behaviors are built upon the fundamental ones which are Search, Approach, Idle, Ignore, GoTo and Planner. In particular, Idle (robot does nothing), GoTo (robot goes to a waypoint) and Planner (robot plans the path from current position to

any waypoint and uses Floyd-Warshall algorithm for shortest path) behaviors are employed to generate complicated patterns. The *Event Generator* is an observer that implements continuous to discrete-event conversion.

The Stage simulator of the Player/Stage project [64] has been used to simulate the Segway robots. It is a 2D simulator capable of simulating sensors like laser, sonar, camera, pan-tilt-zoom (ptz), and has simple friction models for simulating built-in objects. Stage is capable of simulating a large population of robots with low-fidelity models. To more realistically simulate the dynamics, the model of the Segway obtained from system identification (see Section 3.3.2) has been used in conjunction with Stage as shown in Figure 3.3. Each robot in the Stage simulator runs an instance of Player robot server to provide access to sensors and actuators of the clients.

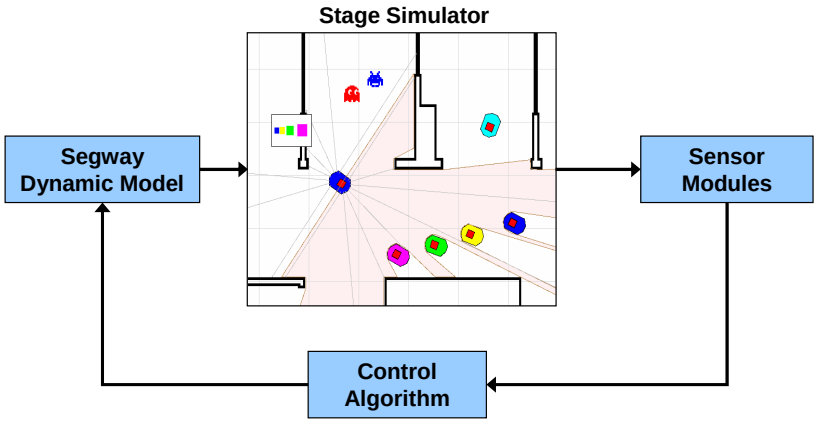


Figure 3.3. Simulation of Segway robot

3.2 Modeling the Robotic System Behavior

In hybrid supervisory setting, the modeling of the discrete-event dynamics of the complex system starts with identification of possible events that are generated implicitly through out the progress of the system. The events are partitioned into two subsets as controllable events and uncontrollable events. In this respect, robotic system modeling consists of controllable robot behaviors and uncontrollable reactions of the robot to the external stimuli.

A behavior is a set of processes involving sensing and action against the environ-

ment of the robot. Behaviors can be designed at a variety of levels of abstraction. In general, they are made to be higher than the robots atomic actions, e.g., turn left by 30 degree, and they extend in time and space. Some commonly implemented behaviors in literature include: go home, search object, avoid obstacle, pick up object, etc. In the discrete event setting, a behavior is a controllable event. The union of these behaviors is the controllable event set Σ_c. The set of uncontrollable events Σ_u is the set of all possible responses of the robot during the interaction with its environment. It consists of the reactions of the robot's behaviors, including a successful or unsuccessful completion of the controllable events (e.g., approach a target, grab an object) and an interruption of the robots current behaviors (e.g., find an object, detect obstacle). The C/D and D/C blocks are the interface between the discrete event dynamics and continuous time dynamics of the robot system. The D/C block is a map $\kappa : Q \times \Sigma \rightarrow \mathbb{R}^m$ that converts discrete-event information into the continuous reference signal as follows.

$$r(t) = \kappa(q_i, \sigma_k) \qquad t_k \leq t < t_{k+1} \tag{3.1}$$

where $q_i \in Q$ is the last automaton state of the DFSA model and $\sigma_k \in \Sigma$ is the most recently generated event. The C/D block is a map λ that converts the state space X of the continuous time system into the set of discrete events Σ. An event σ_k is generated at time $t = \tau$ if there exist an ϵ such that

$$\sigma_k = \lambda(x) \tag{3.2a}$$

$$f(x, u, t) > \epsilon \quad \text{for} \quad t < \tau \tag{3.2b}$$

$$f(x, u, \tau) \leq \epsilon \tag{3.2c}$$

Equation 3.2b and 3.2c define a closed set $\Omega \in X$. When the state trajectory enters Ω from outside for the first time, an event is generated at the time of entrance. The C/D block (also known as event generator) is fully specified by the pair (λ, f).

Based on all possible robot behaviors limited by its hardware and all possible reactions of the robot during the interaction with its environment, the set of discrete events Σ is identified as listed in Table 7.1. It should be noted that two

events cannot be generated at the same time instant.

Event	Explanation	Status
a	Approach to target	Controllable
e	Enemy detected	Uncontrollable
g	GoTo a waypoint	Controllable
i	Idle	Controllable
m	Mission accomplished	Uncontrollable
o	Obstacle detected	Uncontrollable
p	Planner (plan the path to a waypoint)	Controllable
s	Search the environment	Controllable

Table 3.1. Explanation of events based on robot behaviors

Discrete-event dynamical behavior of physical plants is often modeled as regular languages that can be realized by finite-state automata [15]. The open-loop discrete event dynamics are modeled as a DFSA based on the postulated experiment scenario and robot behavior. The model may vary for different mission scenarios. The DFSA plant model assumes that a mobile robotic system executes the assigned tasks independently and autonomously and also has the capability to perform complicated missions in coordination with other autonomous units.

As stated early in this chapter, the robot can perform a set of behaviors, including search, approach and idle. The experiment scenario is designed to give two tasks to the robot: one is searching for enemies in the environment and the second is going to the specified target location. The mission of robot is to explore the environment as much as possible before completing the mission at the target location. Therefore, the robot is able to chose from several behaviors to execute independently, yet it can still accomplish a cooperative mission. Without any discrete event supervisory control, the robot starts the operation by either searching or planning to go to target. During the mission, sensors can detect obstacles and enemies, which may trigger new planning to approach to target. Otherwise, it will approach to target to accomplish the mission after exploring the environment. The robot repeats the same procedures until stopped by human operator or higher level of control. Given the above experiment scenario, the robot plant model G is designed and shown in Figure 3.4.

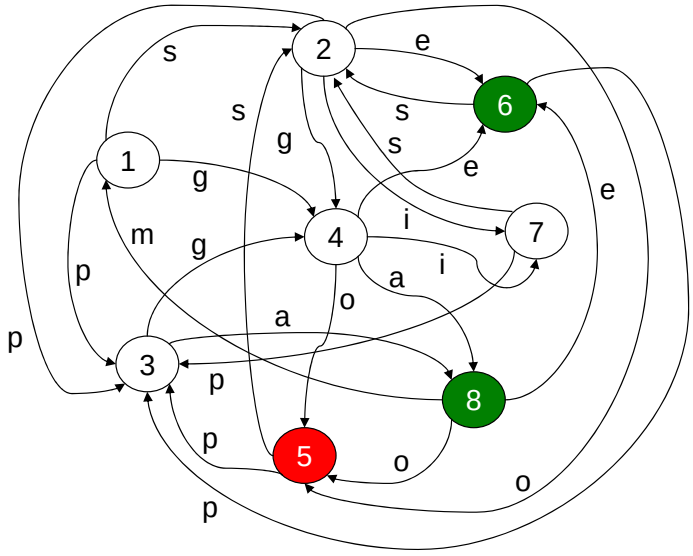


Figure 3.4. DFSA model of the open loop robot behaviors

3.3 Identification of Plant Parameters

3.3.1 Identification of Event Probabilities

Due to the probabilistic interpretation of the event cost matrix $\tilde{\mathbf{\Pi}}$, the idea of online parameter identification is to estimate the frequencies of events observed at corresponding states.

Let $\tilde{\pi}_{ij}$, representing the ij^{th} element of the event cost matrix $\tilde{\mathbf{\Pi}}$, be defined as the transition probability of event σ_j on the state q_i, i.e.,

$$\tilde{\pi}_{ij} = \begin{cases} P[\sigma_i|q_j], & \text{if } \exists q \in Q \text{ s.t. } q = \delta(q_i, \sigma_j) \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

The indexing function $I_k(i,j)$ is defined as the occurrence of event σ_j at time k if the system is in state q_i at time $t < k$. Formally, $I_k(i,j)$ is expressed as:

$$I_k(i,j) = \begin{cases} 1, & \text{if } \sigma_j \text{ is observed at state } q_i \\ 0, & \text{otherwise} \end{cases} \tag{3.4}$$

Let $N_k(i)$, denoting the number of incidents of reaching the state q_i up to the time instant k, be a random process mapping the time interval up to the instant k into the set of nonnegative integers $\mathbb{Z}^+$. Similarly, let $n_k(i,j)$ denote

the number of occurrences of the event σ_j at the state q_i up to the time instant k and $J_k(i) \equiv \Sigma_j I_k(i,j)$. The recursive algorithm of computing p_{ij} is formulated as a stochastic approximation at the starting time instant $t = 0$ with the initial conditions: $p_0(i,j) = 0$ for all $q_i \in Q$, $\sigma_j \in \Sigma$; and $n_0(i) = 0$, $J_0(i) = 0$ for all $q_{\in}Q$. Upon occurrence of the event σ_j at the state q_i, the recursive algorithm is incremented as:

$$
\begin{aligned}
N_k(i) &= N_{k-1}(i) + I_k(i,j) && (3.5)\\
J_k(i) &= J_{k-1}(i) + I_k(i,j) && (3.6)\\
\hat{p}_k(i,j) &= \hat{p}_{k-1}(i,j) + J_k(i) N_k(i)^{-1} (I_k(i,j) - \hat{p}_{k-1}(i,j)) && (3.7)\\
n_k(i,j) &= n_{k-1}(i,j) + I_k(i,j) && (3.8)
\end{aligned}
$$

The recursive stochastic approximation of $\hat{p}_k(i,j)$ is the frequency estimator $\hat{p}_{ij}(k)$ of independent Bernoulli trials at time k, i.e.,

$$
\begin{aligned}
\hat{p}_k(i,j) &= \frac{n_k(i,k)}{N_k(i)} && (3.9)\\
\lim_{N_k(i)\to\infty} \hat{p}_k(i,j) &= \tilde{\pi}_{ij} && (3.10)
\end{aligned}
$$

Therefore, $\tilde{\pi}_{ij}$ is the asymptotic value of the estimated probabilities $\hat{p}_k(i,j)$ as if the state q_i is visited infinitely many times. However, dealing with finite amount of data, the objective is to obtain a good estimate $\hat{p}_{ij}$ of $\tilde{\pi}_{ij}$ from independent Bernoulli trials of generating events.

It is necessary to have a stopping rule to determine a bound on the number of experiments to be conducted for identification of the $\tilde{\mathbf{\Pi}}$ matrix parameters. The objective is to achieve a trade-off between the number of experimental observations and the estimation accuracy.

The stopping rule, which is used for the parameter identification process, is based on the properties of irreducible stochastic matrices. Following Equations 3.9 and 3.10 and the state transition function δ of the DFSA, the state transition matrix is constructed at the k^{th} iteration as $\mathbf{P}(k)$ that is an irreducible $n \times n$ stochastic matrix under stationary conditions. Similarly, the state probability

vector $\bar{p}(k) \equiv \begin{bmatrix} p_1(k) & p_2(k) & \cdots & p_n(k) \end{bmatrix}$ is obtained by following Equation 3.9:

$$p_i(k) = \frac{N_i(k)}{\sum_{j \in I_Q} N_j(k)} \tag{3.11}$$

The stopping rule makes use of the Perron-Frobenius theorem [62] to establish a relation between the state probability vector $\bar{p}(k)$ and the irreducible stochastic matrix $\mathbf{P}(k)$. There is a unique unity eigenvalue of $\mathbf{P}(k)$ and the corresponding left eigenvector $\bar{p}(k)$ (normalized to unity in the sense of absolute sum) representing the state probability vector, provided that the matrix parameters have converged after a sufficiently large number of iterations. That is,

$$\|\bar{p}(k)(I - \mathbf{P}(k))\|_\infty \leq \frac{1}{k} \rightarrow 0 \text{ as } k \rightarrow \infty \tag{3.12}$$

Equivalently,

$$\|\bar{p}(k) - \bar{p}(k+1)\|_\infty \leq \frac{1}{k} \rightarrow 0 \text{ as } k \rightarrow \infty \tag{3.13}$$

Taking the expected value of $\|\bar{p}(k)\|_\infty$ to be $1/n$, a threshold of η/n is specified, where n is the number of states and $0 < \eta \ll 1$ is a constant. A lower bound on the required number of samples is determined from Equation 3.13 as

$$N \equiv \text{Integer}\left(\frac{n}{\eta}\right) \tag{3.14}$$

based on the number of states n and the specified tolerance η.

Based on above procedure, the experiments were conducted on Segway RMP to identify the probabilities of occurrence of events. To realize the mission scenario described in Section 3.2, the robot was driven by an input vector of linear and angular velocity commands in its typical work environment and the generated events were recorded. The controllable behaviors of the robot were generated using a pseudo-random number generator and the uncontrolled behaviors are observed during robot's interaction with the environment. The environmental variables such as the number of enemies, the number of target locations and obstacle map were fixed through out the experiments to guarantee the convergence of the asymptotic values of event probabilities. Rather than running the robotic experiment over and over again, couple of long-run experiments were conducted till all the event prob-

abilities converged in the Cauchy sense using one enemy and two target locations. Event probabilities constitute the elements of $\tilde{\mathbf{\Pi}}$ matrix and Figure 3.5 shows the convergence of some non-zero elements in $\tilde{\mathbf{\Pi}}$ matrix obtained in real experiments according to the parameter identification procedure. Table 3.2 lists all the event probabilities.

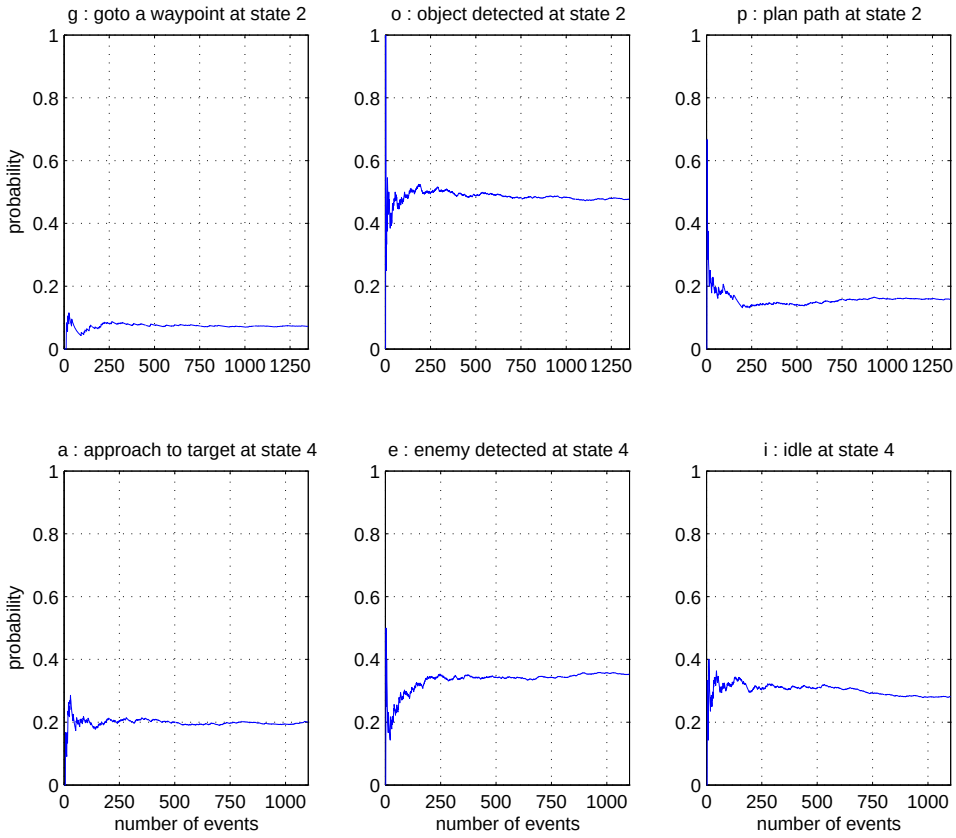


Figure 3.5. Convergence of some non zero $\tilde{\pi}$ values

$\tilde{\mathbf{\Pi}}$	a	e	g	i	m	o	p	s
State 1	0	0	0.1147	0	0	0	0.2782	0.6071
State 2	0	0.1711	0.0719	0.1222	0	0.4771	0.1577	0
State 3	0.3049	0	0.6951	0	0	0	0	0
State 4	0.1996	0.3521	0	0.2813	0	0.1670	0	0
State 5	0	0	0	0	0	0	0.3291	0.6709
State 6	0	0	0	0	0	0	0.6837	0.3163
State 7	0	0	0	0	0	0	0.5137	0.4863
State 8	0	0.1073	0	0	0.8375	0.0552	0	0

Table 3.2. Identified values for event probabilities of robot behavior

3.3.2 System Identification of Segway Dynamics

This section describes the modeling of a Segway RMP. The model was obtained by using the system identification toolbox of MATLAB. The robot was excited using a pseudo random input in its typical work environment. The input was constrained as follows

$$1 \text{ sec} \leq t_d \leq 6 \text{ sec} \tag{3.15}$$

$$-0.8 \text{ m/s} \leq v_c \leq 0.8 \text{ m/s} \tag{3.16}$$

$$-0.4 \text{ rad/s} \leq \omega_c \leq 0.4 \text{ rad/s} \tag{3.17}$$

v_c and ω_c are the commanded linear and angular velocities of the robot which are held constant for a period of t_d seconds. The measured outputs for linear velocity v and angular velocity ω of the onboard sensors were recorded. Since the Segway has the dynamics of an inverted pendulum, an onboard balancing controller is always active which introduces high frequency noise to the response. The subspace method for system identification [65] is performed to eliminate the effects of this noise. To choose the model order we follow standard techniques such as the Akaike's Information Criterion (AIC) [65]. A fourth order model was found to be a good balance between prediction error and modeling complexity as shown in Figure 3.6. The resulting state space model with the input vector $u = [v_c \quad \omega_c]^T$ and output vector $y = [v \quad \omega]^T$ is given by

$$x_{n+1} = Ax_n + Bu_n \tag{3.18a}$$

$$y_n = Cx_n + Du_n \tag{3.18b}$$

where the matrices A, B, C, D are as follows:

$$A = \begin{bmatrix} 0.96243 & -0.026344 & 0.11904 & 0.01982 \\ -0.029337 & 0.61297 & 0.024228 & 0.21747 \\ -0.12771 & -0.079849 & 0.76951 & -0.25586 \\ -0.0052022 & -0.47935 & -0.217 & -0.4911 \end{bmatrix}$$

$$B = \begin{bmatrix} -0.0081757 & -0.00027395 \\ 0.001303 & 0.085203 \\ 0.036843 & 0.080659 \\ -0.024035 & 0.40325 \end{bmatrix}$$

$$C = \begin{bmatrix} 5.3435 & -0.39613 & -0.14567 & 0.092586 \\ 0.12038 & 1.4111 & 0.0012965 & -0.2697 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Figure 3.7 shows the input, measured response and the response of the 4^{th} order model to this input.

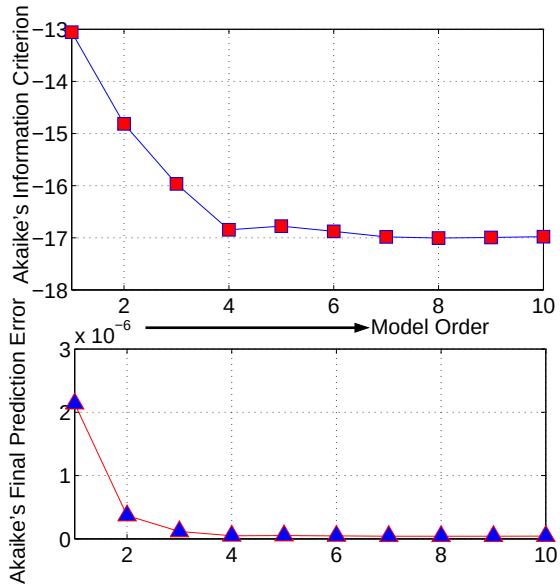


Figure 3.6. Model order selection

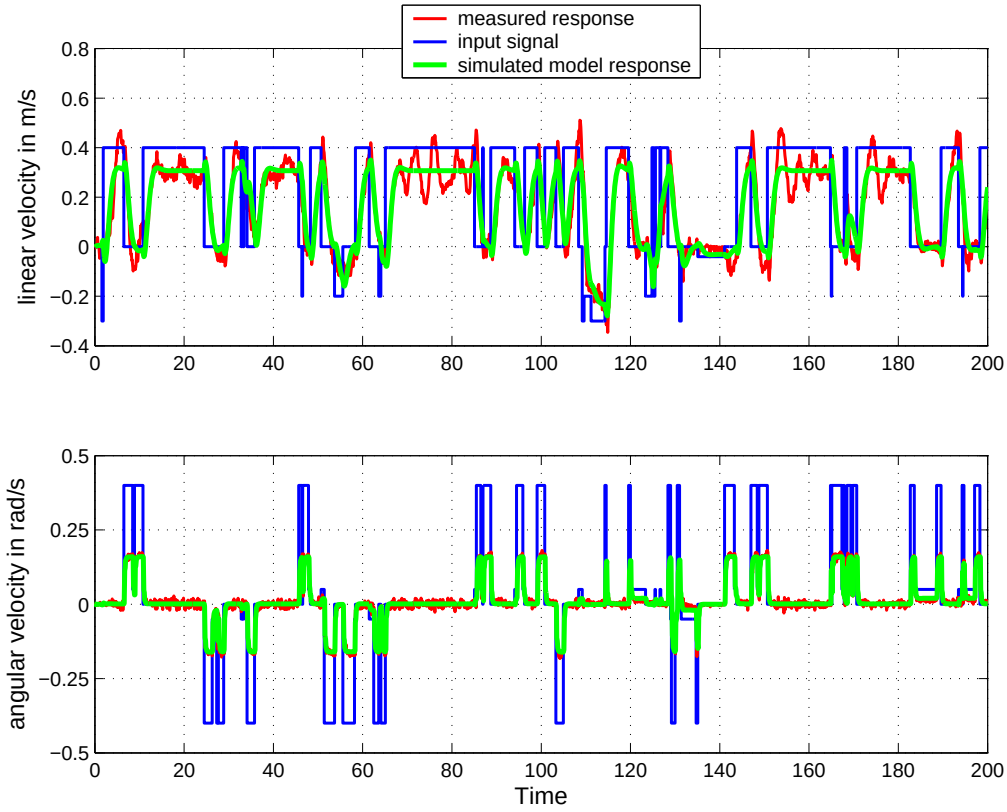


Figure 3.7. Response plots for system identification

Chapter 4

Advanced Analytical Methods for Event Generation

This chapter presents the application of advanced pattern recognition techniques for event generation in complex dynamical systems, where the physical process is approximately stationary in the fast time scale of the process dynamics and any non-stationarity due to gradually evolving degradation occur in the slow time scale [66]. The anomaly detection methods based on Symbolic Dynamic Filtering (SDF) and chaos theory, and their efficacy has been examined on the simulation test bed of a twin-engine propulsion system. Time series data of observed macroscopic variables, generated on the fast time scale from the simulation model, are analyzed at slow time scale epochs for event generation. The results are compared with those derived from traditional pattern recognition tools, such as Principal Component Analysis (PCA) and Artificial Neural Network (ANN).

4.1 Introduction

One of the major tasks of the supervisory decision making is fusion of the (possibly) redundant, conflicting, and incomplete information to make timely decisions. Such information can be derived from different types of sensor data [67] as well as operational history and the knowledge base generated from operator's personal experience. Computer-based advanced analytical techniques are necessary for fusion of the time series data available from multiple sensors and other relevant non-sensor-

based information to make specific inferences that could not be achieved through the sole usage of the available sensory information. Improved performance may not result simply from an increased volume of sensor data and system information unless the ensemble of information is systematically processed in the context of the operational conditions and mission objectives [68]. In order to achieve the desired performance of a Discrete Event Supervisory (DES) controller, it is essential to have an effective event generation algorithm to ensure fusion of the heterogeneous information for:

- enhanced resolution and reduced ambiguity in decision and control
- advantageous trade-offs between probability of false alarms and missed detection [69]

Recently, Ray [44] has proposed a concept of anomaly detection, based on Symbolic Dynamic Filtering (SDF) of measured macroscopic observables, which can be employed for advanced event generation and information fusion. The algorithm is built upon two-time-scale analysis of the stationary behavior of dynamical systems using the principles of symbolic dynamics [50], information theory [5], automata theory [70], and pattern recognition [71]. Symbolic dynamics captures the essential dynamical features of the physical process through space partitioning. Information theory allows modeling of incipient events and chaotic behavior that are analogous to thermodynamic phase transitions. Automata theory generates finite-state machine models of the dynamical system behavior under different operating conditions. Pattern discovery methods infer events through quantitative evaluation of the deviations in statistical patterns of the respective state machines. Combining these efforts with ergodic theory of chaos and thermodynamic perspective is essential when the dynamic system at hand exhibits a nonlinear behavior, especially in a chaotic manner. Following Eckmann and Ruelle [72], it is possible to use these theories in event generation.

The algorithms of the real-time event generation method has been tested on a simulation model of a generic gas turbine engine [73], and its efficacy is evaluated relative to other existing pattern recognition techniques from the perspectives of early detection of degradation in turbine efficiency, which can be deemed as an indicator of the turbine's health state. The results are compared with those derived

from traditional pattern recognition tools, such as Principal Component Analysis (PCA) and Artificial Neural Network (ANN).

4.2 Brief Review of Symbolic Dynamic Filtering

As a consequence of its simplicity and ability to capture macroscopic dynamical behavior in a low dimensional state space, symbolic dynamics has been used as a tool for pattern recognition [74]. In general, the state space of a physical process may have a very high dimension, which is difficult to handle numerically. Symbolic dynamics uses both spatial partitioning of the data sequence to represent the continuous dynamics with a predefined alphabet of symbols [4][50]. The underlying concept of wavelet space partitioning for symbolic time series analysis is introduced in [45]. Further details on symbolic dynamics are available in [70][44], and on space partitioning in [75][76].

Time series data are obtained from sensors and/or from information fusion of sensor signals and analytical models at the time scale of system dynamics, which is called the fast scale. Sets of (fast scale) time series data are generated at slow scale (discrete) time epochs, over which statistics of the dynamical system may become non-stationary, possibly due to occurrence of anomalies. Each set of (fast scale) time series data of macroscopic observables are converted to a symbol sequence by partitioning a compact (i.e., closed and bounded) region Ω in the phase-space of system dynamics, over which the time series data evolves, into finitely many discrete cells [77][66]. Each cell is labeled as a symbol $\sigma \in \Sigma$, where the symbol set Σ is called *alphabet* consisting of $|\Sigma|$ different symbols. As the system evolves in time, the trajectory of the dynamical system travels through various cells in its phase space and the corresponding symbol is assigned to it, thus converting the time series data sequence to a symbol sequence $\{\sigma_1, \sigma_2, \sigma_3, \ldots\}$ that characterizes the system dynamics represented by the data sequence. Thus, generation of a symbol sequence represents coarse-graining of the trajectory's time evolution [4]. Critical steps in the symbol generation process are:

- partitioning of a finite region in the phase space;
- construction of a mapping from the partitioning into the symbol alphabet,

which becomes a representation of the system dynamics defined by the trajectories.

Partitioning of the phase space is a crucial step in SDF. Kennel and Buhl [75] have formulated a phase-space partitioning method based upon the concept of Symbolic False Nearest Neighbors (SFNN), where an algorithm is introduced to refine empirical partitions for symbolic state reconstruction. This method avoids topological degeneracy that is an essential feature of a generating partition [66]. The major advantage of this method is that the partitioning is accomplished by an algorithm based on the time series data. For example, the partitions are defined with respect to a set of radial-basis influence functions,

$$f_k(x) = \frac{\alpha_k}{\|x - z_k\|^2} \tag{4.1}$$

each associated with a symbol σ_k with the center z_k and weight α_k. For each element x of the time series data set, one function $f_m(x)$ is generally expected to be greater than the other $f_k(x)$ with $k \neq m$. Then, the data point x in the phase space is transformed to a symbol σ in the symbol space. The parameters z_k and α_k are the free optimization variables, with the constraint $\alpha_k \geq 0 \ \forall k$. There may be one or more influence functions assigned to each of the symbols in the alphabet. The partitions remain invariant at all epochs of the slow time scale. Partitioning of the phase space by the SFNN approach may often be difficult if the time series data are noise-corrupted [77][75]. This problem is circumvented by if denoising is possible without any significant loss of dynamical information in the time series data.

In an alternative partitioning scheme, referred to as maximum entropy wavelet partitioning (MEWP) in the sequel, the symbol sequence(s) that represent the state of the dynamical system are derived from the wavelet transform coefficients of time series instead of obtaining them directly from the time series itself. The wavelet coefficients not only help represent patterns at different scales, but also effectively attenuate the measurement noise and spurious signals [78]. Thus, the probability of occurrence of false events, which may degrade the performance of DES controller, is significantly reduced.

The core concept of entropy-based partitioning is that regions with more (less)

information are segmented finer (coarser). In other words, more symbols are allocated to information-rich regions and less symbols to information-sparse regions. That is, given the alphabet size, the partition that maximises the entropy of the symbol sequence is chosen [76]. The proposed scheme produces a graph of coefficients versus scale at each time shift. Since the wavelet transform is a function of two-variables: scale and time, these bivariate graphs are converted to univariate by stacking from end to end, starting with the smallest value of scale and ending with the largest value. For example, the wavelet coefficients versus scale at time shift t_k are stacked after those at time shift t_{k-1} to obtain the so-called *scale series data* in the wavelet space, which is analogous to the time series data in the phase space. The number of blocks in a partition is equal to the size of the alphabet and each block of the partition is associated with a symbol in the alphabet. For a given stimulus, the partitioning of wavelet space must remain invariant at all epochs of the slow time scale. For the wavelet-based maximum entropy partitioning of multi-dimensional data, a phase space constructed with wavelet coefficients of multiple time series data are partitioned to generate the symbol sequences. This approach compresses the joint statistics into univariate statistics when co-located heterogeneous sensors and/or analytical measurement(s) are available for monitoring purposes.

The partitioning is performed at the time epoch of the nominal condition that is chosen to have zero anomaly measure. A finite state machine, called D-Markov Machine, is then constructed, where the states of the machine are defined corresponding to a given alphabet Σ and window length D. The alphabet size $|\Sigma|$ is less than or equal to the total number of partitions, while D is the length of consecutive symbol words forming the states of the machine. Hence, the number of states is less than or equal to the total permutations of the alphabet symbols within a word of length D. The choice of $|\Sigma|$ and D depends on specific experiments, noise level and also on the available computation power. Using the symbol sequence, generated from time series data, the state machine is constructed on the principle of sliding block codes [50].

The probability distribution p_t of patterns, represented as histograms in Figure 4.1, is recursively computed as an approximation of the natural invariant density of the dynamical system at the slow time epoch t, which is a fixed point of the

local Perron-Frobenius operator [62]. The process continues with generation of the state probability vectors $p_1, p_2, \ldots, p_k$ at slow time epochs from the respective symbolic sequences at $t_1, t_2, \ldots, t_k$ by using the finite state machine at a time epoch t_{nom}. The last step is computation of anomaly measures at time epochs $t_1, t_2, \ldots, t_k$, relative to the nominal probability vector p_{nom} at time epoch t_{nom} via an appropriate diversity distance function

$$\mathcal{M}_k \equiv d(p_k, p_{nom}) \tag{4.2}$$

Based on the progressive anomaly, the event is generated for the dynamic process which exhibit anomalous behavior beyond a threshold. Threshold values may depend on many factors such as the significance and redundancy of component, signal to noise ratio of the sensor and decisions that will based on the generated event. The step-by-step process is depicted in the flow chart shown in Figure 4.1.

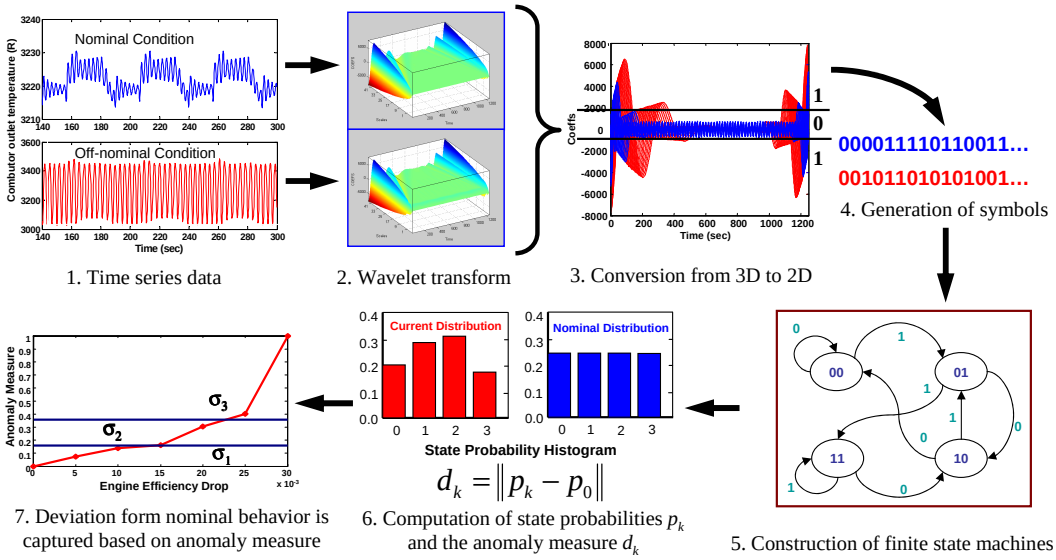


Figure 4.1. Event generation via wavelet-based SDF

The SDF has a unique advantage of detecting an incipient event if its frequency band overlaps with the spectrum of the system dynamics and existing disturbances. The rationale is that the algorithm makes use of the evolving statistics in the pattern vectors instead of relying solely on their frequency characteristics.

4.3 Radial Basis Function Neural Network

Neural networks are capable of learning complex input-output relationships, using sequential training procedures, and adapting to the data [79]. The most commonly used family of neural networks for pattern classification is the feed-forward network. The learning process involves updating the network architecture and its connection weights, so that the network can efficiently perform a specific classification/clustering task. Although neural networks provide a suite of nonlinear algorithms for feature extraction and classification, they are implicitly equivalent or similar to classical statistical pattern recognition methods [71]. Despite these similarities, neural networks do offer several advantages such as, unified approaches for feature extraction and classification and flexible procedures for finding moderately nonlinear solutions. Among the most commonly used family of feed-forward neural networks for pattern classification, the Radial Basis Function Neural Network (RBFNN) was found to be the most suitable for anomaly detection based on time series data of observables(s). RBFNN is essentially a nearest neighbor type of classifier, where the activation of a hidden unit is determined by the distance between the input vector and the prototype vector [80].

The RBFNN technique used herein for the event generation is an extension of the standard RBFNN to form a statistical model of nominal data. As new data enters into the anomaly detection system, it is compared with the RBFNN model. If it falls within the boundaries defined by the model, then it is considered as a nominal data; otherwise, the data is considered as anomalous. The approach is generic and has been applied to a variety of problems, including advanced military aircraft subsystems [81]. A key requirement for RBFNN is appropriate selection of the radial basis function and the order of the statistics of the model. From this perspective, a radial basis function for anomaly detection is chosen as:

$$f(x) = \exp\left(-\frac{1}{\theta_\alpha}\sum_k |x_k - \mu|^\alpha\right) \tag{4.3}$$

where the parameter $\alpha \in (0, \infty)$; and μ and θ_α are the center and α^{th} central moment of the data set, respectively. For $\alpha = 2$, $f(\bullet)$ becomes Gaussian, which is a typical radial basis function used in the neural network literature, and $\alpha = 1$

gives Logistic RBF.

From a sampled time series data under the nominal condition, the mean μ and the central moment θ_α are calculated as:

$$\mu = \frac{1}{N}\sum_{k=1}^{N} x_k \text{ and } \theta_\alpha = \sum_{k=1}^{N} |x_k - \mu|^\alpha \tag{4.4}$$

The distance between any vector $\bar{x}$ and the center μ is obtained as:

$$\|\bar{x} - \mu\|_{\ell_\alpha} = \left(\sum_{k} |x_k - \mu|^\alpha\right)^{\frac{1}{\alpha}} \tag{4.5}$$

Hence, at the nominal condition, the radial basis function $f_{nom} = f(x)$. For different anomalous conditions, the parameters, μ and θ, are kept fixed; and the radial basis function f_k is evaluated from the data set under the (possibly anomalous) condition at the slow time scale. Then, the anomaly measure at the k^{th} epoch is defined as a distance function

$$\mathcal{M}_k \equiv d(f_k, f_{nom}) \tag{4.6}$$

4.4 Principal Component Analysis

In the statistical approach, each pattern is represented in terms of d features or measurements and is viewed as a point in a d-dimensional space. The goal is to choose those features that allow pattern vectors belonging to different categories to occupy compact and disjoint regions in the d-dimensional feature space. The effectiveness of the representation space, generated from the feature set, is determined by how well patterns belonging to different classes can be separated. Given a set of training patterns from each class, the objective is to establish decision boundaries in the feature space to separate patterns belonging to different classes. In the statistical decision-theoretic approach, the decision boundaries are determined by the specified or learnt probability distributions of the patterns belonging to each class.

Feature extraction methods determine an appropriate subspace of dimension

m in the original feature space of dimension d with $m \leq d$. The best known linear feature extraction technique is the Principal Component Analysis (PCA) [71] that makes use of Karhunen-Love expansion to compute the m largest eigenvectors of the $d \times d$ covariance matrix of the N patterns, each of which is d-dimensional. Since PCA uses the most expressive features (eigenvectors with the largest eigenvalues), it effectively approximates the data on a linear subspace using the mean squared error criterion.

To detect deviation in time-series data, the PCA is performed for dimensionality reduction and thus serves as a feature selector in the pattern analysis. For a time series data with length L, an $M \times N$ data matrix is created by dividing the time-series data string of length L into substrings of length $M = L/N$, where each row of the data matrix is a substring of length N. Then, the $N \times N$ covariance matrix, obtained from the data matrix, generates the orthonormal eigenvectors $\nu_1 \ldots \nu_N$ and the corresponding non-negative real eigenvalues $\lambda_1 \geq \cdots \lambda_N \geq 0$, where the eigenvalues are arranged in the increasing order of magnitude. The m largest eigenvalues and associated eigenvectors are selected such that

$$\sum_{i=1}^{m} \lambda_i > \eta \sum_{i=1}^{N} \lambda_i \tag{4.7}$$

where $\eta < 1$ is a real positive number close to 1 (e.g., $\eta = 0.95$). The feature matrix F is defined as:

$$F = \left[\begin{array}{ccc} \sqrt{\frac{\lambda_1}{\sum_{k=1}^{N} \lambda_k}} \nu_1 & \cdots & \sqrt{\frac{\lambda_m}{\sum_{k=1}^{N} \lambda_k}} \nu_m \end{array} \right] \tag{4.8}$$

The feature matrix F_{nom} represents the engine health status derived from the time series data at the nominal condition. Similarly, feature matrices $F_1, F_2 \ldots$ are obtained from time series data at different epochs in the slow time scale. The anomaly measure for at a slow time k is obtained as:

$$\mathcal{M}_k \equiv d(F_k, F_{nom}) \tag{4.9}$$

where d is a metric signifying the distance between the nominal and the k^{th} epoch. Metric was chosen to be the Euclidian norm for the results shown in this work.

4.5 Brief Review of Ergodic Theory of Chaos

Eckmann and Ruelle [72] reviewed main mathematical ideas and their concrete implementation in analyzing experiments. The main subjects are the theory of dimensions, entropy and characteristic exponents.

Many physical systems exhibit the same qualitative features as the nonlinear evolution equation of the form:

$$\dot{x}(t) = F_\mu(x(t)),\ x \in \mathbb{R}^m \tag{4.10}$$

in a space of small dimension m. x is the set of coordinates describing the system, and F_μ determines the nonlinear time evolution where μ corresponds to control parameter. Consequently,

$$x(t) = f_\mu^t(x(0)) \tag{4.11}$$

and time evolution of the equation can be described as a function of initial condition.

As the parameter μ changes the asymptotic behavior may change. The μ values where these changes occur are defined as bifurcation points. Geometrically, the asymptotic motion follows an attractor in phase space which becomes more complicated with increasing μ.

The general idea for linear systems is to represent the system as a collection of independent oscillators or modes. Each mode is periodic such that the global system is quasiperiodic, i.e. a collection of periodic motions. However, for the case of nonlinear systems the concept of mode loses its importance. Even a finite dimensional nonlinear system does not need to be quasiperiodic. The concept of mode should be replaced by number of nonnegative characteristic exponents or information dimension.

After the transients die out the solution of the equation of the dynamical system settles down to a certain region in the phase space which is called an attractor. For dissipative systems, the volume of the attractor is usually very small compared to $\mathbb{R}^m$. But this does not necessarily mean the system contracts length in every direction, but some directions may stretch. It implies the asymptotic behavior may be unstable within the attractor. This can be seen as exponential separation of

orbits of points which were very close to each other. The separation takes place in the direction of stretching, and this attractor is called strange attractor. Moreover, a system with a strange attractor is chaotic or has sensitive dependence on initial conditions.

Ergodic theory can be used to measure the dimension and other quantities of the system. Ergodicity implies that the time average equals to the space average. Space average can be calculated using an invariant measure ρ which satisfies

$$\rho[f^{-1}(E)] = \rho(E),\ t > 0 \tag{4.12}$$

where E is any subset of $\mathbb{R}^m$. There may be many invariant measures associated with a system. The invariant measure which is physically relevant is called physical measure and if it exists, it is unique and represents time averages.

The invariant measure ρ can be decomposed into several different pieces, unless it is ergodic. Therefore, an invariant measure can be represented as a superposition of ergodic measures. If ρ is ergodic

$$\lim_{T\to\infty} \frac{1}{T} \int_0^T \varphi[f^1 x(0)]dt = \int \rho(dx)\varphi(x) \tag{4.13}$$

with respect to ρ.

For a discrete time system with evolution equation

$$x(n+1) = f(x(n)),\ x(i) \in \mathbb{R} \tag{4.14}$$

the separation of two initial conditions $x(0)$ and $x(0)'$ after time N is

$$x(N) - x(N)' = f^N(x(0)) - f^N(x(0)') \approx D_x f^N(x(0)) \cdot [x(0) - x(0)'] \tag{4.15}$$

where $D_x f = (\partial f_i/\partial x_j)$. Therefore the separation grows or decays exponentially with N. The average rate of growth is defined as:

$$\lambda = \lim_{N\to\infty} \frac{1}{N} \log |D_{x(0)} f^N \delta x(0)|,\ \ \delta x(0) = x(0) - x(0)' \tag{4.16}$$

If ρ is ergodic, the largest λ is independent of $x(0)$ and called the largest *Lyapunov*

exponent of the map f with respect to ρ.

The Lyapunov (characteristic) exponents and quantities derived from them gives bounds on the dimension of attractors, and on the production of information by the system. The average rate of information production $h(\rho)$ in an ergodic state ρ is related to sensitive dependence on initial conditions. $h(\rho)$ is called the (Kolmogorov-Sinai) entropy of the measure ρ, and in general

$$h(\rho) \leq \sum \textit{positive characteristic exponents}.$$

If a physical measure ρ can be identified associated with the system then

$$h(\rho) = \sum \textit{positive characteristic exponents}.$$

The dimension of a set is the amount of information needed to specify points on it accurately. Then a dimension, called capacity of set S, is defined as

$$\dim_k(S) = \limsup_{\varepsilon \to \infty} \frac{\log N(\varepsilon)}{\log \varepsilon} \tag{4.17}$$

The information dimension $\dim_H \rho$ of a probability measure ρ is defined as the minimum of the Hausdorff dimensions of the sets S for which $\rho(S) = 1$.

Information dimension can be defined as

$$\dim_H = \lim_{r \to 0} \frac{\log C(r)}{|\log r|} \tag{4.18}$$

where $C(r) = \frac{1}{N^2} \sum_{i,j} \Theta[r - |x(j) - x(i)|]$, N large and $\Theta(u) = (1 + \frac{|u|}{u})/2$.

4.5.1 Experimental Aspects of Chaos Theory

Starting with an experimental time series $u(1), u(2), \ldots$ corresponding to measurements regularly spaced in time, assume that $u(i) \in \mathbb{R}^v$. From the $u(i)$, a sequence of points $x(1), x(2), \ldots$ in $\mathbb{R}^{mv}$ is obtained by taking $x(i) = [u(i+1), \ldots, u(i+m-1)]$. This construction associates with points $X(i)$ in the phase space of the system and their projections $x(i) = \pi_m X(i)$ in $\mathbb{R}^{mv}$. If ρ is the physical measure describing the system, the points $x(i)$ are equidistributed with respect to the projected measure

$\pi_m\rho$ in $\mathbb{R}^{mv}$.

If $\dim_H \rho \leq M$, for most M dimensional projections π_m, $\dim_H \pi_m\rho$ for large enough m.

Using sequence $x(1), x(2), \ldots, x(N)$ in $\mathbb{R}^{mv}$, it is possible to construct the functions C_i^m and C^m as follows:

$$C_i^m(r) = \frac{1}{N}\{\text{number of } x(j) \text{ s.t. } d[x(i), x(j)] \leq r\},$$

$$C^m(r) = \frac{1}{N^2}\{\text{number of ordered pairs } [x(i), x(j)] \text{ s.t. } d[x(i), x(j)] \leq r\},$$

When $N \to \infty$,

$$\lim C_i^m = (\pi_m\rho)[B_{x(i)}(r)] \tag{4.19}$$

Suppose

$$\lim_{r\to 0}\lim_{N\to\infty} \frac{\log C_i^m(r)}{\log r} = \lim_{r\to 0} \frac{(\pi_m\rho)[B_{x(i)}(r)]}{\log r} = \alpha_m \tag{4.20}$$

Then $\dim_H \pi_m\rho = \alpha_m$. We choose m such that $\alpha_m < mv$, then we have $\dim_H \rho = \alpha_m$.

$C_i^m(r)$ is the probability that $x(j)$ satisfies $d[x(i), x(j)] \leq r$. By taking

$$d[x(i), x(j)] = \max\{|u(j) - u(i)|, \ldots, |u(j+m-1) - u(i+m-1)|\},$$

it is possible to interpret $C_i^m(r)$ as the probability that the signal $u(j+k)$ remains in the ball $B_{u(j+k)}(r)$ for m consecutive units of time.

Defining

$$\Phi^m(r) = \frac{1}{N}\sum_i \log C_i^m(r) \tag{4.21}$$

entropy can be obtained by

$$\lim_{r\to 0}\lim_{m\to 0\infty}\lim_{N\to\infty}[\Phi^{m+1}(r) - \Phi^m(r)] = \Delta t \cdot h(\rho) \tag{4.22}$$

Following the ergodic theory of chaos review, an intuitive way of defining the degradation is using information dimension. Using the obtained information dimension for nominal ($\dim_H^{nom}$) and off-nominal cases at different time epochs ($\dim_H^k$), it is expected that the information dimension increases with increasing

deviation, since the system shows more chaotic behavior as the anomaly increases. Therefore, the anomaly measure can be formulated as the distance function of information dimensions as:

$$\mathcal{M}_k \equiv d(\dim_H^k, \dim_H^{nom}) \tag{4.23}$$

Another approach, which is more rigorous than information dimension is using entropy or Kolmogorov-Sinai invariant $h(\rho)$ as anomaly measure. From thermodynamic point of view, the entropy of a state can be calculated as a sum of entropy of the initial state and entropy difference between the initial and the last states. With a similar argument, the entropy for nominal condition can be defined as:

$$h_{nom}(\rho) = h_0(\rho) + \Delta h(\rho)|_0^{nom} \tag{4.24}$$

Since at the nominal condition the system is already persistently excited, $h_0(\rho) > 0$; but value is unknown. Similarly, for off-nominal condition at time epoch k, entropy can be defined as:

$$h_k(\rho) = h_0(\rho) + \Delta h(\rho)|_0^{k} \tag{4.25}$$

Therefore, the deviamtion measure can be formulated as the distance function of Kolmogorov-Sinai invariant as:

$$\mathcal{M}_k \equiv d(h_k(\rho), h_{nom}(\rho)) \tag{4.26}$$

4.6 Experimental Results

This section presents the results of the application of various anomaly detection methods on the simulation test bed of a generic gas turbine engine, similar to that reported in [73]. (See Appendix A for details.) The model contains steady state performance maps for all the components and has control volumes where continuity and energy balances are maintained. Rotor dynamics and the duct momentum dynamics are also included in this model.

In the health monitoring of the engine, usual quantitative approach to identify anomalous condition is to measure the deviation of the efficiency values from

nominal state (brand new engine). For implementation purposes, the high and low pressure turbine efficiencies, fan and compressor tip velocity ratios are reduced to observe the effects on the time series data. Here, the degradation in the efficiencies of different components of the engine are assumed to be uniform. This assumption is reasonable for the comparative study presented herein, although it may not hold true for a real engine because each component may deteriorate at different rates. Health condition (i.e. the efficiencies) of an engine changes in a very slow time scale therefore, for a short time period (in the order of minutes) the efficiency values are constant for all practical purposes. Replication of these conditions may require hundreds of hours of engine simulation. Since this is not a feasible solution, engine component efficiencies are reduced for each run of the simulation and a certain period of operation is observed for each health condition. This is conceptually similar to experimental methods where engine is tested in extreme conditions to simulate years of operation in a few days. Also, transient (0-100 sec) data was not used for the analysis.

For event generation, the time series data from a number different sensor sources was analyzed. Based on these simulation experiments it was found that the combustor outlet temperature and the main burner fuel flow rate variables best captured the degradation of the engine. From thermodynamic perspective, it was expected that these two variables would be affected by the change in engine efficiencies. The results of the simulation corroborate this fact. Upon further investigation, it was found that the combustor outlet temperature essentially captured the dynamics of degradation and using the fuel flow rate did not add anything significant to the analysis. Therefore, the results on event generation are based on the time series analysis of the combustor outlet temperature. The sets of time-series data were collected after the dynamic response attained the stationary behavior. These data sets were used to compare the detection capability of the SDF and chaos-theoretic approaches relative to that of two existing pattern recognition techniques, PCA and RBFNN [82]. Since symbol generation from time series data is the crucial step in SDF, two alternative approaches are investigated: Symbolic False Nearest Neighbors (SFNN) and maximum entropy wavelet partitioning (MEWP).

Time series data were gathered by executing the simulation model under dif-

ferent health conditions, by altering the efficiency values and flow constants to generate simulated degradation. The nominal relative parameter for each efficiency was set to 1.0 (100%) and, under anomalous conditions; the parameters were reduced down to 0.97 (97%) gradually. For implementation purposes, the degradation in efficiencies of different engine components is assumed to be uniform. This assumption is reasonable for the comparative study presented herein. The effects of the decreasing efficiency on the output data were observed under persistent excitation. For example, in these simulation experiments, output data was observed by perturbing the booster vane angle. The perturbation used for this purpose was a square wave with amplitude 10% of the nominal value (which is computed by the feedforwad controller of the engine's actuator).

Figure 4.2 shows the time series data belonging to different health conditions up to 2.5% efficiency degradation. It is noticed that the output data are almost impossible to distinguish with a simple threshold check or visual inspection, or a peak pass test in frequency domain. However, when the saturation occurs as shown in Figure 4.3, high fluctuations in temperature are observed and visual inspection is sufficient for detection of anomaly, but our goal here is to capture the deviation from the nominal condition as the health parameter changes gradually.

For the MEWP method, the alphabet size used in partitioning the wavelet coefficient space is 8. For the SFNN method, the alphabet size to generate the finite state machine is 4. (Ideally, the alphabet size should be 8 for a proper comparison. Unfortunately, SFNN becomes extremely computation-intensive if the dimension of the phase space is large.) For the RBFNN method, exponent value $\alpha = 1$ (Logistic RBF) is used. Results are normalised with respect to maximum degradation condition. For the PCA method, time series data is divided into 10 subsections to form the data matrix.

Experimental formulation of information dimension is given as $\dim_H \rho = \alpha_m$. However, we must be careful in implementing the theory since the time series data to obtain α_m is not infinite and the limit may not be found exactly. Therefore, dimension m is chosen such that $\alpha_m < mv$, where m and v are corresponding data dimensions. When m increases beyond a value, α_m becomes independent of m. Experimentally, α_m can be obtained by plotting $\log[C_i^m(r)]$ vs $\log r$ and determining the slope of the curve. Figure 4.4(a) shows these curves for increasing

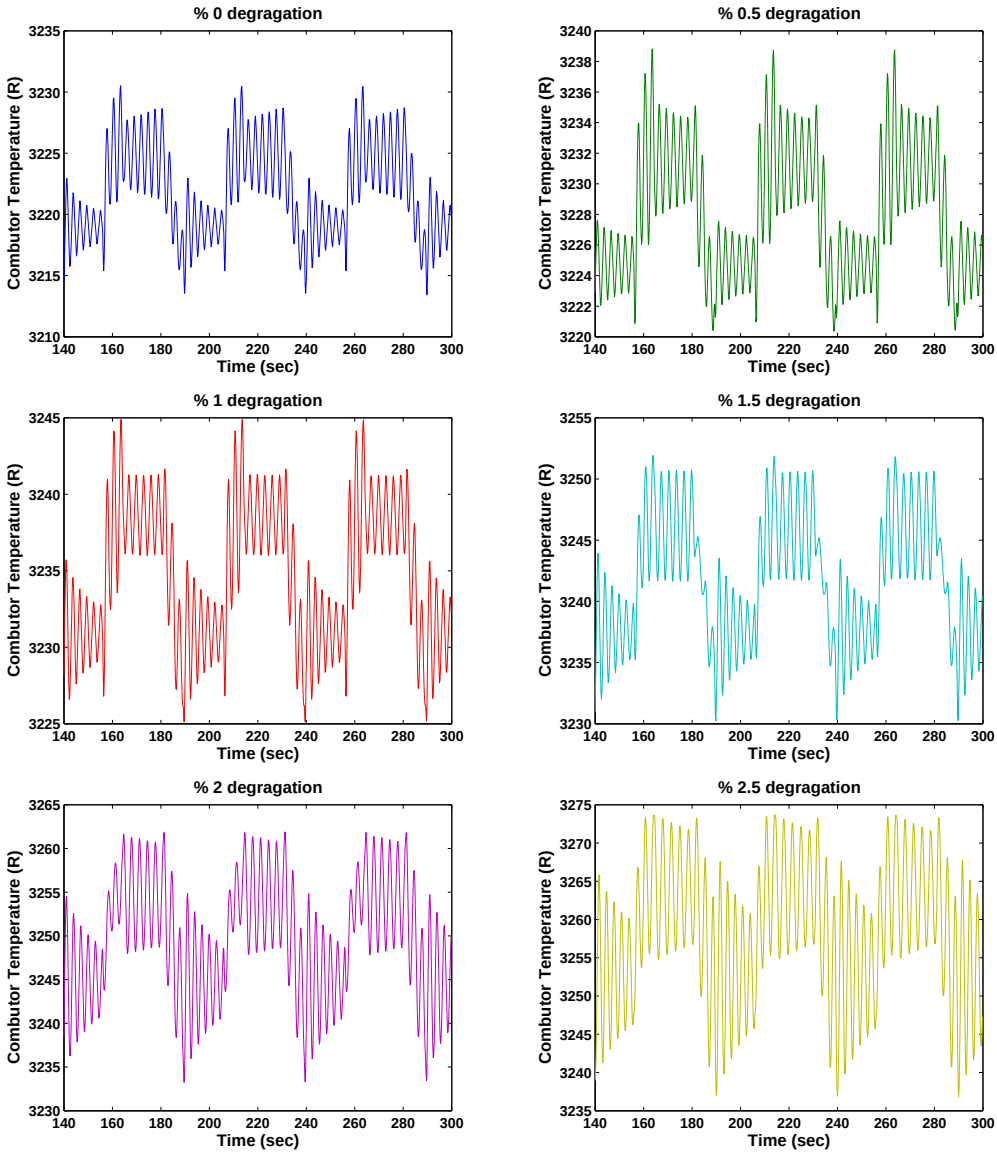


Figure 4.2. Temperature data for different health conditions

m values at the nominal condition. For small r, there exists large scatter of points due to poor statistics; then there is a range $[r0, r1]$ of near consistency (the constant is the information dimension if m is suitable large). For r larger, there is a deviation from consistency due to nonlinear effects. The "*meaningful range*" is that the distribution of distances between pairs of points is statistically useful (see figure 4.4(b)). In the figure, one standard deviation distance from the mean is marked as useful range of data points.

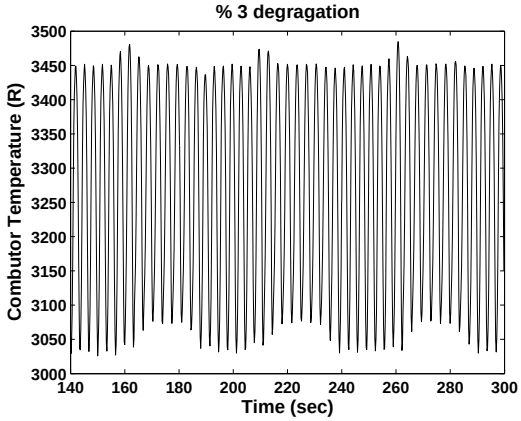
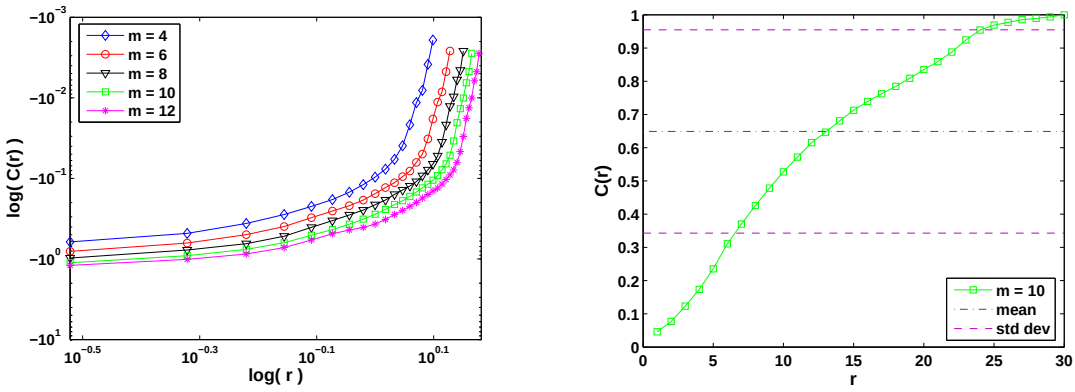


Figure 4.3. Temperature data for 3% efficiency degradation



(a) C^m vs r (b) "*Meaningful range*"

Figure 4.4. Information dimension plots for nominal condition and statistically consistent data range

Figure 4.5 compares the SDF methods, two methods on ergodic theory of chaos and traditional pattern recognition tools of PCA and RBFNN. Results show that the chaos-theoretic methods are marginally better than the SDF methods with respect to combustor temperature data sets. Nevertheless, the anomaly assessment curves are in general consistent with one another and the data, where saturation occurs at 3% efficiency drop. We have argued before that there are inherent problems obtaining information dimension and in most of the cases, to find the correct dimension is not possible. On the other hand, to evaluate the entropy of the system we need to evaluate three limits simultaneously. However, this is much easier to handle than achieving a statistically *meaningful range* to get information dimension. In this respect, it is more convenient to use entropy as anomaly

measure. The events are generated based on the off-nominal deviation, and are clustered in three regions of the curves corresponding to "healthy", "degraded" and "unhealthy" conditions. The particular thresholds are chosen as 0.2 and 0.4.

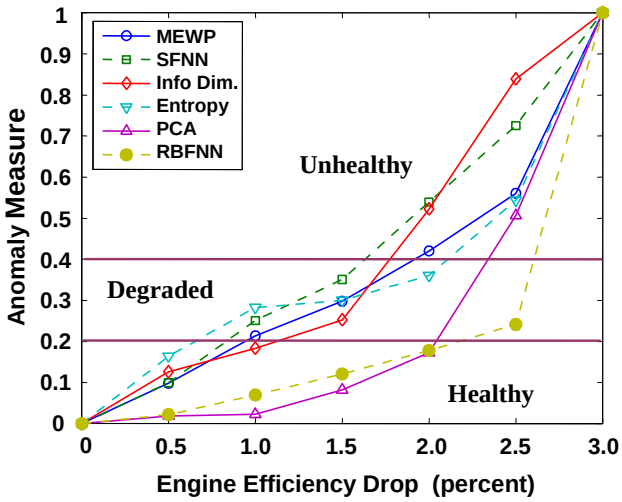


Figure 4.5. Comparisons of various methods for event generation

Computational complexity is an important issue for online implementation. The symbolic dynamic filtering (SDF) method based on both SFNN and MEWP are compared with more traditional techniques using RBFNN and PCA, and chaos theoretic methods. MEWP and Kolmogorov-Sinai entropy based method performs significantly efficient than the others. The SFNN method, on the other hand, have proven to be the most computationally intensive one among the investigated methods.

Chapter 5

Optimal DES Control of Aircraft Propulsion Systems

This chapter presents an application of the recently developed theory of language-measure-based discrete event supervisory (DES) control to aircraft propulsion systems. A two-layer hierarchical architecture is proposed to coordinate the operations of a twin-engine propulsion system. The two engines are individually controlled to achieve enhanced performance and reliability, as necessary for fulfilling the mission objectives. Each engine, together with its continuously varying control system, is operated at the lower level under the supervision of a local discrete-event controller for condition monitoring and life extension; the gain-scheduled feedback controller that is used to maintain the specified performance of the turbofan engine is kept unaltered. A global DES controller at the upper level coordinates the local DES controllers for load balancing and health management of the propulsion system.

5.1 Introduction

As stated in Chapter 3, discrete-event dynamical behavior of physical plants is often modeled as regular languages that can be realized by finite-state automata [14]. This chapter focuses on development of intelligent decision and control algorithms based on the theory of Discrete Event Supervisory (DES) control [16, 15] for a twin-engine aircraft propulsion system.

The DES control system is designed to be hierarchically structured in the following sense. The continuously varying control system of each engine interacts with its own local DES controller for health monitoring and intelligent control; and the operational information is abstracted and reported to the propulsion-level DES controller that coordinates the operation of two engines. Furthermore, the propulsion-level supervisory control system allows interactions with exogenous inputs, such as human operators and inputs from other units (e.g., flight control, structural control, energy management, and avionic systems) of the vehicle management system for flexibility of making on-line modifications in the mission objectives. A feature of the proposed DES control system is that the supervisory control policy can be adaptively updated on-line at both engine and propulsion levels and that the system is tolerant of small anomalies and component faults [68].

Although the theory of DES control has been developed for almost two decades [15], only very few applications have been reported in literature. An apparent reason is that, until recently, no quantitative analytical tool was available for design and evaluation of DES controllers. The work reported here makes use of a quantitative measure of regular languages [46, 12], and is a novel application of hierarchical DES control synthesis for the non-linear complex dynamical system of twin-engine aircraft propulsion. The real-time implementation of the DES scheme is challenging because it requires integration of several disciplines [25] such as systems theory, computer hardware and software, and domain knowledge of gas turbine engine propulsion.

The chapter presents synthesis of a Discrete Event Supervisory (DES) system for intelligent decision and control of twin-engine aircraft propulsion. The DES control of the propulsion system is validated on a simulation test bed. Thus, feasibility of the DES concept is demonstrated for enhanced operation and control of twin-engine aircraft propulsion in the following areas:

- real-time decision-making for propulsion control (e.g., load balancing between engines);
- damage reduction (with no significant loss of performance) via life extending control;
- improvement performance, and reliability of the mission.

5.2 Description of the Test Bed for Propulsion System Simulation

This section presents the implementation of Discrete Event Supervisory (DES) control on a real-time simulation test bed of a twin-engine aircraft propulsion system. The objective is to validate the theory of optimal Discrete Event Supervisory (DES) control for a real-world nonlinear complex dynamical system.

The previously validated engine model is similar in complexity and details to that reported by Diao and Passino [73] and Modular Aero Propulsion System Simulation (MAPSS) model [83]. The reader may refer to Appendix A for details of the engine simulation. Given the proper inputs of throttle position, also known as power lever angle (PLA), and ambient conditions (e.g., altitude (h), Mach number ($\mathbf{M}$), ambient temperature (T_a)), nonlinear dynamics of real-time turbofan engine operation are represented as a component level model in the simulation test bed. Both steady-state and transient operations of the gas turbine engine are simulated in the continuous-time setting. The DES system treats the engine model together with its continuously varying multivariable controller as the plant for open loop (i.e., unsupervised) discrete-event behavior. The continuous-time gain-scheduling robust controller of the turbofan engine is kept unaltered in the DES control system.

A typical high-pressure ratio, dual-spool, low bypass, gas turbine engine is represented with a nonlinear, low bandwidth, performance model, which is used in applications of intelligent engine control. The components of this engine model consist of a single stage high-pressure ratio fan with variable inlet stator vanes, booster with independent hub and tip stator vanes, high-pressure mixed flow compressor, double-annular combustor, high- and low-pressure turbines, afterburner, and nozzle components [83, 84]. The open-loop engine model has three state variables, which are the low-pressure and the high-pressure rotor speeds, as well as the average metal (wall) temperature. Together with its ten actuators, total number of the states associated with the augmented plant model is twenty three.

A DES controlled propulsion system has been designed and implemented on a simulation test bed that consists of three networked computers using the client-server concept. One of the three computers hosts the propulsion system coor-

dinator for health monitoring of the engines and accordingly making intelligent decisions (e.g., load balancing and turning on/off of individual engines in the extreme cases). The other two computers run separate copies (which may or may not be different depending on the health of the individual engines) of the aircraft gas turbine engine simulation model including its (continuously varying) control system and a local discrete-event supervisor. The test bed is capable of simulating different dynamics for individual engines due to non-uniform operating conditions. Each of the engine simulators integrates the event-driven discrete dynamics modeled by finite-state automaton as well as time-driven continuous dynamics modeled by ordinary differential equations through continuous-to-discrete and discrete-to-continuous interfaces [60].

Figure 7.1 shows the supervisory control architecture of the engine propulsion control system. This software architecture is flexible to adapt other DFSA models and controller designs for other complex dynamic systems. Each major function in the simulation program has a modular structure as implemented on the three computers of the simulation test bed.

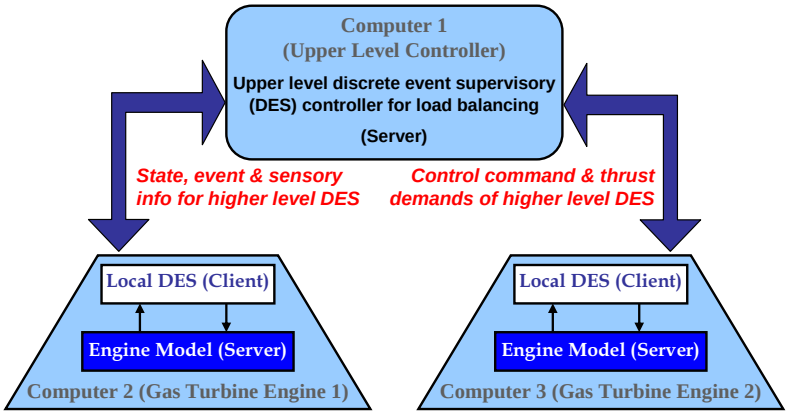


Figure 5.1. Supervisory Control Architecture of the Propulsion System

The C++ code of the multi-layer DES control system is superimposed on the existing FORTRAN simulation code of the turbofan engine model and the associated continuously varying engine control system through another C++ program. Specifically, the wrapper program interfaces the major inputs and outputs of the FORTRAN simulation code with the rest of the program in the C++ environment. This approach takes advantage of the available FORTRAN models as individual modules of the integrated C++ program without making any significant changes.

The FORTRAN code of the turbofan engine simulation program, which consists of high-order nonlinear differential and difference equations and supporting algebraic equations, has been designed for both steady-state and transient operations of a generic jet engine [73, 83]; and different control strategies have been reported in the literature [73, 60, 85]. This simulation code is a stand-alone program with a gain-scheduled feedback controller. The engine simulation model provides various sensor data (e.g., combustion chamber temperature and high-pressure and low-pressure turbine speeds) together with other critical information (e.g., simulation step size and simulation cycle number), which are collected by the C++ wrapper program and exchanged with the DES controllers through a typical message application protocol interface (API) communication routine. This communication protocol sends and receives message packages through TCP and/or UDP networks. The typical delay in this protocol interface is mainly due to the network communications and is found to be less than 1 ms. Since the engine and flight simulations use integration step sizes in the order of 20 ms, the communication delays do not have a major bearing on performance of the proposed control architecture.

Figure 5.2 shows the architecture of the engine level plant and DES controller implementation, which has two replicas, with possible different parameters and initial conditions, in two different computers. Figure 5.3 shows the organization of the propulsion level DES controller together with its own (discrete) event generator, which is implemented on a third computer and makes use of the messaging interface to communicate with the other two computers.

The DES controller design has two important components that serve as interfaces between the continuously-varying control system and the discrete-event supervisory controller - one is Event Generator and the other is Action Generator. Event Generator receives continuously varying sensor data from the engines. The data along with other information like estimated state and external inputs are used to generate events that, in turn, are inputs to unsupervised Deterministic Finite State Automata (DFSA) model of engine operation. The unsupervised DFSA model is constructed based on the operation scenario and the details are discussed later in Section 5.3. The state-based DFSA model serves as state estimator and provides information on engine states and (both controllable and uncontrollable) events for the discrete-event supervisor to take appropriate actions. Event behav-

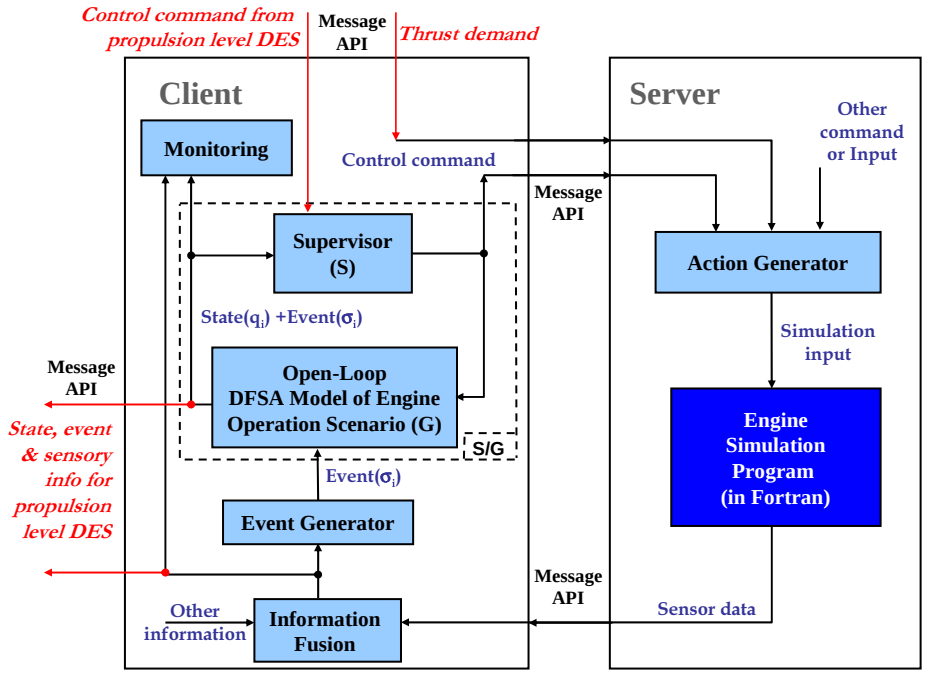


Figure 5.2. Engine Level Plant/DES Controller Implementation

ior in the state-based DFSA model is dependent on the state where the event is generated and not on the history or the path of how the state is reached.

The DES controller represents the control policy applied to the DFSA model of engine operation, and it could be a conventional DES controller based on the control specifications [15] provided by an experienced designer; alternatively, an optimal discrete-event supervisor can be designed by the quantitative method introduced in Chapter 2. In both cases, the DES controller takes the estimated states as inputs and generates control commands (of controllable event disabling or enabling) as outputs. The control commands are transmitted through a Message API communication routine to the Action Generator. The primary task of the action generator is to convert control commands from the supervisor into necessary input functions for the continuously varying plant.

5.3 Synthesis of DES Controllers

The unsupervised dynamics of engine operations are modeled as a DFSA, based on postulated engine operation scenarios. (Note that the model may change for

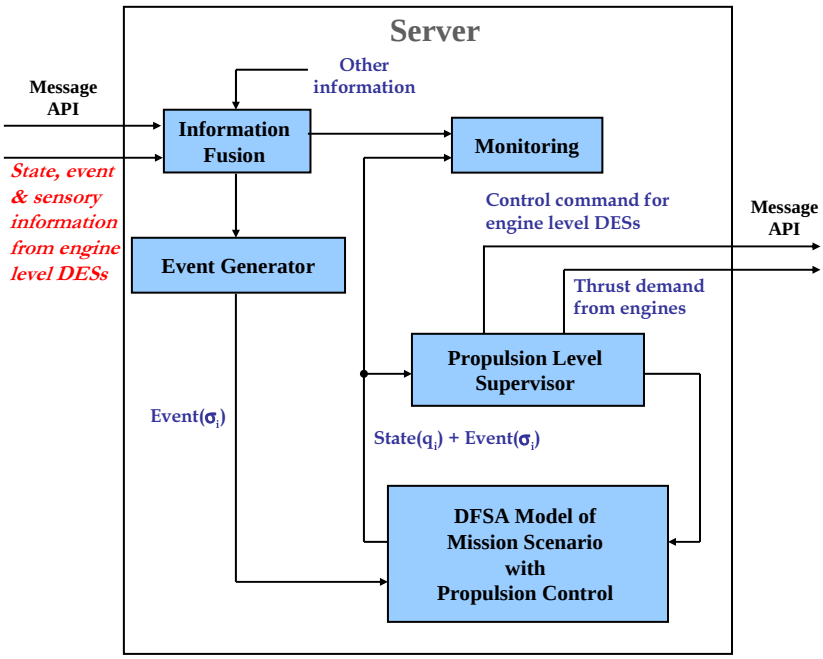


Figure 5.3. Propulsion Level DES Controller Implementation

different mission objectives.) The DFSA model assumes that a twin-engine aircraft is carrying out a routine surveillance mission. The mission abortion is allowed at certain states according to the operation scenario. Each engine of the aircraft is equipped with a continuous time controller which is supervised by a local DES controller. The primary objective of the local engine level DES controller is to strike the right balance between the conflicting demands of higher performance from upper level supervisor and limiting the damage to the engine. The upper (i.e., propulsion) level DES controller redistributes the load depending on the health of the engine and thrust demand placed by the pilot.

5.3.1 Engine Level DES Control

Figure 5.4 presents the DFSA model of the unsupervised engine operations for discrete-event supervisory control. Table 5.1 lists the events and Table 5.2 lists the plant states. The events and states for the DFSA models at the engine level are denoted by lower case letters (e.g., '**a**' is the start event and $\mathbf{q_1}$ is the engine start state). The engine can operate in two regimes, one is high performance regime (state $\mathbf{q_3}$), where the damage rate is also high. The other is low performance

regime (state $\mathbf{q_5}$) where the damage rate is low. In the high performance regime the engine has a tendency of going to state $\mathbf{q_4}$, where engine variables like combustor temperature have been observed to have oscillatory behavior. Temperature oscillations could be extremely harmful for engine health [86] and therefore must be avoided at all costs. Engine level controller chooses the regime of operation (state $\mathbf{q_3}$ or $\mathbf{q_5}$) depending on two factors: thrust requirement at the propulsion level and health of the engine.

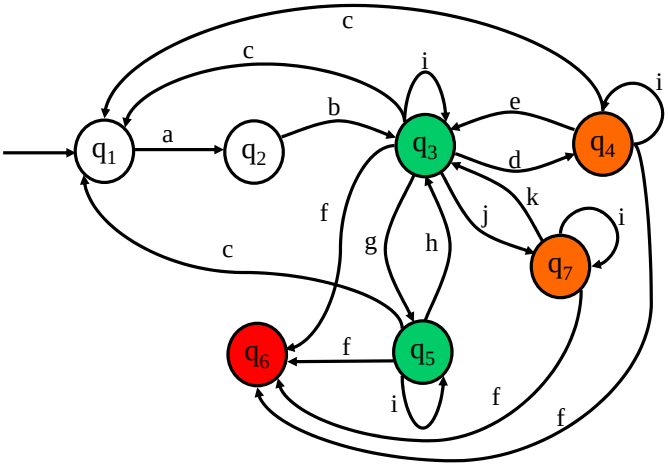


Figure 5.4. Unsupervised Plant DFSA Model at the Engine Level

Events	Description	Status
a	Start	Controllable
b	Warm up complete	Uncontrollable
c	Shut the engine	Uncontrollable
d	Detection of oscillations	Uncontrollable
e	Nozzle area reduction	Controllable
f	Engine fails	Uncontrollable
g	Reduce performance / reduce damage	Controllable
h	Increase performance / increase damage	Controllable
i	Remain in the state	Controllable
j	Reduce performance	Controllable
k	Increase performance	Controllable

Table 5.1. Event List for the Unsupervised Engine Model

Health of the engine is determined from the damage accumulation that is a function of high-pressure turbine gas inlet temperature and shaft speed; and in addition, at random time intervals spikes (i.e., sudden jumps) are introduced to simulate real world damage impacts in an engine. The DFSA model of the supervised engine operations is shown in Figure 5.5, where the dashed lines indicate the

State	Description	Status
$\mathbf{q_1}$	Engine Start	Unmarked
$\mathbf{q_2}$	Engine Warm up	Unmarked
$\mathbf{q_3}$	High Performance / High damage rate	Marked (good)
$\mathbf{q_4}$	Oscillations	Marked (bad)
$\mathbf{q_5}$	Low Performance / Low damage rate	Marked (good)
$\mathbf{q_6}$	Engine inoperable	Marked (bad)
$\mathbf{q_7}$	Low Performance / High damage rate	Marked (bad)

Table 5.2. State List for the Unsupervised Engine Model

controllable events that are disabled by the optimal control algorithm, *Absolute Disabling*, described in Chapter 2. The state $\mathbf{q_7}$ in Figure 5.5 becomes unreachable following the disabling action of the supervisor. Therefore, all transitions, originating from the state $\mathbf{q_7}$, are also shown with dashed lines as well as the state itself.

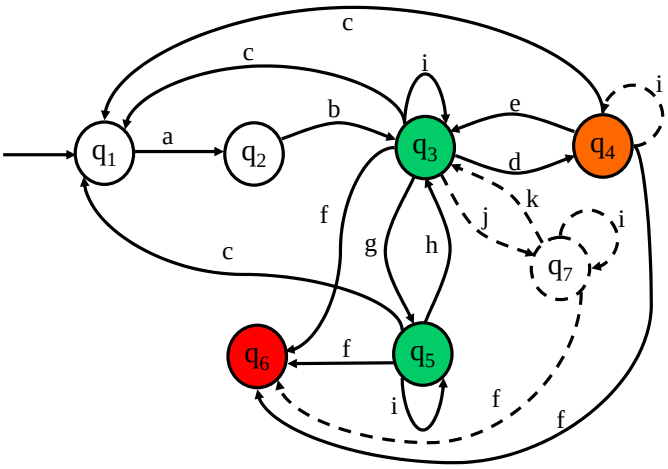


Figure 5.5. Supervised Plant DFSA Model at the Engine Level

5.3.2 Propulsion Level DES Control

One of the main tasks of the propulsion-level DES controller is to redistribute the load between two engines depending on the current health condition of each engine and the thrust demand. A DES controller is expected to ensure that this requirement is satisfied, regardless of being optimal or not. The propulsion-level DES controller acts in an advisory role for the mission-related decisions to enhance the mission performance. However, the ultimate decision for mission-related operations is left for the pilot.

Since the propulsion-level supervisor is designed to execute the key decisions at the engine level, the model for operating regimes of the propulsion system include the Cartesian product of the state sets of two (locally supervised) engine models. However, model order reduction via aggregation and deletion of unrealizable states is needed because the above Cartesian product will cause a very large number of resulting states. The events and states of the unsupervised DFSA model at the propulsion level are listed in Table 5.3 and Table 5.4, respectively. Events and states for the DFSA model at the propulsion level are denoted by upper case letters (e.g., **A** is the start event and $\mathbf{Q_1}$ is the aircraft on the ground state). Multiple occurrences of an event have been indicated as single events as seen in three instances (i.e., events **D**, **O** and **P**) in Table 5.3. The event "request to abort mission" is controllable while the events such as "request accepted", "abort the mission" and "shut down the engine" are uncontrollable events from the supervisor's perspectives.

Events	**Description**	**Status**
A	Start engines	Controllable
B	Warm up complete	Uncontrollable
C	One engine deteriorates	Uncontrollable
D	Redistribute the load (one engine is unhealthy)	Controllable
E	Both engines deteriorate	Uncontrollable
F	One good engine fails	Uncontrollable
G	Both engines fail	Uncontrollable
H	Increase performance	Controllable
I	Reduce performance	Controllable
J	Request to abort mission	Controllable
K	Request accepted	Uncontrollable
L	Request rejected (both engines are running)	Uncontrollable
M	Mission accomplished	Uncontrollable
N	Turn off engines	Uncontrollable
O	Redistribute load (both engines are unhealthy)	Controllable
P	Redistribute load (one engine has failed)	Controllable
Q	Request rejected (one engine has failed)	Uncontrollable
R	One bad engine fails	Uncontrollable

Table 5.3. Event List for the Propulsion Level DFSA Model

State	Description	Status
$\mathbf{Q_1}$	Aircraft on ground	Unmarked
$\mathbf{Q_2}$	Engines warming up	Unmarked
$\mathbf{Q_3}$	Both engines in High Performance operation	Unmarked
$\mathbf{Q_4}$	One engine in High one engine in Low Performance	Unmarked
$\mathbf{Q_5}$	Both engines in Low Performance operation	Unmarked
$\mathbf{Q_6}$	One engine stopped one engine in High Performance	Unmarked
$\mathbf{Q_7}$	One engine stopped one engine in Low Performance	Unmarked
$\mathbf{Q_8}$	Both engines failed	Marked (bad)
$\mathbf{Q_9}$	Decision for abort mission	Marked (bad)
$\mathbf{Q_{10}}$	Mission successful	Marked (good)
$\mathbf{Q_{11}}$	High damage detected for one engine	Unmarked
$\mathbf{Q_{12}}$	High damage detected for both engines	Unmarked

Table 5.4. State List for the Propulsion Level DFSA Model

5.4 Results of Simulation Experiments

Experiments were conducted on the simulation test bed, described in Section 5.2, to validate the DES control concept. Upon successful implementation of the software modules on the client and server computers, several sets of simulation experiments were performed. The first set of experiments was performed on the engine level DES controller to demonstrate how the engine component damage is reduced and consequently the engine life is enhanced. Then, the propulsion level DES controller that is built upon the engine level DES controllers is investigated.

5.4.1 Effects of Engine Level Supervision

Figure 5.6 exhibits the predetermined input profile that excites both unsupervised and supervised propulsion systems. Time responses of several outputs (e.g., combustor outlet temperature, high pressure turbine speed, net thrust of the engine, and fuel flow through the main burner) over a period of 12 minutes were observed. The four plates in each of Figure 5.7 and Figure 5.8 exhibit the response profiles of the above set of plant variables for unsupervised and supervised engines respectively. A comparison of plots in Figure 5.7 and Figure 5.8 indicates that the DES control at the engine level eliminates the high frequency oscillations that are present in the unsupervised plant responses in Figure 5.7. The supervisor takes actions immediately upon detection of oscillations by an FFT algorithm. Without making any alterations in the gain-scheduled controller of the engine, the supervi-

sory actions are implemented as a piecewise constant term in the reference input to the (continuous gain-scheduled) controller. In the continuous control system, there are seven summation points for the feedback loop, each providing a reference signal for a specific actuator. It is found that the booster vane angle and nozzle area manipulations have the most significant effects on the response of engine variables [68]. In the work reported by Yasar and Ray [87], nozzle area is decreased by 20% of the nominal value. Due to supervisors actions, the potentially sustained oscillations are quenched in less than 30 s after oscillations are observed, and the engine operation is brought to steady state. Thus, sustained oscillations are practically non-existent in the supervised plant responses in Figure 5.8. Since high-frequency oscillations of temperature and pressure are the primary sources of fatigue crack damage in the turbine blades, disks, and stationary vanes, the supervisory control becomes very effective for mitigation of structural damage in the engine components. In contrast, in the unsupervised case, the engine health would be adversely affected if the propulsion system is operated in this way to achieve the mission objectives.

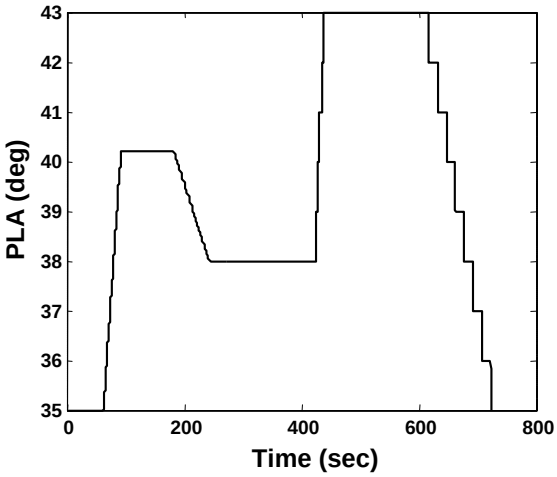


Figure 5.6. Power Lever Angle Input

5.4.2 Effects of Propulsion Level Supervision

The propulsion-level DES controller has three main tasks. The first task is the intelligent decision making and control of the twin-engine aircraft propulsion systems for mission execution; the second task is to improve the overall mission and

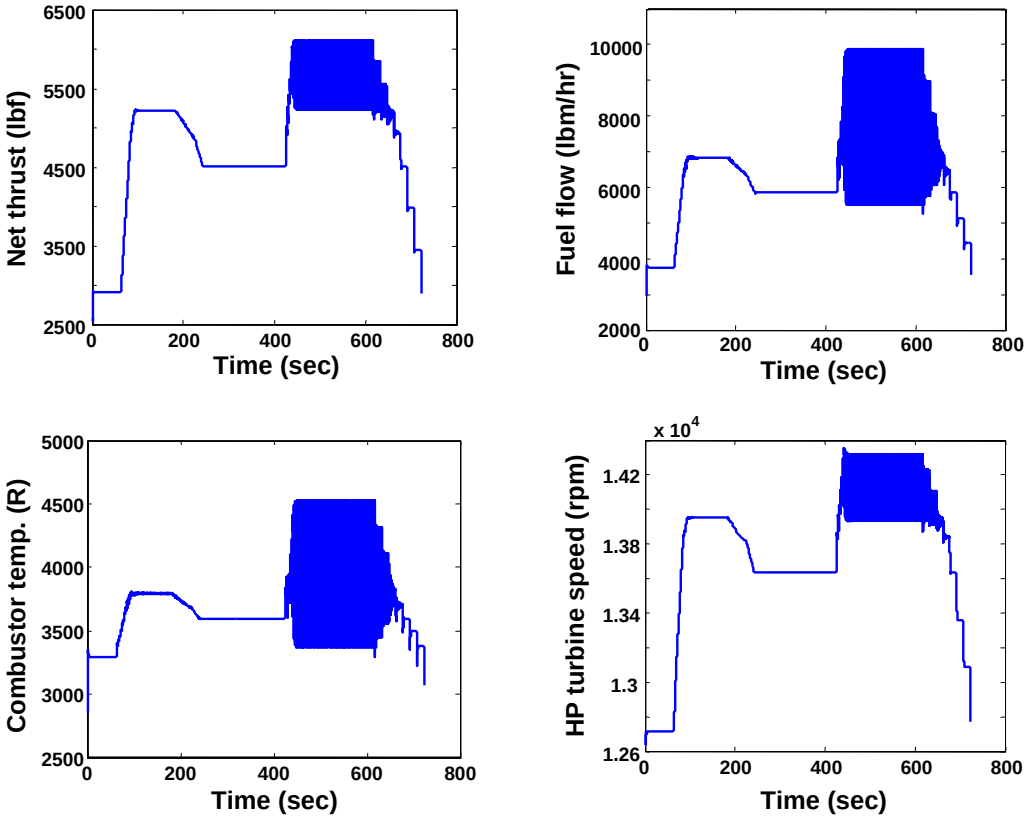


Figure 5.7. Simulation Output for the Unsupervised Case

operational behavior so that engine health can be enhanced via damage reduction; and the third task is load balancing between two engines so that the propulsion system produces the thrust demanded by the flight control system while attempting to enhance engine life. The issue of load balancing becomes even more important when the health conditions of two engines are significantly different (one can be in "bad condition" and the other in "good condition"). If the situation comes to this point, then the aim of the DES controller is judicious redistribution of the load between two engines such that the "bad engine" carries lower load than the "good engine", subject to the condition that the total thrust output of the engines satisfies the mission and safety requirements. To satisfy these design requirements, the propulsion-level DES controller [25, 15], which may or may not be optimal, is designed and implemented over the unsupervised plant model.

Figure 5.9 and Figure 5.10 shows the simulated outputs of two engines, where both engines are in "good" health condition at the start of simulation. The dam-

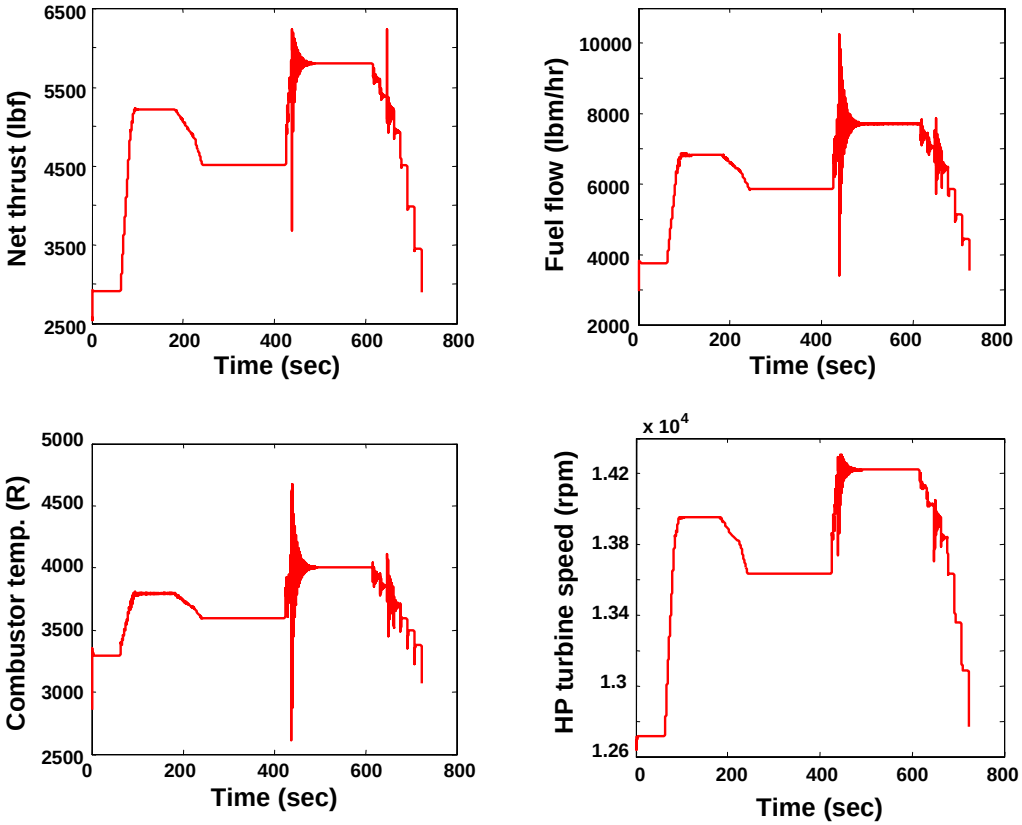


Figure 5.8. Simulation Output for the Supervised Case

age increment is a dimensionless quantity denoting the damage accumulation as percentage of the nominal value at each simulation cycle and is attributed to the turbine blades. As the mission progresses, due to an injected fault in Engine 2, the engine deteriorates and starts to run at "bad" health condition after 330 s. After 630 s, another fault is injected in Engine 1 and this engine also deteriorates. Thus, the load distribution of the engines varies in three regions (see Figures 5.9 and 5.10). In the beginning, the load is equally distributed in Region 1. Then the "good engine" (Engine 1) takes the responsibility of producing higher thrust as seen in Region 2 in the plates of Figures 5.9 and 5.10. Later on, both engines are again loaded equally as seen in Region 3 when it is not advisable to impose uneven thrust requirements. The first damage event affects the performance of Engine 2 after approximately 50 s in order to realize the thrust demand of the pilot. At this point, the full thrust demand cannot be satisfied if the load in Engine 2 is reduced. While the thrust produced in Region 2 by each engine is not the same, the

most critical impact of the uneven load distribution is formation of excess moment along the yaw axis of the aircraft. However, it might be possible to counteract this situation by adjusting the control surfaces of aircraft [88].

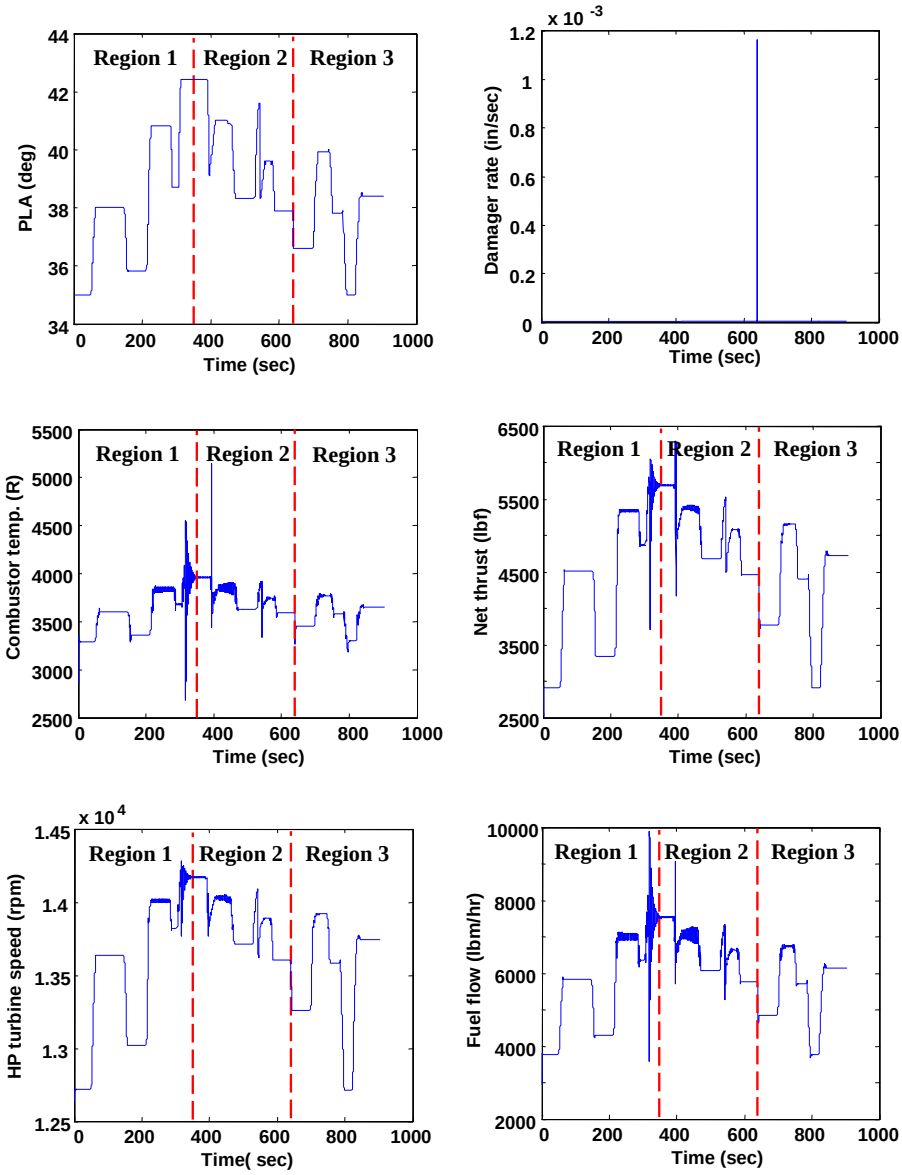


Figure 5.9. Effect of Conventional Propulsion Level DES Controller on Engine 1

5.4.3 Evaluation of DES Controllers

The language measure, described in [46, 12], has been used to quantitatively evaluate the impact of the propulsion-level DES controller on the overall mission be-

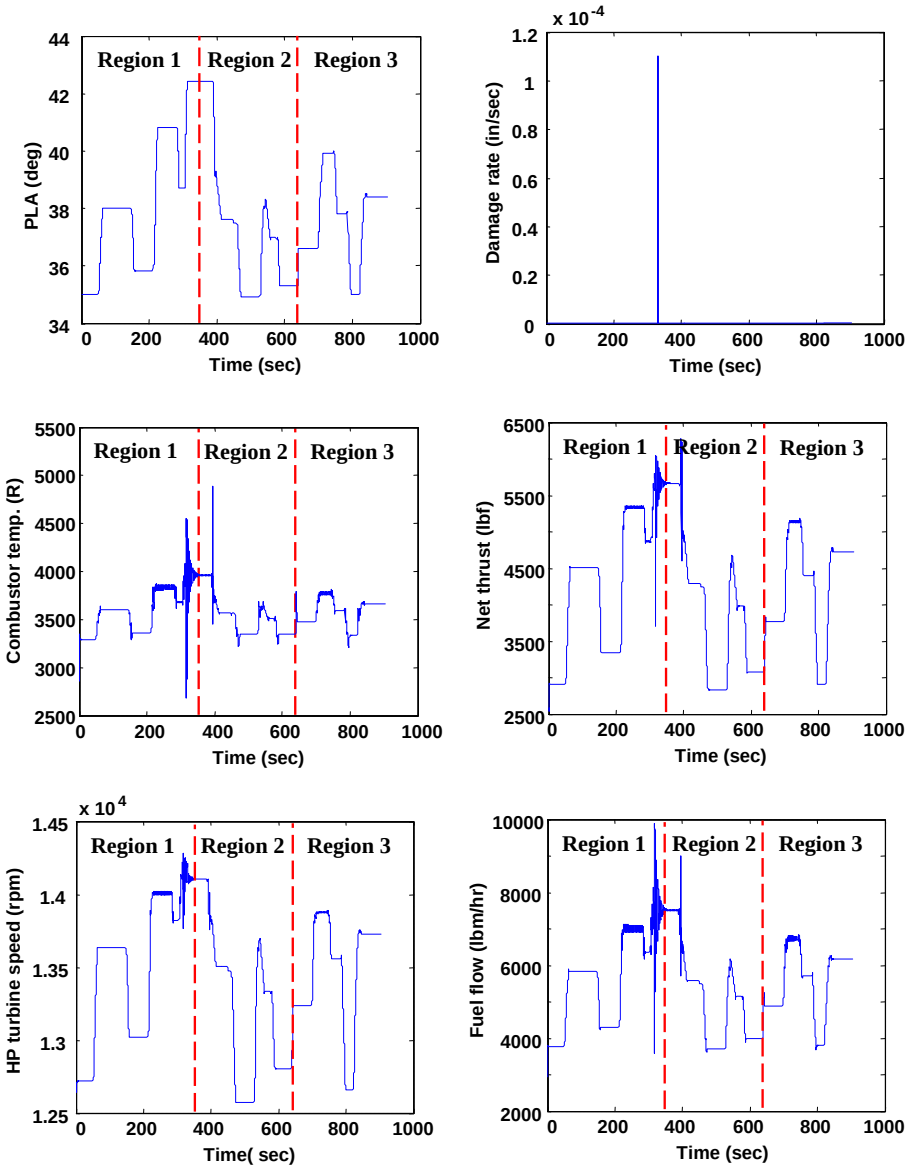


Figure 5.10. Effect of Conventional Propulsion Level DES Controller on Engine 2

havior. Given the state weights and the state transition probabilities, the language measure provides a quantitative performance measure of the controllers. Supporting information and pertinent mathematical background are given in the Chapter 2.

5.4.3.1 Identification of state transition probabilities

Quantitative analysis of DES controllers and synthesis of optimal DES controller require the identification of the state transition cost matrix. Similar to continuously varying dynamical systems (CVDS), one must use the techniques of system identification [65] to evaluate the parameters of the unsupervised DFSA plant model, i.e., the elements $\tilde{\pi}_{ij}^{0}$ of the event cost matrix $\tilde{\mathbf{\Pi}}$ (see Definition 2.4.4). As the number of experiments increases, the identified event costs tend to converge within an appropriately specified tolerance. For stationary operation of the engine, since conditional probabilities of the events are assumed to be time-invariant, the identified event costs and their uncertainty bounds can be determined. The probabilities of the events such as deterioration and failure of an engine are triggered by the event generation algorithm, based on the sensory information. Nevertheless, it is not possible to simulate some events, such as acceptance or rejection of the request by the pilot, without a random event generator. The randomization used in triggering this type of events has certainly an effect on the identified event costs, but not directly, since sensor-based events are not induced by this randomization [46]. As a typical case, Figure 5.11 presents identification of event costs at state $\mathbf{Q_5}$, where both engines are in low-performance operation.

Fifty experimental runs were conducted for both the unsupervised plant G and the supervised plant S/G to construct the respective state transition cost matrices $\mathbf{\Pi}^{0}$ and $\mathbf{\Pi}^{S}$. The different visited states and the triggered events were monitored and plotted to obtain the particular $\tilde{\mathbf{\Pi}}$-matrices. After identifying the respective event cost matrices $\tilde{\mathbf{\Pi}}^{0}$ and $\tilde{\mathbf{\Pi}}^{S}$ of G and S/G, the state transition cost matrices $\mathbf{\Pi}^{0}$ and $\mathbf{\Pi}^{S}$ are formed using Definition 2.4.5. Table 5.5 lists the $\mathbf{\Pi}^{0}$-matrix of the unsupervised DFSA model at the propulsion level.

5.4.3.2 Selection of characteristic values

Given the state characteristic values and the state transition probabilities, the language measure serves as a theoretical performance measure for quantitative evaluation of DES controllers. The characteristic values are assigned based on the designer's perception of the importance of terminating on specific marked states. For the propulsion level DES controller, the weights of the states are selected

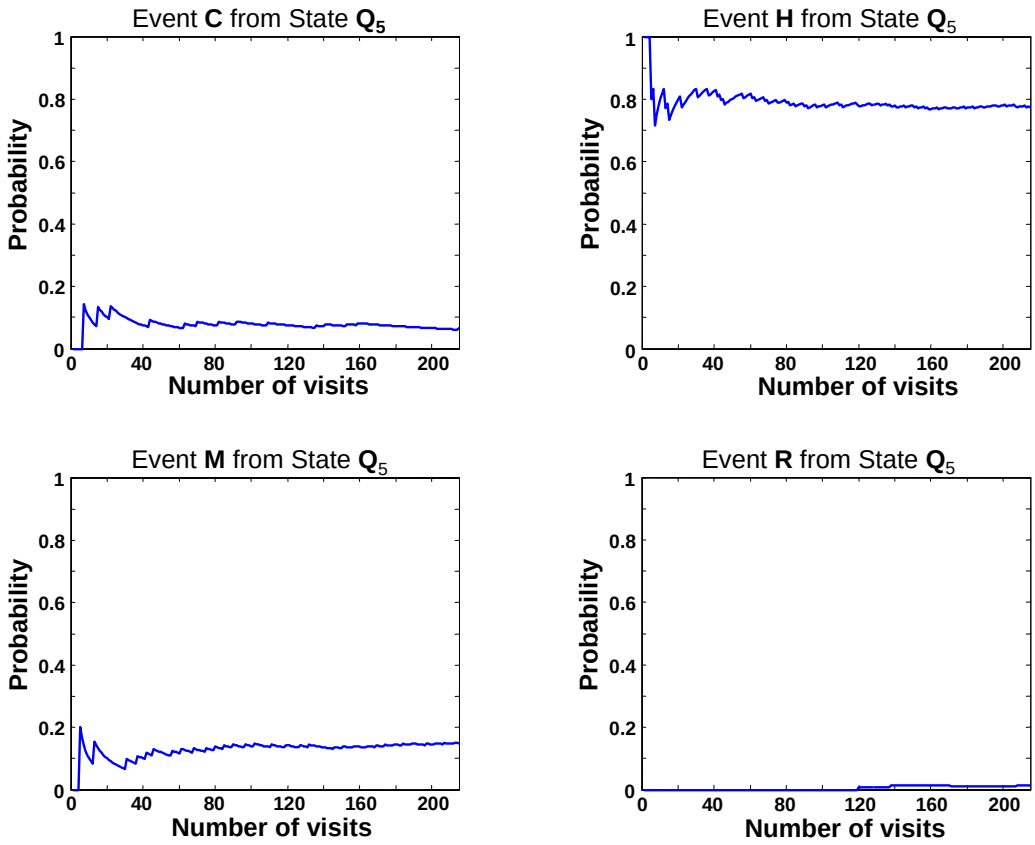


Figure 5.11. Convergence of Event Cost Identification

States	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}	Q_{11}	Q_{12}
Q_1	0	1.0	0	0	0	0	0	0	0	0	0	0
Q_2	0	0	1.0	0	0	0	0	0	0	0	0	0
Q_3	0	0	0	0	0.899	0.037	0	0	0	0	0.051	0.014
Q_4	0	0	0.500	0	0.500	0	0	0	0	0	0	0
Q_5	0	0	0.772	0	0	0	0.014	0	0	0.149	0.065	0
Q_6	0	0	0	0	0	0	0.250	0.125	0.625	0	0	0
Q_7	0	0	0	0	0	0	0	0	1.0	0	0	0
Q_8	0	0	0	0	0	0	0	0	0	0	0	0
Q_9	0.415	0	0	0	0	0	0	0	0	0	0.512	0.073
Q_{10}	1.0	0	0	0	0	0	0	0	0	0	0	0
Q_{11}	0	0	0	0.044	0.370	0	0	0	0.587	0	0	0
Q_{12}	0	0	0	0	0.333	0	0	0	0.667	0	0	0

Table 5.5. Π matrix of the Propulsion Level DFSA Model

according to each state's importance (contribution) to the mission management as:

$$\bar{\chi} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1.000 & -0.188 & 0.300 & 0 & 0 \end{bmatrix}.$$

The non-marked states have no direct bearing on the propulsion performance and hence each state (i.e., states $\mathbf{Q_1}$ to $\mathbf{Q_7}$, $\mathbf{Q_{11}}$ and $\mathbf{Q_{12}}$) has been assigned zero

weight. The marked states, $\mathbf{Q_8}$, $\mathbf{Q_9}$ and $\mathbf{Q_{10}}$, which do have direct bearing on the propulsion performance, are assigned non-zero weights as follows:

$\mathbf{Q_8}$: *Both engines failed*, State $\mathbf{Q_9}$: *Mission abort*, State $\mathbf{Q_{10}}$: *Mission successful*. They have relative weights of -1, -0.188 and 0.3 respectively. The bad marked state, $\mathbf{Q_8}$, is assigned the characteristic value of -1.0 because the aircraft will most likely be destroyed (because of both engines being non-functional) if the DFSA terminates on this state. On the other hand, the good marked state, $\mathbf{Q_{10}}$: *Mission successful*, is assigned the characteristic value of +0.3 based on its relative importance to the loss of the aircraft. Therefore, by choosing the characteristic values in this way, loosing one aircraft is made equivalent to approximately three successful missions. The other bad marked state, $\mathbf{Q_9}$: *Decision to abort mission*, has also a negative characteristic value which signifies the importance of this state relative to a successful mission and a possible loss of the aircraft. The selection of characteristic value for mission abort is to quantitatively match the theoretical (i.e., language measure-based) performance of the unsupervised plant with its experiment-based performance (which will be introduced in the next section). The corresponding performance measures for the supervised cases must also match with this selection provided that the plant is modeled appropriately and the language measure parameters, $\mathbf{\Pi}$-matrix and $\bar{\chi}$-vector, are correctly assigned.

Using the relation $\bar{\mu} = [I - \mathbf{\Pi}]^{-1}\bar{\chi}$ derived in Chapter 2, the language measures (i.e., the theoretical performance of the propulsion system) are calculated for unsupervised plant model G and DES controlled plant S/G as $\mu_{unsupervised} = 0.1260$ and $\mu_{supervised} = 0.2055$ respectively. It is seen that the DES controller used in the propulsion level has a positive effect on the mission behavior of the propulsion system.

5.4.3.3 Optimal DES controller synthesis and evaluation

After identifying the event cost matrix $\tilde{\mathbf{\Pi}}^0$ of the unsupervised plant G, the state transition cost matrix $\mathbf{\Pi}^0$ is constructed using Definition 2.4.5. State transition cost matrix is basically the only unknown input for the optimal control algorithm to design the optimal DES controller, since the other necessary parameters, such as the characteristic vector $\bar{\chi}$, are selected by the designer based on the design requirements. Therefore, given the state transition cost matrix $\mathbf{\Pi}^0$ of the unsuper-

vised plant and the state characteristic vector $\bar{\chi}$, the optimal DES controller can be synthesized as described in Chapter 2.

Table 5.6 lists the iterations of optimal control synthesis for the propulsion level supervisory control where the first column belongs to the unsupervised plant G. The performance measure of the unsupervised plant is negative at the states $\mathbf{Q_6}$, $\mathbf{Q_7}$, $\mathbf{Q_8}$, $\mathbf{Q_9}$, $\mathbf{Q_{11}}$, and $\mathbf{Q_{12}}$ as indicated by the bold scripts in Table 5.6. All controllable events leading to these states are disabled and the resulting performance measure at Iteration 1 shows sign change at states $\mathbf{Q_7}$, $\mathbf{Q_{11}}$, and $\mathbf{Q_{12}}$ as indicated by italics in Table 5.6. All controllable events leading to these states are now re-enabled for further increase in performance at Iteration 2. However, there is no sign change in the performance vector between Iteration 1 and Iteration 2, which immediately shows that the algorithm converged to the optimal solution after this iteration. The synthesis is complete in Iteration 2 (i.e., there is no need to go for the Iteration 3) because there is no sign change; moreover, the performance vector at Iteration 2 shows also no improvement after the previous iteration.

	Unsupervised Plant	**Iteration 1**	**Iteration 2**
State Q_1	0.1260	0.2452	0.2452
State Q_2	0.1273	0.2477	0.2477
State Q_3	0.1286	0.2502	0.2502
State Q_4	0.1415	0.2617	0.2617
State Q_5	0.1572	0.2785	0.2785
State Q_6	**-0.2555**	**-0.1238**	**-0.1238**
State Q_7	**-0.1510**	***0.0000***	0.0000
State Q_8	**-1.0000**	**-1.0000**	**-1.0000**
State Q_9	**-0.1525**	**-0.0353**	**-0.0353**
State Q_{10}	0.4248	0.5428	0.5428
State Q_{11}	**-0.0250**	***0.1132***	0.1132
State Q_{12}	**-0.0488**	***0.0919***	0.0919

Table 5.6. Iterations for Optimal DES Controller Synthesis

The performance of the optimal controller was compared with that of unsupervised plant G and the supervisor S that was designed using the conventional procedure [16, 15]. Theoretical performance of the supervisors can be associated with the language measure of each supervisor, as described previously in Chapter 2. The language measure of the unsupervised plant and that of the supervised plants at the propulsion level are listed in Table 5.7 that also shows a close agreement between the analytically generated language measures and experimentally

determined values.

	Unsupervised	**Supervised**	**Optimal**
Language Measure (μ)	0.1260	0.2055	0.2452
Performance Index (ν)	0.1256	0.1959	0.2480

Table 5.7. Language Measure and Performance for Different Supervisors

Results of simulation experiments have been used to validate the DES controller performance based on the language-theoretic analysis. The experimental performance index is determined as a function of the relative weights of the visited states. The mission outcomes of the unsupervised and supervised propulsion systems were recorded during the simulated missions. The numbers of missions ending at both good and bad marked states were multiplied by the respective relative weights of the states. That is, the experimental performance measure is calculated as

$$\nu = \frac{\sum_{i=1}^{N} n_i \cdot \chi_i}{N} \tag{5.1}$$

where n_i is the number of terminal visits to the i^{th} state, and N is the total number of experiments. The performance of the unsupervised plant and the other two supervisors, namely, conventionally designed supervisor [15] and language-based optimal supervisor [12], are compared based on the observations of mission execution on the simulation test bed. The experimental evaluations of the performance for different supervisors are presented in Table 5.7. Both theoretical and experimental (simulation) evaluations of DES controllers provide better mission management under optimal supervision. It is seen that the theoretical performance of the supervisors is in quantitative agreement with the experimental results, presented in Table 5.7. Optimal DES controller, synthesized using the *Absolute Disabling* algorithm described in Chapter 2, has the highest theoretical performance (i.e., highest language measure) among all controllers. The results of the simulation experiments concur with the theoretical measure of the controllers in the sense that optimal supervisor yields the best mission performance.

Remarks on the optimal supervisor design: At the first step of the iterations, the performance measure of the unsupervised plant is negative at states $\mathbf{Q_6}$, $\mathbf{Q_7}$, $\mathbf{Q_8}$, $\mathbf{Q_9}$, $\mathbf{Q_{11}}$, and $\mathbf{Q_{12}}$. Therefore, controllable transitions to these states should be disabled. It is observed that one engine is lost at the states $\mathbf{Q_6}$ and $\mathbf{Q_7}$, im-

plying that the events leading to these states would cause the "one engine failed" condition which is uncontrollable and hence cannot be disabled. Similarly, the only transition to the state $\mathbf{Q_8}$ causes the "both engines failed" condition, which is also uncontrollable; hence, this transition cannot be disabled either. The transitions leading to states $\mathbf{Q_{11}}$ and $\mathbf{Q_{12}}$ are the "deterioration of engines" condition, which is directly related to the damage information received from the engine-level DES control system. Based on this information, the propulsion-level supervisor makes a decision on the health condition of the individual engines. Evidently, this kind of sensory events are uncontrollable, so is the engine deterioration event. The remaining state that should be investigated is the state $\mathbf{Q_9}$: *Mission abort*. The optimal control algorithm, *Absolute Disabling*, disables all mission abort requests, which significantly increases the performance of the mission behavior with increased risk of losing the aircraft. The simulation experiments show that the mission performance at the propulsion level increases under the optimal DES controller, albeit with an increased probability of aircraft loss. Table 5.7 shows a close agreement between the analytically generated language measures and experimentally determined performance data. It should be noted that, the optimal control policy is likely to change if the elements (i.e., individual state weights) of the $\bar{\chi}$-vector are altered.

Chapter 6

Hybrid Integration of Flight and Propulsion Systems

Integration of flight and propulsion control systems in advanced aircraft has attracted much attention because of ever increasing demand on enhancement of performance and reliability. Since the underlying dynamic couplings and nonlinear interactions of flight and propulsion systems are too complex for real-time computation, hierarchical hybrid (i.e., combined continuously-varying and discrete-event) architecture is proposed for development of future generation control systems that will take advantage of these interactions for mission enhancement. While the original structures of continuously varying control systems for propulsion and flight are retained, Discrete Event Supervisory (DES) control facilitates decision making for aircraft operation. DES decisions regarding propulsion and flight control influence the performance and reliability of the entire vehicle control system due to interactions at the level of continuously varying dynamics. A two-level hierarchical DES control system is designed to supervise and coordinate the operation of twin-engine aircraft propulsion with flight dynamics. In essence, the propulsion system is integrated with the flight dynamical system such that the DES controller at the propulsion level of hierarchy provides load balancing of the engines as well as overall health and mission management of the aircraft propulsion. The parameter-scheduling dynamic-inversion controller stabilizes and drives the flight system in the vehicle operation envelope and compensates for potential unbalance and any other undesirable action, resulting from discrete event supervision of the

propulsion system. Results of real-time simulation on a test bed are presented to demonstrate efficacy of the proposed control concept.

6.1 Introduction

Development of intelligent decision and control algorithms, based on the theory of Discrete Event Supervisory (DES) control [26, 16], has paramount importance in many military and commercial applications. Especially, since interactions among complex systems may not be adequately described solely through an understanding of individual component's dynamics [2], human reasoning and tools of artificial intelligence are often applied for modeling and control of such integrated systems.

Recent theoretical developments in the DES control theory have led to several application examples emerged in the fields of aerospace and robotics [46, 89, 87]. However, these applications are largely confined to interactions between the DES control module and the continuous-time plant under discrete event supervision. Therefore, these DES control policies can be implemented without considering dynamical effects of the supervisory decisions on other continuous-time systems, which are coupled with the DES controlled plant.

For intelligent decision and control of a twin-engine aircraft propulsion system, a hierarchical structure is employed in the following sense: continuously varying control of an engine [90] interacts with its own local DES controller for detailed health monitoring and life extending control. The operational information is abstracted and reported to the coordinator for propulsion-level DES control that considers the operation of two engines as detailed in Chapter 5. DES control is also used for the problem of load balancing between engines during various operating conditions. Nevertheless, load distribution of the engines has impact on flight dynamics as well as engine dynamics.

The effects of dynamic coupling between the engine model and flight model may not be clearly understood unless these two systems are examined together [91]. Particularly, the effects of coupling between altitude and Mach number for a given input throttle position cannot be ignored. In Chapter 5 DES control of aircraft propulsion systems for intelligent decision-making regarding engine health monitoring and load balancing is examined; however, the objective of this inves-

tigation was to provide a baseline for design of aircraft control systems [92] and this study did not address the effects of coupling between the engine and flight parameters. Integration of flight and propulsion systems has been studied earlier [93, 94, 95] to take advantage of favorable interactions and to mitigate the effects of adverse interactions between the two systems. The available flight system models usually incorporate simplified engine models that often consist of only static functions and tabulated thrust values adjoined with low order linear dynamics. In order to study the interacting effects where propulsion is directly involved in controlling the aircraft, it is essential that the nonlinear dynamic response of the engines be adequately modeled [96, 97, 98].

Above discussions evince that it is logical to combine the DES decision and control of the propulsion system with the flight control system to enhance the interactions between propulsion dynamics and aerodynamics. It is also imperative to show that the propulsion-level supervisor is not affected adversely from the coupling between engine and flight dynamics. The objective of this paper is to observe the effects of supervisory decisions, which are taken in the propulsion level, on the aerodynamic control surfaces and develop a comprehensive control architecture to stabilize and drive the integrated system while operational behavior is supervised in discrete-event setting. The linear model-inverting architecture [99] is incorporated to stabilize the aircraft by adjusting the control surfaces to offset the possible adverse effects of discrete event supervision.

6.2 Engine Model and Supervisory Control

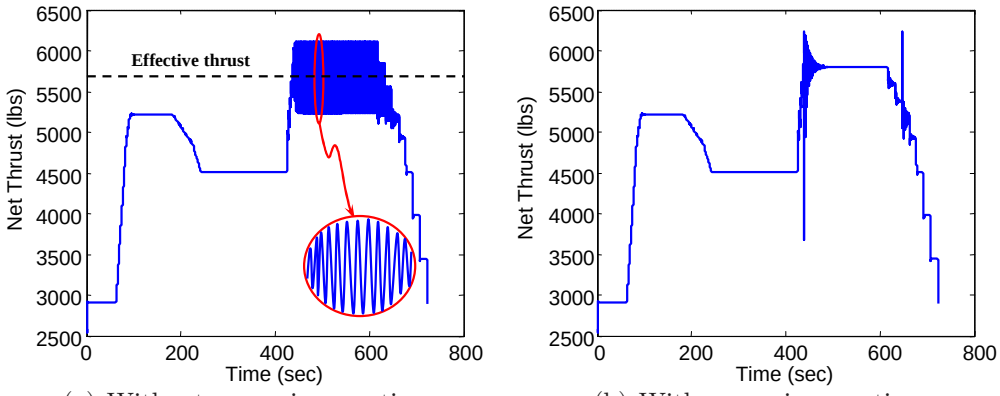
This section present the features of the generic gas turbine engine model employed as the plant for the DES controlled propulsion system, and the supervisory decisions for life extending control of the underlying system. The details of the hierarchical DES control system, its optimization procedure and implementation details are given in Chapter 5. The reader may refer to Appendix A for details of the engine simulation.

The DES control law has been validated on a networked test bed that simulates dynamic flight and propulsion conditions. The network consists of four computers communicating via a network hub, three of which host the continuously varying

plants and DES control units and the other being the pilot station. The plant dynamics in the simulation test bed are built upon the model of a generic turbofan gas turbine engine and a generic fighter aircraft model. The software architecture of the test bed is flexible to adapt piloted and autonomous flights with open-source flight simulator program FlightGear [100].

Regarding the engine operation, there are two types of possible supervisory decisions in the present architecture. The local supervisory decisions, including the adjustments of the booster inlet guide vane angle, variable nozzle area, bypass injector area, and propulsion level decisions including the equal or unequal distributions of the thrust demands from the engines. As reported in Chapter 5, local supervisor acts as a decision-maker for actuator positions under the thrust demand coming from propulsion-level supervisor while limiting the component damage. As an example, nozzle outlet area manipulations are introduced in the system to quench the instabilities when the engine is operated in elevated throttle positions. While stabilizing the engine dynamics, the net output thrust of the engine is not adversely affected by this action. Figure 6.1(b) presents the case where local DES controller decides to reduce the variable nozzle area of the engine by 20% upon the detection of high frequency instabilities in combustor temperature and high pressure turbine speed. High frequency oscillations are modeled as the primary cause of damage to the turbine blades and degrade the engine health. A comparison of the engine performance profiles in the two plates of Figure 6.1 indicates that combustion map instabilities are quenched in less than one minute by reducing the nozzle area, which is implemented as an additional bias term to the (electronic) summation junction in the actuator control software. In this way, the control law and the control command are left unaltered.

The propulsion-level DES controller behaves in an advisory role to enhance the mission performance for the mission-related decisions, such as to abort the mission. However, the ultimate decision for mission-related operations is left for the pilot. The other regulation imposed on the propulsion system by the propulsion-level supervisor, which also acts as a risk assessment unit for the health of overall propulsion, is based on the load entrusted to each engine [87]. Based on analytical tools developed for anomaly detection [68], the propulsion-level supervisor decides on the health conditions of individual engines, therefore determines the thrust

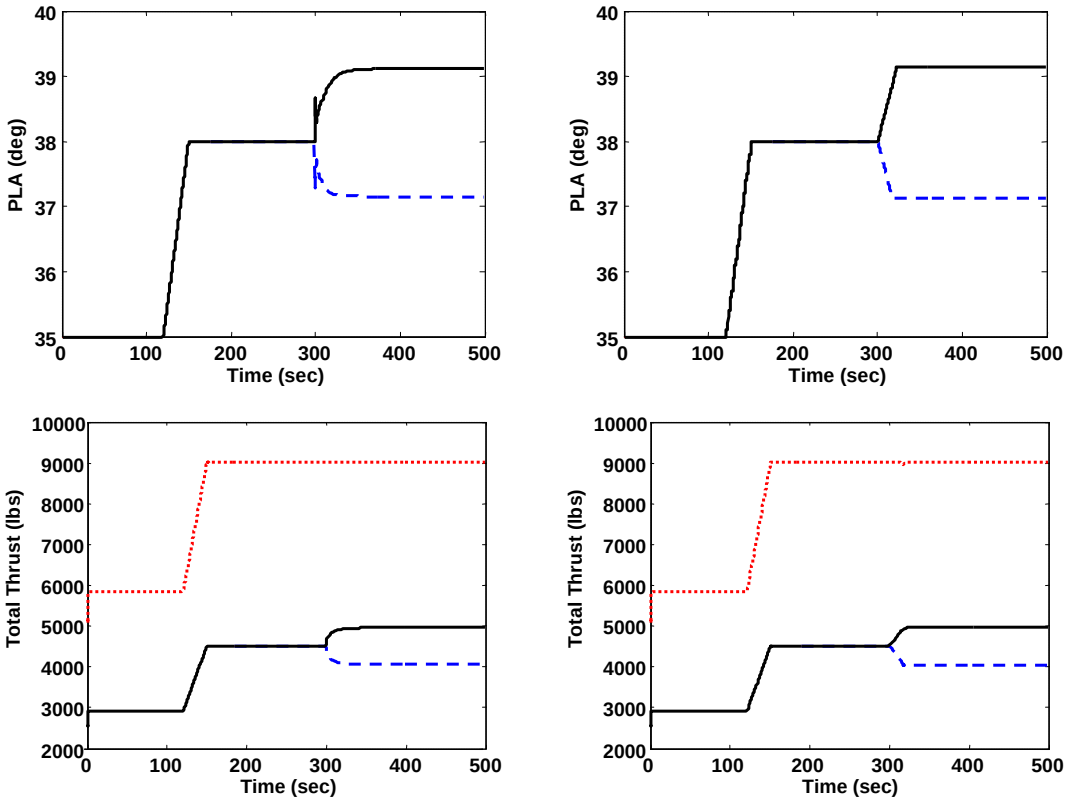


(a) Without supervisory action (b) With supervisory action

Figure 6.1. Thrust output of the engine with and without supervisory action

demands for local supervisors. The issue of load balancing becomes important when the healths of two engines are significantly different. For example, one engine could be in "bad health" while the other is in "good health". When such a situation is observed, the aim of the DES controller is judicious redistribution of the load between two engines such that the "unhealthy" engine carries lower load than the "healthy" one, subject to the condition that the total thrust output of the engines remains same. Simulation experiments on the test bed have been conducted for load distribution of 55% and 45% for healthy and unhealthy engines, respectively.

Load distribution can be achieved using either an outer feedback loop over the existing engine controller, or a feedforward structure For the feedback loop, a proportional and integral control is designed to track the reference thrust value, which is 45% or 55% of the total produced thrust just before load redistribution is enforced depending on the health condition of the engine. On the other hand, a proportional control is sufficient in the feedforward loop, which is designed to adjust the output thrusts to be the values mentioned before. Figure 6.2 shows the response of the system under these two structures, where one of the engines (dotted curve) becomes unhealthy at 300 seconds. From that point onwards, the healthy engine (solid curve) carries more responsibility in terms of thrust load, and the total thrust produced by the engines (dashed curve) before and after the distribution of the load is equal. It is concluded that both feedback control (Figure 6.2(a)) and feedforward control (Figure 6.2(b)) effectively serve the purpose of load distribution without violating the total thrust output constraint.



(a) Feedback (b) Feedforward

Figure 6.2. Thrust distribution response of engines

While the load is redistributed among the engines, another effect of this action is to introduce an excess moment along the yaw axis of the aircraft. Therefore, it is the responsibility of the flight control system to take care of this yaw moment. The details of this subject will be discussed later.

6.3 Aircraft Model and Integration

This section discusses the generic aircraft model including full-envelope, nonlinear aerodynamics, which is integrated with the existing propulsion system, together with the aerodynamic (control) surfaces of the model. Integration of the aircraft and engine models is also briefly described. The model represents a high performance supersonic aircraft, which is powered by two afterburning turbofan engines [101]. Figure 6.3 depicts a three dimensional view of the aircraft together with its

control surfaces and locations.

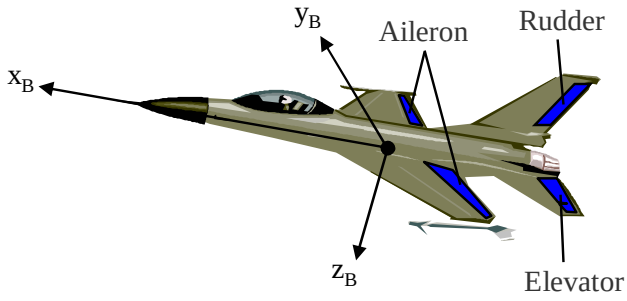
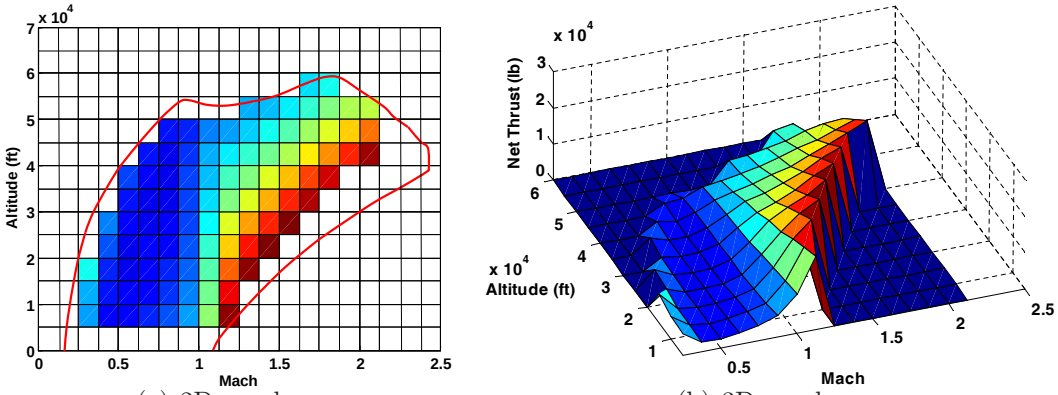


Figure 6.3. 3-D view of the aircraft and its control surfaces

The operational envelope of the aircraft for trimmed, straight-and-level, 1g flight is provided by Brumbaugh [101], however the actual model is tested for the full flight envelope to determine the real trim conditions and trust requirements to maintain these conditions. Figure 6.4 shows the provided flight envelope with solid curve and actual trimmed points of the model in 6.4(a) together with the net thrust output of the engines at these trimmed points in 6.4(b).



(a) 2D envelope (b) 3D envelope

Figure 6.4. Provided and the actual operational envelopes of the aircraft

The primary control surfaces of the aircraft consist of right and left ailerons, two horizontal elevators (stabilator) and a single vertical rudder, where elevators are capable of symmetric and differential motion. All the five actuators in the model corresponding to each control surface have identical first order response given by

$$G(s) = \frac{20}{s + 20}.$$

The actuator commands are not easily predictable because of the nonlinear interactions in the command paths.

The equations defining the aerodynamic model provide non-dimensional force and moment coefficients, which are functions of Mach number ($\mathbf{M}$), angle of attack (α), and angle of sideslip (β). The nonlinear equations of motion, used in the model, are general six degree of freedom equations representing the flight dynamics of a rigid aircraft flying in a stationary atmosphere over a flat, non-rotating earth. There are 13 state variables in the model for which the details of the derivation of state equations can be found in [102]. The details of the aircraft simulation program is also provided in Appendix B.

The linear gain-scheduled control law of the aircraft is synthesized based on the linearized models of the plant dynamics, which are obtained at various points of flight operation. An equally spaced grid of 36 points of operation is chosen in order to derive the linearized models of the flight dynamics including the stability derivatives and trim states as well as the trim values of the controls. Integration of the flight and engine models necessitates replacement of the propulsion model embedded in the flight dynamics with the DES controlled propulsion system in hand.

The aircraft under consideration consists of two engines, where each engine thrust vector is aligned with the body axis and acts at a point located 10 ft ($\sim$3 m) behind the vehicle center of gravity and 4 ft ($\sim$1.2 m) laterally from the centerline. The thrust produced by each engine is a function of altitude, Mach number, and throttle setting and is observed to have a maximum value of 24000 lbf ($\sim$10,900 kgf). Throttle position inputs to the engines are in degrees between the minimum of 20° and the maximum of 127°. The afterburner section begins to respond at a throttle position of 91° [101].

The generic model of the turbofan engine performs between throttle settings (or power lever angle (PLA)) of 21° to 50°, and the afterburner section of the model responds after 42.5°. The engine model can produce thrust output of 12600 lbf ($\sim$5730 kgf) at the maximum throttle setting. Therefore, it is mandatory to closely match the input-output relationships of the aircraft engine model and the generic engine model; the diagram in Figure 6 addresses this issue for integration of the aircraft and engine models. However, there is more than one option to

use the engine model as an integral part of the aircraft model. It is essential to decide on the functions $f_1(\bullet)$ and $f_2(\bullet)$, which map the power lever angles, PLA^{Flight} (PLA^F) to PLA^{Engine} (PLA^E), and thrusts, F_N^{Engine} (F_E^N) to F_N^{Flight} (F_F^N), respectively. One of the ways to obtain these mappings is first to decide on function $f_1(\bullet)$, and then match up the 3D thrust profiles of the models using the function $f_2(\bullet)$.

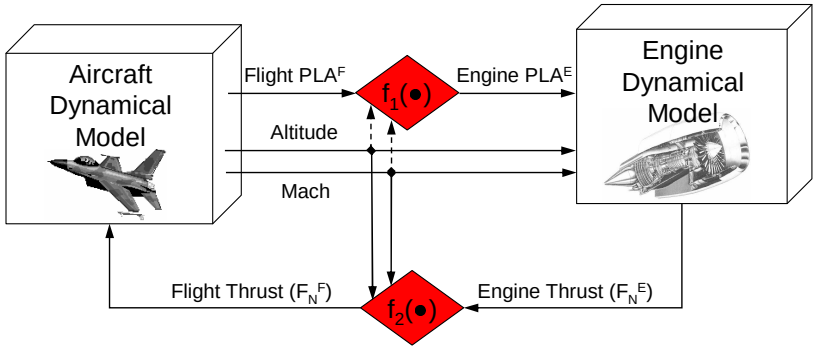
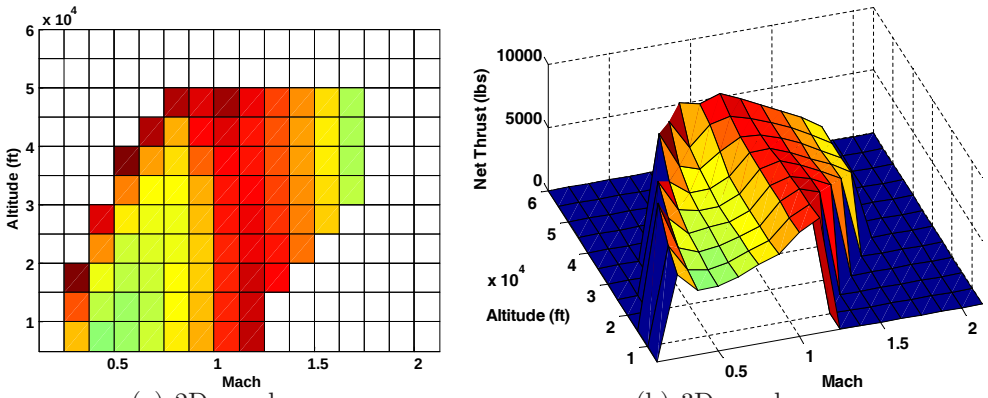


Figure 6.5. Integration of aircraft and engine models while preserving the input-output relationships

In practice the easiest way of mapping PLA^F to PLA^E is by an affine function $f_1(x) = ax + b$. Therefore, it is possible to attain the maximum and minimum values of the throttle settings. This mapping is used to convert all recorded PLA^F values to achieve the necessary thrust requirements which are provided in Figure 6.4(b). Net thrust output of the engine model for the whole flight envelope of the aircraft is given in Figure 6.6(b). As it is seen in Figure 6.6(a), engine model cannot be trimmed at some of the operating points of the flight model. These points should be avoided during the integrated flight. A comparison of Figure 6.4 and Figure 6.6 reveals that the function $f_2(\bullet)$, mapping F_N^E to F_N^F, is not easy to obtain, although point by point ratio of $K = F_N^F/F_N^E$ is readily available. Therefore it is useful to describe the relation between PLA^E and ratio K.

Figure 6.7 is a good representative of the engine systems in terms of thrust output and PLA input. Due to the controller dynamics embedded in the engine model, the response of the engine remains linear with respect to PLA input even if the afterburner starts to take action. In the aircraft model, however, the afterburner response of the engine is nonlinear. The effects of afterburner response can easily be observed around 42.5° of throttle position where the response of the



(a) 2D envelope (b) 3D envelope

Figure 6.6. Engine thrust outputs obtained in the aircraft flight envelope

models start to separate from each other considerably. As a result of the separation between models at the point where afterburner effects are observable, the curve fitting to the K vs. PLA relation needs a piecewise linear or an exponential form if more information is preserved. However, this method for the integration of the models involves additional nonlinear dynamics to the response of the overall system. This can be circumvented by using a linear fit to the data points of the relation K vs. PLA with the penalty of increasing scaling errors for especially high PLA values.

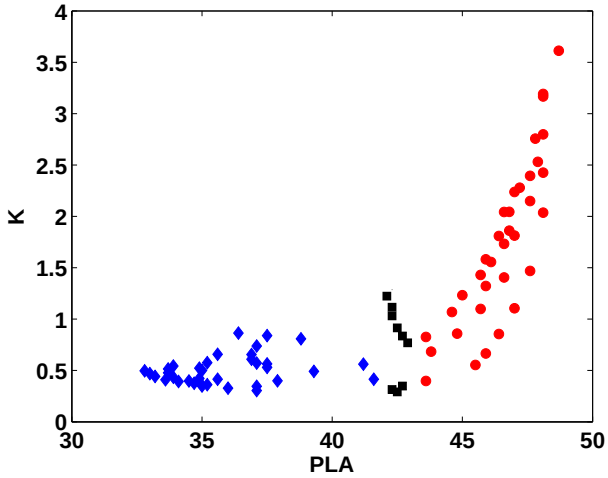


Figure 6.7. Thrust ratio relation as a function of throttle position

6.4 Parameter Scheduling - Dynamic Inversion Control Law

This section elaborates the control law that is used to maneuver the aircraft under different engine loading conditions. The parameter-scheduling type dynamic-inversion control law effectively stabilizes the aircraft in full operating envelope. The linear model inverting control architecture is often used in nonlinear aircraft control applications along with some adaptation rule [99, 103].

The short-period longitudinal dynamics of an aircraft can be approximated by the linear system:

$$\begin{bmatrix} \Delta\dot{\alpha} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} Z_\alpha & 1 \\ M_\alpha & M_q \end{bmatrix} \begin{bmatrix} \Delta\alpha \\ q \end{bmatrix} + \begin{bmatrix} Z_{\delta_e} \\ M_{\delta_e} \end{bmatrix} \Delta\delta_e + \begin{bmatrix} Z_{\delta_t} \\ M_{\delta_t} \end{bmatrix} \Delta\delta_t \tag{6.1}$$

This system of first order equations can be transformed to a single second order equation (by dropping the control rate terms)

$$\Delta\ddot{\alpha} = \left(Z_\alpha^2 + M_\alpha\right)\Delta\alpha + (Z_\alpha + M_q)q + (Z_\alpha Z_{\delta_e} + M_{\delta_e})\Delta\delta_e + (Z_\alpha Z_{\delta_t} + M_{\delta_t})\Delta\delta_t \tag{6.2}$$

If the following control law for the elevator deflection is used

$$\Delta\delta_e = \frac{\nu_\alpha - (Z_\alpha^2 + M_\alpha)\Delta\alpha - (Z_\alpha + M_q)q - (Z_\alpha Z_{\delta_t} + M_{\delta_t})\Delta\delta_t}{Z_\alpha Z_{\delta_e} + M_{\delta_e}} \tag{6.3}$$

then the pitch dynamics are reduced to $\Delta\ddot{\alpha} = \nu_\alpha$ where ν_α is called the pseudo-control and represents the desired angle-of-attack acceleration. The pseudo-control can be defined to follow an ideal response to pilot commands using a command filter. In addition, it can include feedback terms in order to reject disturbances with desired error dynamics.

The method described above is shown for a linear system. However, in practice the method can be applied to non-linear aircraft dynamics with reasonable accuracy using either an adaptive scheme [103, 104] or by scheduling the stability derivatives, trim states and controls with flight condition. In this case, a scheduling approach is used where the flight condition is represented by the altitude, h,

and Mach number, $\mathbf{M}$.

$$Z_\alpha, Z_{\delta_e}, Z_{\delta_t}, M_\alpha, M_q, M_{\delta_e}, M_{\delta_t}, \alpha_0, \delta_{e0}, \text{ and } , \delta_{t0} = f(h, \mathbf{M})$$

$$\text{where } \Delta\alpha = \alpha - \alpha_0, \Delta\delta_e = \delta_e - \delta_{e0}, \text{ and } \Delta\delta_t = \delta_t - \delta_{t0}$$

The desired angle-of-attack is given by the command filter

$$\begin{bmatrix} \dot{\alpha}_d \\ \ddot{\alpha}_d \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & 2\xi\omega_n \end{bmatrix} \begin{bmatrix} \alpha_d \\ \dot{\alpha}_d \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_n^2 \end{bmatrix} \alpha_{cmd} \tag{6.4}$$

The command filter is used to calculate the desired values of angle-of-attack and its first and second derivative ($\alpha_d, \dot{\alpha}_d,$ and $\ddot{\alpha}_d$) in order to achieve the desired second order response to the angle-of-attack commands from the pilot (α_{cmd}). The pseudo-control is then calculated using the following control law

$$\nu_\alpha = \ddot{\alpha}_d + K_P(\alpha_d - \alpha) + K_D(\dot{\alpha}_d - \dot{\alpha}_d) \tag{6.5}$$

Defining the tracking error as $\tilde{\alpha} = \alpha_d - \alpha$, the error dynamics are governed by

$$\ddot{\tilde{\alpha}} + K_D\dot{\tilde{\alpha}} + K_P\tilde{\alpha} = 0 \tag{6.6}$$

The proportional and derivative gains, K_P and K_D, can be selected to ensure stable second order error dynamics. A reasonable method of selecting the gains is such that the error dynamics have the same natural frequency and damping as the command filter $K_P = \omega_n^2$ $K_D = 2\xi\omega_n$. In this way, the dynamic response to a disturbance will be similar to pilot commands.

The lateral directional controller is designed using a similar method as described previously for longitudinal dynamics of an aircraft except the fact that the dynamic

inversion of the model is required. The dynamics care represented by:

$$\begin{bmatrix} \dot{p} \\ \dot{\beta} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} L_p & L_\beta & L_r \\ Y_p + \tan\alpha & Y_\beta & Y_r - 1 \\ N_p & N_\beta & N_r \end{bmatrix} \begin{bmatrix} p \\ \beta \\ r \end{bmatrix} + \begin{bmatrix} L_{\delta_a} & L_{\delta_r} \\ Y_{\delta_a} & Y_{\delta_r} \\ N_{\delta_a} & N_{\delta_r} \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} + \begin{bmatrix} L_{\delta_{dt}} \\ Y_{\delta_{dt}} \\ N_{\delta_{dt}} \end{bmatrix} \delta_{dt} + \begin{bmatrix} 0 \\ \frac{g}{V}\cos\theta\sin\phi \\ 0 \end{bmatrix} \tag{6.7}$$

Note that the term, δ_{dt}, represents differential thrust. It is important to account for this term so that the controller performs well when the engine load balancing changes.

The objective of the lateral-directional controller is to track commands for sideslip angle, β, and stability axis roll rate, $p_s = p\cos\alpha + r\sin\alpha$. As with the longitudinal controller, the last two equations are converted to a single second order equation in β. The linearized model is parameterized by angle-of-attack to account for stability axis roll:

$$\begin{bmatrix} \dot{p}_s \\ \ddot{\beta} \end{bmatrix} = \mathbf{A} \begin{bmatrix} p \\ \beta \\ r \end{bmatrix} + \mathbf{B_1} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} + \mathbf{B_2}\delta_{dt} + \mathbf{g} \tag{6.8}$$

where

$$\mathbf{A} = \begin{bmatrix} L_p\cos\alpha + N_p\sin\alpha & L_\beta\cos\alpha + N_\beta\sin\alpha & L_r\cos\alpha + N_r\sin\alpha \\ Y_\beta Y_p + (Y_p + \tan\alpha)L_p + (Y_r - 1)N_p & Y_\beta^2 + (Y_p + \tan\alpha)L_\beta + (Y_r - 1)N_\beta & Y_\beta Y_r + (Y_p + \tan\alpha)L_r + (Y_r - 1)N_r \end{bmatrix}$$

$$\mathbf{B_1} = \begin{bmatrix} L_{\delta_a}\cos\alpha + N_{\delta_a}\sin\alpha & L_{\delta_r}\cos\alpha + N_{\delta_r}\sin\alpha \\ Y_\beta Y_{\delta_a} + (Y_p + \tan\alpha)L_{\delta_a} + (Y_r - 1)N_{\delta_a} & Y_\beta Y_{\delta_r} + (Y_p + \tan\alpha)L_{\delta_r} + (Y_r - 1)N_{\delta_r} \end{bmatrix}$$

$$\mathbf{B_2} = \begin{bmatrix} L_{\delta_{dt}}\cos\alpha + N_{\delta_{dt}}\sin\alpha \\ Y_\beta Y_{\delta_{dt}} + (Y_p + \tan\alpha)L_{\delta_{dt}} + (Y_r - 1)N_{\delta_{dt}} \end{bmatrix}$$

$$\mathbf{g} = \begin{bmatrix} 0 \\ \frac{g}{V}(Y_\beta\cos\theta\sin\phi + \cos\theta\cos\phi\dot{\phi} - \sin\theta\sin\phi\dot{\theta}) \end{bmatrix}$$

The inversion control law is then given by:

$$\begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} = \mathbf{B_1}^{-1} \left(\begin{bmatrix} \nu_p \\ \nu_\beta \end{bmatrix} - \mathbf{A} \begin{bmatrix} p \\ \beta \\ r \end{bmatrix} - \mathbf{B_2} \delta_{dt} - \mathbf{g} \right) \tag{6.9}$$

The yaw axis controller is designed to achieve a second order sideslip response using a second order command filter and proportional derivative controller similar to that used for angle-of-attack. The desired stability axis roll rate response is first order. Therefore, a first order command filter is selected and a proportional plus integral compensator is used:

$$\nu_\beta = \ddot{\beta}_d + K_P(\beta_d - \beta) + K_D(\dot{\beta}_d - \dot{\beta}) \tag{6.10}$$

$$\nu_p = \dot{p}_{s_d} + K_P(p_{s_d} - p_s) + K_I \int (p_{s_d} - p_s)dt \tag{6.11}$$

The overall architecture of the parameter-scheduled dynamic-inversion control law for the flight dynamics is given in Figure 6.8.

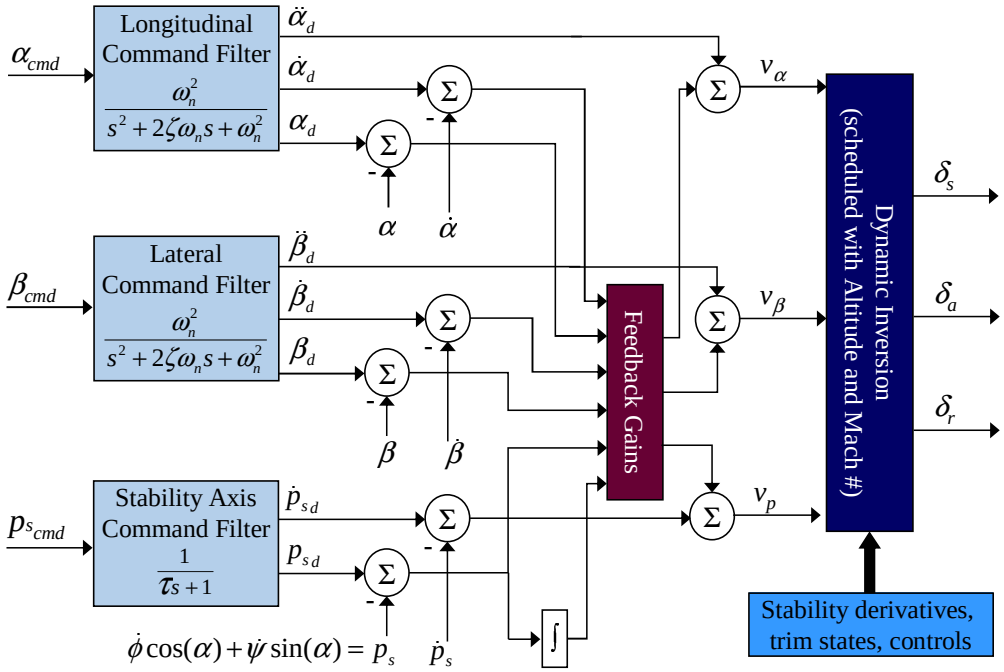


Figure 6.8. Schematic of the flight control law

6.5 Results of the Integrated System Simulation

The experimentation setup for the integrated flight system is designed on the simulation test bed to validate the seamless operation of DES controlled propulsion system with other modules of the aircraft. Upon successful implementation of the software modules, several sets of experiments were conducted.

6.5.1 Evaluation of Discrete Event Dynamics

The quantitative evaluation of discrete event dynamics are performed using the concepts of language measure and renormalized measure, which are introduced in Chapter 2, and optimal supervisor syntheses, based on *Absolute Disabling* and *Relative Disabling* criteria, are compared. The same automaton structure from Chapter 5 is utilized for integrated flight and propulsion simulation with one modification. Although that finite state model of the propulsion has accessibility and co-accessibility properties [16], which is an important condition for supervisor design [15], the deadlock situation existed at the state $\mathbf{Q_8}$ [16]. To avoid this, a transition from the marked state $\mathbf{Q_8}$: *Both engines failed* to initial state is added to the automata. Using the same conditional event probabilities as in Chapter 5 and the modified state transition function δ, the state transition probability matrix $\mathbf{P}$ is obtained as:

$$\begin{bmatrix}
0 & 1.000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1.000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.899 & 0.037 & 0 & 0 & 0 & 0 & 0.051 & 0.014 \\
0 & 0 & 0.500 & 0 & 0.500 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.772 & 0 & 0 & 0 & 0.014 & 0 & 0 & 0.149 & 0.065 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.250 & 0.125 & 0.625 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.000 & 0 & 0 & 0 \\
1.000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.415 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.512 & 0.073 \\
1.000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.044 & 0.370 & 0 & 0 & 0 & 0.587 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.333 & 0 & 0 & 0 & 0.667 & 0 & 0 & 0
\end{bmatrix}$$

For the propulsion level DES controller, the weights of the states are selected according to each state's importance to the mission management as:

$$\bar{\chi} = \begin{bmatrix} 0 & 0 & 0.05 & 0.02 & 0.05 & -0.05 & -0.05 & -1.00 & -0.20 & 0.30 & 0 & 0 \end{bmatrix}^T$$

State	$\bar{\mu}^{\mathbf{0}}$	$\bar{\mu}^{\mathbf{norm}}$	$\bar{p}$
$\mathbf{Q_1}$	0.293	0.029	0.074
$\mathbf{Q_2}$	0.326	0.033	0.074
$\mathbf{Q_3}$	0.362	0.036	0.322
$\mathbf{Q_4}$	0.356	0.036	0.003
$\mathbf{Q_5}$	0.384	0.038	0.320
$\mathbf{Q_6}$	-0.163	-0.016	0.012
$\mathbf{Q_7}$	-0.072	-0.007	0.007
$\mathbf{Q_8}$	-0.736	-0.074	0.002
$\mathbf{Q_9}$	-0.025	-0.003	0.061
$\mathbf{Q_{10}}$	0.564	0.056	0.048
$\mathbf{Q_{11}}$	0.129	0.013	0.068
$\mathbf{Q_{12}}$	0.100	0.010	0.009

Table 6.1. Language measure vectors and state probability vector of unsupervised plant

$\bar{\mu}^{\mathbf{norm}}$	**Open loop**	**Iteration 1**	**Iteration 2**
State $\mathbf{Q_1}$	0.0293	0.0300	0.0300
State $\mathbf{Q_2}$	0.0326	0.0333	0.0333
State $\mathbf{Q_3}$	0.0362	0.0370	0.0370
State $\mathbf{Q_4}$	0.0356	0.0363	0.0363
State $\mathbf{Q_5}$	0.0384	0.0391	0.0391
State $\mathbf{Q_6}$	**-0.0163**	**-0.0132**	**-0.0132**
State $\mathbf{Q_7}$	**-0.0072**	**-0.0050**	**-0.0050**
State $\mathbf{Q_8}$	**-0.0736**	**-0.0730**	**-0.0730**
State $\mathbf{Q_9}$	**-0.0025**	**-0.0014**	**-0.0014**
State $\mathbf{Q_{10}}$	0.0564	0.0570	0.0570
State $\mathbf{Q_{11}}$	0.0129	0.0144	0.0144
State $\mathbf{Q_{12}}$	0.0100	0.0117	0.0117

Table 6.2. Optimal supervisor iterations for *Absolute Disabling*

The language measure of the above automaton is computed by $\bar{\mu} = [I - (1 - \theta)\mathbf{P}]^{-1}\bar{\chi}$ and renormalized as $\bar{\mu}^{norm} = \theta\bar{\mu}$ by $\theta = 0.1$. The fixed point of the above operator $\mathbf{P}$ gives the steady-state probability vector $\bar{p}$. Table 6.1 provides the measure and state probability vectors for the unsupervised discrete event dynamics.

Therefore the average performance of the open loop plant can be computed as

$$\mu^{ave} = \langle \bar{p}, \bar{\chi} \rangle \equiv \bar{p}^T \bar{\chi} = 0.03178$$

Optimal supervisor iterations for *Absolute Disabling* and *Relative Disabling* strategies are presented in Table 6.2 and Table 6.3 respectively.

Table 6.4 presents the disabling action of the optimal supervisors synthesized by *Absolute Disabling* (denoted by circle - ○: enabled, ●: disabled) and *Relative Dis-*

$\bar{\mu}^{\mathbf{norm}}$	**Open loop**	**Iteration 1**	**Iteration 2**
State Q_1	0.0293	0.0420	0.0420
State Q_2	0.0326	0.0467	0.0467
State Q_3	0.0362	0.0518	0.0518
State Q_4	0.0356	0.0500	0.0500
State Q_5	0.0384	0.0548	0.0548
State Q_6	**-0.0163**	***0.0007***	0.0007
State Q_7	**-0.0072**	***0.0113***	0.0113
State Q_8	**-0.0736**	**-0.0622**	**-0.0622**
State Q_9	**-0.0025**	***0.0181***	0.0181
State Q_{10}	0.0564	0.0678	0.0678
State Q_{11}	0.0129	0.0428	0.0428
State Q_{12}	0.0100	0.0411	0.0411

Table 6.3. Optimal supervisor iterations for *Relative Disabling*

abling (denoted by square - □: enabled, ■: disabled). The resulting supervised plant behaviors are significantly different since the afore mentioned algorithms converge to two different optimal control solutions. However, if the average performance of the supervised plants are compared, the results reveal that none of the algorithms are expected to perform superior to each other. The respective average measures for the supervisors synthesized by *Absolute Disabling* and *Relative Disabling* are given as

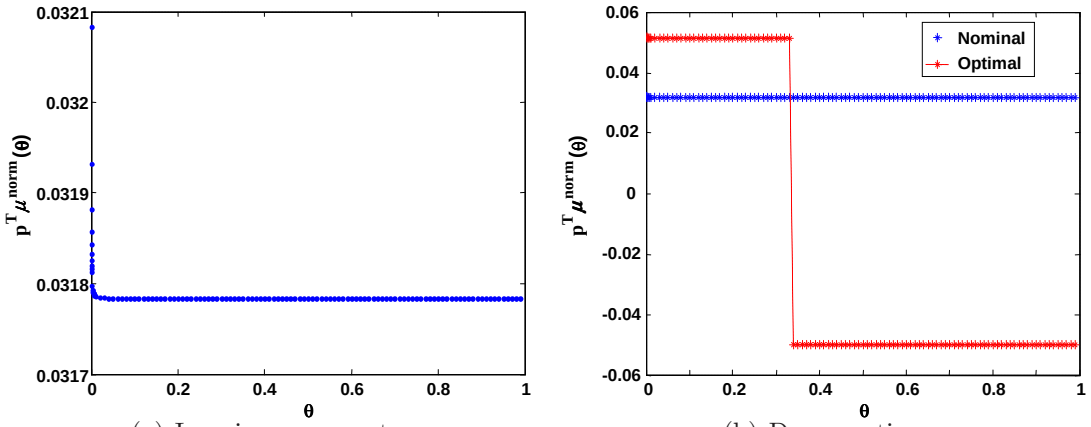
$$\mu^{ave,\Diamond} = \langle \bar{p}^{\Diamond}, \bar{\chi} \rangle = 0.05152$$
$$\mu^{ave,\blacklozenge} = \langle \bar{p}^{\blacklozenge}, \bar{\chi} \rangle = 0.05152$$

State	Event **H**		Event **I**		Event **J**		Event **P**	
Q_5	○	■	○	□	○	□	○	□
Q_6	○	□	●	□	●	□	○	□
Q_7	●	■	○	□	●	□	○	□
Q_{11}	○	□	○	□	●	■	●	■
Q_{12}	○	□	○	□	●	■	○	□

Table 6.4. Disabling map for two strategies

The expected long term behavior of the system is associated with $\mu^{ave} = \bar{p}^T \bar{\chi}$, which can be calculated using the invariance property of $\bar{p}^T \bar{\mu}^{norm}(\theta)$ for all $\theta \in (0, 1)$; and $\bar{p}^T \bar{\mu}^{norm}(\theta) = \bar{p}^T \bar{\chi}$. This property of the normalized measure is demonstrated in Figure 6.9(a). It should be noted that the value of $\bar{p}^T \bar{\mu}^{norm}(\theta)$ is not exactly equal to $\bar{p}^T \bar{\chi}$ for very small values of θ because of the numerical error in the

calculations. However, the optimal control algorithm may change the properties of the **P** matrix due to disabling of the controllable events such that the matrix may have multiple fixed points, $\bar{p}$. This degeneration of the **P** matrix indicates that design procedure will no longer yield to the optimal control law. In practice, it is possible to find the critical value of for which the optimal control design still works. Figure 6.9(b) depicts the computation of the critical value of θ, and it is observed that selection of $\theta = 0.1$ is valid for optimal control design.



(a) Invariance property (b) Degeneration

Figure 6.9. μ^{ave} is invariant for unsupervised plant but may change during optimal control synthesis

6.5.2 Evaluation of Continuous Dynamics

Loads of the two engines are redistributed when health condition of one engine is degraded [68] and the other engine functions normally. Due to this imbalance of the generated thrust, the response of the aircraft to the aerodynamic forces may change drastically. As a typical case, Figure 6.10 and Figure 6.11 show the changes in the attitude of the closed loop dynamical behavior of the aircraft when DES controller decides to change the load distribution due to health conditions of the engines. In Figure 6.10, thrust produced by individual engines is depicted (solid curve for healthy engine and dashed curve for unhealthy engine) as well as the total thrust (thick curve), and it is seen that load distribution is successfully accomplished in integrated system. It is also observed that the nominal controller successfully stabilizes the plant dynamics since both angle of attack (α) and angle

of sideslip (β) reach steady state values. However, the nominal flight control design for the aircraft does not account for the differential thrust. Therefore, in Figure 6.11, the flight orientation angles are adversely affected from load distribution; and the situation manifests in both roll and yaw angles as a shooting behavior from the nominal condition immediately after the thrust redistribution. As a result, the yaw angle settles to a steady state value of $-38°$ which leads to divergence from nominal path of the aircraft.

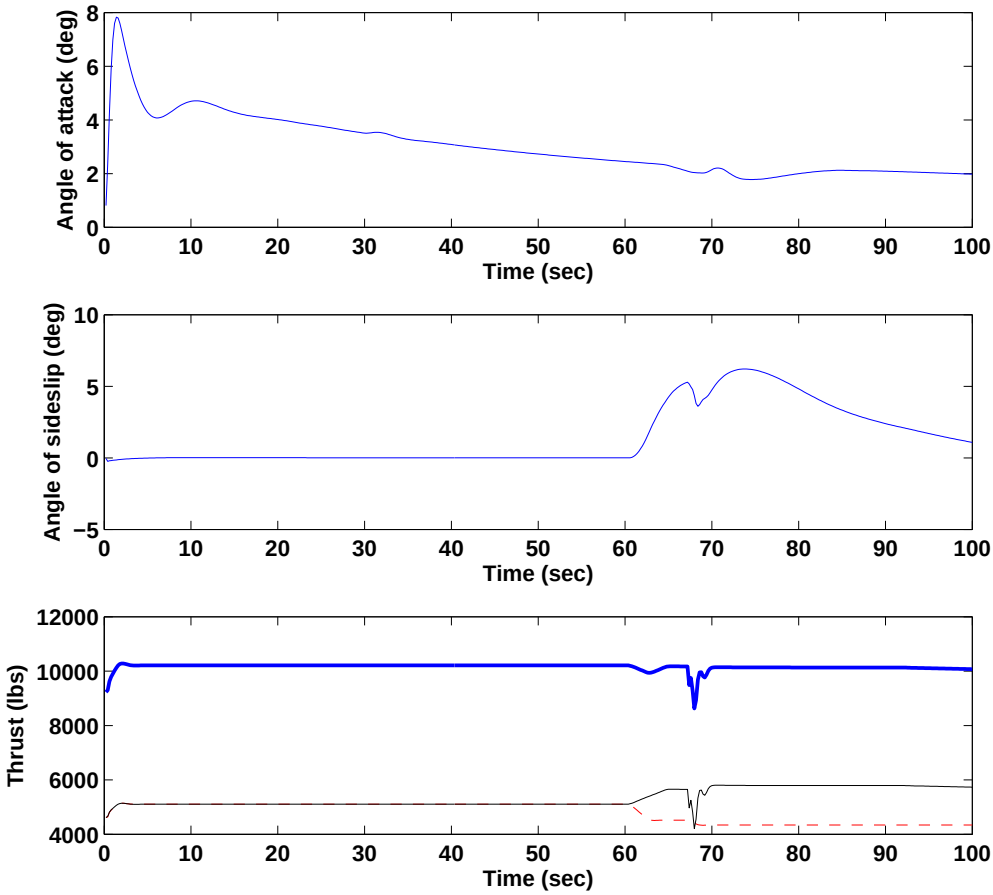


Figure 6.10. Response of the aircraft to load distribution

Response of the engines in the same flight is presented in Figure 6.12, and the solid curve represents the healthy engine and the dashed curve represents the response of the unhealthy engine. Thick curve in the upper left plate of Figure 6.12 shows the total thrust output of the engines, which is kept unaltered before and after the load redistribution. This shows that DES control logic of the propulsion system, which incorporates the detection of engine degradation and the assessment

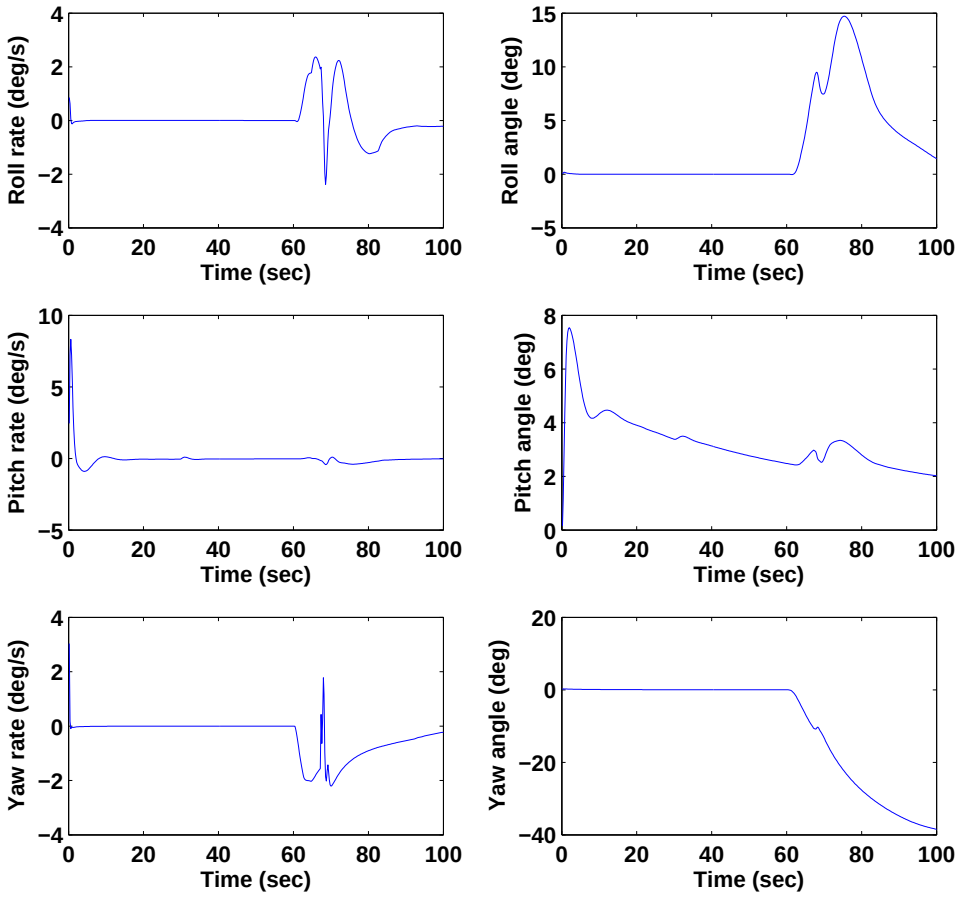


Figure 6.11. Changes in the attitude of the aircraft after load distribution

of relative health status of engines, issues the load distribution command and works effectively under dynamic flight conditions. It is also important to note that afterburner of the engine steps in after PLA of 42.5° is applied, which results in discontinuous behavior of the response for both engine and aircraft dynamics. It should also be noted that the thrust produced by the engines is scaled before going through the aircraft model.

To observe the effects of load redistribution on the aircraft control surfaces and, the dynamic-inversion controller with differential thrust accommodation is used. The controller manipulates the control surfaces (ailerons, elevators and rudder) for regulating the roll angle and sideslip. Under this controller action, Figure 6.13 and Figure 6.14 show the changes in the attitude of the closed loop dynamical behavior of the aircraft when DES controller decides to change the load distribution due to health conditions of the engines. It is seen that roll angle and yaw angle respond

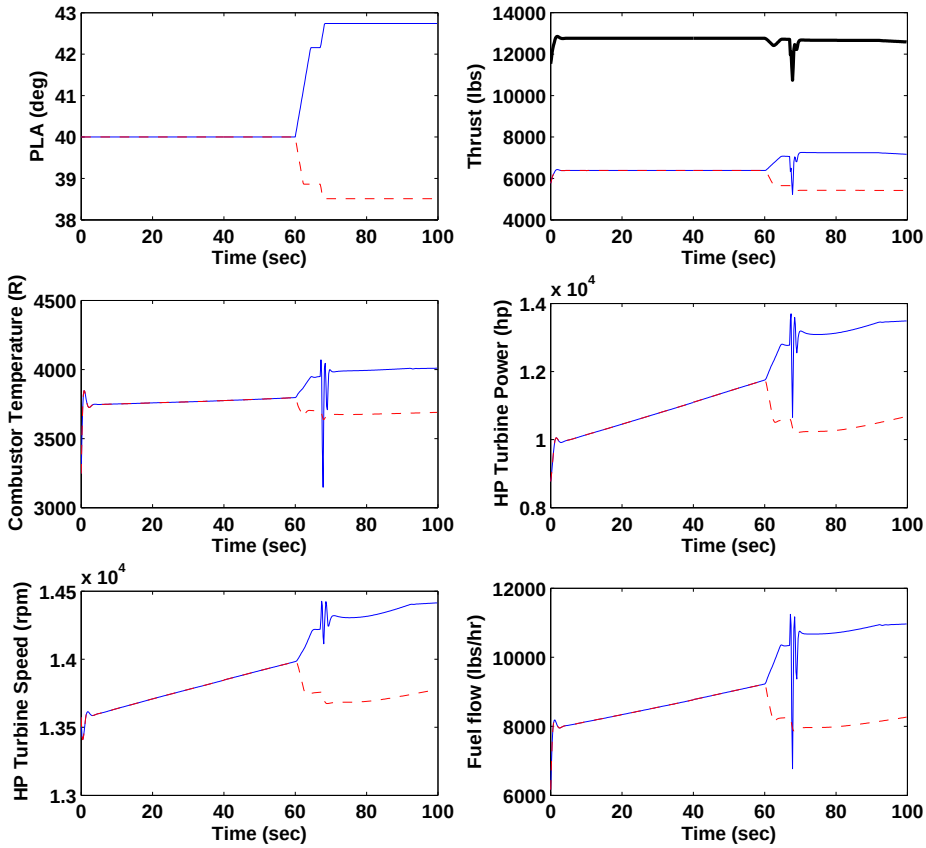


Figure 6.12. Response of the engines under dynamic flight conditions

to load distribution much better than the nominal control case which indicates that flight control with differential thrust accommodation can handle the effects of DES decision reasonably. However, the controller still may cause a possible divergence from the desired path in the long term because the dynamic-inversion law is capable of controlling the yaw angle with a steady state error of $-0.17°$. The rationale is that the dynamic-inversion control variable is the sideslip angle β instead of yaw angle ψ and the control law does not contain a heading control per se. A possible and easily implementable solution could be introducing an outer feedforward bias to apply a corrective heading action, since the steady state error of the yaw motion is constant.

Figure 6.15 depicts the deflection of the control surfaces due to control commands to overcome the effects of load imbalance and resulting moments created on the aircraft. Small deflections, which are much less than the provided limits [101], result in superior controlled behavior of the aircraft in terms of roll and yaw

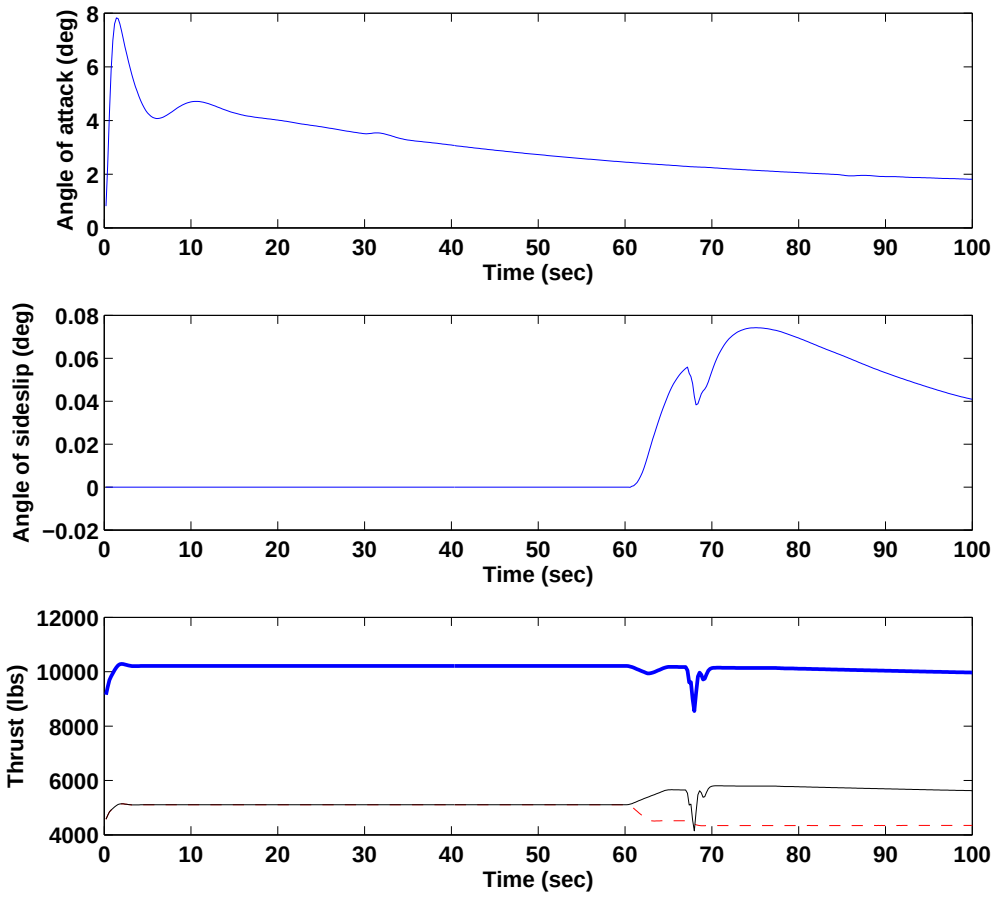


Figure 6.13. Response of the aircraft to load distribution under dynamic-inversion control

angles.

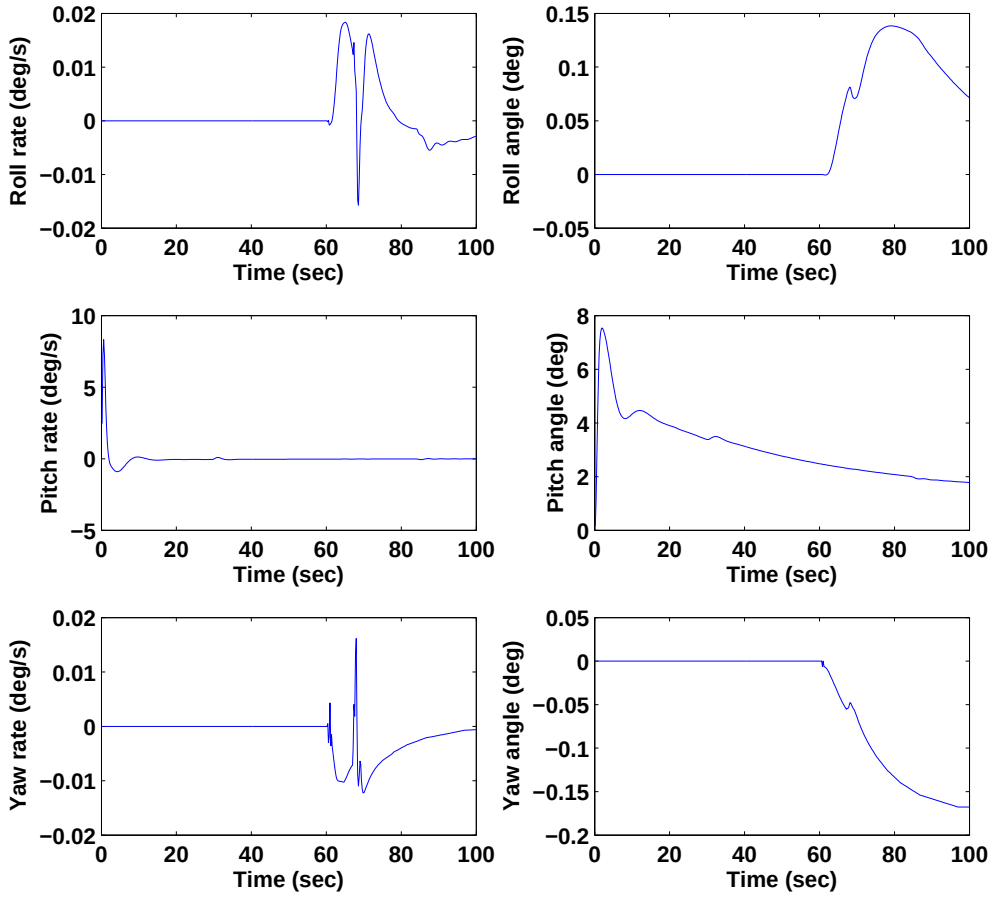


Figure 6.14. Changes in the attitude of aircraft under dynamic-inversion control

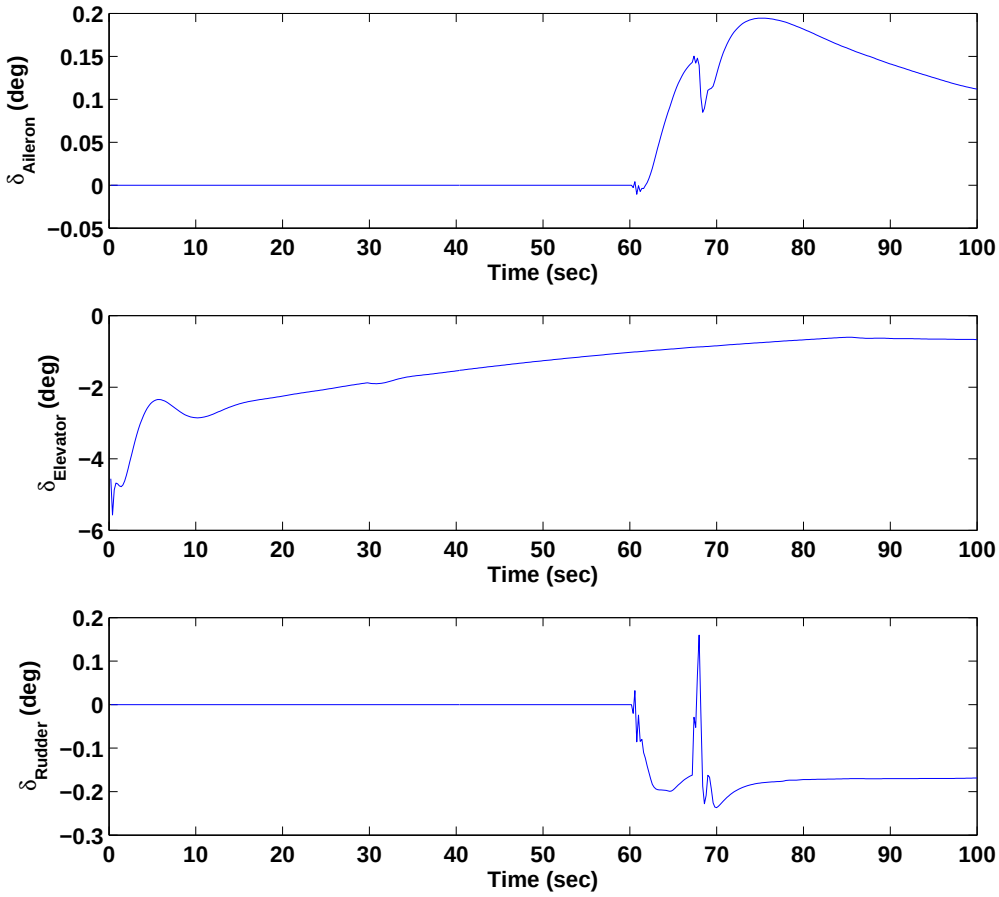


Figure 6.15. Aileron, differential elevator and rudder deflections

Chapter 7

Control and Coordination of Unmanned Vehicles

This chapter presents a hierarchical and hybrid architecture for control and coordination of future generation unmanned vehicles. Specifically, the coordination of unmanned aerial vehicles (UAV) and unmanned ground vehicles (UGV) to develop a command, control, computer, communication, intelligence, surveillance and reconnaissance (C^4ISR) system is demonstrated. The overall system features coordination of a rotary-wing unmanned aerial vehicle (RUAV) and multiple unmanned ground vehicles (UGVs) by integrating a high-fidelity rotorcraft model with actual ground-based robots that emulate UGVs. The RUAV component of the system is realized on a networked computer simulation, whereas networked robotics hardware is employed for the UGV representation. The RUAV dynamics run a high-fidelity nonlinear simulation model of a UH-60A Black Hawk helicopter. To emulate the dynamics of the UGVs, several networked Segway robots are employed. The proposed C^4ISR system undertakes the task of mission management from the top level discrete-event supervision (DES) to the bottom level continuous-time regulation, therefore employs an hierarchical hybrid supervisory architecture.

7.1 Introduction

Simulation technologies have become an integral part in the development of unmanned vehicles, systems and operations. Therefore, it is crucial to establish

standards for simulation and visualization instruments, data transfer and communication protocols and high levels of simulation fidelity. Substantial research is in progress for all phases of simulation technologies to support the decision-making process of an operator.

Future autonomous unmanned aerial vehicles (UAV) will need to work in teams with other UAVs and unmanned ground vehicles (UGVs) to share information and coordinate activities in much the same way as current manned systems. For this purpose, mathematical models and simulation studies have been developed to understand and ultimately provide this future unmanned vehicle capability.

The research focuses on coordination and cooperation for autonomous UAVs and UGVs engaged in information gathering and data fusion tasks, including cooperative tracking, area exploration, and target search. The underlying mathematical model for coordination and cooperation employs quantitative measure of regular languages [14] and discrete event supervisory (DES) control [15, 16]. This language-theoretic approach builds on established principles of sensor data fusion for event generation, and extends the ideas to problems for control of heterogenous agents. The research is envisioned to make substantial progress towards formulating, solving, and demonstrating these methods for UAV-UGV coordinated operation. In particular, distributed algorithms that enable UAV based surveillance and exploration operations to support UGV units have to be developed. These surveillance and exploration algorithms can incorporate realistic constraints on platforms and sensors because of the operation of hardware-in-the-loop UGVs.

Further research on this subject will provide significant scientific and technical advancement in the cooperative control of autonomous systems. The availability of autonomous UAV-UGV teams capable of complex cooperative behavior will enable military and civilian personnel to execute highly complex missions effectively and remotely.

7.2 Overall Architecture and Mission Objectives

The C^4ISR research aims to simulate real-time autonomous operation and coordination of air and ground units under certain mission objectives. The simulation test bed is built upon a high fidelity simulation model of a rotorcraft and hardware-

in-the-loop emulation of robotic systems.

The architecture of the The C[4]ISR system is provided in Figure 7.1, where the C[4]ISR system incorporates a high-fidelity simulation model of a UH-60A Black Hawk helicopter known as GenHel [105] and Segway robotic systems. The architecture is divided into three components based on the logical perception of the systems and considering the physical locations of the blocks. The *Helicopter* component is to simulate the RUAV operations of the test bed whereas the *Robots* component is used for UGV emulation. The *Command and Control* (C&C) component receives the abstracted information from other components and serves for decision making based on the overall mission objectives and coordination of RUAV and UGV units. The communications between the blocks and different components are through Ethernet and wireless networks depending on the data size and network traffic requirements.

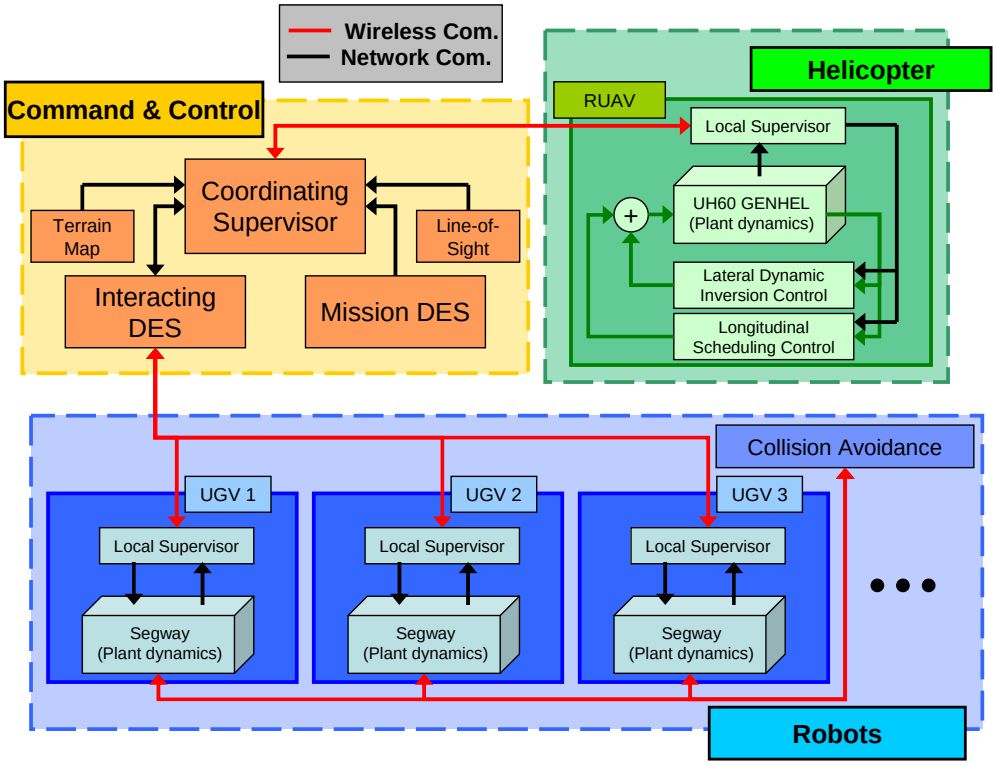


Figure 7.1. Overall architecture of the C[4]ISR system

The *Helicopter* component employs the nonlinear GenHel simulation model of a UH-60 helicopter and accompanying continuously varying controllers together

with a local discrete event supervisory (DES) controller (also denoted as "supervisor"). The GenHel model and its (continuous and DES) controllers are simulated in one computer. Although they could be implemented as a single program, the supervisor and the GenHel model (with continuous controllers) are disjoined and talk to each other through socket communication. This method is preferred to separate the discrete event and continuous dynamics of the system in a modular structure. Similarly, the *Robots* component emulates the dynamics of UGVs using several networked Segway robots. When emulating many UGVs, it is also possible to use simulated Segway robots due to the limited number of actual robots, hardware limitations, and restricted availability of laboratory space. For this purpose, system identification is performed for the governing dynamics of the robotic system, and the identified dynamics are utilized as if the actual hardware is operating. Moreover, a separate collision avoidance algorithm continually supports the autonomous operation of the robotic systems, since utilization of actual hardware brings the danger of collision and necessitates an avoidance algorithm.

In the formulation of intelligent decision-making policies, the supervision of the entire fleet is managed by a chain of multi-tier discrete-event supervisors, while the interactions between the supervisors in the hierarchy are obtained from the relationship between the discrete events and continuously-varying sensory information [87]. The C&C component of the test bed obtains the information from the local supervisors of the RUAV and UGVs through a wireless network. The C&C computer receives constant updates on the current location of each vehicle, selects a desired path for each vehicle based on the current mission status, known enemy positions and expected behavioral patterns of enemy and friendly units, and issues corresponding waypoint commands.

The C^4ISR system is used to manage a cooperative autonomous mission that requires intelligent control and decision-making. It is assumed that a highly maneuverable rotorcraft is used to search for enemy units and to coordinate with the operation of available ground units. The RUAV also attempts to avoid detection and enemy fire by flying in a lower altitude than enemy sensing capability. Each UGV aims to reach the target location while avoiding enemy fire by keeping a safe distance with enemies. Judicious distribution of limited resources to accomplish the mission objectives has utmost importance to the mission. The

proposed C^{4}ISR system undertakes the task of mission management from the top level discrete-event supervision [16] to the bottom level continuous-time regulation [10]. The goal of the supervisory decision and control system is to autonomously achieve predefined mission objectives using a hierarchical structure of:

- continuous-time control at the lower level for high precision maneuvering;
- discrete-event supervisory control at the upper level for intelligent decision-making [15].

7.3 Unmanned Rotorcraft Simulation

7.3.1 Flight Dynamics Model

The RUAV component of the C^{4}ISR system is realized using a networked computer simulation with one computer running the flight dynamics model and another computer running a visualization of the rotorcraft flying over virtual terrain. The RUAV flight dynamics are simulated by a high-fidelity nonlinear model of a UH-60A Black Hawk helicopter known as GenHel [105]. The code models non-linear aerodynamic effects, and includes fuselage rigid body dynamics, rotor blade flapping and lagging dynamics, rotor inflow dynamics, engine/fuel control dynamics, actuators, and a model of the existing UH-60A automatic flight control system (AFCS). Modifications to the code allow for the replacement of the existing AFCS channels by the controllers presented in this chapter, as well as networked communication (using UDP sockets) with the C&C and rotorcraft visualization computers.

7.3.2 Visualization

Visualization of the simulated RUAV is performed by an open source flight simulation program called FlightGear [100]. Although FlightGear is capable of simulating flight dynamics, its role in this research is limited solely to the visualization of data provided by an external simulation. The features of FlightGear relevant to this chapter include accurate terrain data for the entire Earth and the ability to place objects (e.g., buildings, bridges, and static vehicles that may be used as targets)

within the terrain. A multiplayer feature allows simultaneous operation of multiple vehicles that interact within the same virtual world.

The RUAV and the UGVs are linked in two ways: the C&C computer communicates with each vehicle to issue commands and receive position data and each vehicle has its own instance of FlightGear which communicates with every other instance of FlightGear so that all of the vehicles can be visualized in the virtual world. The network communication with the C&C computer and among the separate instances of FlightGear is performed using UDP sockets.

In addition to coordinating the network communication, each of the instances of FlightGear as well as the C&C computer must have the same data regarding terrain and placement of static objects (the positioning of dynamic objects is handled by the continuous network communication). After the location of each of the static objects is determined, this information must be transmitted to each instance of FlightGear and the C&C computer to ensure that the virtual world is identical in each of its representations in the test bed.

Moreover, the terrain database is needed to model the sensors required for a terrain following controller used on the RUAV. A terrain database was extracted for the State College, Pennsylvania area, where the University Park campus of the Pennsylvania State University is located and the simulated mission will be executed. The terrain profile of State College obtained from FlightGear and a topological map of the same area is given in Figure 7.2.

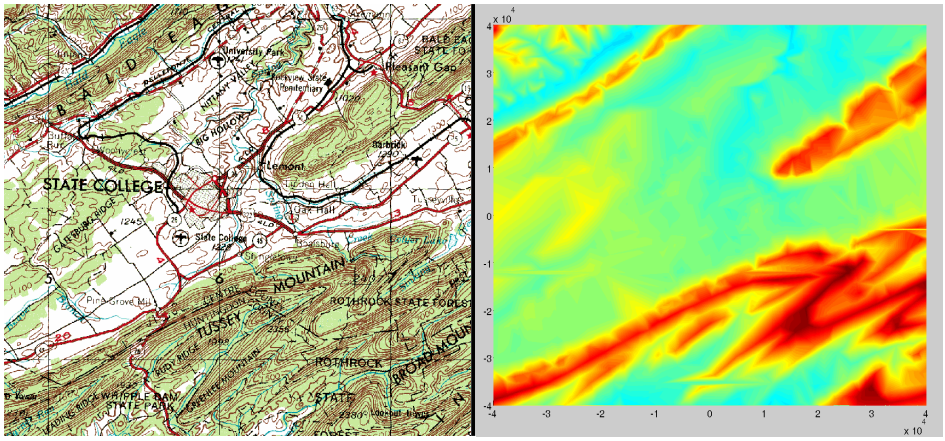


Figure 7.2. State College area topological map and extracted terrain profile

7.4 Unmanned Ground Vehicles

Four Segway robotic mobile platforms (RMPs) have been used as a simulation for unmanned ground vehicles (UGVs). Details of the Segway RMPs, modeling and model parameter identification in discrete-event and continuous settings are given in Chapter 3.

The commercial Segway RMPs are customized at the Networked Robotic and Sensor Intelligence Lab at Penn State. Each of the Segway RMPs is equipped with a SICK LMS200 laser range finder, six Devantech SRF05 sonars and six Sharp GP2D12 infrared sensors for obstacle avoidance. The laser range finder with a range of 80 m and resolution of the order of millimeters is used for both long range and short range obstacle avoidance. Three sonars in the front and three in the back are used to provide short range obstacle avoidance information with a range of about 3 meters. The IRs are placed on the sides to enable the robot to see if it is safe to make turns. The sonars and IRs are controlled by a dedicated network of Brainstem General Purpose (GP) modules based on the PIC18C252 micro-controller. The brain of the entire system is a Dell Latitude D410 laptop which is connected to the Brainstem Network and Sick Laser and the RMP's on board controller through universal serial bus (USB) port. The robot itself is connected to the rest of the network through the use of IEEE 802.11b/g wireless network.

The robot was operated in a restricted envelope due to space limitations and measurement scaling for matching the response of the UGVs and UAV in real time. Still, the speed of the UGV is comparable to that of the UAV in simulation environment. The input was constrained as follows:

$$-0.8 \text{ m/s} \leq v \leq 0.8 \text{ m/s} \tag{7.1}$$

$$-0.4 \text{ rad/s} \leq \omega \leq 0.4 \text{ rad/s} \tag{7.2}$$

The Player [64] is the lowest level functional block in the hierarchy which accesses to the robots' physical hardware such as laser, sonar and the actuators, and is usually run on the onboard computer. The *Action Generator* is the block that consists a set of fundamental behaviors that the robot can perform. More sophisti-

cated behaviors are built upon the fundamental ones which are Search, Approach, Idle, Ignore, GoTo and Planner. In the context of this chapter, Idle (robot does nothing), GoTo (robot goes to a waypoint) and Planner (robot plans the path from current position to any waypoint and uses Floyd-Warshall algorithm for shortest path) actions are employed. The *Event Generator* is an observer that implements continuous to discrete-event conversion. The Stage simulator of the Player/Stage project [64] has been used to simulate the Segway robots. It is a 2D simulator capable of simulating sensors like laser, sonar, camera, pan-tilt-zoom (ptz), and has simple friction models for simulating built-in objects. Stage is capable of simulating a large population of robots with low-fidelity models. To more realistically simulate the dynamics, the model of the Segway obtained in Chapter 3 has been used in conjunction with Stage. Each robot in the Stage simulator runs an instance of Player robot server to provide access to sensors and actuators to the clients.

7.5 Hybrid Supervisory Controller Architecture

7.5.1 Directional Controllers for RUAV

Both the longitudinal and lateral-directional controllers for the RUAV are based on control design presented in [39]. The longitudinal controller uses an explicit model-following control scheme with a feedback portion designed using an LQR solution [39]. The LQR gains are scheduled with airspeed to account for variations in the plant model. As illustrated in Figure 7.3(a), the controller is designed to regulate rotor speed, minimize transients in engine torque, and follow commands in vertical speed and pitch attitude. Lastly, the controller aims to minimize torque loads to the main transmission based on the damage weight, which acts to reduce the damage to the transmission.

One of the side objectives of this chapter is to investigate the feasibility and potential benefits of implementing a damage mitigating control (DMC) system on an operational RUAV. At the most basic level, the DMC system uses a dynamic gain-scheduled controller [104]. This controller includes parameters to vary the controller with flight condition, as in traditional gain scheduling; however, the controller also includes a parameter (subsequently referred to as the Damage

Weight (Dw)) that adjusts the level of damage mitigation in the controller (see Figure 7.3(a)). Normalized Dw varies between 0 and 10. The value 10 implies maximum emphasis is being given to damage mitigation and thus the response of controller is relatively sluggish. On the other extreme the controller with $Dw = 0$ is an extremely agile controller where the main emphasis is performance and these may cause an increase in damage rate. As damage begins to accumulate, the level of damage mitigation can be increased, possibly to a level such that handling qualities are diminished, and the aircraft may need to operate in a degraded mode with a restricted flight envelope. This is accomplished via discrete event supervision of DMC controller using the damage weight.

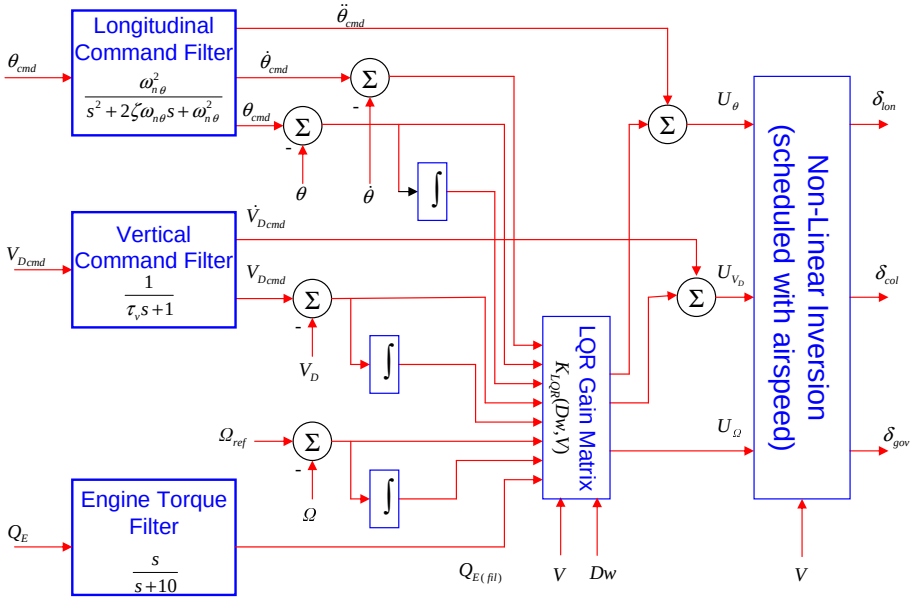
The lateral-directional controller, shown in Figure 7.3(b), also uses model-following with feedback designed using an LQR solution scheduled with airspeed [104]. Inputs to the lateral-directional controller for high speed (above 30 knots) are commanded roll attitude and lateral acceleration for coordinated flight; for low speed, the commands are roll attitude and yaw rate. In this application, the yaw rate command is obtained by passing a heading command through a proportional-integral (PI) filter.

$$\psi_{err} = \psi_{cmd} - \psi \tag{7.3}$$

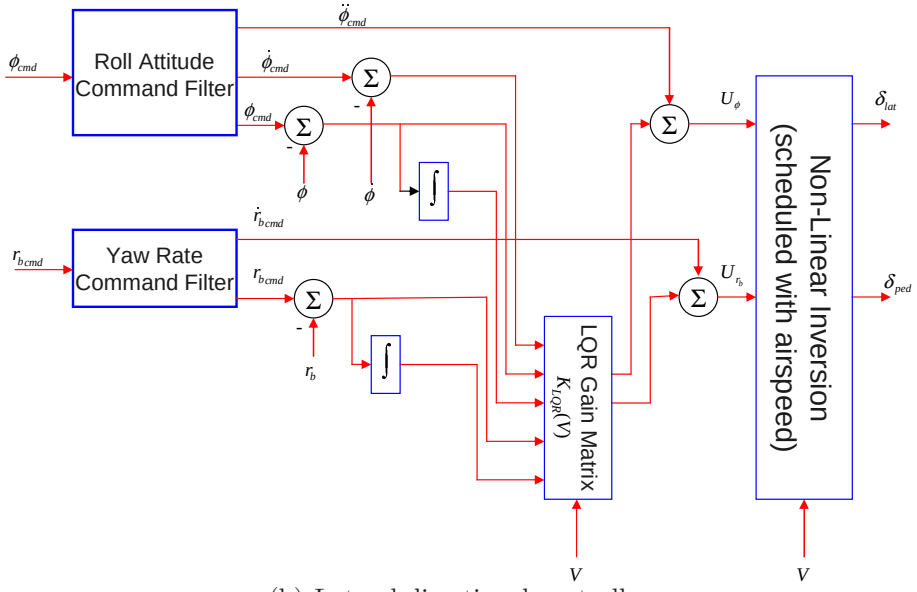
$$r_{b_{cmd}} = K_{P_\psi}\psi_{err} + K_{I_\psi}\int \psi_{err}\, dt \tag{7.4}$$

7.5.2 Terrain Following and Waypoint Navigation of RUAV

Terrain following is accomplished by converting altitude commands from the discrete-event supervisor of the RUAV into vertical speed commands used by the low-level longitudinal controller using a proportional-derivative (PD) filter. The vertical speed commands are limited to prevent the helicopter from entering an autorotative state (descent rate limit) and to prevent violating the continuous engine torque limit of the helicopter (climb rate limit); as with the low-level controllers, the vertical speed limits are scheduled with airspeed. Airspeed regulation converts the total airspeed command from the discrete-event supervisor of the RUAV into



(a) Longitudinal controller



(b) Lateral-directional controller

Figure 7.3. RUAV controllers

a pitch attitude command via a proportional-integral controller.

$$h_{err} = h_{cmd} - h \tag{7.5}$$

$$V_{D_{cmd}} = K_{P_h} h_{err} + K_{D_h} \dot{h}_{err} \tag{7.6}$$

$$V_{err} = V_{cmd} - V \tag{7.7}$$

$$\theta_{cmd} = K_{P_V} V_{err} + K_{I_V} \int V_{err}\, dt \tag{7.8}$$

Waypoint navigation is based on commands from the global discrete-event supervisor; specifically, the global DES provides a waypoint location (x-y position), a navigation mode command, and a heading command which is used only at low speed. The waypoint navigation system then directs the RUAV to the given waypoint.

The control portion of the navigation system is a proportional controller with $\phi_{cmd} = K_{nav}\chi_{err}$, which relates a roll angle command to the difference between the rotorcraft's current flightpath angle and the flightpath angle required to fly to the desired point (χ_{err}). The roll angle command is limited to 30 degrees, and is also rate limited to prevent sharp changes in roll attitude. Since the navigation system takes in flightpath angle and outputs a roll angle command, waypoint navigation only operates at medium to high speeds.

For hover mode, the RUAV switches from the model-following controllers described above to proportional-integral-derivative (PID) controllers with command filters, which are used to convert longitudinal position into a pitch angle command and lateral position into a roll angle command. The desired heading of the RUAV is converted into a yaw rate command through a PI controller [106].

7.5.3 Local Discrete Event Supervisor for RUAV

The local discrete event supervisor for RUAV maneuvers is used to decide on the set points for the continuous controllers depending on the operation and mission objectives [89]. An FSA model of the RUAV operation is created considering the requirements to accomplish the mission and maneuvering capability of the RUAV. The local supervisor restricts the RUAV behavior while making the optimal tradeoff such that the maneuver is completed with minimal effort and mission objectives are still satisfied.

This section presents an elaborate scenario where controllers are tested. Suppose the RUAV is flying over a terrain which is divided into two territories: Enemy territory and friendly territory. When flying in enemy territory the RUAV has to fly close to the ground and at a high speed (nap of the earth flight) to avoid be-

ing shot. Damage mitigation is not an option here therefore the most aggressive controllers (low Dw) are employed in this case for better performance.

The power versus airspeed relationship for helicopter typically exhibit a "power bucket" i.e., the minimum torque required for level trimmed flight occurs in the intermediate speed range (shown by the green region in Figure 7.4). Therefore for the case when the unmanned helicopter is flying in friendly territory, the RUAV flies at an intermediate speed and high altitude to minimize damage. Flying at high altitude is preferred (for damage mitigation) because the torque requirements do not fluctuate with terrain.

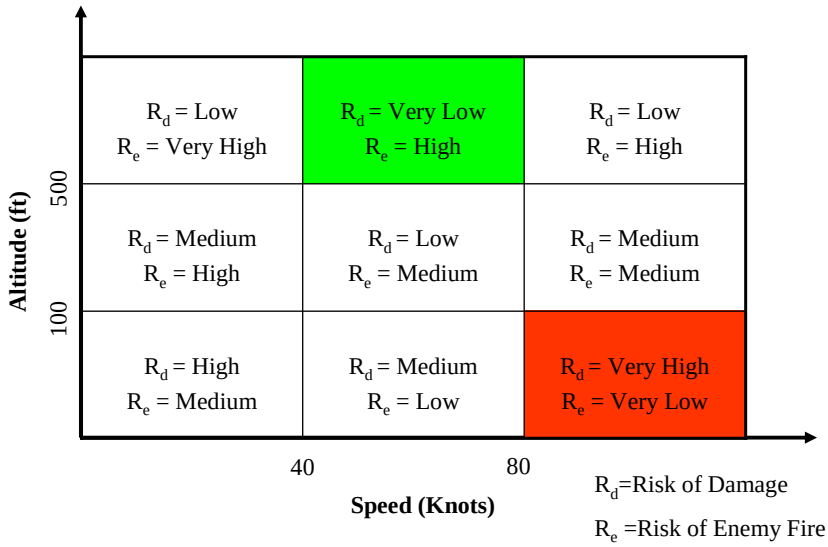
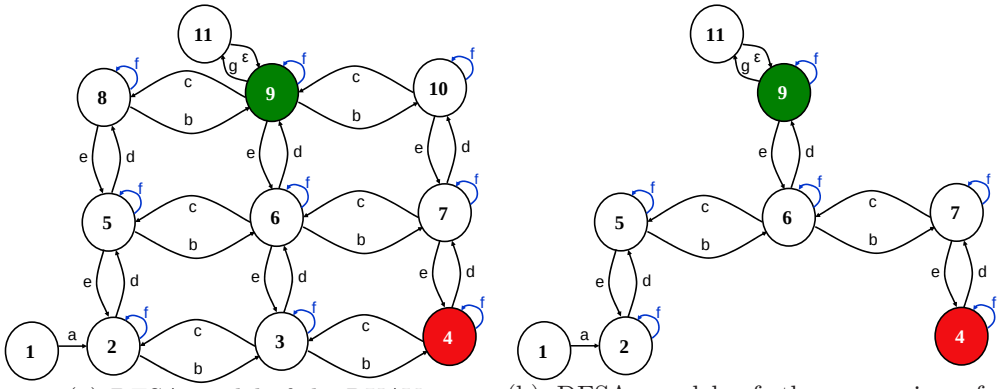


Figure 7.4. Partitioning of flight regime

The local DES controller of the RUAV determines the range of operation that serves the mission objectives best. In terms of damage mitigation, it is preferable to fly at an intermediate speed and high altitude whereas the nap-op-the earth flight forces the transmission components more, resulting in higher damage. However, this situation is unavoidable if the risk of enemy fire exists. Figure 7.5(a) depicts the DFSA model of the RUAV operation and Figure 7.5(b) shows the supervisor DFSA that is applied on the plant. The list of events are provided in Table 7.1. Based on the DFSA state of the model, the action generator creates the necessary reference points for the airspeed, altitude and damage weight. These reference signals are fed to the continuous domain controllers for regulation of the continuous inputs to the rotorcraft.



(a) DFSA model of the RUAV

(b) DFSA model of the supervisor for RUAV

Figure 7.5. Supervision of discrete event dynamics of the RUAV

Event	Explanation	Status
a	Start the rotorcraft operation	Controllable
b	Increase rotorcraft speed	Controllable
c	Reduce rotorcraft speed	Controllable
d	Increase rotorcraft altitude	Controllable
e	Decrease rotorcraft altitude	Controllable
f	High damage observed	Uncontrollable
g	Enemy fire risk detected	Uncontrollable

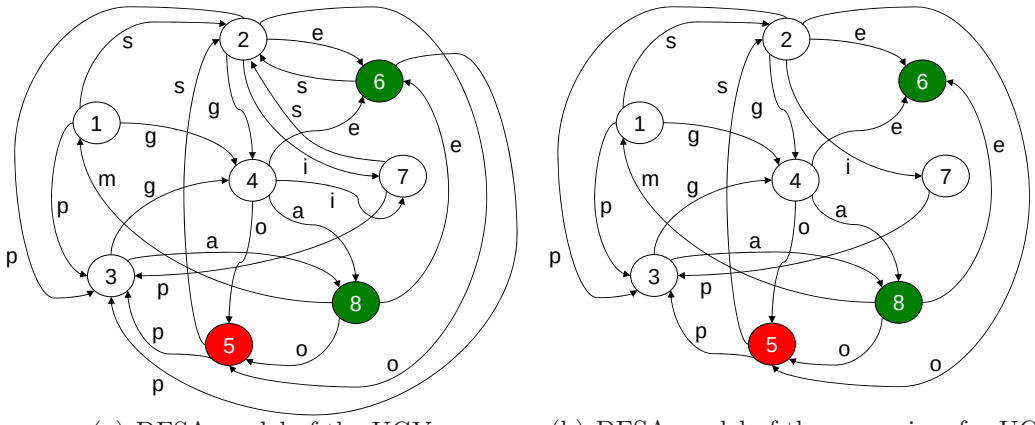
Table 7.1. Explanation of events based on rotorcraft operation envelope

7.5.4 Local Discrete Event Supervisor for UGV

The aim of the supervisor for unmanned ground vehicle operations is to restrict the controlled plant behavior by disabling or enabling certain controllable events based on observed event strings and optimal control policy [30]. The optimal control policy determines the expected long term behavior of the finite state model of the plant and tries to maximize the performance of this behavior by selectively disabling or enabling certain controllable events. At each state of the operation, the supervisor selects the states that is not allowed to go.

The local DES controller for the UGV operations is designed using the optimal control procedure that is described in Chapter 2. Based on their importance on the overall mission management, the states of the automata model, given in Figure 7.6(a), are assigned relative weights of significance. These relative state weights are selected as follows:

$$\bar{\chi} = [\ 0 \quad 0 \quad 0 \quad 0 \quad -0.2 \quad 0.2 \quad 0 \quad 1.0\ ]$$



(a) DFSA model of the UGV (b) DFSA model of the supervisor for UGV

Figure 7.6. Supervision of discrete event dynamics of the UGV

Using the event probability information of the DFSA model for the Segway, which is provided in Chapter 3, the optimal control design iterations are performed. The iterations for the *Relative Disabling* criteria are provided in Table 7.2. The iterations converge in four steps without further improvement in measure vector. Resulting optimal supervisor is shown in Figure 7.6(b).

	Open Loop	**Iteration 1**	**Iteration 2**	**Iteration 3**	**Iteration 4**
μ_1	8.374	11.669	18.781	18.819	18.819
μ_2	8.321	11.578	18.821	18.833	18.833
μ_3	8.710	11.994	19.242	19.260	19.260
μ_4	8.573	11.909	19.316	19.328	19.328
μ_5	8.165	11.398	18.570	18.584	18.584
μ_6	8.701	12.111	20.000	20.000	20.000
μ_7	8.436	11.763	18.871	18.889	18.889
μ_8	9.313	12.585	19.711	19.743	19.743

Table 7.2. Optimal control design iterations for robot behavior

7.5.5 Global Discrete Event Supervisor for C&C

The global discrete-event supervisor for command and control is designed to accomplish the mission objectives by further restricting the cartesian product of controlled plant behaviors. The global events are generated from the information of local supervisors, mission objectives and simulated sensor information. The supervisor selects a desired path for each vehicle based on the current mission status, known enemy positions and expected behavioral patterns of enemy and friendly

units, and issues corresponding waypoint commands for the RUAV and the UGV units. The aim is judicious distribution of limited resources towards the goals of mission, which has utmost importance for mission management, therefore to increase the likelihood of accomplishing the mission objectives successfully.

The global discrete event model is constructed upon the cartesian product of individual local supervisors and specializes based on mission scenario and current status of the mission. Therefore, the global supervisor might be different for different mission scenarios and objectives. Furthermore at each state of the operation, the global supervisor selects the states of local supervisors that are not allowed to go rather then selecting a single state for each unit, which will be forced to go. This approach is especially useful to increase the options that the C&C system may choose while the expected performance is still increased. Moreover, there usually exits a cost associated with disabling an event (e.g. halting the mission during execution phase has a cost since the resources are already dispatched for the mission). Therefore, minimizing the restricted behavior (rather than forcing one path of operation) is also inherently aimed in the design of control policy.

The global supervisor further restricts the cartesian product of behaviors by enforcing the mission objectives on the local supervisors as additional disabling actions. The defined mission objectives (see Section 7.2) are translated into language specifications as:

- Disable searching behavior of UGV and send it to target
- Keep the UAV at a medium speed and altitude range
- Synchronize the operations of UAV and UGV at the start

Resulting global supervisor is shown in Figure 7.7, where the states are paired (first one from RUAV, the second from the UGV) in the product automata. The events belonging to UGV dynamics are highlighted as italics to distinguish between events of RUAV and UGV. It should also be noted that the total number of states are less than 64, which is the product of number of states for two individual supervisors, since the disabling action of global supervisor results in unreachable states, which are deleted from the automata.

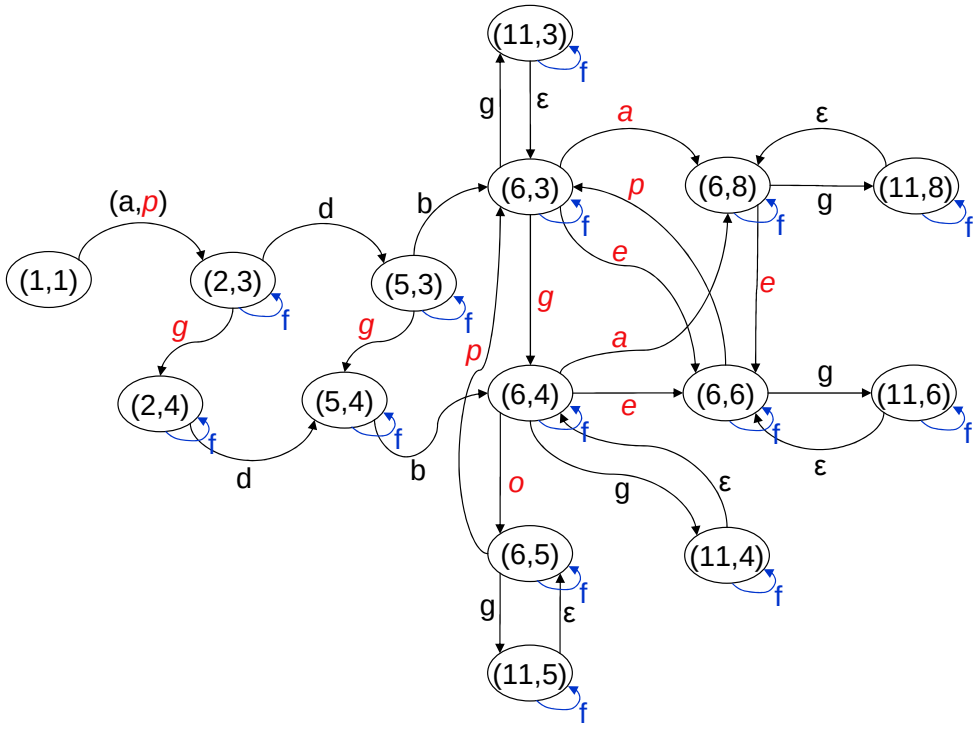


Figure 7.7. Product automata structure for global supervisor of C&C

7.6 Results of RUAV-UGV Coordination

The initial set of results are presented for the RUAV operation during simulated missions under complete autonomy, where local DES control and waypoint navigation takes active part in the simulation. There exists three enemy territories in the path of UAV and the map has both flat and steep terrain profiles throughout the flight path. It is assumed that an unmanned helicopter caries out a surveillance mission starting from the base, following various waypoints and ending at the target. Over enemy territories, the RUAV is supposed to follow the terrain very closely, (nap of the earth flight) as dictated by the supervisor, otherwise it flies at a high altitude and medium speed. The Damage Weight, Dw, which varies between 0-10 is chosen based on the flight requirements: High damage weight controllers are used in friendly territory (higher damage mitigation and a relatively sluggish response) and low damage weight controllers are used in enemy territory (lower damage mitigation and a relatively fast response).

Figure 7.8 presents the behavior of the RUAV when performing under local discrete event supervision and without supervision that uses fixed altitude and

damage weight. It can be seen that supervisor commands that guides the RUAV through enemy and friendly territories are followed with high precision. First plate of Figure 7.9 represents the damage increase (in terms of crack growth) for the two simulation runs, with and without local supervisory decisions. The improvement in terms of damage mitigation is apparent from the figure. The bottom plate of Figure 7.9 depicts the change of Damage Weight over the simulation time.

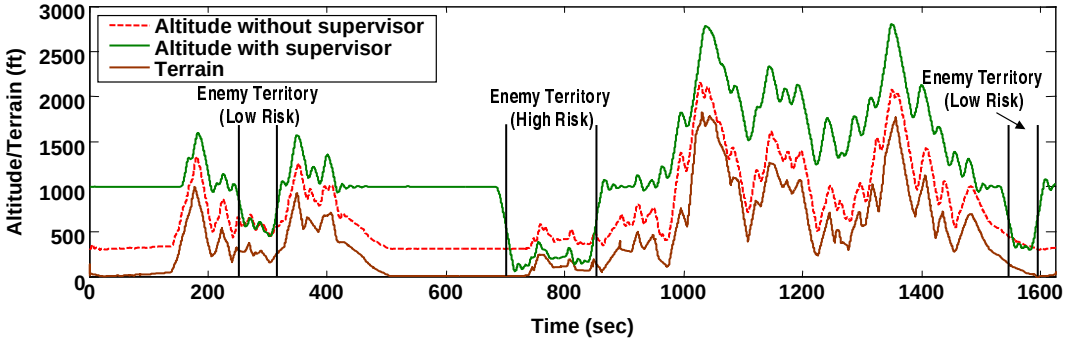


Figure 7.8. Autonomous flight of RUAV with and without supervisor

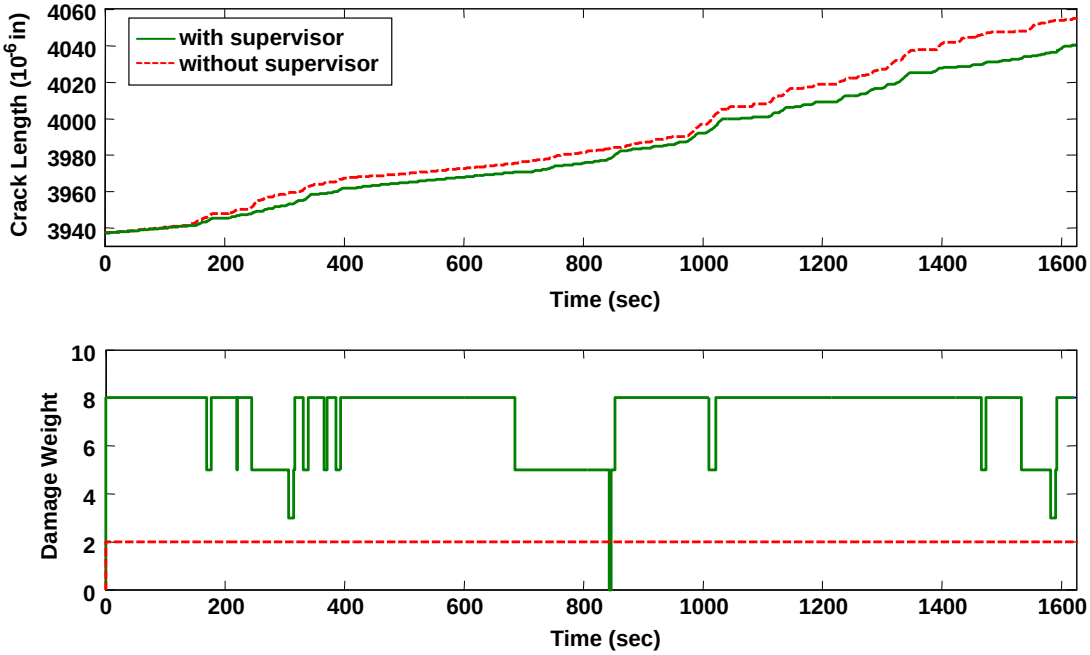


Figure 7.9. Crack length progression and damage weight changes during flight

The results presented herein also demonstrate the seamless integration of unmanned helicopter simulation and robotic mobile platform. The goal is to verify the benefits of cooperative control of heterogenous agents such as RUAV and UGV

in enemy environment. The experiment is defined as following: The UGV starts from the base and aims to reach to the target point which lays in enemy territory. The shortest path seems to be the best strategy of the UGV if there is no enemy unit between base and target and the UGV follows its best strategy under limited information. However, the enemy unit can detect the UGV, which is undesirable. The RUAV has a better sensing capability than UGV and can search possible enemy locations before the UGV approaches those points. If the enemy is detected by the RUAV, the local strategy of the UGV is changes due to enemy information sent by higher-level C&C controller and UGV follows possibly a longer but safer route to the target.

Figure 7.10 shows the mission map for the operations with and without an enemy threat. The solid line is the path of the RUAV while searching the possible enemy locations in the map. The dashed line is the path of the UGV when RUAV does not detect enemy presence and the dotted line is the path when RUAV detects the enemy by its sensor. The shaded region is the fire range of the enemy that UGV has to avoid. Figure 7.11 exhibits the behavior of RUAV and UGV for same mission executions in time domain.

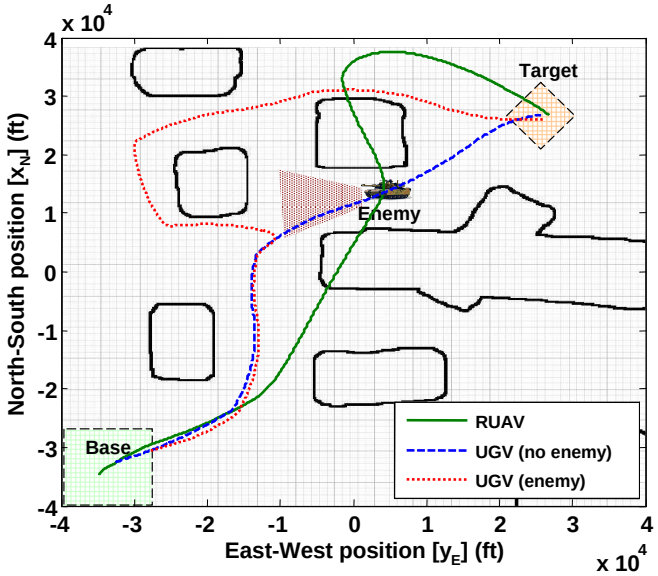


Figure 7.10. Spatial navigation of RUAV and UGV in the presence and absence of enemy threat

The robot and helicopter begin the mission at the same time. When simulation starts, the UGV initiates its Planner behavior while the RUAV searches for possible

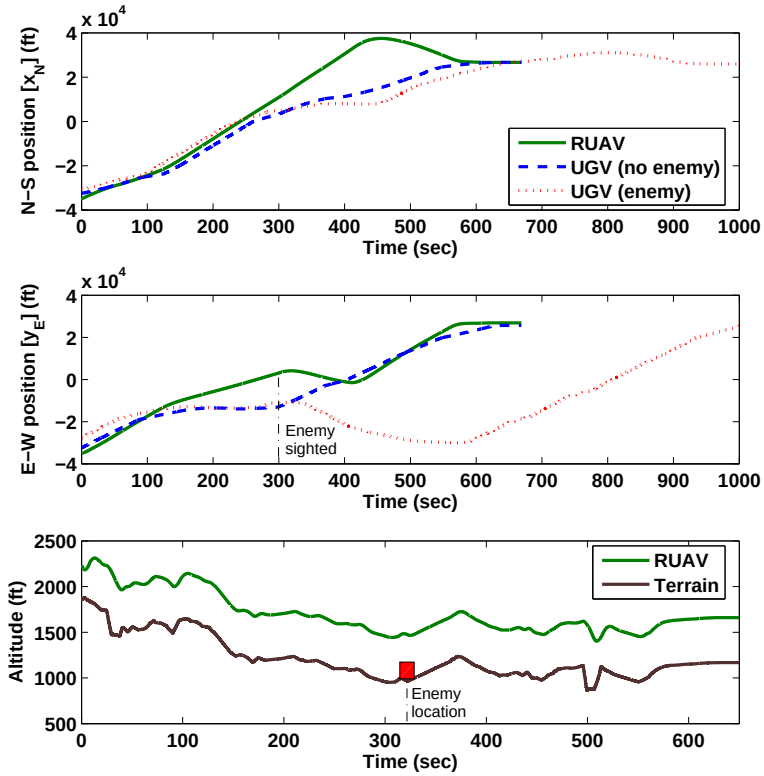


Figure 7.11. Temporal navigation of RUAV and UGV in the presence and absence of enemy threat

enemy locations as seen in Figure 7.10. From the sensor data of the RUAV, "enemy detected" event might be generated. Based on this event, the UGV may proceed with its original path (if the event is not generated) or deviate for a longer but safer path (if the event is generated). Both RUAV and UGV rendezvous at the target location at the end of the simulation. It is seen in Figure 7.11 that behaviors of UGV with and without enemy presence differs at 300 seconds, when the enemy is detected by the RUAV, although RUAV passes over the actual location of the enemy 25 seconds later.

Chapter 8

Summary, Conclusions, and Recommendations for Future Research

Large-scale systems have been traditionally characterized by a large number of variables, nonlinearities and uncertainties. Their decomposition into smaller, more manageable subsystems, possibly organized in a hierarchical form, has been associated with intense and time-critical information exchange and with the need for efficient coordination mechanisms. The problems regarding the large-scale systems manifest themselves mainly in the fields of modeling and control. Consequently, the recent research trend is towards the design of control laws and decision algorithms for large-scale systems using the principles and methods of intelligent control.

Today, the paradigms of discrete event and hybrid dynamic systems and control applications of these systems take their place at the focal point of developments in both theory and practice. The states of such systems change in asynchronous moments and the events created in the system (e.g. failures of actuators and sensors, saturation of some variable, etc.) can be represented by a set of discrete variables. It is necessary to compute methodic and mathematical models, and develop methods and algorithms of control and decision for such models. The control of hybrid dynamic systems requires a theory which will include both continuous and discrete variables, at which in some cases must consider also an influence of human

factors. The development of control theory methods of hybrid complex systems should integrate methods of control theory (decentralized optimal and adaptive algorithms of control) with methods from the area of formal logic, decision theory, graph theory, etc., and should be closely connected with the methods of computer science, information theory and artificial intelligence. Consequently, this thesis attempts to integrate the theoretical developments in Discrete Event Supervisory (DES) control theory and continuous control laws in an hierarchical architecture to synthesize and implement hybrid control schemes for complex dynamical systems.

This chapter summarizes the contributions of this dissertation and outlines some of the future research directions that is related to this thesis.

8.1 Contributions of the Dissertation

The foremost theoretical contribution of this dissertation is the introduction of different optimal control strategies for discrete event systems in a comparative manner. The algorithms for computation of optimal design iterations are developed for two different strategies, i.e. *Absolute Disabling* and *Relative Disabling*. The simulation studies showed that neither of the strategies is always superior to the other one. Nonetheless, the first strategy is considered to be less aggressive since it tries to avoid the negative performing states only. This means, if every state has a positive measure, this algorithm does not interfere with the plant dynamics. On the other hand, the second strategy usually further improves the plant behavior by comparing relative performances of states in disabling decisions.

Development of hybrid supervisory control of complex dynamical systems starts with modeling of the discrete event behaviors of the actual system, and the identification of model parameters follows. This dissertation demonstrates the modeling and model parameter identification stages for developing a supervisory control scheme on a mobile robotic behavior experiment and an aircraft gas turbine engine simulation testbed. Therefore, it is envisioned that the theoretical developments in this thesis provide implementable solutions to real-world applications.

The impact of this dissertation is mainly on the application side of hybrid supervisory control solutions. One of the major shortcomings of DES control is that there are only a few examples in technical literature reporting applications

of DES control in complex dynamical processes. An apparent reason is that, until recently, no quantitative analytical tool was available for design and evaluation of DES controllers. In this dissertation, however, a quantitative approach to analysis and synthesis of hierarchical DES control laws for aircraft propulsion systems has been successfully employed. The practical accomplishments are:

- Intelligent decision and control of distributed propulsion management systems, where each of the engines has its own local DES control.
- Structural damage reduction and life extension of aircraft engines without any significant loss of the system performance.
- Decision making and mission planning modifications through a high-level DES coordinator.
- Incorporation of optimal control laws for enhanced mission management.

A decision and control architecture has been proposed and implemented to coordinate the operations of a twin-engine propulsion system. The DES control law has been validated for a twin-engine aircraft propulsion system on a networked simulation test bed. The plant dynamics in the simulation test bed is built upon the model of a generic turbofan gas turbine engine. The software architecture of the simulation test bed is flexible for adaptation to arbitrary automata models and a variety of DES control laws, including those that are quantitatively analyzed using a language measure.

However, the applications of DES control theory, even the ones that use the quantitative method, are largely confined to interactions between the DES control module and the continuous-time plant under discrete event supervision. Therefore, these control policies can only be implemented without considering dynamical effects of the supervisory decisions on the other continuous-time systems, which are coupled with the DES controlled plant. Therefore, it is also imperative to design continuous controllers that regulate the dynamics of the system based on supervisory decisions of the DES controller. By integrating DES controlled propulsion system with a flight control system, it is demonstrated that the propulsion level supervisor is not affected adversely from the dynamic coupling between engine

and flight models. Other accomplishments are demonstration of the dynamic effects of DES control system on nonlinear flight dynamics and responses of the aircraft as well as the capabilities of the flight control system to neutralize the undesired consequences of DES control commands. The results indicate that DES control is an effective tool for mission and operation scheduling and evaluation when complemented by a successful continuous-time control law to regulate the agile maneuvering and quick responses of the plant.

Another application example, which is studied in this dissertation, is cooperative control of heterogenous automata. A comprehensive simulation and hardware testbed has been developed for validation of various control strategies on coordinated and cooperative control of rotary-wing unmanned aerial vehicle (RUAV) and unmanned ground vehicle (UGV) operations. The testbed aims to incorporate high fidelity simulation model of a rotorcraft and hardware of robotic mobile platforms to establish a framework for C^4ISR systems. The seamless integration of the RUAV component and a single robotic mobile platform has been achieved to successfully demonstrate the capabilities of continuous-domain lower level and discrete-event-domain higher level controllers. Intelligent decision and control of distributed autonomous systems, where each component has its own local control, and decision making and mission planning modifications through a high-level C&C coordinator are demonstrated successfully. Optimal DES control laws for enhanced mission management are incorporated and the software architecture is flexible to employ arbitrary models and controller designs.

One of the major tasks of the supervisory decision making is fusion of the (possibly) redundant, conflicting, and incomplete information to make timely decisions. Advanced analytical techniques are necessary for fusion of the time series data available from multiple sensors to make specific inferences and generate discrete events. In order to achieve the desired performance of a DES controller, it is essential to have an effective event generation algorithm. The algorithms of the real-time event generation method are built upon two-time-scale analysis of the stationary behavior of dynamical systems using the principles of symbolic dynamics, information theory, automata theory, and pattern recognition. Combining these techniques with ergodic theory of chaos and thermodynamic perspective is essential when the dynamic system at hand exhibits a nonlinear behavior, espe-

cially in a chaotic manner. The developed event generation algorithms are tested on a simulation model of a generic gas turbine engine, and their efficacy is evaluated relative to other existing pattern recognition techniques. Time series data of observed macroscopic variables, generated on the fast scale from the simulation model, are analyzed at slow time epochs. The results are compared with those derived from traditional pattern recognition tools, such as Principal Component Analysis (PCA) and Artificial Neural Network (ANN). Computational complexity is an important issue for online implementation. The symbolic dynamic filtering (SDF) method and chaos theoretic Kolmogorov-Sinai entropy method perform significantly efficient than the others.

8.2 Future Research Directions

The work reported in this thesis, has the potential to be extended both in scope and size. A few key areas of interest are listed below:

Intelligent Resilient Control: The objectives of intelligent resilient control is to reduce (or eliminate) loss-of-control incidents and ensure safe operation under safety-critical adverse, upset, and hazard conditions in the current and next-generation complex systems. The long-term goals are to develop technologies to prevent or recover from loss-of-control by detecting, characterizing, and mitigating historical and emerging precursors to these events, and to provide onboard control resilience functions for continuously assessing and managing system performance and control capability to ensure operational safety and recoverability under multiple and cascading adverse, upset and hazard conditions. For this purpose, it is necessary to develop integrated multidisciplinary methods, tools, and techniques for the characterization, detection, and prediction of coupled adverse/upset/hazard conditions and their effects on operational safety of systems; loss-of-control prevention, mitigation, recovery, and adaptive planning under multiple and cascading adverse/upset/hazard conditions; and assessment of complex integrated adaptive systems (analytical, simulation, and experimental validation, verification, and predictive capability assessment; software safety assurance). Hybrid supervisory control

might provide an answer for intelligent resilient control applications because of inherent inference and adaptability aspects of supervision.

C^{4}ISR: The coordination of unmanned aerial vehicles (UAV) and unmanned ground vehicles (UGV) to develop a command, control, computer, communication, intelligence, surveillance and reconnaissance (C^{4}ISR) system is demonstrated in this thesis as an application to hybrid supervisory control. Further research on this subject will provide significant scientific and technical advancement in the cooperative control of autonomous systems. The availability of autonomous UAV-UGV teams capable of complex cooperative behavior will enable military and civilian personnel to execute highly complex missions effectively and remotely. Application examples include: Cooperative decision-making and control of UAVs; formation flying for clusters of micro-satellites; and coordination of mobile robots used for search and rescue missions. Of particular interest is the cooperative control of autonomous teams for missions that include: Cooperative search, acquisition, tracking, and rescue; persistent intelligence, surveillance, and reconnaissance; task decomposition among heterogeneous vehicles for coordinated attack; cooperative timing of tasks; rendezvous and simultaneous target intercept; and task sequencing.

Incipient Fault Prognosis and Diagnosis: This subject addresses the issues of fault detection and isolation (FDI) and degradation monitoring in complex dynamical systems. The objective is to enhance operational safety and performance by reducing the risk of system failures. The challenge is to identify the system faults that may not be directly observable from the available sensors but affect the overall performance of the system. The focus is on the development of real-time decision tools for assessment of individual system components in the discrete-event setting, based on the filtered information derived from analytical models of the system and time series data of available sensors. Specific objectives are summarized as: Real-time detection and isolation of component faults in system components, including those faults that are not directly observable from the available sensors and computable parameters (e.g., efficiency); real-time prognostic decision-making for fault

accommodation and reconfiguration at the system level, based on the health status information of individual components; interfacing the plant with hybrid control systems to mitigate the effects of system reconfiguration on the performance. The resulting hybrid architecture will provide the prognosis of fault information on system components, and reconfiguration of the system will provide diagnosis of incipient faults before the faulty system becomes unrecoverable.

Appendix A

Analytical Engine Model

The mathematical model describing the two-spool, low bypass turbofan engine is provided in detail in [83]. The engine model, simulated as a FORTRAN program, is depicted in Figure A.1. Overall performance maps are used to provide steady-state representations of the engine's rotating components. Fluid momentum in the bypass duct and the augmentor, mass and energy storage within control volumes, and rotor inertias are also included to provide transient capability. For completeness of the dissertation, the mathematical model is presented in this appendix.

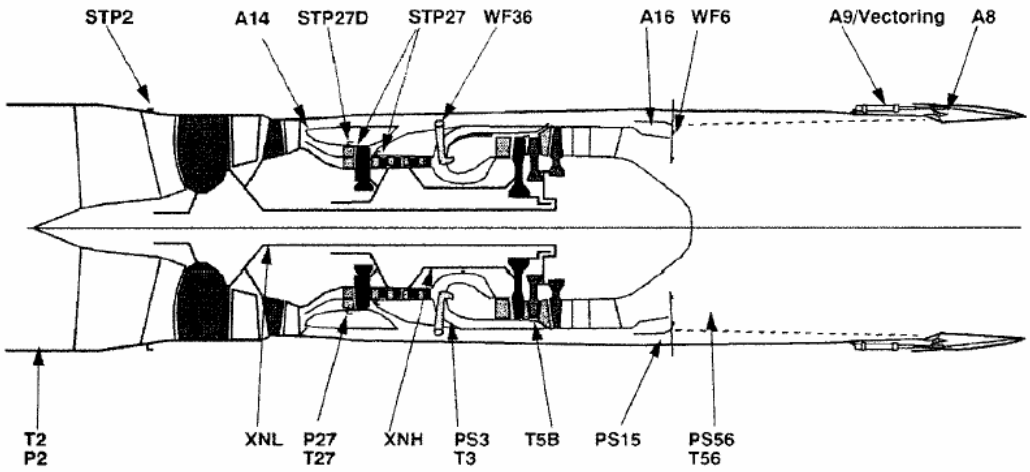


Figure A.1. Schematic of turbofan engine model with labeled actuators (above) and sensors (below)

A.1 Thrust Model and Equations

General thrust equation: Thrust is a mechanical force which is generated through the reaction of accelerating a mass of gas. A gas or working fluid is accelerated to the rear and the engine and aircraft are accelerated in the opposite direction.

For a moving fluid, the important parameter is the mass flow rate. Mass flow rate is the amount of mass moving through a given plane over some amount of time and given by:

$$\dot{m} = \rho \cdot V \cdot A \tag{A.1}$$

Since the mass flow rate already contains the time dependence (mass/time), we can express the change in momentum across the propulsion device in terms of the change in the mass flow rate. We will denote the exit of the device as station "e" and the free stream as station "0". Then

$$F = (\dot{m}V)_e - (\dot{m}V)_0 \tag{A.2}$$

There is an additional effect which we must account for if the exit pressure P_e is different from the free stream pressure. The fluid pressure is related to the momentum of the gas molecules and acts perpendicular to any boundary which we impose. If there is a net change of pressure in the flow there is an additional change in momentum. Across the exit area A_e, we may encounter an additional force term. Then, the general thrust equation is then given by:

$$F = (\dot{m}V)_e - (\dot{m}V)_0 + (P_e - P_0)A_e \tag{A.3}$$

Turbofan thrust: Every gas turbine engine has a combustion section, a compressor and a turbine. The compressor, burner, and turbine are called the core of the engine, since all gas turbines have these components. The core is also referred to as the gas generator since the output of the core is hot exhaust gas.

In the turbofan engine, the engine core is surrounded by a fan in the front and an additional turbine at the rear. The fan and fan turbine are connected by an additional shaft. As with the core compressor and turbine, some of the fan blades turn with the shaft and some blades remain stationary. The fan shaft passes

through the core shaft and this type of arrangement is called a two spool engine; one "spool" for the fan, one "spool" for the core.

The incoming air is captured by the engine inlet. Some of the incoming air passes through the fan and continues on into the core compressor and then into the burner, where it is mixed with fuel and combustion occurs. The hot exhaust passes through the core and fan turbines and then leaves out the nozzle. This airflow is called the core airflow and is denoted by $\dot{m}_c$. The rest of the incoming air passes through the fan and bypasses, or goes around the engine. The air that goes through the fan has a velocity that is slightly increased from free stream. This airflow is called the fan flow, or bypass flow, and is denoted by $\dot{m}_f$. The ratio of $\dot{m}_f$ to $\dot{m}_c$ is called the bypass ratio - BR.

$$BR = \frac{\dot{m}_f}{\dot{m}_c} \tag{A.4}$$

The total mass flow rate through the inlet is the sum of the core and fan flows

$$\dot{m}_0 = \dot{m}_f + \dot{m}_c \tag{A.5}$$

A turbofan gets some of its thrust from the core and some of its thrust from the fan. If we denote the exit of the core as station "e", the exit of the fan as station "f", and the free stream as station "0", we can use the basic thrust equation for each stream to obtain the total thrust:

$$F = (\dot{m}V)_f - \dot{m}_f V_0 + (\dot{m}V)_e - \dot{m}_c V_0 \tag{A.6}$$

We can combine the terms multiplying V_0 and use the definition of the bypass ratio BR to obtain the final thrust equation:

$$F = (\dot{m}V)_e + BR \cdot \dot{m}_c V_f - (\dot{m}V)_0 \tag{A.7}$$

Because the fuel flow rate for the core is changed only a small amount by the addition of the fan, a turbofan generates more thrust for nearly the same amount of fuel used by the core. Many modern fighter planes actually use low bypass ratio turbofans equipped with afterburners. They can then cruise efficiently but still

have high thrust when dogfighting. Even though the fighter plane can fly much faster than the speed of sound, the air going into the engine must travel less than the speed of sound for high efficiency. Therefore, the airplane inlet slows the air down from supersonic speeds.

A.2 Steady-State Engine Model

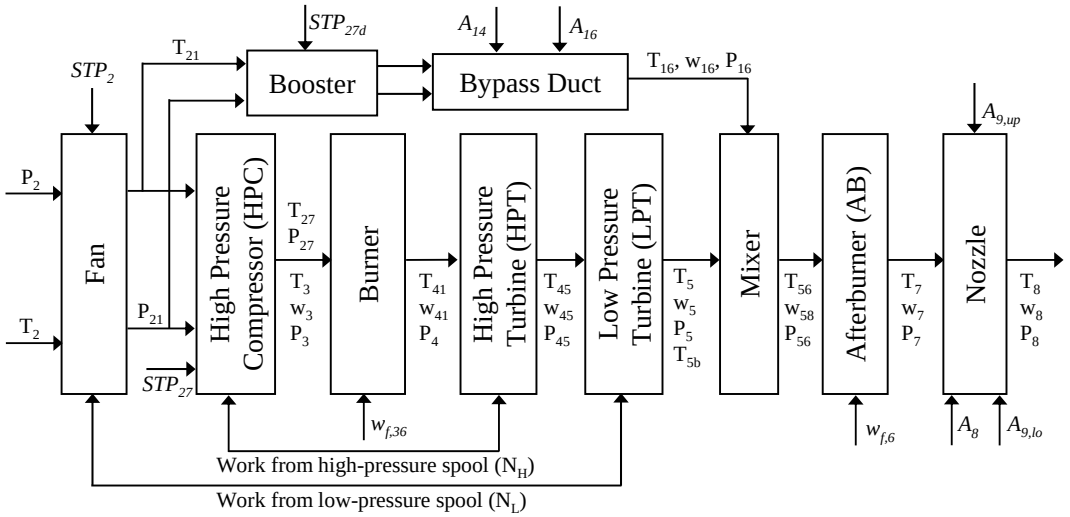


Figure A.2. Flow diagram and components of engine model with actuators (italic)

The components of the engine model and station numbering is provided in Figure A.2[83]. Air is brought into the turbojet through the inlet at the left of the drawing. At the rear of the inlet, the air enters the compressor. The compressor acts like many rows of airfoils, with each row producing a small increase in pressure. At the exit of the compressor, the air is at a much higher pressure than free stream. In the burner a small amount of fuel is combined with the air and ignited at near constant pressure. Leaving the burner, the hot exhaust is passed through the turbine. Energy is extracted from the flow by the turbine to turn the compressor, which is linked to the turbine by a central shaft. Some pressure is lost in the hot exhaust during this process, but the pressure entering the nozzle is still greater than free stream. The nozzle then converts the high pressure and temperature into high velocity. Because the exit velocity is greater than the free stream velocity, thrust is created as described by the thrust equation (Equation A.7).

The engine pressure ratio (EPR) is defined to be the total pressure ratio across the engine. Using our station numbering, EPR is the ratio of nozzle total pressure to compressor face total pressure. The EPR is simply the product of the pressure ratios across all of the engine components:

$$\begin{aligned} EPR &= \frac{P_8}{P_2} = \frac{P_3}{P_2} \cdot \frac{P_4}{P_3} \cdot \frac{P_5}{P_4} \cdot \frac{P_8}{P_5} \\ &= CPR \times BPR \times TPR \times NPR \end{aligned} \tag{A.8}$$

Similarly, the engine temperature ratio (ETR) is defined to be the total temperature ratio across the engine. Using our station numbering , ETR is the ratio of nozzle total temperature to compressor face total temperature. The ETR is simply the product of the temperature ratios across all of the engine components:

$$ETR = \frac{T_8}{T_2} = \frac{T_3}{T_2} \cdot \frac{T_4}{P_3} \cdot \frac{T_5}{T_4} \cdot \frac{T_8}{T_5} \tag{A.9}$$

Along with a diagram of the implemented engine model, Figure A.2 also lists the actuators. The actuators are for the fan variable inlet stator vanes (STP_2), forward blocker door area positioning (A_{14}), high-pressure compressor and booster hub stators vanes (STP_{27} and STP_{27d}), burner fuel flow ($w_{f,36}$), aft variable bypass area positioning (A_{16}), afterburner fuel flow ($w_{f,6}$), and nozzle throat and exit area positioning (A_8 and A_9, respectively).

Gas properties: Curve fits of data are used to compute variable thermodynamic gas properties. For each control volume the following equations are used:

$$c_p = f(T, f/a) \tag{A.10}$$

$$R = f(f/a) \approx R_a \tag{A.11}$$

$$c_v = c_p - \frac{R}{J} \tag{A.12}$$

$$\gamma = \frac{c_p}{c_v} \tag{A.13}$$

$$h = f(T, f/a) \tag{A.14}$$

Flight conditions and inlet: All jet engines have an inlet to bring free stream air into the engine. Because the inlet does no thermodynamic work, the total temperature through the inlet is constant.

The total pressure through the inlet changes because of several flow effects. Inlet pressure performance is characterized by the inlet total pressure recovery, or ram recovery, which measures the amount of the free stream flow conditions that are "recovered". The pressure recovery P_2/P_0 depends on a wide variety of factors, including the shape of the inlet, the speed of the aircraft, the airflow demands of the engine, and aircraft maneuvers. Recovery losses associated with the boundary layers on the inlet surface or flow separations in the duct are included in the inlet efficiency factor η_1.

For subsonic flight speeds, these losses are the only losses. At supersonic flight speeds, there are additional losses created by the shock waves necessary to reduce the flow speed to subsonic conditions for the compressor. The magnitude of the recovery loss depends on the specific inlet design and is usually determined experimentally.

The following conditions are provided as inputs to the engine model to define the flight conditions and inlet model:

$$
\begin{aligned}
P_{amb} &= f_1(a) && \text{(A.15)}\\
T_{amb} &= f_2(a) + dT_{amb} && \text{(A.16)}\\
\eta_1 &= 1.0 \quad \text{if} \quad M \leq 1.0 &&\\
&= 1.0 - 0.075(M-1.0)^{1.35} \quad \text{if} \quad M > 1.0 && \text{(A.17)}\\
T_2 &= T_{amb}\left[1.0 + \frac{(\gamma_1 - 1)M^2}{2}\right] && \text{(A.18)}\\
P_2 &= P_{amb} \cdot \eta_1 \left[1.0 + \frac{(\gamma_1 - 1)M^2}{2}\right]^{\gamma_1/(\gamma_1 - 1)} && \text{(A.19)}\\
\gamma_1 &= 1.4 && \text{(A.20)}
\end{aligned}
$$

where functions f_1 and f_2 are curve fits to atmospheric data.

Fan: Fan performance is represented by a set of overall performance maps. Separate maps are used for the tip and hub sections. The maps are assumed to represent fan performance with variable geometry at nominal, scheduled positions. Map-generated, fan-corrected airflow is adjusted to account for off-schedule

geometry effects. The following equations describe the fan model:

$$P_{21} = P_2 \cdot f\left(\frac{P_2}{P_{amb}}, \sqrt{\theta_2}, N_L\right) \quad (A.21)$$

$$h_2 = f(T_2) \quad (A.22)$$

$$h_{21} = h_2 + 5.858 \times 10^{-5} T_2 \cdot f(SEDM_2, ZSW_2) \quad (A.23)$$

$$T_{21} = f(h_{21}) \quad (A.24)$$

$$\eta_2 = \frac{h_{21} - h_2}{h_{21}} \quad (A.25)$$

$$w_2 = \sqrt{\theta_2}\frac{P_2}{P_{amb}} \cdot f(N_L) \quad (A.26)$$

$$PW_2 = h_2 \cdot w_2 \quad (A.27)$$

$$SM_2 = \left(k_2 \cdot \frac{w_2}{P_2/P_{amb}} - 1\right) \times 100 \quad (A.28)$$

Compressor and Booster: All turbine engines have a compressor to increase the pressure of the incoming air before it enters the combustor. Modern large turbofan engines usually use axial compressors.

The job of the compressor is to increase the pressure of the flow. We measure the increase by the compressor pressure ratio (CPR), which is the ratio of the air total pressure exiting the compressor to the air pressure entering the compressor. Since no external heat is being added to or extracted from the compressor during the pressure increase, the process is isentropic. From the conservation of energy, the compressor work per mass of airflow W_{27} is equal to the change in the specific enthalpy of the flow from the entrance to the exit of the compressor.

An average, single-stage axial compressor increases the pressure by only a factor of 1.2. But it is relatively easy to link together several stages and produce a multistage axial compressor. In the multistage compressor, the pressure is multiplied from row to row (8 stages at 1.2 per stage gives a factor of 4.3). Therefore, most high performance, high compression turbine engines use multi-staged axial compressors.

Overall performance maps are used for the compressor with a shift in the corrected airflow based on off-schedule values of variable-geometry position. The

following equations describe the compressor and booster models:

$$
\begin{aligned}
h_{27} &= 5.858 \times 10^{-5} \cdot f(SEDM_{27}, ZSW_{27}) && \text{(A.29)}\\
CPR &= \frac{P_3}{P_2} \geq 1.0 \\
&= \left(\frac{h_{27}}{0.23995} + 1\right)^{\gamma_1/(\gamma_1 - 1)} && \text{(A.30)}\\
P_3 &= P_2 \cdot CPR && \text{(A.31)}\\
\frac{T_3}{T_2} &= \left(\frac{P_3}{P_2}\right)^{(\gamma_{27}-1)/\gamma_{27}} && \text{(A.32)}\\
h_3 &= h_{27} + f(T_{27}, h_{27}) && \text{(A.33)}\\
T_3 &= f(h_3) && \text{(A.34)}\\
\eta_{27} &= \frac{h_{27} - h_{21}}{h_{27}} && \text{(A.35)}\\
w_{27} &= \sqrt{\theta_{27}} \frac{P_{27}}{P_{amb}} \cdot f(N_H) && \text{(A.36)}\\
w_3 &= w_{27} && \text{(A.37)}\\
PW_{27} &= h_3 \cdot w_{27} && \text{(A.38)}\\
W_{27} &= \frac{c_p \cdot T_2}{\eta_{27}} \left[CPR^{(\gamma_{27}-1)/\gamma_{27}} - 1\right] && \text{(A.39)}\\
SM_{27} &= \left(k_{27} \cdot \frac{w_{27}}{P_{27}/P_{amb}} - 1\right) \times 100 && \text{(A.40)}
\end{aligned}
$$

$$
\begin{aligned}
h_{27d} &= 5.858 \times 10^{-5} \cdot f(SEDM_{27d}, ZSW_{27d}) && \text{(A.41)}\\
\eta_{27} &= \frac{h_{27d} - h_{21}}{h_{27d}} && \text{(A.42)}\\
w_{27d} &= \sqrt{\theta_{27d}} \frac{P_{27d}}{P_{amb}} \cdot f(N_H) && \text{(A.43)}\\
PW_{27d} &= h_3 \cdot w_{27d} && \text{(A.44)}\\
SM_{27d} &= \left(k_{27d} \cdot \frac{w_{27d}}{P_{27d}/P_{amb}} - 1\right) \times 100 && \text{(A.45)}
\end{aligned}
$$

Main Burner: All turbine engines have a combustor, or burner, in which the fuel is combined with high pressure air and burned. The resulting high temperature exhaust gas is used to turn the power turbine and produce thrust when passed through a nozzle.

The burner sits between the compressor and the power turbine, and the burner is arranged like an annulus, or a doughnut. The central shaft that connects the turbine and compressor passes through the center hole. The burning occurs at a higher pressure than free stream because of the action of the compressor. The pressure in the burner remains nearly constant during burning, decreasing by only 1 to 2 per cent while the burner pressure ratio BPR is nearly equal to one.

As opposed to the compressor and power turbine, we cannot simply relate the total temperature ratio in the burner to the total pressure ratio. Under the adiabatic conditions of the compressor and turbine, the pressure ratio and temperature ratio are related. In the burner, heat is released in the combustion process, and the energy equation must be used to determine the temperature change.

In engine operation, we can set the fuel flow rate which determines a value for the fuel/air ratio f/a and sets the temperature ratio in the burner. The burner temperature ratio and pressure ratio determine a value for the engine temperature ratio, ETR, and engine pressure ratio, EPR, which in turn determine the theoretical engine thrust.

Total pressure losses are included in the models of the main combustor, bypass duct, mixer entrance, and augmentor. Heat addition associated with the burning of fuel in the main combustor is assumed to take place in volume v_4.

$$BPR = \frac{P_4}{P_3} \approx 1.0 \tag{A.46}$$

$$P_4 = P_3 - 7.57 \times 10^{-4} w_3^2 \cdot \frac{T_3}{P_3} \tag{A.47}$$

$$\begin{aligned} [1 + (f/a)_4]\, h_4 &= h_3 + (f/a)_4 \cdot \eta_4 \cdot Q_4 \\ [1 + (f/a)_4]\, c_p T_4 &= c_p T_3 + (f/a)_4 \cdot \eta_4 \cdot Q_4 \end{aligned} \tag{A.48}$$

$$\frac{T_4}{T_3} = \frac{1 + [(f/a)_4 \cdot \eta_4 \cdot Q_4]\,/(c_p T_3)}{1 + (f/a)_4} \tag{A.49}$$

Power Turbines: All gas turbine engines have a power turbine located downstream of the burner to extract energy from the hot flow and turn the compressor. The job of the turbine is to extract energy from the heated flow exiting the burner. The turbine is connected to the shaft, which is also connected to the compressor. As the flow passes through the turbine, the total pressure and temperature decrease.

We measure the decrease in pressure by the turbine pressure ratio (TPR), which is the ratio of the air pressure exiting the turbine to the air pressure entering the turbine. Since no external heat is being added to or extracted from the turbine during this process, the process is isentropic. The temperature ratio across the turbine is related to the pressure ratio by the isentropic flow equations.

The turbine, like the compressor, is composed of several rows of airfoil cascades. Some of the rows, called rotors, are connected to the central shaft and rotate at high speed. Other rows, called stators, are fixed and do not rotate. The job of the stators is to keep the flow from spiraling around the axis by bringing the flow back parallel to the axis.

Turbofan engines usually employ a separate turbine and shaft to power the fan. Such an arrangement is termed a two spool engine. Since the turbine extracts energy from the flow, the pressure decreases across the turbine. Because of the high pressure change across the turbine, the flow tends to leak around the tips of the blades.

From the conservation of energy, the turbine work per mass of airflow (W_T) is equal to the change in the specific enthalpy h of the flow from the entrance to the exit of the turbine.

The efficiency factor is included to account for the actual performance of the turbine as opposed to the ideal, isentropic performance. Because of mechanical inefficiencies, you cannot get 100% of the available work from the turbine.

Turbine blades exist in a much more hostile environment than compressor blades. Sitting just downstream of the burner, the blades experience flow temperatures of more than a thousand degrees Fahrenheit and sometimes they must be actively cooled.

Overall performance of the high and low-pressure turbines is represented by bivariate maps. Cooling bleed for each turbine is assumed to reenter the cycle at the turbine discharge although a portion of each bleed is assumed to do-work:

$$TPR_{HP} = \frac{P_{45}}{P_4} \leq 1.0 \tag{A.50}$$

$$\frac{T_{42}}{T_{41}} = \left(\frac{P_{42}}{P_4}\right)^{(\gamma_{41}-1)/\gamma_{41}} \tag{A.51}$$

$$h_{42} = h_{41} \cdot T_{41} \tag{A.52}$$

$$\eta_{42} = f(h_{41}, ZSE_{41}) \tag{A.53}$$

$$w_{41} = f\left(N_H, \frac{\sqrt{T_{41}}}{P_4}, ZSW_{41}\right) \tag{A.54}$$

$$PW_4 = w_{41} \cdot h_{42} \tag{A.55}$$

$$\begin{aligned} W_{HP} &= h_{42} - h_4 = c_p(T_{42} - T_4) \\ &= \eta_{42} \cdot c_p T_4 \left(1 - TPR_{HP}^{(\gamma_{41}-1)/\gamma_{41}}\right) \end{aligned} \tag{A.56}$$

$$TPR_{LP} = \frac{P_5}{P_{42}} \leq 1.0 \tag{A.57}$$

$$h_5 = h_{49} \cdot T_{49} \tag{A.58}$$

$$\eta_{49} = f(h_{49}, ZSE_{49}) \tag{A.59}$$

$$w_{49} = f\left(N_L, \frac{\sqrt{T_{49}}}{P_{42}}, ZSW_{49}\right) \tag{A.60}$$

$$PW_{48} = w_{49} \cdot h_5 \tag{A.61}$$

Nozzle: All jet engines have a nozzle which produces the thrust as described on the thrust section. The nozzle also sets the total mass flow rate through the engine. The nozzle sits downstream of the power turbines and does no work on the flow.

Afterburning turbofans require a variable geometry convergent-divergent (CD) nozzle. In this nozzle, the flow first converges down to the minimum area or throat, then is expanded through the divergent section to the exit. The variable geometry causes these nozzles to be heavier than a fixed geometry nozzle, but variable geometry provides efficient engine operation over a wider airflow range than a simple fixed nozzle.

Because the nozzle does no thermodynamic work, the total temperature through the nozzle is constant. The total pressure pt across the nozzle is constant as well. The static pressure at the exit of the nozzle is equal to free stream static pressure, unless the exiting flow is expanded to supersonic conditions. The ratio of the nozzle total to static pressure ratio is called the nozzle pressure ratio NPR.

A convergent-divergent nozzle configuration is assumed. The following equations define the basic nozzle model. Simplifications to the basic model, intended

to reduce computation time, are noted:

$$\frac{T_8}{T_5} = 1 \tag{A.62}$$

$$\frac{P_8}{P_5} = \left(\frac{T_8}{T_5}\right)^{\gamma_8/(\gamma_8-1)} \tag{A.63}$$

$$\gamma_8 = f\left(T_8, (f/a)_{68}\right) \tag{A.64}$$

$$NPR = \frac{P_8}{P_0} \tag{A.65}$$

$$h_8 = h_{8s} + \frac{V_8^2}{2\eta_8} \tag{A.66}$$

$$V_8 = \sqrt{2\eta_8 \cdot c_p T_8 \left[1 - \left(\frac{1}{NPR}\right)^{(\gamma_8-1)/\gamma_8}\right]} \tag{A.67}$$

$$F_G = \frac{w_8(1.0244 - 0.6067 \cdot (f/a)_{68})V_8}{32.17} \tag{A.68}$$

$$F_{ram} = w_2 \cdot M \cdot 1.5238\sqrt{T_{amb}} \tag{A.69}$$

$$F_N = F_G - F_{ram} \tag{A.70}$$

A.3 Symbols and Nomenclature

A	cross-sectional area
a	altitude
c_p	specific heat at constant pressure
c_v	specific heat at constant temperature
dT	temperature difference
F	thrust
f/a	fuel-to-air ratio
h	specific enthalpy
J	mechanical equivalent of heat
k	stall constant

M	Mach number
N	rotational spool speed
P	total pressure
PW	power
Q	heat
R	gas constant
T	total temperature
$SEDM$	efficiency scalar
SM	stall margin
V	Velocity
W	work
$\dot{w}$	mass flow rate
ZSW	scalar for ratio of tip rotor speed to axial flow velocity
ZSE	scalar for turbine efficiency
γ	ratio of specific heats
η	efficiency
ρ	density
θ	ratio of total temperature to standard-day temperature

Subscripts

a	air
amb	ambient
f	fuel

G	gross
H	high-pressure spool
HP	high-pressure turbine
j	station number
L	low-pressure spool
LP	low-pressure turbine
N	net

Appendix B

Aircraft Model

This appendix describes a generic, high-performance aircraft model, including detailed, full-envelope, nonlinear aerodynamics. The models is a collection of modules each performing a specific function. The primary modules are the aircraft actuator and surface command inputs, aircraft mass and geometry modeling, the atmospheric model the aerodynamics, the propulsion system and the observation variable modeling. Although the details of the model are provided in [101], the mathematical model is presented in this appendix for completeness of the dissertation.

B.1 Aircraft Forces and Motion

There are four forces that act on an airplane. The motion of an aircraft depends on the relative magnitude of the forces.

Weight is a force that is always directed toward the center of the earth. The weight is distributed throughout the airplane, but often assumed to act through a single point called the center of gravity. In flight, the airplane rotates about the center of gravity (cg).

To overcome the weight force, airplanes generate an opposing force called lift. Lift is generated by the motion of the airplane through the air and is an aerodynamic force. Lift is directed perpendicular to the flight direction. Aircraft lift acts through a single point called the center of pressure. The distribution of lift around the aircraft is important for solving the control problem. Aerodynamic surfaces

are used to control the aircraft in roll, pitch, and yaw.

As the airplane moves through the air, the air resists the motion of the aircraft and the resistance force is called drag. Drag is directed along and opposed to the flight direction. Like lift, drag acts through the aircraft center of pressure.

To overcome drag, airplanes use a propulsion system to generate a force called thrust. The direction of the thrust force depends on how the engines are attached to the aircraft.

In an ideal situation, the forces acting on an aircraft in flight can produce no net external force. In this situation the lift is equal to the weight, and the thrust is equal to the drag. The aircraft maintains a constant airspeed called the cruise velocity. Any force acting at some distance from the center of gravity produces a torque about the cg. When there is no rotation about the cg the aircraft is said to be trimmed.

During flight, it is necessary to control the attitude or orientation of a flying aircraft in all three dimensions. In flight, any aircraft will rotate about its center of gravity. We can define a three dimensional coordinate system through the center of gravity We can then define the orientation of the aircraft by the amount of rotation of the parts of the aircraft along these principal axes.

The yaw axis is defined to be perpendicular to the plane of the wings with its origin at the center of gravity and directed towards the bottom of the aircraft. A yaw motion is a movement of the nose of the aircraft from side to side. The pitch axis is perpendicular to the yaw axis and is parallel to the plane of the wings with its origin at the cg and directed towards the right wing tip. A pitch motion is an up or down movement of the nose of the aircraft. The roll axis is perpendicular to the other two axes with its origin at the cg, and is directed towards the nose of the aircraft. A rolling motion is an up and down movement of the wing tips of the aircraft.

In flight, the control surfaces of an aircraft produce aerodynamic forces. These forces are applied at the center of pressure of the control surfaces which are some distance from the aircraft cg and produce torques (or moments) about the principal axes. The torques cause the aircraft to rotate. The elevators produce a pitching moment, the rudder produces a yawing moment, and the ailerons produce a rolling moment. The ability to vary the amount of the force and the moment allows the

pilot to maneuver or to trim the aircraft.

B.2 Aircraft Description and Analytical Model

Figure B.1 shows the individual modules of the aircraft model and the flow of information between these modules that would be connected with user synthesized control laws to form a complete system model.

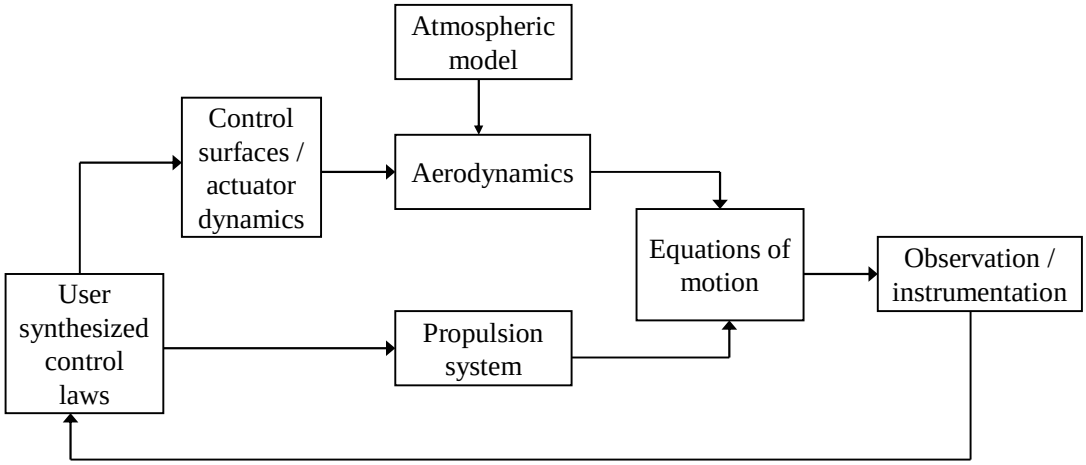


Figure B.1. Aircraft model structure and modules

Control surfaces and actuators: The aircraft model is a high performance, supersonic vehicle representative of current-day fighters. It is powered by two afterburning turbofan engines, each capable of producing approximately 32000 lb of thrust. The operational envelope for this vehicle, for trimmed, straight-and-level, $1g$ flight is provided in [101] for specified weight of 45000 lb. The envelope includes a maximum Mach number of 2.3 and an altitude limit in the 50000 to 60000 ft range.

The aircraft primary flight control surfaces consist of two horizontal stabilators which are capable of symmetric and differential movement, two conventional ailerons, and a single vertical rudder. The model includes identical actuators for all surfaces. These actuators are rate limited at 24°/sec and have a first order response modeled by

$$G(s) = \frac{20}{s + 20} \tag{B.1}$$

The command inputs to the aileron and stabilator surfaces are differential and

symmetric commands, which are separated into inputs to each of the surface actuator models. The resulting surface positions are then recombined to obtain the command response. Because of the nonlinearities in the command paths, the commands interact in a hard-to-predict way.

Aerodynamic model: The aerodynamics are modeled for the full vehicle envelope using multidimensional tables and linear interpolation to form nonlinear function generators. In general, these aerodynamic quantities are functions of Mach number **M** and some combination of angle of attack (α), angle of sideslip (β) and symmetric stabilator deflection.

The equations defining the aerodynamic model provide nondimensional force and moment coefficients. The longitudinal parameters are in the stability axis system; the lateral-directional parameters are given with respect to the body axis system. The equations used for this model are given in the following:

$$C_L = C_{L_0} + \Delta C_{L_n} n \tag{B.2}$$

$$C_m = C_{m_0} + \Delta C_{m_n} n + \frac{\bar{c}}{2V}\left(C_{m_q} q + C_{m_{\dot{\alpha}}} \dot{\alpha} + C_{L_0} \Delta N_0\right) \tag{B.3}$$

$$C_D = C_{D_0} + \Delta C_{D_h} + \Delta C_{D_{\mathbf{M}}} \tag{B.4}$$

$$C_y = C_{y_0} + C_{y_{\delta_A}} \delta_A + C_{y_{\delta_D}} \delta_D - \Delta C_{y_{\delta_R}} K_{\delta_{Ry}} \tag{B.5}$$

$$C_\ell = C_{\ell_0} + C_{\ell_{\delta_A}} \delta_A + C_{\ell_{\delta_D}} \delta_D - \Delta C_{\ell_{\delta_R}} K_{\delta_{R\ell}} + \frac{b}{2V}\left(C_{\ell_p} p + C_{\ell_r} r\right) \tag{B.6}$$

$$C_n = C_{n_0} + C_{n_{\delta_A}} \delta_A + C_{n_{\delta_D}} \delta_D + \Delta C_{n_{\delta_R}} K_{\delta_{Rn}} + \frac{b}{2V}\left(C_{n_p} p + C_{n_r} r\right) \tag{B.7}$$

Propulsion system model: The propulsion system model consists of two distinct engine models. The engines are similar, but not identical; thrust produced for identical throttle settings is not symmetrical. Each engine thrust vector is aligned with the body axis, and acts as a point located 10 ft behind the vehicle center of gravity and 4 ft laterally from the centerline. The thrust produced by each engine is a function of altitude (h), **M**, and throttle setting. Each engine is modeled as a nonlinear system having two separate sections; a core engine and an afterburner (augmentor) section with associated sequencing logic.

Throttled position inputs to the engine model are in degrees, with a minimum position of 20° and a maximum of 127°. The core section responds to throttle inputs up to 83°. The afterburner section begins to respond at a throttle position

of 91°. The core model has first order dynamics and rate limiting to model spool-up effects. The afterburner has a rate limiter and sequencing logic to model the fuel pump and pressure regulator effects.

Observation model: The observation variables provided by this model represents a broad class of parameters useful for vehicle analysis and control synthesis problems. These variables include the state and time derivatives of state, and control variables. Air data parameters, accelerations, flightpath terms are also included. These observation equations implicitly utilize an atmospheric model.

There body axis angular rates and three translational accelerations are available as observation variables. These include the x-body axis rate (u), the y-body axis rate (v) and the z-body axis rate (w). The equations defining these quantities together with their time derivatives are:

$$u = V\cos\alpha\cos\beta \tag{B.8}$$

$$v = V\sin\beta \tag{B.9}$$

$$w = V\sin\alpha\cos\beta \tag{B.10}$$

$$\dot{u} = \frac{1}{m}(X_T - gm\sin\theta - D\cos\alpha + L\sin\alpha) + rv - qw \tag{B.11}$$

$$\dot{v} = \frac{1}{m}(Y_T + gm\cos\theta\sin\phi + Y) + pw - ru \tag{B.12}$$

$$\dot{w} = \frac{1}{m}(Z_T + gm\cos\theta\cos\phi - D\sin\alpha - L\cos\alpha) + qu - pv \tag{B.13}$$

The vehicle body axis accelerations constitute the set of observation variables that, except for state variables, are most commonly used in aircraft control analysis and synthesis problems. These accelerations are measured in g units and are derived directly from the body axis forces. The equations used for the body axis accelerations are:

$$a_x = \frac{1}{g_0 m}(X_T - gm\sin\theta - D\cos\alpha + L\sin\alpha) \tag{B.14}$$

$$a_y = \frac{1}{g_0 m}(Y_T + gm\cos\theta\sin\phi + Y) \tag{B.15}$$

$$a_z = \frac{1}{g_0 m}(Z_T + gm\cos\theta\cos\phi - D\sin\alpha - L\cos\alpha) \tag{B.16}$$

where subscript 0 denotes standard-day, sea level conditions. The equations for

the output of the body axis accelerometers (denoted by subscript n) that are at vehicle center of gravity are:

$$a_{nx} = \frac{1}{g_0 m}(X_T - D\cos\alpha + L\sin\alpha) \quad \text{(B.17)}$$

$$a_{ny} = \frac{1}{g_0 m}(Y_T + Y) \quad \text{(B.18)}$$

$$a_{nz} = \frac{1}{g_0 m}(Z_T - D\sin\alpha - L\cos\alpha) \quad \text{(B.19)}$$

$$a_n = -a_{nz} \quad \text{(B.20)}$$

For orthogonal accelerometers that are aligned with the vehicle body axes but not at vehicle cg, the following equations apply:

$$a_{nx,i} = a_{nx} + \frac{1}{g_0}[(q^2 + r^2)x_x - (pq - \dot{r})y_x - (pr + \dot{q})z_x] \quad \text{(B.21)}$$

$$a_{ny,i} = a_{ny} + \frac{1}{g_0}[(pq + \dot{r})x_y - (p^2 + r^2)y_y + (qr - \dot{p})z_y] \quad \text{(B.22)}$$

$$a_{nz,i} = a_{nz} + \frac{1}{g_0}[(pr - \dot{q})x_z + (qr + \dot{p})y_z - (q^2 + p^2)z_z] \quad \text{(B.23)}$$

$$a_{n,i} = -a_{nz,i} \quad \text{(B.24)}$$

The load factor $n = L/W$, L being the total aerodynamic lift and W being the vehicle weight, is used in the set of acceleration equations. Flightpath related parameters such as flightpath angle (γ), flightpath acceleration (fpa), vertical acceleration ($\ddot{h}$), flightpath angle rate ($\dot{\gamma}$) are also computed as observation variables. The following equations are used to determine these quantities:

$$\gamma = \sin^{-1}\left(\frac{\dot{h}}{v}\right) \quad \text{(B.25)}$$

$$fpa = \frac{\dot{v}}{g_0} \quad \text{(B.26)}$$

$$\ddot{h} = a_x \sin\theta - a_y \sin\phi\cos\theta - a_z \cos\phi\cos\theta \quad \text{(B.27)}$$

$$\dot{\gamma} = \frac{V\ddot{h} - \dot{h}\dot{V}}{V\sqrt{V^2 - \dot{h}^2}} \quad \text{(B.28)}$$

Two energy related terms are computed as observation variables; specific energy

(E_s) and specific power (P_s), defined as:

$$E_s = h + \frac{V^2}{2g} \tag{B.29}$$

$$P_s = \dot{h} + \frac{V\dot{V}}{g} \tag{B.30}$$

Four force parameters, total aerodynamic lift L, total aerodynamic drag D, total aerodynamic normal force N and total aerodynamic axial force A, are defined by the following equations:

$$L = \bar{q}SC_L \tag{B.31}$$

$$D = \bar{q}SC_D \tag{B.32}$$

$$N = L\cos\alpha + D\sin\alpha \tag{B.33}$$

$$A = -L\sin\alpha + D\cos\alpha \tag{B.34}$$

The air data parameters that have impact on aircraft dynamics and control problems are the sensed parameters and the reference and scaling parameters. The sensed parameters are impact, free-stream and total pressures (q_c, p_a, p_t), and free-stream and total temperatures (T, T_t). The selected reference and scaling parameters include Mach number, dynamic pressure and speed of sound ($\mathbf{M}, \bar{q}, a$). The equations are given by:

$$a = \left(1.4\frac{p_0}{\rho_0 T_0}T\right)^{1/2} \tag{B.35}$$

$$\mathbf{M} = \frac{V}{a} \tag{B.36}$$

$$Re = \frac{\rho V \ell}{\mu} \tag{B.37}$$

$$\bar{q} = \frac{\rho V^2}{2} \tag{B.38}$$

$$q_c = \begin{cases} \left[(1.0 + 0.2\mathbf{M}^2)^{3.5} - 1.0\right] p_a, & (\mathbf{M} \le 1.0) \\ \left[1.2\mathbf{M}^2 \left(\frac{5.76\mathbf{M}^2}{5.6\mathbf{M}^2 - 0.8}\right)^{2.5} - 1.0\right] p_a, & (\mathbf{M} > 1.0) \end{cases} \tag{B.39}$$

$$p_t = p_a + q_c \tag{B.40}$$

$$T_t = T(1.0 + 0.2\mathbf{M}^2) \tag{B.41}$$

Equivalent airspeed (V_e) and calibrated airspeed (V_c) are calculated in knots. Calibrated airspeed is computed from an iterative procedure. The equations used for velocity calculations are:

$$V_e = \sqrt{\frac{2\bar{q}}{\rho_0}} = 17.17\sqrt{\bar{q}} \tag{B.42}$$

$$V_c = \begin{cases} 1479.116\sqrt{\left(\frac{q_c}{p_0}+1.0\right)^{2/7}-1.0}, & (V_c \le a_0) \\ 582.95174\sqrt{\left(\frac{q_c}{p_0}+1.0\right)\left[1.0-\frac{1.0}{7.0(V_c/a_0)^2}\right]^{2.5}}, & (V_c > a_0) \end{cases} \tag{B.43}$$

The final set of observation variables consists of measurements from sensors not located at the vehicle cg. These represent angle of attack (α_i), angle of sideslip (β_i), altitude (h_i) and altitude rate ($\dot{h}_i$) measurements displaced from cg by some distance. The following equations are used in the computation:

$$\alpha_i = \alpha - \left(\frac{qx - py}{V}\right) \tag{B.44}$$

$$\beta_i = \beta + \left(\frac{rx - pz}{V}\right) \tag{B.45}$$

$$h_i = h + x\sin\theta - y\sin\phi\cos\theta - z\cos\phi\cos\theta \tag{B.46}$$

$$\begin{aligned} \dot{h}_i = {} & \dot{h} + \dot{\theta}(x\cos\theta + y\sin\phi\sin\theta + z\cos\phi\sin\theta) \\ & -\dot{\phi}(y\cos\phi\cos\theta - z\sin\phi\cos\theta) \end{aligned} \tag{B.47}$$

The remaining parameters are total angular momentum (T), stability axis roll, pitch and yaw rates (p_s, q_s, r_s), defined as:

$$T = \frac{1}{2}\left(I_x p^2 - 2I_{xy}pq - 2I_{xz}pr + I_y q^2 - 2I_{yz}qr + I_z r^2\right) \tag{B.48}$$

$$p_s = p\cos\alpha + r\sin\alpha \tag{B.49}$$

$$q_s = q \tag{B.50}$$

$$r_s = -p\sin\alpha + r\cos\alpha \tag{B.51}$$

Equations of motion and atmospheric model: The nonlinear equations of motion used in this model are general six degree-of-freedom equations representing the flight dynamics of a rigid aircraft flying in a stationary atmosphere over a flat,

non-rotating Earth. The equations for each variable in the state vector are given in the following. The following equations for rotational accelerations are used:

$$\begin{aligned}
\dot{p} &= \frac{1}{\det I}[(\Sigma L)I_1 + (\Sigma M)I_2 + (\Sigma N)I_3 - p^2(I_{xz}I_2 - I_{xy}I_3) \\
&\quad +pq(I_{xz}I_1 - I_{yz}I_2 - D_zI_3) - pr(I_{xy}I_1 + D_yI_2 - I_{yz}I_3) \\
&\quad +q^2(I_{yz}I_1 - I_{xy}I_3) - qr(D_xI_1 - I_{xy}I_2 + I_{xz}I_3) - r^2(I_{yz}I_1 - I_{xz}I_2)] \quad \text{(B.52)} \\
\dot{q} &= \frac{1}{\det I}[(\Sigma L)I_2 + (\Sigma M)I_4 + (\Sigma N)I_5 - p^2(I_{xz}I_4 - I_{xy}I_5) \\
&\quad +pq(I_{xz}I_2 - I_{yz}I_4 - D_zI_5) - pr(I_{xy}I_2 + D_yI_4 - I_{yz}I_5) \\
&\quad +q^2(I_{yz}I_2 - I_{xy}I_5) - qr(D_xI_2 - I_{xy}I_4 + I_{xz}I_5) - r^2(I_{yz}I_2 - I_{xz}I_4)] \quad \text{(B.53)} \\
\dot{r} &= \frac{1}{\det I}[(\Sigma L)I_3 + (\Sigma M)I_5 + (\Sigma N)I_6 - p^2(I_{xz}I_5 - I_{xy}I_6) \\
&\quad +pq(I_{xz}I_3 - I_{yz}I_5 - D_zI_6) - pr(I_{xy}I_3 + D_yI_5 - I_{yz}I_6) \\
&\quad +q^2(I_{yz}I_3 - I_{xy}I_6) - qr(D_xI_3 - I_{xy}I_5 + I_{xz}I_6) - r^2(I_{yz}I_3 - I_{xz}I_5)] \quad \text{(B.54)}
\end{aligned}$$

where ΣL, ΣM and ΣN are the aerodynamic total moments about the $x-$, $y-$ and $z-$ body axes respectively, including power plant induced moments, and

$$\begin{aligned}
\det I &= I_xI_yI_z - 2I_{xy}I_{xz}I_{yz} - I_xI_{yz}^2 - I_yI_{xz}^2 - I_zI_{xy}^2 \\
I_1 &= I_yI_z - I_{yz}^2 \\
I_2 &= I_{xy}I_z + I_{yz}I_{xz} \\
I_3 &= I_{xy}I_{yz} + I_yI_{xz} \\
I_4 &= I_xI_z - I_{xz}^2 \\
I_5 &= I_xI_{yz} + I_{xy}I_{xz} \\
I_6 &= I_xI_y - I_{xy}^2 \\
D_x &= I_z - I_y \\
D_y &= I_x - I_z \\
D_z &= I_y - I_x
\end{aligned}$$

The following are the equations for translational accelerations:

$$\begin{aligned}
\dot{V} &= \frac{1}{m}[-D\cos\beta + Y\sin\beta + X_T\cos\alpha\cos\beta + Y_T\sin\beta + Z_T\sin\alpha\sin\beta \\
&\quad -mg(\sin\theta\cos\alpha\cos\beta - \cos\theta\sin\phi\sin\beta - \cos\theta\cos\phi\sin\alpha\cos\beta)] \quad \text{(B.55)}
\end{aligned}$$

$$\begin{aligned}\dot{\alpha} &= \frac{1}{Vm\cos\beta}[-L + Z_T\cos\alpha - X_T\sin\alpha + mg(\cos\theta\cos\phi\cos\alpha + \sin\theta\sin\alpha)] \\ &\quad +q - \tan\beta(p\cos\alpha + r\sin\alpha) \qquad \text{(B.56)} \\ \dot{\beta} &= \frac{1}{Vm}[D\sin\beta + Y\cos\beta - X_T\cos\alpha\sin\beta + Y_T\cos\beta - Z_T\sin\alpha\sin\beta \\ &\qquad +mg(\sin\theta\cos\alpha\sin\beta + \cos\theta\sin\phi\cos\beta - \cos\theta\cos\phi\sin\alpha\sin\beta)] \\ &\quad +p\sin\alpha - r\cos\alpha \qquad \text{(B.57)}\end{aligned}$$

where X_T, Y_T and Z_T are thrust along the $x-$, $y-$ and $z-$ body axes respectively, D is drag force, L is total aerodynamic lift, V is total velocity and Y is sideforce.

The equations defining the vehicle attitude rates and Earth-relative velocities are:

$$\begin{aligned}\dot{\theta} &= q\cos\phi - r\sin\phi \qquad \text{(B.58)} \\ \dot{\psi} &= q\sin\phi\sec\theta + r\cos\phi\sec\theta \qquad \text{(B.59)} \\ \dot{\phi} &= p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \qquad \text{(B.60)} \\ \dot{h} &= V(\cos\beta\cos\alpha\sin\theta - \sin\beta\sin\phi\cos\theta - \cos\beta\sin\alpha\cos\phi\cos\theta) \qquad \text{(B.61)} \\ \dot{x} &= V[\cos\beta\cos\alpha\cos\theta\cos\psi + \sin\beta(\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi) \\ &\quad + \cos\beta\sin\alpha(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi)] \qquad \text{(B.62)} \\ \dot{y} &= V[\cos\beta\cos\alpha\cos\theta\sin\psi + \sin\beta(\sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi) \\ &\quad + \cos\beta\sin\alpha(\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi)] \qquad \text{(B.63)}\end{aligned}$$

B.3 Symbols and Nomenclature

A	axial force
a	speed of sound in air
b	wingspan
C	force or moment coefficient
$\bar{c}$	mean aerodynamic chord
D	drag force

g	gravitational acceleration
h	altitude
I	aircraft inertia tensor
L	total aerodynamic lift or total body axis aerodynamic rolling moment
ℓ	generalized length
$\mathbf{M}$	Mach number
M	total body axis aerodynamic pitching moment
m	aircraft total mass
N	total body axis aerodynamic yawing moment
n	load factor
p	roll rate
q	pitch rate
Re	Reynolds number
r	yaw rate
T	ambient temperature or total angular momentum
V	total velocity
W	vehicle weight
α	angle of attack
β	angle of sideslip
γ	flightpath angle
θ	pitch angle

μ	coefficient of viscosity
ρ	density of air
ϕ	roll angle
ψ	heading angle

Subscripts

i	measurement not at aerodynamic reference
ℓ	rolling moment
m	pitching moment
n	yawing moment
s	stability axis
x	along the x-body axis
y	along the y-body axis
z	along the z-body axis
0	standard day, sea level conditions

Bibliography

[1] L. Catana, "The concept "system of philosophy": The case of Jacob Brucker's historiography of philosophy," *History and Theory*, vol. 44, no. 1, p. 72, 2005.

[2] *Science*, vol. 284, no. 5411, 1999.

[3] J. Banks, V. Dragan, and A. Jones, *Chaos : A Mathematical Introduction*. Cambridge University Press, 2003.

[4] C. Beck and F. Schögel, *Thermodynamics of Chaotic Systems an introduction*. Cambridge University Press, 1997.

[5] T. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley, 1991.

[6] H. Khalil, *Nonlinear Systems*, 3rd ed. Prentice Hall, 2002.

[7] S. Strogatz, *Sync: The Emerging Science of Spontaneous Order*. Theia Press, 2003.

[8] K. Zhou and J. Doyle, *Essentials of Robust Control*. Prentice Hall, 1997.

[9] F. Lewis and V. Syrmos, *Optimal Control*, 2nd ed. John Wiley & Sons, 1995.

[10] J. Bay, *Fundamentals of Linear State Space Systems*. McGraw-Hill, 1999.

[11] P. Jackson, *Introduction to Expert Systems*, 3rd ed. Harlow, England: Addison-Wesley, 1998.

[12] A. Ray, V. Phoha, and S. Phoha, Eds., *Quantitative Measure for Discrete Event Supervisory Control: Theory and Applications*. Springer, 2005.

[13] S. Phoha, S. Sircar, A. Ray, and I. Mayk, "Discrete event control of warfare dynamics," in *Symposium on Command and Control Research and the 9th Annual Decision Aids Conference*, Monterey, CA, 1992.

[14] J. Hopcroft, R. Motwani, and J. Ullman, *Introduction to Automata Theory, Languages, and Computation*, 2nd ed. Addison-Wesley, 2001.

[15] P. Ramadge and W. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control and Optimization*, vol. 25, no. 1, pp. 206–230, 1987.

[16] C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Kluwer Academic, 1999.

[17] A. Gollu and P. P. Varaiya, "Hybrid dynamical systems," in *Proceedings of the IEEE Conference on Decision and Control*, Tampa, FL, 1989, pp. 2708–2712.

[18] A. Balluchi, L. Benvenuti, G. M. Miconi, U. Pozzi, T. Villa, M. D. DiBenedetto, H. Wong-Toi, and A. L. Vincentelli, "Maximal safe set computation for idle speed control of an automotive engine," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science, N. Lynch and B. H. Krogh, Eds. Springer-Verlag, 2000, vol. 1790, pp. 32–44.

[19] G. Meyer, "Design of fight vehicle management systems," in *Proceedings of the IEEE Conference on Decision and Control*, Lake Buena Vista, FL, 1994.

[20] A. Back, J. Guckenheimer, and M. Myers, "A dynamical simulation facility for hybrid systems," in *Hybrid Systems*, ser. Lecture Notes in Computer Science, R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, Eds. Springer-Verlag, 1993, vol. 736, pp. 255–267.

[21] T. Bak, J. Bendtsen, and A. P. Ravn, "Hybrid control design for a wheeled mobile robot," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science, O. Malr and A. Pnueli, Eds. Springer-Verlag, 2003, vol. 2623, pp. 50–65.

[22] D. N. Godbole, J. Lygeros, and S. Sastry, "Hierarchical hybrid control: A case stuy," in *Proceedings of the IEEE Conference on Decision and Control*, Lake Buena Vista, FL, 1994, pp. 1592–1597.

[23] P. P. Varaiya, "Smart cars on smart roads: Problems of control," *IEEE Transactions on Automatic Control*, vol. 38, no. 2, pp. 195–207, 1993.

[24] M. S. Branicky, "Studies in hybrid systems: Modeling, analysis, and control," Ph.D. dissertation, Massachusetts Institute of Technology, 1995.

[25] R. Kumar and V. Garg, *Modeling and Control of Logical Discrete Event Systems*. Kluwer Academic, 1995.

[26] P. Ramadge and W. Wonham, "The control of discrete event systems," *Proceedings of IEEE*, vol. 77, no. 1, pp. 81–98, January 1989.

[27] A. Ray, A. Surana, and S. Phoha, "A language measure for supervisory control," *Applied Mathematics Letters*, vol. 16, pp. 985–991, 2003.

[28] A. Ray and S. Phoha, "Signed real measure of regular languages for discrete-event automata," *International Journal of Control*, vol. 76, no. 18, pp. 1800–1808, 2003.

[29] X. Wang and A. Ray, "A language measure for performance evaluation of discrete event supervisory control systems," *Applied Mathematical Modeling*, vol. 28, no. 9, pp. 817–883, 2004.

[30] A. Ray, J. Fu, and C. Lagoa, "Optimal supervisory control of finite state automata," *International Journal of Control*, vol. 77, no. 12, pp. 1083–1100, 2004.

[31] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. C. Teneketzis, "Failure diagnosis using discrete-event models," *IEEE Transactions on Control Systems Technology*, vol. 4, no. 2, pp. 105–124, 1996.

[32] R. Sengupta and S. Lafortune, "A deterministic optimal control theory for discrete event systems," in *Proceedings of the IEEE Conference on Decision and Control*, San Antonio, TX, 1993, pp. 1182–1187.

[33] F. Lin, "A note on optimal supervisory control," in *Proceedings of the IEEE Symposium on Intelligent Control*, Arlington, VA, 1991, pp. 227–232.

[34] W. Wonham and P. Ramadge, "On the supremal controllable sublanguage of a given language," *SIAM J. Control and Optimization*, vol. 25, no. 3, pp. 637–659, 1987.

[35] K. Passino and P. Antsaklis, "On the optimal control of discrete event systems," in *Proceedings of the Conference on Decision and Control*, Tampa, FL, 1989, pp. 2713–2718.

[36] Y. Li, "Hybrid synthesis of optimal control for discrete event systems," in *Proceedings of the IEEE Symposium on Intelligent Control*, Chicago, IL, 1993, pp. 308–313.

[37] S. Mohanty, V. Chandra, and R. Kumar, "A computer implementable algorithm for the synthesis of an optimal controller for acyclic discrete event processes," in *Proceedings of the IEEE International Conference on Robotics & Automation*, Detroit, MI, 1999, pp. 123–130.

[38] Y. Brave and M. Heyman, "On optimal attraction of discrete event processes," Technion, Haifa, Israel, Tech. Rep. Center for Intelligent Systems Report 9010, 1990.

[39] D. Bridges, J. Horn, and A. Ray, "Model-following control of a military helicopter with damage mitigation," in *AIAA Guidance, Navigation, and Control Conference*, San Francisco, CA, 2005.

[40] A. Ray and S. Phoha, "Integrated prognostics, maintenance, and life extending control," in *Computer-Aided Maintenance, Methodologies and Practice*, J. Lee and B. Wang, Eds. Dordrecht, The Netherlands: Kluwer Academic, 1999, pp. 136–169.

[41] P. Kallappa and A. Ray, "Life-extending load regulation of power plants," in *Encyclopedia of Electrical and Electronic Engineering*, J. Webster, Ed. New York, NY: Wiley, 1999, vol. 11, pp. 494–511.

[42] C. Lorenzo, M. Holmes, and A. Ray, "Nonlinear control of a reusable rocket engine for life extension," *AIAA Journal of Propulsion and Power*, vol. 17, no. 5, pp. 998–1004, 2001.

[43] A. Ray and J. Caplin, "Life extending control of aircraft: Trade-off between flight performance and structural durability," *The Aeronautical Journal*, vol. 104, no. 1039, pp. 397–408, 2000.

[44] A. Ray, "Symbolic dynamic analysis of complex systems for anomaly detection," *Signal Processing*, vol. 84, no. 7, pp. 1115–1130, 2004.

[45] S. Chin, A. Ray, and V. Rajagopalan, "Symbolic time series analysis for anomaly detection: A comparative evaluation," *Signal Processing*, vol. 85, no. 9, pp. 1859–1868, September 2005.

[46] A. Ray, "Signed real measure of regular languages for discrete event supervisory control," *International Journal of Control*, vol. 78, no. 12, pp. 949–967, 2005.

[47] X. Wang, A. Ray, and A. Khatkhate, "On-line identification of language measure parameters for discrete event supervisory control," *Applied Mathematical Modeling*, vol. 29, no. 6, pp. 597–613, 2005.

[48] G. Kaiser, *A Friendly Guide to Wavelets*. Birkhauser, 1994.

[49] B. Kitchens, *Symbolic Dynamics: One Sided, Two Sided and Countable State Markov Shifts*. Springer-Verlag, 1998.

[50] D. Lind and M. Marcus, *An Introduction to Symbolic Dynamics and Coding*. Cambridge University Press, 1995.

[51] H. Lewis, *The Foundations of Fuzzy Control*. Plenum Press, 1997.

[52] *IEEE Transactions on Systems, Man, and Cybernetics*, October 2000.

[53] A. Ray, "Characterization and mitigation of service failures in complex dynamical systems," The Pennsylvania State University, Tech. Rep. CSF MURI Third Annual Review, 2004.

[54] R. Kalman, P. Falb, and M. Arbib, *Topics in Mathematical System Theory*. McGraw-Hill, 1969.

[55] Y. Ho and X. Cao, *Perturbation Analysis of Discrete Event Dynamical Systems*. Kluwer Academic, 1991.

[56] A. Ray, J. Fu, and C. Lagoa, "Optimal control of regular languages with event disabling cost," *Demonstratio Mathematica*, vol. 37, no. 4, pp. 991–1013, 2004.

[57] W. Rudin, *Real and Complex Analysis*, 3rd ed. McGraw-Hill, 1987.

[58] C. Lagoa, J. Fu, and A. Ray, "Robust optimal control of regular languages," *Automatica*, vol. 41, no. 8, pp. 1339–1445, 2005.

[59] X. Wang, A. Ray, P. Lee, and J. Fu, "Optimal control of robot behavior using language measure," *International Journal of Vehicle Autonomous Systems*, vol. 2, no. 3/4, pp. 147–167, 2004.

[60] M. Yasar, J. Fu, and A. Ray, "Optimal discrete event control of gas turbine engines," in *Quantitative Measure for Discrete Event Supervisory Control: Theory and Applications*, V. P. A. Ray and S. Phoha, Eds. Springer, 2005, ch. 7, pp. 183–205.

[61] V. Phoha, A. Nadgar, A. Ray, S. Phoha, and V. Jain, "Supervisory control of software systems," *IEEE Transactions on Computers*, vol. 53, no. 9, pp. 1187–1199, 2004.

[62] R. Bapat and T. E. S. Raghavan, *Nonnegative Matrices and Applications*. Cambridge University Press, 1997.

[63] I. Chattopadhyay and A. Ray, "Renormalized measure of regular languages," *International Journal of Control*, vol. 79, no. 9, pp. 1107–1117, 2006.

[64] B. P. Gerkey, R. T. Vaughan, and A. Howard, "The Player/Stage project: Tools for multi-robot and distributed sensor systems," in *Proceedings of the International Conference on Advanced Robotics*, Coimbra, Portugal, June 2003, pp. 317–323.

[65] L. Ljung, *System Identification: Theory for the User*, 2nd ed. Prentice Hall, 1999.

[66] R. Badii and A. Politi, *Complexity, Hierarchical Structures and Scaling in Physics*. Cambridge, U.K: Cambridge University Press, 1997.

[67] A. Volponi, T. Brotherton, R. Luppold, and D. Simon, "Development of an information fusion system for engine diagnostics and health management," in *JANNAF Airbreathing Propulsion Subcommittee Meeting*, Colorado Springs, CO, 2003.

[68] D. Tolani, M. Yasar, A. Ray, and V. Yang, "Anomaly detection in aircraft gas turbine engines," *Journal of Aerospace Computing, Information, and Communication*, vol. 3, no. 2, pp. 44–51, 2006.

[69] M. Basseville and I. Nikiforov, *Detection of Abrupt Changes: Theory and Application*. New Jersey: PTR Prentice-Hall, 1993.

[70] C. R. Shalizi, K. Shalizi, and J. Crutchfeld, "An algorithm for pattern discovery in time series," *ArXiv Computer Science e-prints*, vol. cs/0210025, pp. 1–26, 2002.

[71] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, 2nd ed. New York, NY: John Wiley & Sons, 2001.

[72] J. Eckmann and D. Ruelle, "Ergodic theory of chaos and strange attractors," *Review of Modern Physics*, vol. 57, pp. 617–659, 1985.

[73] Y. Diao and K. Passino, "Stable fault-tolerant adaptive fuzzy/neural control for a turbine engine," *IEEE Transactions on Control Systems Technology*, vol. 9, no. 3, pp. 494–509, 2001.

[74] C. Daw, C. Finney, and E. Tracy, "A review of symbolic analysis of experimental data," *Review of Scientific Instruments*, vol. 74, no. 2, pp. 915–930, 2003.

[75] M. Kennel and M. Buhl, "Estimating good discrete partitions from observed data: Symbolic false nearest neighbors," *Physical Review Letters*, vol. 91, no. 8, p. 084102, 2003.

[76] V. Rajagopalan and A.Ray, "Symbolic time series analysis via wavelet-based partitioning," *Signal Processing*, vol. 86, no. 11, pp. 3309–3320, 2006.

[77] H. Abarbanel, *The Analysis of Observed Chaotic Data*. New York, NY: Springer-Verlag, 1996.

[78] D. Donoho, "De-noising by soft-thresholding," *IEEE Transactions on Information Theory*, vol. 41, no. 3, pp. 613–627, 1995.

[79] S. Haykin, *Neural Networks : A Comprehensive Foundation*. New Jersey: Prentice Hall, 1999.

[80] C. Bishop, *Neural Networks for Pattern Recognition*. New York: Oxford University Press, 1995.

[81] T. Brotherton and T. Johnson, "Anomaly detection for advanced military aircraft using neural networks," in *Proceedings of IEEE Aerospace Conference*, Montana, March 2001.

[82] S. C. Chin, "Real-time anomaly detection in complex dynamical systems," Ph.D. dissertation, The Pennsylvania State University, 2004.

[83] K. Parker and T. Guo, "Development of a turbofan engine simulation in a graphical simulation environment," in *JANNAF Aero-Propulsion Subcommittee Meeting*, Destin, FL, 2002.

[84] S. Adibhatla and K. Johnson, "Evaluation of nonlinear PSC algorithm on a variable cycle engine," in *AIAA Joint Propulsion Conference and Exhibit*, Monterey, CA, 1993.

[85] J. Litt, K. Parker, and S. Chatterjee, "Adaptive gas turbine engine control for deterioration compensation due to aging," in *International Symposium on Air Breathing Engines*, Cleveland, OH, 2003.

[86] J. DeLaat and C. Chang, "Active control of high frequency combustion instability in aircraft gas-turbine engines," in *International Symposium on Air Breathing Engines*, Cleveland, OH, 2003.

[87] M. Yasar and A. Ray, "Hierarchical control of aircraft propulsion systems: Discrete event supervisor aproach," *Control Engineering Practice*, vol. 15, no. 2, pp. 149–162, 2007.

[88] M. Yasar, J. F. Horn, and A. Ray, "Effects of supervisory decisions on nonlinear aircraft dynamics," in *Proceedings of American Control Conference*, Minneapolis, MN, 2006, pp. 4154–4159.

[89] D. Tolani, J. Horn, M. Yasar, and A. Ray, "Hierarchical control of rotorcraft for enhanced performance and structural durability," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, San Francisco, CA, 2005.

[90] A. Kreiner and K. Lietzau, "The use of onboard real-time models for jet engine control," MTU Aero Engines, Germany, Tech. Rep., 2003.

[91] S. Garg, D. L. Mattern, and R. E. Bullard, "Integrated flight/propulsion control system design based on a centralized approach," *AIAA Journal of Guidance, Control, and Dynamics*, vol. 14, no. 1, pp. 107–116, 1991.

[92] B. Stevens and F. Lewis, *Aircraft Control and Simulation.* Wiley-IEEE, 2003.

[93] P. Menon, V. Iragavarapu, and S. Garg, "Enhancing aircraft performance through flight/propulsion system integration," in *AIAA Guidance, Navigation and Control Conference*, San Diego, CA, 1996.

[94] L. McWilliams, E. Raven, and P. Sain, "A study of a propulsion control system for a VATOL aircraft," in *Proceedings of the IEEE National Aerospace and Electronics Conference*, vol. 1, 1989, pp. 356–363.

[95] D. Bates, S. Gatley, and I. Postlethwaite, "Integrated flight and propulsion control system design using H_∞ loop-shaping techniques," in *Proceedings of the Conference on Decision and Control*, Phoenix, AZ, 1999, pp. 1523–1528.

[96] S. Garg, "Partitioning of centralized integrated flight/propulsion control design for decentralized implementation," *IEEE Transactions on Control Systems Technology*, vol. 1, no. 2, pp. 93–100, 1993.

[97] M. Harefors and D. Bates, "Integrated propulsion-based flight control system for a civil transport aircraft," in *Proceedings of the IEEE International Conference on Control Applications*, Glasgow, Scotland, 2002, pp. 132–137.

[98] P. Schmidt, S. Garg, and B. Holowecky, "A parameter optimization approach to controller partitioning for integrated flight/propulsion control application," *IEEE Transactions on Control Systems Technology*, vol. 1, no. 1, pp. 21–36, 1993.

[99] R. Rysdyk and A. Calise, "Robust nonlinear adaptive flight control for consistent handling qualities," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 6, pp. 896–910, 2005.

[100] A. Perry, "The FlightGear flight simulator," in *USENIX Annual Techincal Conference*, Boston, MA, 2004.

[101] R. Brumbaugh, "An aircraft model for the AIAA controls design challenge," *Journal of Guidance, Control, and Dynamics*, vol. 17, no. 4, pp. 747–752, 1994.

[102] E. Duke, R. Antoniewicz, and K. Krambeer, "Derivation and definition of a linear aircraft model," NASA, Tech. Rep. RP-1207, 1998.

[103] A. Calise, S. Lee, and M. Sharma, "Development of a reconfigurable flight control law for the X-36 tailless fighter aircraft," in *AIAA Guidance, Navigation and Control Conference*, Denver, CO, 2000.

[104] N. Sahani and J. Horn, "Adaptive model inversion control of a helicopter with structural load limiting," in *AIAA Guidance, Navigation, and Control Conference*, Providence, RI, 2004.

[105] J. Howlett, "UH-60A Black Hawk engineering simulation program: Volume I - mathematical model," NASA, Tech. Rep. CR-177542, USAAVSCOM TR 89-A-001, Septemper 1989.

[106] E. N. Johnson and A. J. Calise, "A six degree-of-freedom adaptive flight control architecture for trajectory following," in *AIAA Guidance, Navigation, and Control Conference*, Monterey, CA, August 2002.

Vita

Murat Yasar

Murat Yasar was born in Ankara, Turkey on August 27, 1980. He received his baccalaureate in Mechanical Engineering from Orta Doğu Teknik Üniversitesi (Middle East Technical University) in June 2002. Murat Yasar joined the Pennsylvania State University as a graduate student in the Department of Mechanical and Nuclear Engineering in August 2002 and earned two consecutive masters degrees in Electrical Engineering (May 2005) and Mechanical Engineering (August 2005). He continued his doctoral research in Mechanical Engineering simultaneously under the guidance of Professor Asok Ray and successfully defended his thesis on March 1, 2007. His general research interests include control theory, signal processing and discrete event systems. Specific research experience of Murat Yasar includes quantitative evaluation of discrete event supervisors, signal processing and fault detection on complex systems and development of multi-computer simulation networks. He has publications in several international journals and conferences. He currently holds a position of Research Engineer with Techno-Sciences, Inc.