Model-based Estimation of Probabilistic Hybrid Automata

by

MELVIN MICHAEL HENRY

B.S. Systems and Computer Science
Howard University, 1999

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2002

Author_____

Department of Aeronautics and Astronautics
May 23, 2002

Certified by_____

Brian C. Williams
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by_____

Wallace E. Vander Velde
Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

MODEL-BASED ESTIMATION OF PROBABILISTIC HYBRID AUTOMATA

by

MELVIN MICHAEL HENRY

Submitted to the Department of Aeronautics and Astronautics
on May 23, 2002 in Partial Fulfillment of the
Requirements for the Degree of Master of Science in
Aeronautics and Astronautics Engineering

ABSTRACT

The ability to monitor and diagnose complex physical systems is critical for constructing highly autonomous artifacts that can function robustly in harsh environments over a long period of time. To accomplish this, we need to use high fidelity models that describe both the discrete stochastic behavior and the continuous dynamics of these complex systems. These models are used by a hybrid monitoring and diagnosis capability that tracks a system's dynamics as it moves between distinctive behavioral modes. In this thesis, we address the challenge of learning these hybrid discrete/continuous models.

We introduce a Hybrid Parameter Estimation System that extracts parameter estimates from sensor data. First, we review a method for Hybrid Modeling based on *Probabilistic Hybrid Automata* (PHA) [Hofbaur and Williams, 2002]. Second, we introduce *Hybrid Parameter Estimation* as a technique for learning the parameters of a PHA, by unifying standard nonlinear estimation techniques with classical probabilistic estimation techniques. Finally, we introduce the Hybrid Expectation Maximization algorithm for computing hybrid estimates by combining Hybrid Parameter Estimation with prior work on Hybrid State Estimation. This approach tracks the most desirable estimates based on statistical measure of probability. We demonstrate this algorithm on a simulated Mars habitat called BIO-Plex.

Thesis Supervisor: Brian C. Williams

Title: Associate Professor, Department of Aeronautics and Astronautics and
Department of Electrical Engineering and Computer Science

Acknowledgements

Foremost, I want to give thanks to Jehovah God. Great things he has done. He has truly brought me through another milestone in my life. Without health and strength, I would be unable to complete this thesis.

Further, I would like to express my deep appreciation to my thesis supervisor, Brian C. Williams. He has been a supporter but most of all a true scholar. I admire and owe him for his broad knowledge and deep insight – as well as his perseverance. Much credit must be given to him for managing a group where students can explore new directions and for creating opportunities for me to learn and grow as a researcher.

I am particularly indebted to Michael Hofbaur who functioned as my secondary advisor. He offered technical support while here at MIT (and after he returned to Austria). His vast knowledge was always available to me regardless of wherever he was. For that I am truly grateful. He is an incredible individual who I have great respect and admiration for. He has been there to listen and to offer valid (and quick) advice on both academic and professional matters.

I am also grateful to the many student readers in the department of Aeronautics and Astronautics at MIT. Special mention must be made of Seung Chung who read large sections of my thesis and offered helpful comments throughout. Also, I am indebted to Paul Elliott, who assisted with the data analysis of my results. You guys are the best. Your assistance made this thesis the quality it is today.

To Milton and Majorie Samuels, you are God's gift from above to me. Your kindness and love have truly helped me during these tiring times. Thanks for all your support. You made me feel like family.

Most of all, thanks to my wonderful family. Mom, Muriel E. Davis, thank you for your unconditional love and encouragement. To my sisters: Esther, Idetha, Sharlene, Monique, and Patricia, thanks for believing in me. To my brothers, Lesroy, and Livingstone, thanks for keeping me focused. Without all of you this victory would be shallow, if at all possible.

I pay tribute to all my close and special friends: Cherolyn Allen, Kashfia Rahman, and Audrey Cloyd, you are God sent. Majorie Joseph, Agnes Jordon, Charles Washington, and Brian Hall, thanks for all your prayers. Nathalie Liburd, Michel Farrell, Desma Alexander, Kenrick Bell, Merva Lake, Natalie Greaves and Marlo Hudson, thanks for your support.

Finally, I pay tribute to my sponsor, NASA. This work was supported in part by the NASA Cross Enterprise Technology Development Program under contract NAG2-1388, and by the NASA Intelligent Systems Program under contract NCC2-1235.

Melvin M. Henry

Cambridge, MA

23 May 2002

Table of Contents

List of Figures

Chapter 1 – Introduction

1.1 Motivation

Space and planetary vehicles need to be robust. The ability for a vehicle to perform robustly would be advantageous where human supervision and control is not feasible. Onboard human control may be unacceptable, for example, in circumstances where the environment is severe and/or unpredictable. In addition, vehicles may be required to remain in such environmental conditions for considerable periods of time.

In the past, efforts to achieve vehicular robustness have produced rather mediocre results. The failures of unmanned vehicles such as Mars Climate Orbiter and Mars Polar Lander indicates this [Young et al., 2000]. These failures implicate the lack of reactive software onboard such vehicles that has the capability to estimate and diagnose the behavior of the vehicles.

Onboard model-based estimation software is therefore critical for constructing highly autonomous artifacts that can operate robustly in severe environments. Such software has been successfully demonstrated on space vehicles that have the ability to estimate a vehicle's behavior as it moves through a series of discrete modes [Williams and Nayak, 1996; Muscettola et al., 1998]. However, these and other related methods fall short in that they characterized vehicular behavior as either purely discrete or purely continuous. Unfortunately, many real-world systems are defined by both discrete and continuous dynamics. Future estimation capabilities must use high fidelity models to capture the discrete stochastic behavior and the continuous dynamics of these vehicles.

The problem of *learning* a vehicle's behavior and dynamics presents some major challenges. First, the system's designer must understand the vehicle's behavior in order to construct a reasonable model that describes the vehicle. Constructing these models often requires a difficult analysis from empirical data or from first principles of physics. Misunderstanding the vehicle's behavior will result in the construction of an inappropriate vehicular model. Second, the creation of a high fidelity model that represents the correct behavior of these vehicles can be a challenging task [Nicholson, 1980]. For instance, if the model is a poor representation of the vehicle's behavior, then it is useless to experiment on such model.

The problem of learning the parameters of a system's model is known in engineering as the *parameter estimation problem.* We focus on maximum likelihood learning, in which a single set of parameters is estimated. We choose this approach over Bayesian approaches, which treat the parameters as random variables and compute the posterior distribution of the parameters given the data. We choose the former approach since it is a more classical approach to parameter estimation. One can also differentiate between on-line and off-line approaches to learning. On-line recursive algorithms can be obtained by computing the gradient or the second derivatives of the log likelihood [Ljung and Söderström, 1983]. Similar gradient-based methods can be obtained for offline methods. An alternative method for offline learning makes use of the Expectation Maximization (EM) algorithm [Dempster et al., 1977].

The EM algorithm is an iterative procedure for maximum likelihood parameter estimation from a data set. The algorithm has two steps: the E-Step and the M-step. The EM algorithm iterates between an E-step and a M-step. The E-step fixes the current

parameters and computes posterior probabilities over the hidden states given the posterior distributions. On the other hand, the M-step fixes current posterior probabilities and computes the parameters. For linear dynamic system models, the E-step is exactly the Kalman smoothing problem, and the M-step simplifies to a linear regression problem [Shumway and Stoffer, 1982; Digalakis et al., 1993].

In this thesis we focus on a Hybrid Expectation Maximization (Hybrid EM) algorithm that analyzes quantitative details through statistical analysis. This algorithm utilizes continuous/discrete automata and learns the equation parameters and transition probabilities of the automata. We develop this algorithm as a generalization of the EM algorithm to these automata. We demonstrate this algorithm on our target application of this research thesis and address this application in the next section. The major challenge of creating the Hybrid EM algorithm is the size of the automata. Hence central to this approach is a learning method based on automated model decomposition. We address this model decomposition capability in Chapter 7.

1.2 Application

The target application of the research in this thesis is the model learning of the (BIO-Plex) Bioregenerative Planetary Life Support System Test Complex (see Figure 1.1). For the purpose of this research, BIO-Plex can be broadly defined as a closed artificial environment that evaluates life support technologies pertaining to both biological and physiochemical life. It provides air, water and up to 90 % of the food necessary for a crew of four astronauts to survive on a continuous basis.

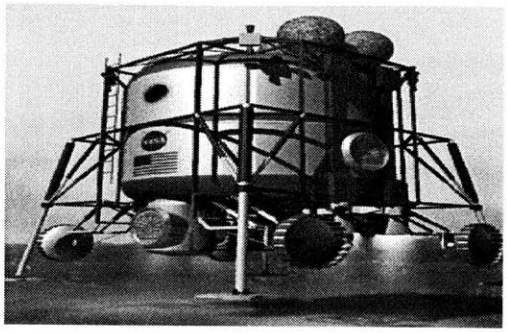


Figure 1.1: BIO-Plex complex

The actual form of the BIO-Plex system, its chemical composition, and the specific resources available to it were abstracted to a large degree. Instead, the focus was on constructing a Hybrid EM algorithm that can evaluate the behavior of the BIO-Plex complex. The information presented in this section is heavily based on the BIO-Plex complex requirement analysis presented in [Finn, 1999].

1.2.1 BIO-Plex complex

The area of interest includes creating a Hybrid EM algorithm that simulates the BIO-Plex environment currently being researched at NASA Johnson Space Center in Houston, Texas. Such an algorithm uses a model to learn the discrete stochastic behavior and the continuous dynamics of BIO-Plex. In addition, the algorithm uses a variant on the Expectation Maximization (EM) algorithm to accomplish our learning of the BIO-Plex model.

1.3 Problem Statement

The ability to model complex physical systems is critical for constructing highly autonomous artifacts that can function robustly in severe environments for considerable periods of time. Such models must capture both the discrete stochastic behavior and the continuous dynamics of these complex systems. We propose a hybrid model learning capability for physical systems that has the ability to learn a system's dynamics as it moves between distinctive behavioral modes. We address in this thesis, the challenges of learning and refining models of complex physical systems.

1.4 Thesis Layout

The next chapter, Chapter 2, is a literature review of crucial information to this thesis. First, we provide a preliminary review of notation used throughout, as well as ordinary difference equations (ODE) and dynamic systems concepts. Second, we provide basic definitions that are crucial to this thesis. Such definitions include hidden Markov models (HMM), continuous variables, and probabilistic hybrid automata (PHA) [Hofbaur and Williams, 2002].

In Chapter 3, we present a tutorial to learn a system model. We introduce a general overview of the Expectation Maximization *(EM)* algorithm. We show by example how the EM algorithm can be used to estimate the equation parameters (the parameters of equation) of the system.

Chapter 4 discusses two types of discrete/continuous automata. First we discuss deterministic discrete/continuous automata (DHA) using an n-channel enhancement-

mode MOSFET as an example. Second, we introduce Probabilistic discrete/continuous Automata using a Servo valve as an example. Finally, we demonstrate how to model a physical system with discrete/continuous behavior as hybrid automata (discrete/continuous automata).

In Chapter 5, we introduce a Hybrid Parameter Estimation System that extracts parameter estimates from sensor data. First, we introduce the method of *Hybrid Learning*. Hybrid Learning uses the Hybrid EM algorithm. Hybrid EM algorithm as a special case of the EM algorithm with added capabilities for handling Hybrid Systems – complex dynamic systems that have both discrete stochastic behavior and continuous dynamics. Second, we briefly describe the Hybrid Mode/State Estimation technique, which uses the E-step. Third, we introduce a technique called *Hybrid Parameter Estimation,* which uses the M-step. Fourth, we demonstrate how the E-step and the M-step fold together.

In Chapter 6, we discuss two types of applications: (1) Linear time-invariant (LTI) System, and (2) BIO-Plex – an advance life support (ALS) system dynamic simulation testbed. First, we model the behavior of the system as a probabilistic hybrid automaton. Second, we estimate the parameters of the system, given the PHA and measurement data. Finally we assess the quality of our Hybrid Parameter Estimation system comparing the parameter estimates with the real parameters.

In the final chapter, Chapter 7, we briefly mention other areas of inquiry related to parameter estimation. This is followed by a summary of the contributions of the thesis. Finally, we conclude by giving a range of research opportunities that arise from this work.

Chapter 2 – Terminology and Models

2.1 Overview

This chapter provides a literature review of preliminary ideas and definitions that the reader must understand in order to appreciate this work. Section 2.2 provides a preliminary review of the notation used throughout this thesis, as well as ordinary difference equations (ODE) and dynamic systems concepts. Section 2.3 provides basic definitions of related concepts pertaining to this thesis. Such definitions include hidden Markov models (HMM), continuous state variables, and probabilistic hybrid automata (PHA) [Hofbaur and Williams, 2002]. Knowledge of the aforementioned ideas and concepts is required in order to understand later chapters of this thesis.

2.2 Preliminaries

Throughout this thesis, we assume familiarity with the notation and concepts of ordinary difference equations [Arnold, 1973], and dynamic systems [Hirsch and Smale, 1974].

2.2.1 Notation

This thesis adopts the following conventions.

A boldface symbol denotes a matrix or vector, i.e.,

b, B scalars

b, B matrices or vectors

Inn general, we use boldfaced lowercase letters to represent vectors, lowercase italics to represent variables, and boldfaced capital letters to denote matrices. All vectors are assumed to be column vectors, unless explicitly stated.

The following conventions are widely use:

u control input

$\mathbf{v}_{c1}$ input disturbance

$\mathbf{v}_{c2}$ measurement noise

y observations or output vector

x state vector

Time is generally the only independent variable considered.

An estimate of the variable is denoted by a superscript caret; for example, $\hat{x}$ is an estimate of x.

A superscript tilde is used to denote a dummy or related variable. So $\tilde{x}$ denotes a variable similar in character to but different from x.

The equations and the figures are numbered according to the chapter. Section and subsection numbering do not affect the equation and figure numbering. For example, (3.2) denotes the second equation of Chapter 3, while Figure 4.10 denotes the tenth figure of Chapter 4.

2.2.2 Ordinary differential equations

In this thesis, the continuous dynamic behavior of our probabilistic automata are expressed using ordinary differential equations (ODEs):

$$\dot{x}(t) = f\left(x(t)\right), \tag{2.1}$$

where $x(t) \in X \subset \mathbb{R}^n$. Function $f : X \to \mathbb{R}^n$ is called a **vector field** on $\mathbb{R}^n$.

A system of ODEs is called **time-invariant** if its vector field does not depend explicitly on time.

A **plant** or an ODE with **inputs** and **outputs** is given by

$$\begin{aligned} \dot{x}(t) &= f\left(x(t), u(t)\right) \\ y(t) &= g\left(x(t)\right) \end{aligned} \tag{2.2}$$

where the number of components of the state vector, n, is called the **order** of the system. The input $u_{(\cdot)}$ and the output $y_{(\cdot)}$ have p and q components respectively. (That is, $x(t) \in X \subset \mathbb{R}^n$, $u(t) \in U \subset \mathbb{R}^p$, $y(t) \in Y \subset \mathbb{R}^q$, $f : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$, and $g : \mathbb{R}^n \to \mathbb{R}^q$).

2.2.3 Dynamic Systems

Examples of dynamic systems include a traveling space vehicle, a chemical plant, a home heating system, the population growth of a country, and the behavior of a country's economic structure. Some dynamic systems can be understood and analyzed intuitively. However, many dynamic systems, which are unfamiliar and complex, must be systematically analyzed. In order to study such complex dynamic systems, certain basic theory of dynamics must be understood.

2.2.3.1 Terminology

This section defines some basic terms used throughout this thesis to represent a physical system.

System: A system is a generally represented by a mathematical model, which can take many forms. Such forms include algebraic equations, finite state automata, difference equations, ordinary differential equations, and partial differential equations.

State x: The system's state summarizes the effects of all the past inputs to the system. Assuming that the system model is accurate, the current evolution of the system is specified by the previous value of the state and current inputs.

Control input, u: The control input is the quantity that can be manipulated (within constraints) to control the state of the system x and hence the output y.

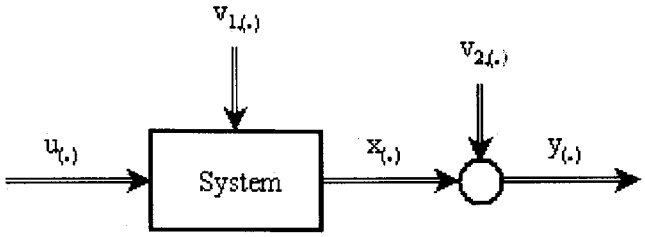
Noise or disturbance v_1 and v_2: The input disturbance v_1 models the uncertainty or noise in the inputs to the system. On the other hand, the measurement noise v_2 conceptually models the noise introduced by the measuring device (sensor).

Observation y: The observation is the output of the sensor.

2.2.3.2 Dynamic System model

Mathematical models, which include state space models can represent the behavior of a dynamic system. State Space models represent information about the past through a real-valued hidden state variable. The dependency between the present state variable and the previous state variable is specified through the dynamic equations of the systems and the noise model.

For example, Figure 2.1 represents a dynamic system that has an input $u_{(\cdot)}$ and an observed output $y_{(\cdot)}$. Where $u_{(\cdot)}$ can be either $u_{(k)}$ for discrete-time systems or $u_{(t)}$ for continuous-time systems.



$x_{(\cdot)}$	state of the system
$u_{(\cdot)}$	control input
$v_{1(\cdot)}$	input disturbance
$v_{2(\cdot)}$	measurement noise
$y_{(\cdot)}$	observation

Figure 2.1: Dynamic system model

In this thesis, algebraic, difference, or differential equations are use to represent the behavior of dynamic systems. The use of either differential or difference equations to represent the dynamic behavior of the system corresponds respectively to whether the behavior of the system is viewed as occurring in continuous or discrete time. In general, for a discrete-time dynamic system model, the evolution of the system's state x and the output of the system y are governed by the following difference equations:

$$\begin{aligned} x_{(k+1)} &= f(x_{(k)}, u_{(k)}, v_{1,(k)}) \\ y_{(k)} &= g(x_{(k)}, u_{(k)}, v_{2,(k)}) \end{aligned} \tag{2.3}$$

Whereas for a continuous-time dynamic system model, x and y are designated by the following differential equations:

$$\begin{aligned} \dot{x}_{(t)} &= f(x_{(t)}, u_{(t)}, v_{1,(t)}) \\ y_{(t)} &= g(x_{(t)}, u_{(t)}, v_{2,(t)}) \end{aligned} \tag{2.4}$$

If the *state transition function* $\boldsymbol{f}$ and *the output function* $\mathbf{g}$ of the dynamic system are linear, the following linear time-varying differential equations are obtained:

$$\begin{aligned} \dot{x}_{(t)} &= A_{(t)}x_{(t)} + B_{(t)}u_{(t)} + v_{1,(t)} \\ y_{(t)} &= C_{(t)}x_{(t)} + Du_{(t)} + v_{2,(t)} \end{aligned} \tag{2.5}$$

where A, B, C, and D are $n \times n$, $n \times p$, $q \times n$, and $q \times p$ respectively. If A, B, C, and D are constant, the dynamic system can be adequately approximated by a set of linear time-invariant difference equations:

$$\begin{aligned} x_{(k+1)} &= Ax_{(k)} + Bu_{(k)} + v_{1,(k)} \\ y_{(k)} &= Cx_{(k)} + Du_{(k)} + v_{2,(k)} \end{aligned} \tag{2.6}$$

These disturbances, $\mathbf{v}_1$ and $\mathbf{v}_2$, can be modeled as a random, uncorrelated sequence with zero-mean and Gaussian distribution.

2.3 Hybrid Systems Concepts

This section provides a summary of hybrid systems concepts, which include hidden Markov models, Continuous State variables, and Probabilistic Hybrid Automata [Hofbaur and Williams, 2002].

2.3.1 Hidden Markov Models

For a hidden Markov model, estimation is framed as the problem of determining the probability distribution $b_{(k)}$ over the modes $\mathcal{M}$ at time-step k. The probability of being in a mode m_i at time-step k is denoted $b_{(k)}[m_i]$. This problem is also called the belief-state update problem.

Definition 1:

The tuple, $\langle \mathcal{M}, y_d, \mathcal{U}_d, P_\Theta, P_\tau, P_o \rangle$, describes a *Hidden Markov Model (HMM).* Where $\mathcal{M}$, y_d and $\mathcal{U}_d$ represent finite sets of feasible modes[1] m_i, *observations* y_{di} and *control values* u_{di}, respectively. The *initial state function,* $P_\Theta[m_i]$, denotes the probability that m_i is the initial mode. The probability of transitioning from mode $m_{j,(k-1)}$ to $m_{i,(k)}$ at time step k given a discrete control action $u_{d,(k-1)}$ is denoted by the *mode transition function,* $P_\tau(m_i \mid u_d, m_j)$. The *observation function* $P_o(y_d \mid m_i)$ describes the probability that a discrete value $y_{d,(k)}$ is observed at k given $m_{i(k)}$.

In general, standard belief update for an HMM is an incremental process. This process computes the belief-state $b_{(k)}$ at the present time-step given the present observations $y_{d,(k)}$, the belief-state $b_{(k-1)}$ and the discrete control action $u_{d,(k-1)}$ from the previous time-step. Belief update is a two-step process. First, it uses the previous belief-state and the probabilistic transition function to estimate the belief-state denoted $b_{(\bullet k)}[m_i]$. Second, it updates this estimation to account for the present observations at time-step k resulting in the final belief-state $b_{(k)}[m_i]$:

$$
\begin{aligned}
b_{(\bullet k)}[m_i] &= \sum_{m_j \in M} P_\tau(m_i \mid u_{d,(k-1)}, m_j) b_{(k-1)}[m_j] \\
b_{(k)}[m_i] &= \frac{b_{(\bullet k)}[m_i] P_o(y_{d,(k)} \mid m_i)}{\sum_{m_j \in M} b_{(\bullet k)}[m_j] P_o(y_{d,(k)} \mid m_j)}
\end{aligned}
\tag{2.7}
$$

[1] To avoid confusion in terminology, the HMM state is refer to as the system's mode, and the term state is reserved for the state of a probabilistic hybrid automaton.

2.3.2 *Continuous State Variables*

To estimate the state of a continuous dynamic system, a state observer is generally used. One approach uses a discrete-time Kalman filter [Gelb, 1974] that captures the continuous dynamics based on a discrete-time model of the dynamic system.

Definition 2:

The tuple, $\langle \mathbf{x}_c, \mathbf{y}_c, \mathrm{u}_c, \mathrm{v}_c, \mathbf{f}_c, \mathbf{g}_c \rangle$, describes a *discrete-time model (DTM).* $\mathbf{x}_c$, $\mathbf{y}_c$, $\mathbf{u}_c$, and $\mathbf{v}_c$ represent the finite sets of *independent state-variables* x_{ci}, *observed variables* y_{ci}, *control variables* u_{ci}, and *exogenous input variables* v_{ci}, respectively. The *state transition function* $\mathbf{f}_c$ predicts the evolution of the state-variables $\mathbf{x}_{c,(k+1)} = \mathbf{f}_c(\mathbf{x}_{c,(k)}, \mathbf{u}_{c,(k)}, \mathbf{v}_{c,(k)})$ and the *output function* $\mathbf{g}_c$ specifies the observed variables $\mathbf{y}_{c,(k)} = \mathbf{g}_c(\mathbf{x}_{c,(k)}, \mathbf{v}_{c,(k)})$.

2.3.3 *Probabilistic Hybrid Automata*

A Probabilistic Hybrid Automata is a hidden Markov model encoded as a finite set of modes that exhibit continuous dynamic behavior, which can be expressed by difference, differential or algebraic equations [Hafbaur and Williams, 2002]:

Definition 3

A *probabilistic hybrid automata (PHA)* can be defined as a tuple $\langle \mathcal{M}, \mathbf{x}_c, \mathbf{y}_c, \mathcal{U}, \mathcal{F}_c, \mathcal{G}_c, \mathcal{T} \rangle$:

- The finite set $\mathcal{M}$ denotes the modes $m_i \in \mathcal{M}$ of the automaton.
- $\mathbf{x}_c$ and $\mathbf{y}_c$ denotes the set of independent continuous *state-variables* and *output variables* respectively. The set of input variables, $\mathcal{U} = \mathbf{u}_c \cup \mathbf{u}_d \cup \mathbf{v}_c$, is divided into continuous control variables $\mathbf{u}_c$, continuous exogenous variables $\mathbf{v}_c$, and discrete control variables $\mathbf{u}_d$. Components of continuous variables range over $\Re$, whereas components of discrete variables range over finite domains $\mathcal{D}$.
- The sets $\mathcal{F}_c$ and $\mathcal{G}_c$ associate with each mode $m_i \in \mathcal{M}$ functions $\mathbf{f}_{ci}$ and $\mathbf{g}_{ci}$ that govern the continuous dynamics exhibited at mode m_i by $\mathbf{x}_{c,(k+1)} = \mathbf{f}_{ci}(\mathbf{x}_{c,(k)}, \mathbf{u}_{c,(k)}, \mathbf{v}_{c,(k)})$ and $\mathbf{y}_{c,(k)} = \mathbf{g}_{ci}(\mathbf{x}_{c,(k)}, \mathbf{v}_{c,(k)})$.
- $\mathcal{T}$ specifies for each mode $m_{i(k)}$ a set of transition functions $\mathcal{T}_i = \{\tau_{i1}, ..., \tau_{in}\}$. Each transition function τ_{ij} has an associated guard condition $C_{ij}(x_{c,(k)}, u_{d,(k)})$ and specifies the probability distribution over target modes $m_{l(k+1)}$ together with an assignment for $x_{c,(k+1)}$.

Figure 2.2 shows a transition function for a mode m_1 with $\mathcal{T}_1 = \{\tau_{11}, \tau_{12}\}$. The transition function τ_{12} represents a transition to either mode m_3 with probability p_3 or to mode m_4, whenever its guard condition C_{12} is satisfied.

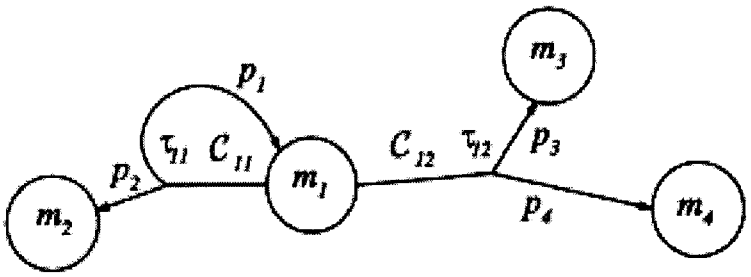


Figure 2.2: PHA showing transition functions associated with mode m_1

The tuple, $\langle m_{(k)}, x_{c,(k)} \rangle$, specifies the **hybrid state** $\mathbf{X}_{(k)}$ of a probabilistic hybrid automaton at time-step k. Where $m_{(k)} \in \mathcal{M}$ denotes the mode of the automaton and $x_{c,(k)}$ specifies the values of the state-variables. The shorter notation $m_{i,(k)}$ was used to denote $m_{(k)} = m_i$.

A probabilistic hybrid automaton is a model of a system with inputs $\mathbf{u}_c$, $\mathbf{u}_d$ and $\mathbf{v}_c$; output $\mathbf{y}_c$; internal hybrid state $\langle m, x_c \rangle$. The behavior of the $\mathcal{PHA}$, also called the **trajectory,** is represented by the sequence of hybrid states $\mathbf{t} = \{\mathbf{X}_{(0)}, \mathbf{X}_{(1)}, ..., \mathbf{X}_{(k)}\}$.

Chapter 3 – A Tutorial On Learning A System Model

3.1 Overview

The purpose of this thesis is to propose a model learning capability for complex physical systems. This learning capability is a variant on the Expectation Maximization (EM) algorithm. The EM algorithm is a procedure that is used in many fields of study. It is an efficient algorithm that solves the problem of learning the parameters of a system's model.

In this chapter, we provide a simple tutorial that demonstrates how the EM algorithm learns the parameters of a model. This algorithm has an E-step and a M-step. The E-step obtains measurements at each time step and estimates the mode of the system. This operation labels the measurement data with the most likely modes of the system. In the M-step, the labeling is used to separate the data according to the system's modes. Having done so allows us to estimate the equation parameters for each mode by using standard nonlinear estimation techniques [Shumway and Stoffer, 1982].

3.2 Motivating Example: Electrical System

Consider a system that is given an electrical current i as input and produces a voltage V as output. The behavior of the system (see Figure 3.1) can be approximated as having three components: a resistor R, a DC voltage source V_0 and an AC voltage source

V_d. Furthermore, the system's noise is modeled as an additive noise with a Gaussian distribution, specified by standard deviation σ.

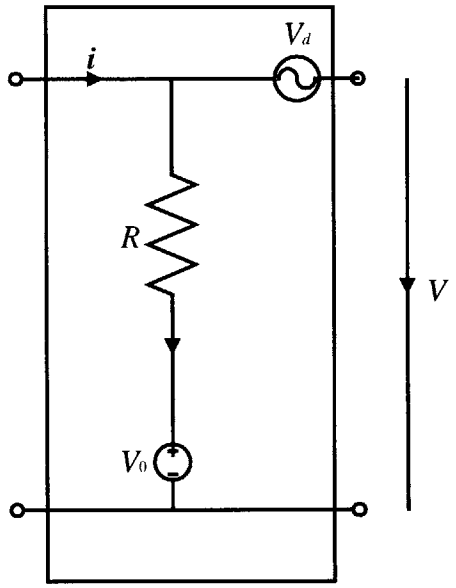


Figure 3.1: The System's behavior

The system has two operational modes, mode 1 and 2. Mode j is expressed by the linear relationship:

$$V = iR_j + V_{j,0} + V_d \tag{3.1}$$

where $j \in \{1,2\}$. In addition, the pair of parameters, $(R_j, V_{j,0})$, describes mode j. We assume that each mode of the system is affected by the same noise value V_d. For simplicity, we assume that the system has no failure modes.

3.3 Estimation Problem

Consider the problem of estimating the mode parameters, given a data set that does not specify the system's current mode. According to the EM algorithm, we can estimate the mode of the system by repeatedly performing the following two tasks: (1) estimate values of the pair of parameters, R and V_0, of a particular mode, given data for that mode, and (2) assign each data point to the mode that most likely generated that point. Task 1 is simple provided that task 2 is already solved, and vice versa. For example, assuming that we know the labeling of each data point, we can then calculate R and V_0 of a particular mode, by taking into account only those data points that are labeled with that particular mode. Similarly, if we know R and V_0 of each mode of the system, we can then label every data point in the data set with the most likely system's mode that generated the point.

3.4 EM algorithm

The basic structure of a typical EM algorithm is as follows:

- Initialize the modes of the system with random equation parameter values.
- Iterate through the data set until the parameter values converge:
  - E step: Label each data point to the most likely mode it belongs to.
  - M step: Update the parameter values of each mode using only the data points associated with that mode.

3.4.1 The Expectation (E) step:

We discuss the E step of the EM algorithm as two subtasks: (A) estimating probabilities and (B) estimating system modes.

(A) Estimating the mode probability of a data point

In the E step, we assume that the parameter values are known for each mode in the system. For every data point and system mode, we calculate a residual value that the data point belongs to the system mode. The residual is then used to compute the probability of the data point in that mode. Generally, the residual value of the *k*th data point is the numerical difference between the observation and the prediction of data point *k* for each mode. For our electrical example, we can simply calculate the residual value $r_{j,(k)}$ of a data point *k* being in mode *j* by the following linear equation:

$$r_{j,(k)} = i_k R_j + V_{j,0} - V_k \tag{3.2}$$

where R_j and $V_{j,0}$ represent the equation parameters of mode j that we need to estimate. Let us assume for example that modes 1 and 2 of the system are represented by equations: (1) $V = 5i + 2$ and (2) $V = 2i + 4$, respectively, and that the *k*th data point is $i = 2$ and $V = 10.2$. Then the residual value of the *k*th data point for mode 1 is $r_{1,(k)} = (5 \times 2 + 2) - 10.2 = 1.8$, while $r_{2,(k)} = (2 \times 2 + 4) - 10.2 = -2.2$ represents the residual value of the *k*th data point for mode 2.

Having calculated the residual values of every data point in the given data set for each mode, these values are then used to estimate the probabilities of each data point belonging to each mode of the system. For example, the probabilities of the *k*th data

point being in mode 1 or 2 can be derived using Bayes' Theorem [Kinney, 1997]. The probability of being in mode 1, $p_{1,(k)}$, given that the input is i_k and we observe V_k, is:

$$p_{1,(k)} = p(m_1 \mid i_k, V_k) = \frac{p(m_1, i_k, V_k)}{p(m_1, i_k, V_k) + p(m_2, i_k, V_k)} \tag{3.3}$$

This probability is assumed to be Gaussian and can be calculated by equation (3.4):

$$p_{1,(k)} = \left(\frac{\frac{1}{\sigma\sqrt{2\pi}} e^{-r_1^2/2\sigma^2}}{\frac{1}{\sigma\sqrt{2\pi}} e^{-r_1^2/2\sigma^2} + \frac{1}{\sigma\sqrt{2\pi}} e^{-r_2^2/2\sigma^2}} \right) \tag{3.4}$$

Likewise, $p_{2,(k)}$ denotes the probability of being in mode 2 given input i_k and observation V_k:

$$p_{2,(k)} = p(m_2 \mid i_k, V_k) = \frac{p(m_2, i_k, V_k)}{p(m_1, i_k, V_k) + p(m_2, i_k, V_k)} \tag{3.5}$$

This probability is assumed to be Gaussian and can be calculated by equation (3.6):

$$p_{2,(k)} = \left(\frac{\frac{1}{\sigma\sqrt{2\pi}} e^{-r_2^2/2\sigma^2}}{\frac{1}{\sigma\sqrt{2\pi}} e^{-r_1^2/2\sigma^2} + \frac{1}{\sigma\sqrt{2\pi}} e^{-r_2^2/2\sigma^2}} \right) \tag{3.6}$$

We can cancel $\frac{1}{\sigma\sqrt{2\pi}}$ from the equations (3.4) and (3.6) to produce equations (3.7) and (3.8), respectively, because both modes of the system have the same noise source:

$$p_{1,(k)} = \left(\frac{e^{-r_1^2(k)/2\sigma^2}}{e^{-r_1^2(k)/2\sigma^2} + e^{-r_2^2(k)/2\sigma^2}} \right) \tag{3.7}$$

$$p_{2,(k)} = \left(\frac{e^{-r_2^2(k)/2\sigma^2}}{e^{-r_1^2(k)/2\sigma^2} + e^{-r_2^2(k)/2\sigma^2}} \right) \tag{3.8}$$

(B) Estimating the system mode

For our example, given that the *k*th data point is $(i_k, V_k) = (2, 10.2)$, the probability of the *k*th data point being in modes 1 and 2 is $p_{1,(k)} \simeq 1$ and $p_{2,(k)} \simeq 0$, respectively. We can conclude that the most likely mode that generated the *k*th data point is mode 1. Assuming that the variance σ^2 for the two modes of the system are equivalent, if $r^2_{1,(k)}$ is smaller than $r^2_{2,(k)}$, then $\mathrm{p}_{1,(k)}$ is greater than $p_{2,(k)}$. Whenever $p_{1,(k)}$ is greater than $p_{2,(k)}$, we can conclude that the most likely system's mode of data point *k* is mode 1. This approach produces a data set, "labeled data", that relates each data point to the most likely mode of the system.

3.4.2 The Maximization (M) step:

The E step computed for every data point in the labeled data set, a probability of being in a particular mode of the system. The M step uses this labeling to update the equation parameters. The method we employ to estimate the equation parameters of each mode in the system's model is call a "weighted least squares fit". The weighted least squares fit method is a special case of the least square fit method [Strang, 1986]. An example of the least squares estimation problem is the linear regression problem in which we fit *N* data points (i_k, V_k) for $k = 1,...,N$, in the data set to a model of the system. For instance, mode *j* has two adjustable parameter values $(R_j, V_{j,0})$. We use equation (3.9), along with the data, to calculate the best estimated parameter values for mode *j*:

$$V_k = V(i_k) = V(i_k; R_j, V_{j,0}) = i_k R_j + V_{j,0} \tag{3.9}$$

For example, the parameters $(R_j, V_{j,0})$ of mode *j* are the solution to equation (3.10):

$$\begin{bmatrix} \sum_k i_k^2 & \sum_k i_k \\ \sum_k i_k & \sum_k 1 \end{bmatrix} \begin{bmatrix} R_j \\ V_{j,0} \end{bmatrix} = \begin{bmatrix} \sum_k i_k V_k \\ \sum_k V_k \end{bmatrix} \tag{3.10}$$

However, all the data points may not be equally weighted, and the above model would not accurately fit the data. To obtain a more accurate measure of how the model fits the data, we use the weighted least squares fit method. In the weighted least squares fit method, we are given the probability $p_j(k)$ for each data point in the data set. Equation (3.10) generalizes to:

$$\begin{bmatrix} \sum_k p_j(k) i_k^2 & \sum_k p_j(k) i_k \\ \sum_k p_j(k) i_k & \sum_k p_j(k) \end{bmatrix} \begin{bmatrix} R_j \\ V_{j,0} \end{bmatrix} = \begin{bmatrix} \sum_k p_j(k) i_k V_k \\ \sum_k p_j(k) V_k \end{bmatrix} \tag{3.11}$$

Therefore, for modes 1 and 2 in the above example, the following equations are obtained:

$$\begin{bmatrix} \sum_k p_1(k) i_k^2 & \sum_k p_1(k) i_k \\ \sum_k p_1(k) i_k & \sum_k p_1(k) \end{bmatrix} \begin{bmatrix} R_1 \\ V_{1,0} \end{bmatrix} = \begin{bmatrix} \sum_k p_1(k) i_k V_k \\ \sum_k p_1(k) V_k \end{bmatrix} \tag{3.12}$$

$$\begin{bmatrix} \sum_k p_2(k) i_k^2 & \sum_k p_2(k) i_k \\ \sum_k p_2(k) i_k & \sum_k p_2(k) \end{bmatrix} \begin{bmatrix} R_2 \\ V_{2,0} \end{bmatrix} = \begin{bmatrix} \sum_k p_2(k) i_k V_k \\ \sum_k p_2(k) V_k \end{bmatrix} \tag{3.13}$$

Calculating a set of parameter values is not the final outcome of the parameter estimation problem [Press, 1992]. Recall that a set of parameter values represents a system mode and that all the mode parameters of the system describe the model of the system. Our aim is to calculate the most likely set of parameters that best fit the model of the system. To accomplish this, we need to provide the following when we perform parameter estimation on a system: (1) obtain a set of parameter values which describe the model, (2) provide error estimates on the model parameters, and (3) provide a means to assess whether or not the model of the system is appropriate. A system model is considered appropriate when the model closely resemble a given data set (i.e. the model

can describe or matches the given data set). If item (3) suggests that the model is an unlikely fit to the data, then discard the set of parameter estimates and continue the parameter estimation process. When we obtain a model that closely resemble the given data set, we assume the EM algorithm has converged.

3.5 Summary

To summarize this chapter, we provided a general review of the EM algorithm, which iterates between an E-step and M-step. The E-step fixes the current parameters and computes a probability distribution over the modes given the measurement data. On the other hand, the M-step determines the most likely set of parameters, given the probability distribution produced by the E-step. The M step simplifies to a linear regression problem [Shumway and Stoffer, 1982].

Chapter 4 – Hybrid Automata

4.1 Overview

The goal of this chapter is to model the behavior of complex physical systems, which may be characterized by both discrete and continuous dynamics. For this purpose, we introduce a model called a Hybrid Automaton (*HA*). An HA is a modeling formalism that merges discrete automata (*DA*) with continuous systems models. Hybrid Automata allows us to represent both the discrete stochastic behavior and the continuous dynamics in an expressive way.

In this chapter, we discuss two types of Hybrid Automata. In the first type of HA, Deterministic Hybrid Automata (*DHA*), the regions of operation of a system are modeled as the modes of the HA. Mode changes are triggered whenever the physical system moves from one region of operation to another region. For each mode, there is a set of equations that describes the behavior of the HA within this mode. In this type of HA, all movements between the modes are considered to be deterministic.

In the second type of HA, Probabilistic Hybrid Automata (*PHA*), we model the automata slightly different. Unlikely DHA, PHA's mode changes are triggered whenever the continuous state variable, x, reaches the domain-boundary for a mode. In this type of HA, all the transitions between the modes are considered to be probabilistic.

4.2 Deterministic Hybrid Automata

We model the movement of a system between modes through a set of transitions. We consider the case of deterministic transitions in this section, using a MOSFET as an example.

4.2.1 Motivating Example: n-channel enhancement-mode MOSFET

The Metal-Oxide-Semiconductor Field-Effect Transistor (or MOSFET) has become one of the most important transistors used in Electronics today. Most microcomputer and memory circuits are comprised of thousands of MOSFETs on a small silicon board. MOSFETs are also used as voltage-controlled resistors, switches and in calculator chips [Mims III, 1983].

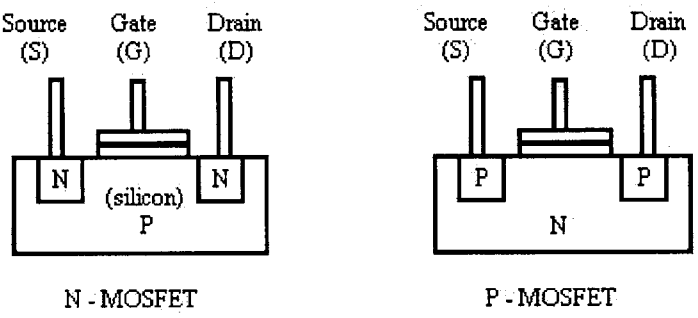


Figure 4.1: The n-channel and the p-channel MOSFET symbols

A MOSFET has three terminals: the source S, the gate G, and the drain D, as shown in Figure 4.1 [Horowitz and Hill, 1989; McWhorter and Evans, 1994].

Enhancement-mode MOSFETs are generally "off" by default and must be switched "on". They are switched "on" by a positive (for n-channel) or negative (for p-channel) bias voltage on the gate.

4.2.2 The behavior of the n-channel enhancement-mode MOSFET

The gate G has no electrical contact with the source S and the drain D. Since there is no electric current flow into the gate, only the gate-source voltage V_{GS} controls the behavior of the gate. A positive gate voltage (gate-source voltage) attracts electrons to the region below the gate. This creates a thin build up of electrons between the source and drain. At this point, current begins to flow through the channel. Also, V_{GS} determines the resistance of the channel. Figure 4.2 describes the overall behavior of the n-channel enhancement-mode MOSFET.

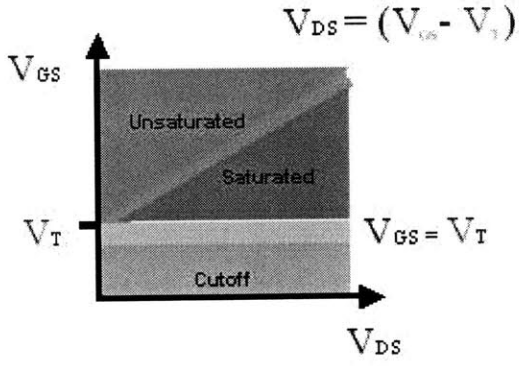


Figure 4.2: Graphical representation of the behavior of the MOSFET

V_{DS} and V_T denote the drain-source voltage and the threshold voltage (the value of the gate-source voltage when the n-channel enhancement-mode MOSFET becomes "on"), respectively.

The schematic of the n-channel enhancement-mode MOSFET is shown in Figure 4.3, where I_G and I_{DS} represent the gate current (the current at the gate), and the drain current, respectively.

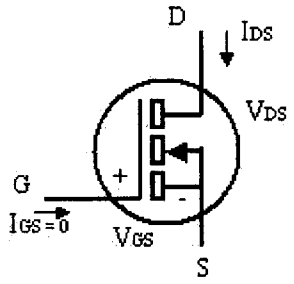


Figure 4.3: The schematic of the n-channel enhancement-mode MOSFET

The n-channel enhancement-mode MOSFET has three regions of operation: "Cutoff", "Unsaturated" and "Saturated". In the Cutoff region, the n-channel enhancement MOSFET is off.

For each region of operation, there is an algebraic equation that describes how the drain current varies with the gate-source voltage [White, 1994]. Whenever the n-channel enhancement-mode MOSFET is operating within the Cutoff region, no electrical current flows from the drain to the source (see equation (4.1)):

$$I_{DS} = 0 \tag{4.1}$$

In addition, the gate-source voltage is always less than the threshold voltage $(V_{GS} < V_T)$. Within the Saturated and the Unsaturated regions, the drain current depends on $V_{GS} - V_T$, the amount by which the gate-source voltage exceeds the threshold voltage. Within the Unsaturated region, current is now flowing from the drain to source. The drain current changes as the drain-source voltage and the gate-source voltage both change. Mathematically,

$$I_{DS} = \frac{K}{2}\left[2(V_{GS} - V_T)V_{DS} - V_{DS}^2\right] \tag{4.2}$$

However, in the saturation region, only changing the gate-source voltage changes the drain current. This relationship is expressed by equation (4.3):

$$I_{DS} = \frac{K}{2}(V_{GS} - V_T)^2 \tag{4.3}$$

In equations (4.2) and (4.3), the parameter K is a product of two factors: (1) the geometry of the n-channel enhancement MOSFET and (2) the capacitance of the silicon. Figure 4.4 is a graphical representation of how drain current varies with gate-source voltage during these three regions of operation.

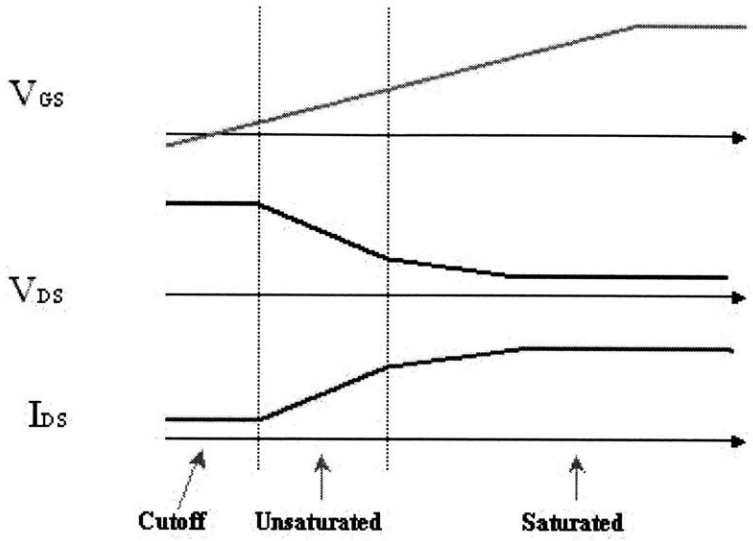


Figure 4.4: The relationship between I_{DS} and V_{DS} with increasing V_{GS}

4.2.3 Model description of the MOSFET system

Next we model the behavior of the n-channel enhancement-mode MOSFET as a hybrid automaton (see Figure 4.5). Our hybrid automaton consists of three nominal operational modes: Cutoff m_c (for the Cutoff region of operation), Unsaturated m_u (for the Unsaturated region) and Saturated m_s (for the Saturated region). We assume the system operates perfectly, i.e. our automaton has no failure modes.

Finally we model the movement of a system between modes through a set of transitions. There are nine possible deterministic transitions. For example, when the n-channel enhancement-mode MOSFET is in Cutoff m_c at time step t_k, and a transition occurs, we can observe one of three scenarios at time step t_{k+1}: the MOSFET remains in

mode m_c, the MOSFET is in mode m_s, or the MOSFET is in mode m_u. When the MOSFET remains in m_c, a self-transition τ_{11} occurs. If the MOSFET moves to mode m_s, transition τ_{12} occurs. If the MOSFET moves to mode m_u, transition τ_{13} occurs.

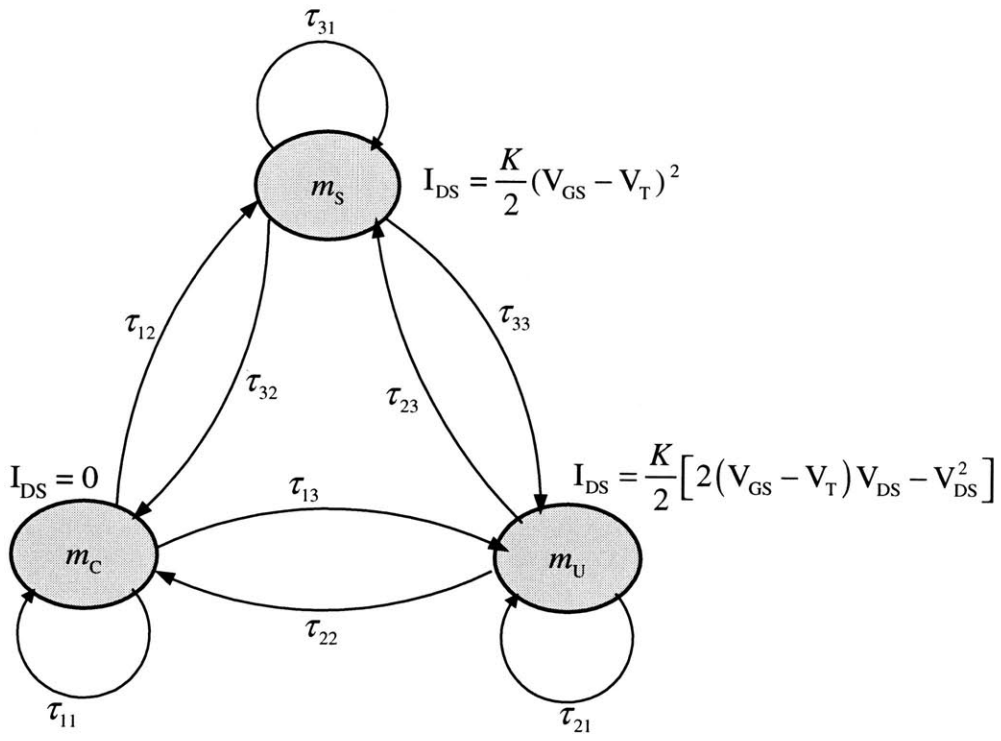


Figure 4.5: Hybrid Automaton of the n-channel enhancement-mode MOSFET

4.3 Probabilistic Hybrid Automata

In this section, we capture the behavior of a servo valve and then model its behavior as a probabilistic hybrid automaton.

4.3.1 Motivation Example: Servo valve

A servo valve is a continuously operated valve, which controls the carbon dioxide (CO_2) flow into the plant growth chamber (PGC) of the Bioregenerative Planetary Life Support System Test Complex (BIO-Plex). The schematic of the valve is shown in Figure 4.6 where P_1, P_2, u, and Q represent the CO_2 pressure at the inlet of the valve, the CO_2 pressure at the outlet of the valve, the continuous command input [0...1] and the CO_2 flow rate, respectively.

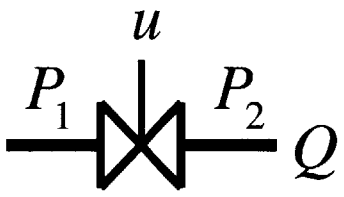


Figure 4.6: The schematic of the Servo valve

The continuous input u controls the opening of the servo valve from $u = 0$ (closed) to $u = 1$ (completely open). The flow rate exhibits a transient behavior whenever we change the value of u abruptly, for example, from $u = 0$ to $u = 0.3$ at time

$t = 1T$ (where T denotes the sampling rate of our model learning and refinement system). In this case and under the assumption that the pressures P_1 and P_2 remain constant, the flow rate Q transitions from $Q = 0$ to $Q = 0.3Q_{max}$, where Q_{max} denotes the flow rate of the fully open valve for the pressure difference $P_1 - P_2$.

4.3.2 Learning the system behavior

The transient behavior is assumed to be fast compared to the sampling rate of our Hybrid EM algorithm, that is the flow rate of the valve settles at its new operational point prior to the next sampling time point (see Figure 4.7). Furthermore, we want to capture the imperfect behavior of the servo valve due to friction, where the flow rate deviates from the desired value by an offset α, for example, $u = 0.3$ can lead to the flow rate $Q = 0.3Q_{max} + \alpha$ as shown in Figure 4.7.

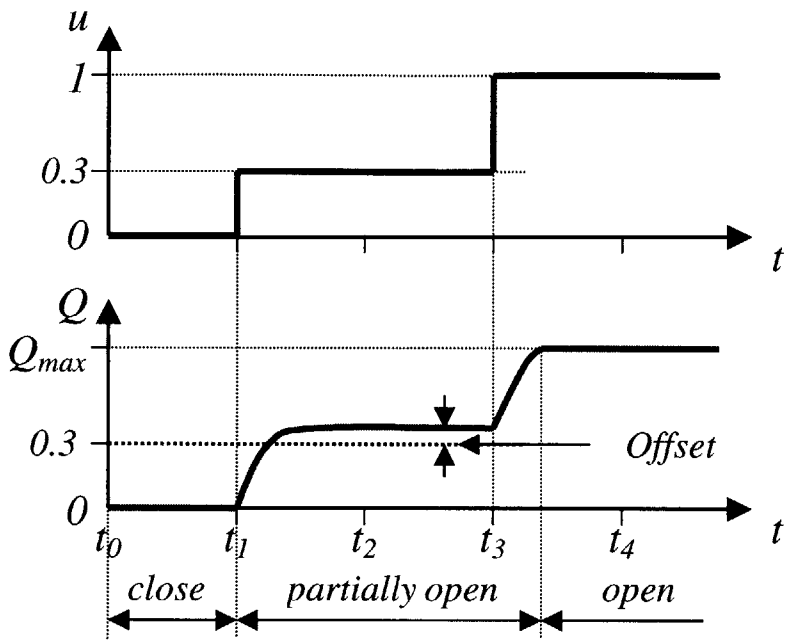


Figure 4.7: command input (u) vs. the valve opening (Q)

Our Hybrid EM algorithm monitors the servo valve by taking sample readings (of the command u, the pressures P_1, P_2 and the flow rate Q) at specific time points $t_k = kT + t_0$, where T represents the sampling rate of the system and t_0 denotes the initial time point. Given the fast transient behavior of the servo valve with respect to the sampling rate, we can model the behavior of the valve as follows: A command u at the time point t_k leads to an opening of the valve of $A_0(u_k + \alpha)$ at the following time point t_{k+1} (see Figure 4.8).

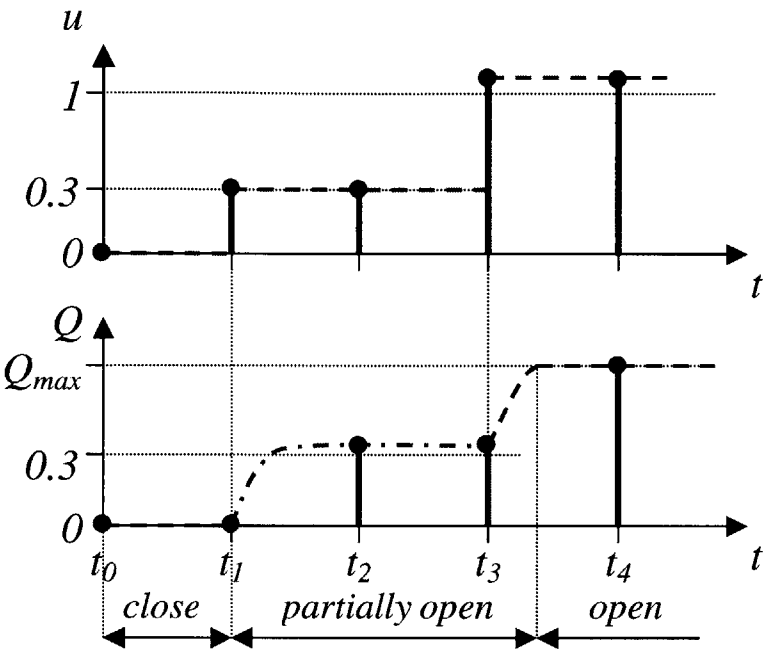


Figure 4.8: The sampling rate of the valve

A_0 denotes the cross-sectional area of the completely open valve. This allows us to state the following difference equation for the partially open servo valve:

$$\begin{aligned} x_k &= u_{k-1} \\ Q_k &= \mu A_0 (x_k + \alpha)\sqrt{P_{1,k} - P_{2,k}} \end{aligned} \tag{4.4}$$

where x_k denotes the state variable at time step t_k and μ represents the rheological resistance of the servo value.

Depending on the value of u at the previous time step t_{k-1}, the servo valve can be close, partially open or completely open at time step t_k. These three cases represent the

nominal operational modes of the valve. Allowing the control command to range over a wider interval (e.g. whenever u is the output of a non-limiting continuous controller), it makes sense to abstract the limiting behavior of the valve by distinguishing among these three modes explicitly. For example, whenever $u_{k-1} \geq 1$, we assume a completely open valve at the next time step leading to the maximal possible flow rate for a given pressure difference:

$$Q_k = \mu \mathrm{A}_0 \sqrt{P_{1,k} - P_{2,k}} \tag{4.5}$$

and when $u_{k-1} \leq 0$, we assume a close valve at time step t_k leading to zero flow rate

$$Q_k = 0 \tag{4.6}$$

irrespective of the pressure difference.

4.3.3 Model description of servo valve

We use the aforementioned description of the servo valve to model the operational modes of the value: when the servo valve is closed, we model our system to be in mode m_c and whenever the valve is partially open, we model our system to be in mode m_{po}. A completely open valve causes us to assume that our system is in mode m_o.

Like most real systems, the servo valve can fail to function properly during its operation. Depending on the value of u at the previous time step t_{k-1}, the valve can become stuck-closed, stuck-partially open, or stuck-open at time step t_k. These three scenarios represent the common failure modes of the servo valve. For example, if we assume a closed valve at time point t_{k-1} and we issue a continuous input of $u_{k-1} > 0$, but we observe a flow rate of $Q_k = 0$, instead of a flow rate of $Q_k > 0$ at time point t_k, then

we model this scenario as one of the common failure modes of our system. Here we assume we have a valve stuck-closed. Similarly, if we assume a completely open valve at time step t_{k-1} and we issue a command $u_{k-1} \leq 0$, but we observe a flow rate at the next sampling point k, we assume the servo valve is stuck-open. The final failure mode of the servo valve exists whenever the valve is assumed to be partially open and a command input of either $u_{k-1} \leq 0$ or $u_{k-1} \geq 1$ is issued, and we observe the same fluid flow rate at time step t_k, then we assume that the valve is stuck-partially open.

We use the following model description to illustrate the common failure modes for the servo valve: when the valve becomes stuck-closed, our system is assumed to be in mode m_{sc}, and whenever the valve is assumed to be stuck-open, we model this valve's behavior as a different failure mode m_{so}. Similarly, whenever the servo valve is stuck-partially open, we model our system to be in mode m_{pso}.

Autonomous mode changes are triggered whenever the continuous state variable x reaches the domain-boundary for a mode. For example, if we assume a closed valve at time t_{k-1} and $u_{k-1} > 0$, then we observe a gas flow at the next sampling point t_k. This can be modeled by transiting among the modes that are triggered by the value of the state-variable x_k, that is, the transitions are guarded by functions of the form $x_k > 0$. The transitions are assumed to take place instantaneously, to be more specific, we model mode transitions to be guarded on the state variable immediately at time step t_k and use the transitioned mode as the valid one for this time step (one could also think that the transition takes place immediately before the time point t_k).

Given the aforementioned model description of the servo valve, we can represent it as a PHA. This automaton can be used to model the modes of the servo valve and how we transition between these system modes whenever we change the command input. In the remaining section, we discuss how to formally model a hybrid system as a Probabilistic Hybrid Automaton using the servo valve as an example.

4.3.4 Hybrid Modeling

A PHA describes a hybrid system as a hidden Markov model encoded as a set of modes that exhibit continuous dynamic behaviors, which are expressed by differential, difference, or algebraic equations. By definition, we frame a single Probabilistic Hybrid Automaton as an automaton that consist of a set of modes, a set of transitions, and a set of variables. The set of variables is comprised of input(s), state variable(s), and output(s) of a PHA (see Figure 4.9).

PHA:

Variables: # input(s), # state(s), # output(s)

Modes: $\mathcal{M} = \{m_{(0)}, ..., m_{(k)}\}$ $\Leftrightarrow$ finite set of modes in the automaton

Transitions: $\tau_i = \{\tau_{i1}, ..., \tau_{in}\}$ $\Leftrightarrow$ finite set of transitions in the automaton

Figure 4.9: PHA Structure of a Hybrid System

The behavior of a servo valve, for instance, can be modeled as a PHA with a continuous command input u, a state variable x, and an output Q, which denotes the

flow rate of the valve. Recall that a servo valve can be modeled with three nominal operational modes, which are closed m_c, partially open m_{po}, and completely open m_o (see Figure 4.10). In addition, a servo valve has three common failure modes, which are stuck-closed m_{sc}, stuck-open m_{so}, and stuck-partially open m_{pso}. Shared among the six distinctive modes of the PHA, are the sixteen common transitions τ_1, τ_2, ... τ_{16}. Using this information and the modeling formalism cited above, we model the servo valve as a PHA in Figure 4.10.

PHA: Sv

Variables: 1 input u, 1 state $\mathbf{x}$, 1 output Q

Modes: m_c, m_{po}, m_o, m_{sc}, m_{so}, m_{pso}

Transitions: τ_1, τ_2, τ_3, τ_4, τ_5, τ_6, τ_7, τ_8, τ_9, τ_{10}, τ_{11}, τ_{12}, τ_{13}, τ_{14}, τ_{15}, τ_{16}

Figure 4.10: PHA Structure of the Servo valve

Next, a mode of a PHA (see Figure 4.11) is comprised of: (1) an associated set of ordinary differential (ODEs), difference, or algebraic equation(s), which describes the behavior of the system in the mode, (2) an associated set of guards that must be satisfied in order for a particular transition function to occur, (3) a set of transition functions that transition from the source modes to the target modes in the PHA whenever a set of particular guards are satisfied. Such mode transitions are a set of outgoing transitions from a source mode.

Mode: $m_{(k)}$

ODE: the set of ordinary differential/difference equations for this mode
Guards: $C_{ij}, \ldots, C_{in}$ $\Leftrightarrow$ guards satisfied when exiting m_k
Transitions: $\tau_r, \ldots, \tau_s$) $\Leftrightarrow$ the set of transitions for mode m_k

Figure 4.11: Mode Structure of the PHA

For instance, in Figure 4.12, when the servo valve is closed and we issue a command u to the valve, we can observe one of three scenarios: the valve remains closed m_c, the valve is stuck-closed m_{sc} due to mechanical failure, or the valve becomes partially open m_{po}. If the issued command u requests the servo valve to remain closed, (a transition from mode m_c to m_c), a self-transition τ_1 occurs. The transition τ_1 is guarded by the function of the form $x_k \leq 0$ (C_{11} satisfied). Once C_{11} remains satisfied, the servo valve remains in m_c (i.e. no modal change occurs). On the hand, if the issued command u is to open the valve, either a transition τ_2 from mode m_c to m_{sc}, or a transition τ_3 from mode m_c to m_{po} must occur. Recall that autonomous mode changes are triggered whenever the continuous state variable x reaches the domain-boundary for a mode. When $x_k \geq 0$ occurs, guard C_{12} is satisfied and either transition τ_2 or τ_3 occurs.

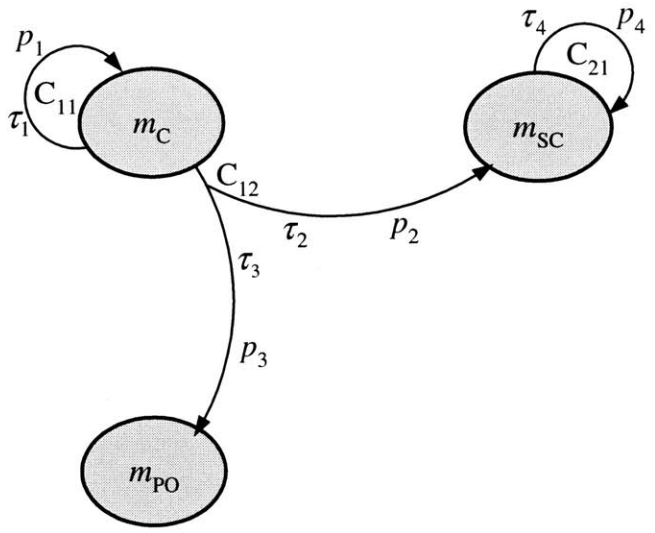


Figure 4.12: Modeling mode transitions from closed valve to either to stuck at close valve or partially open valve

Figure 4.13 illustrates how we model the closed valve mode of the servo valve using the PHA formalism.

Mode: m_c

ODE: $Q_k = 0$

Guards: C_{11}, C_{12}

Transitions: τ_1, τ_2, τ_3

Figure 4.13: Mode Structure of a closed Servo valve

Finally, we frame a transition function of a PHA (see Figure 4.14) to consist of: (1) a source mode that specifies the origin of a particular mode transition, (2) a guard that has to be satisfied, and (3) an associated thread which gives the probability distribution of transiting from the source mode(s) to particular target mode(s).

Transition: τ_r

Source: $m_{(k)}$ ⇔ Source mode of transition τ_r

Guard: C_{ij} ⇔ guard that has to be satisfied whenever transition τ_r is taken

Thread: $\left[p_j \quad m_l \right]$) ⇔ [probability target mode]

Figure 4.14: Transition Structure of τ_r

For example, in Figure 4.15, the transition function τ_3 describes the transition from a close valve m_{c} to a partially open valve m_{po} whenever guard C_{12} is satisfied. Provided that guard C_{12} is satisfied, we can transition to the target mode m_{po} with probability p_3. We model the transition function τ_3 as follows:

Transition: τ_3

Source: m_c

Guard: C_{12}

Thread: $\begin{bmatrix} p_3 & m_{po} \end{bmatrix}$

Figure 4.15: Transition Structure of τ_3 for the Servo valve

Figure 4.16 shows the Servo valve completely modeled as a PHA. In addition, the complete PHA structure of a Servo valve is available in Appendix B.

In general, any hybrid system can be modeled as a PHA. Once the modeling task is accomplished, our system then uses this information along with the measurement data, to learn the parameters of a PHA. We address the learning of the Hybrid Automata in Chapter 5.

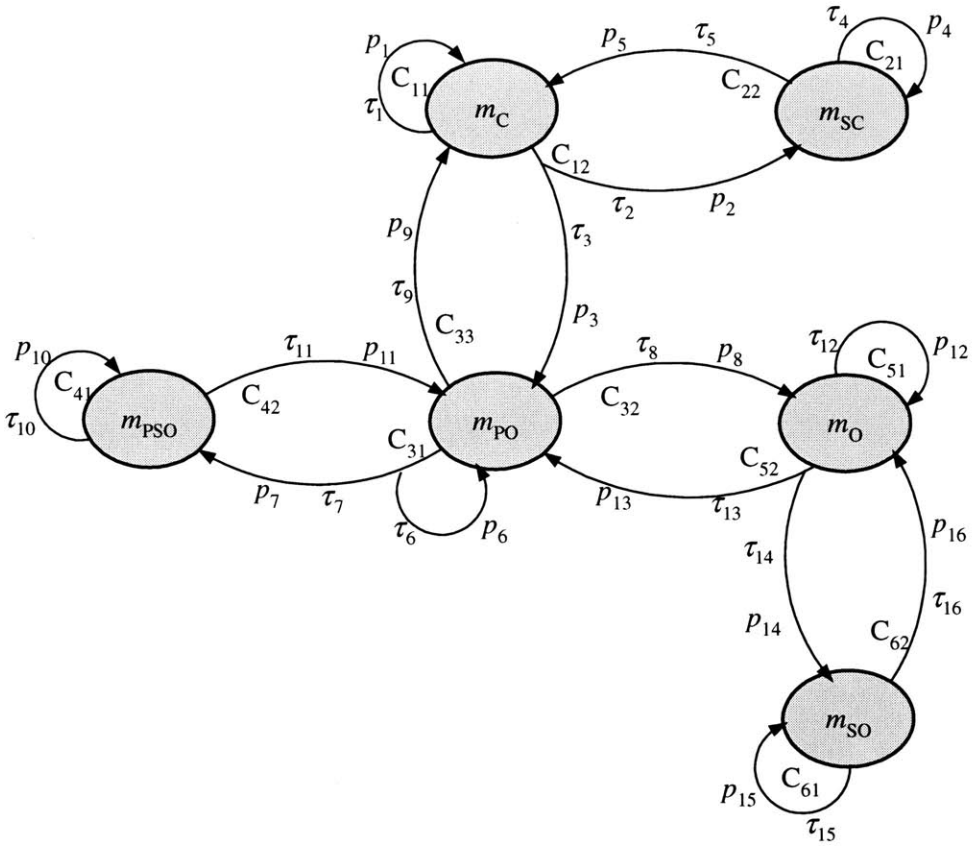


Figure 4.16: PHA of the servo valve

4.4 Summary

In this chapter, we demonstrated how to model a physical system as a Hybrid Automata. A probabilistic hybrid automaton is a Hidden Markov models (*HMM*) represented as a set of modes that exhibit a continuous dynamic behavior, expressed by difference, differential, or algebraic equations, and a discrete behavior, expressed as mode transitions.

First, we discussed Deterministic Hybrid Automata. In this type of automaton, we modeled the movement of a system between modes through a set of deterministic transitions. Second, we introduced Probabilistic Hybrid Automata. We modeled the behavior of a system as a probabilistic hybrid automaton. In addition, we frame the dynamics of the valve as it moves from one mode to another through a set of probabilistic transitions.

Chapter 5 – Learning of Hybrid Automata

5.1 Overview

Large-scale model learning has been considered one of the grand challenges of machine learning. One area where significant progress has been achieved is in the area of Bayes net learning. Learning of Probabilistic Hybrid Automata (PHA) is a new open challenge for machine learning with broad application. In this chapter, we address the challenges of learning Hybrid Automata by introducing a variant of the Expectation Maximization algorithm. This modified algorithm enables learning of a complex physical system as it moves between its distinctive behavioral modes.

Chapter 5 is organized as follows. First, we introduce Hybrid Learning as a Hybrid Expectation Maximization (Hybrid EM) algorithm that folds Hybrid Mode/State Estimation and Hybrid Parameter Estimation together. Hybrid Learning uses high fidelity models to describe the discrete stochastic behavior and the continuous dynamics of hybrid systems. Second, we briefly show how Hybrid Mode/State Estimation tracks and diagnoses a PHA by creating a Hybrid Observer that uses the results of the continuous state to estimate the system's modes. Third, we frame Hybrid Parameter Estimation as a method that unifies standard nonlinear estimation techniques for estimating the equation parameters of the system's modes with classical probabilistic estimation techniques for estimating the transition probabilities of a PHA.

5.2 Hybrid Learning

To detect the onset of failures such as a Servo valve being stuck-closed; it is essential that a learning system be able to accurately extract the parameters of a model from noisy measurement data. The problem of learning the parameters of a hybrid automaton is modeled as the Hybrid Parameter Estimation problem. More precisely,

Definition: Hybrid Parameter Estimation problem

Given a probabilistic hybrid automata PHA for a system, a sequence of observations $\left(y_{(0)},...,y_{(k)}\right)$, *a history of control inputs* $\left(u_{(0)},...,u_{(k-1)}\right)$, *a sequence of state variable estimates* $\left(\hat{x}_{(0)},...,\hat{x}_{(k)}\right)$, *and a sequence of the most likely modes* $\left(m_{(0)},...,m_{(k)}\right)$, *determine the parameters of the PHA.*

The parameters of the PHA consist of the equation parameters of a system mode, for example (μ,α) in equation (4.4), with the transition probabilities over the modes. To determine the parameters of the PHA, we introduce a Hybrid Learning system.

Hybrid Learning is an interactive process, which unifies a Hybrid Mode/State Estimation technique and a Hybrid Parameter Estimation technique. Figure 5.1 is used to illustrate this unification. With each execution of the Hybrid Mode/State Estimation technique, the learning system updates the set of Mode/State estimates and stores the best set of Mode/State estimates (labeled_data). In addition, with each execution of the Hybrid Parameter Estimation technique, the learning system re-estimates the parameter values in order to keep the best set of parameter estimates (Data). Each estimation

process is repeated until the best sets of hybrid estimates, i.e. the best of Mode/State estimates and Parameter estimates are achieved.

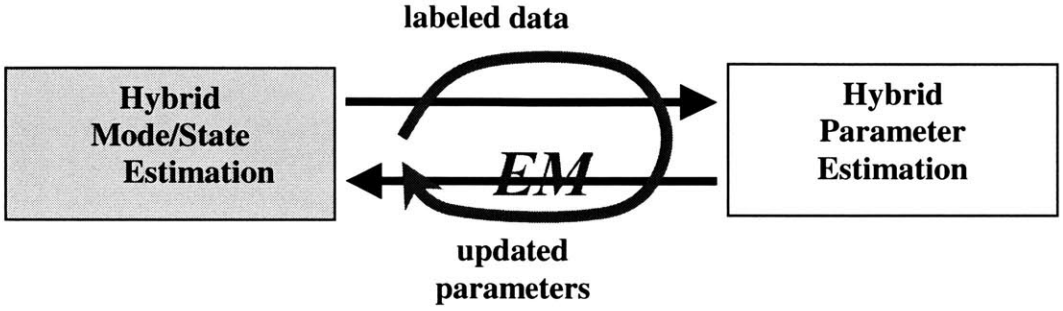


Figure 5.1: Block diagram for the Hybrid Expectation Maximization Algorithm

Our learning system is a variant on EM algorithm. This algorithm we called Hybrid Expectation Maximization (Hybrid_EM). The next section provides a quick overview of the Hybrid EM algorithm.

5.2.1 Hybrid EM algorithm

The Hybrid EM algorithm is a procedure that can be used to solve a large variety of estimation problems in many disciplines. This algorithm modifies the EM algorithm given in Chapter 3. The basic structure of a typical Hybrid EM algorithm is as follows:

- Initialize the modes of the PHA with random parameter values
- Iterate through the data set until the parameter values converge
  - Hybrid E step:

- Detect mode changes
- Assign each data point to the most likely mode it belongs to (labeled data).
- Return the labeled data ⇔ Return mode/state estimates

- Hybrid M step:
  - Update the equation parameter values of each mode using only the data points associated with that particular mode.
  - Update the transition probabilities of the PHA
  - Return Data ⇔ Return parameters estimates

Figure 5.2 provides the pseudo code of the Hybrid_EM. The Hybrid_EM algorithm is introduced as a procedure that accepts a data set "Data" and a PHA. First, the Hybrid_EM algorithm invokes the Hybrid_E step, which labels each data point according to the mostly likely mode it belongs to (labeled_data). This labeled_data is then passed to the Hybrid_M step, which uses the labeling to estimate the parameters of the PHA. The newly updated parameter values are then stored in a text file (Data) and return to the Hybrid_E step. Both the Hybrid_E and the Hybrid_M steps are repeated until the best set of parameters estimates are obtained. When the Hybrid_EM algorithm determines the best set of Hybrid Estimates[1], we conclude that convergence has occurred in our system.

[1] By the best set of Hybrid estimates we mean the best set of most likely modes for the Hybrid_E step and the best set of parameter values for the Hybrid_M step.

```
Hybrid_EM(Data, PHA)
    Begin loop
        labeled_data = Hybrid _E(Data, PHA);
        Update_PHA = Hybrid_M(labeled_data, PHA);
        If converged?(PHA, Update_PHA)
            Return Update_PHA
        else
            PHA = Update_PHA
        End-If
    End loop
End-Hybrid_EM
```

Figure 5.2: Hybrid EM algorithm

The Hybrid_E step is provided by Hybrid Mode/State Estimation technique and the Hybrid_M step is provided by Hybrid Parameter Estimation technique. In the next two sections, 5.3 and 5.4, we specify how the E step uses information from the learning system to guide Hybrid Mode/State Estimation, and how the M step uses results (measurement data) from the learning system to guide Hybrid Parameter Estimation.

5.3 Hybrid Mode/State Estimation

We adopt the technique called *Hybrid Mode/State Estimation,* formulated by [Hofbaur and Williams, 2002] to track and diagnose a PHA. The detailed of such a technique is outside the scope of this thesis; however, a brief summary of this technique is provided here. Hofbaur and Williams first introduced the PHA formalism. They then introduced Hybrid Mode/State Estimation as a technique for tracking and diagnosing a

PHA. This combines two techniques, which are the *Hybrid Mode Estimation* technique and the *Hybrid State Estimation* technique.

The Hybrid Mode estimation technique obtains measurements at each time step and estimates the mode of the system at each step. This technique labels the measurement data with the most likely modes of the system. We use this labeling to separate the data according to the most likely mode of the system. The, Hybrid State Estimation technique maintains a set of likely hybrid state estimates $\hat{\mathbf{X}}$. The **hybrid state** $\mathbf{X}_{(k)}$ of a probabilistic hybrid automaton at time-step t_k is specified by the tuple $\langle m_{(k)}, x_{c,(k)} \rangle$, where $m_{(k)} \in \mathcal{M}$ specifies the mode of the automaton and $x_{c,(k)}$ specifies the values of the continuous state-variables. These state variables and automaton modes can be estimated using a Hybrid observer.

Hybrid Observer:

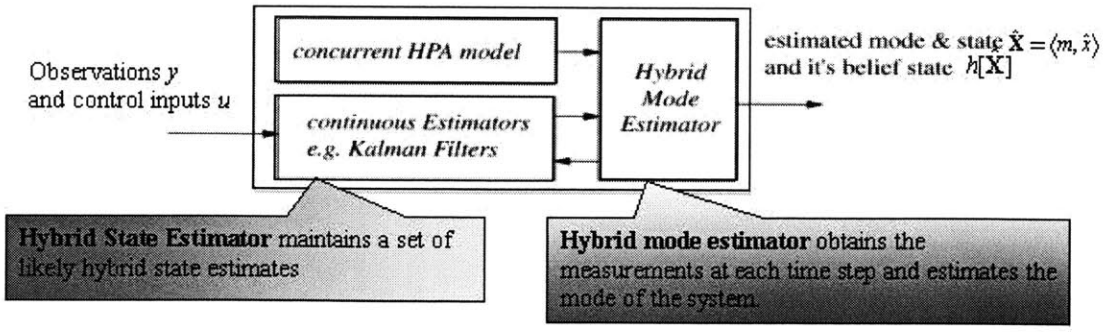


Figure 5.3: Hybrid Observer contains a Hybrid State Estimator and a Hybrid Mode Estimator

The hybrid observer, see Figure 5.3, is composed of two components. Component one, the Hybrid State Estimator, generalizes the Kalman filter, and is responsible for estimating continuous state variables. Component two, the Hybrid Mode Estimator, generalizes the Markov observer, and is responsible for maintaining mode estimates of a hybrid system. The details pertaining to how the Kalman filter and the Mode Estimator components interact are beyond the scope of this thesis. The reader who wishes to learn more about these two components should consult [Williams and Hofbaur, 2002].

In Figure 5.4, we produce the basic pseudo code for the Hybrid_E step.

Hybrid_E(Data,PHA)
    Detect mode changes
    Labels each data point in data set with the most likely mode of Model
    Return labeled_data
End-Hybrid_E

Figure 5.4: Function for the Hybrid E step

In the next section, we show how the Hybrid EM algorithm uses the above labeled_data to determine the parameters of the PHA (Chapter 4).

5.4 Hybrid Parameter Estimation

Hybrid Parameter Estimation is comprised of two estimation techniques: (1) a technique that estimates the equation parameters of each mode of a PHA, and (2) a

technique that estimates all the transition probabilities of a PHA. Recall that the Hybrid_M step uses the Hybrid Parameter Estimation technique. We frame the Hybrid_M step as consisting of two tasks: (1) the Update_Equation_Parameters task, and (2) the Update_Transition_Probabilities task (see Figure 5.5).

Hybrid-M(labeled-data,PHA)

```
    ; Given a set of labeled data, estimated the equation parameters and
    ; transition probabilities for PHA.

    Update_PHA = Update_Equation_Parameters(labeled_data, PHA);
    Update_PHA = Update_Transition_Probabilities(labeled_data, PHA);
    Return Update_PHA
End-Hybrid-M
```

Figure 5.5: Function for the Hybrid M step

In the Update_Equation_Parameters task, the labeling of the measurement data is used to separate the data according to the modes of the PHA. Having done so allows us to estimate the equation parameters for each mode by using standard nonlinear estimation techniques. For example, the labeled_data D_j is assumed to belong to mode m_j, and we use the weighted least squares fitting method to estimate the parameters p_j:

$$p_j^* = \arg\min_{p_j} \sum_{(y_j, x_j) \in D_j} [y_j - \hat{y}(x_j, p_j)]^2 \qquad (5.1)$$

The Update_Equation_Parameters task is then invoked with the variables, labeled_data and PHA. First, we determine the number of modes $\mathcal{M}$ in a PHA. Second, we create data structures, called buckets, for each mode in the PHA. We store data points

belonging to a particular mode in the corresponding bucket. For instance, for mode j, we create bucket j. Then we store all the data points, which belong to mode j into bucket j. We perform this task for all the modes of the PHA. Once all data points are stored in their appropriate buckets, the content of each bucket are placed in a structure called m_data. To reference a mode j, we would reference m_data(j) of the data structure. Similarly, the equations for each mode are stored in a data structure called m_eqn. Both m_data and m_eqn are then used by the function Estimate_Parameters to estimate the parameters of each mode in the PHA (see Figure 5.6).

```
Update_Equation_Parameters(labeled-data, PHA)
    ; Given a set of labeled data, estimates the parameters for the equations
    ; of every mode in PHA.

        For each m in modes(PHA)
                bucket(m) ={ }
        End-For
        For each <data-point, mode> in labeled-data
                add data-point to bucket(mode)
        End-For
        For each m in modes(PHA)
                m-eqn(ID(m)) = equations(m)
                m-data(ID(m)) = bucket(m)
        End-For
        Set_of_Parameters = Estimate_Parameters(m-eqn, m-data)
    return PHA
End-Update_Equation_Parameters
```

Figure 5.6: Update_Equation_Parameters function

Furthermore, labeling the data according to the modes provides us with an estimate for the time points when mode changes occur. Observing the system over a sufficiently large period of time will provide the information necessary to estimate the transition probabilities for a PHA. This estimation technique is discussed in the Update_Transition_Probabilities task. For instance, let us again consider the scenario when a Servo valve is fully close, and we issue a command u to open the valve. The transition function τ_1 specifies a self-transition from mode m_c with probability p_1 when guard C_{11} is satisfied. Similarly, when guard C_{12} is satisfied, τ_2 specifies a transition from mode m_c to mode m_{sc} with probability p_2 or the transition τ_3 from mode m_c to mode m_{po} with probability p_3. There are three approaches we consider: approach one estimates the transition probabilities when the transitions of a PHA do not have guards. Approach two estimates the transition probabilities for unique path transitions from source modes to target modes when satisfied guards are taken into account. Approach three involves estimating the transition probabilities for multiple path transitions from source modes to target modes when the guards are satisfied.

In approach one, we estimate the probability $\tilde{p}_3$ from mode m_c to mode m_{po} when all the guards of the PHA are ignored, by the following method: First, we calculate the number of times a transition occurs from mode m_c to m_{po}, $\tilde{n}_{cp}$. Second, we calculate the number of times a transition occurs from mode m_c to other targets modes of the HPA, $\tilde{n}_{ccps}$. Finally, $\tilde{p}_3$ is the quotient of $\tilde{n}_{cp}$ by $\tilde{n}_{ccps}$:

$$\tilde{p}_3 = \frac{\text{number of times } m_c \to m_{po}}{\text{number of times } m_c \to m_c, m_{po}, m_{sc}} = \frac{\tilde{n}_{cp}}{\tilde{n}_{ccps}} = \frac{\tilde{n}_{cp}}{\tilde{n}_{cc} + \tilde{n}_{cp} + \tilde{n}_{cs}} \qquad (5.2)$$

The following pseudo code, see Figure 5.7, is used to calculate the transition probabilities of modes of the PHA using approach one.

Update-transition-probabilities (labeled-data, PHA)
; Given a set of labeled data
; Estimate the transition probabilities for the model

m = number of modes(PHA)
MTM = matrix(m, m)
MOM = matrix(m, 1)
For each consecutive pair (<data-point1, mode m_c>, <data-point2, mode m_{po}>) of labeled-data
i = mode-index(mode m_c)
j = mode-index(mode m_{po})
MTM(i, j) = MTM(i, j) + 1
MOM(i) = MOM(i)+ 1
End-For
Update_PHA(TP) = Estimate-Transition-Probabilities(MTM, MOM,PHA)
Return Update_PHA
End-Update-transition-probabilities

Estimate-transition-probabilities(MTM, MOM,PHA)
Bin = matrix(size-of-matrix(MTM))
For i = 1 to number-of-rows(MTM)
For j = 1 to number-of-columns(MTM)
If MOM(i) ≠ 0
Bin(i, j) = MTM(i, j) / MOM(i)
End-If
End-For
End-For
Return Bin
End-Estimate-transition-probabilities

Figure 5.7: Function for estimating the Transition Probabilities without guards

---

In approach two, we estimate the transition probabilities of a PHA when the satisfied guards are taken into account and there is a unique path from source mode to target mode(s). For example, to estimate the probability p_3 from mode m_c to mode m_{po} when guard C_{12} is satisfied, we calculate n_{cp}, the number of times a transition occurs from mode m_c to m_{po}. Then we calculate the number of times a transition occurs from mode m_c to other target modes of a PHA n_{cps}. p_3 is the quotient of n_{cp} by n_{cps}.

$$p_3 = \frac{\text{number of times } m_c \rightarrow m_{po}}{\text{number of times } m_c \rightarrow m_{po}, m_{sc}} = \frac{n_{cp}}{n_{cps}} = \frac{n_{cp}}{n_{cp} + n_{cs}} \tag{5.3}$$

By estimating these probabilities $\tilde{p}_3$ and p_3 according to the first two approaches, approaches one and two, it is obvious that $\tilde{n}_{cp} \neq n_{cp}$ nor $\tilde{n}_{ccps} \neq n_{cps}$. Furthermore, these two probabilities are not equivalent to each other, $\tilde{p}_3 \neq p_3$. The Hybrid_EM algorithm currently uses the approach two to estimate the transition probabilities of the modes in the PHA. The following algorithm is used to calculate the transition probabilities of modes using approach two (See Figure 5.8):

```
Update_Transition_Probabilities (labeled_data,PHA)
    ;   Given a set of labeled data
    ;   Estimate the transition probabilities for the model

    m = number of modes(PHA)
    ;     gs = guards satisfied whenever we are in mode m
    MGM = matrix(m, gs)
    ;     ts = transition taken when a particular mode is satisfied
    GSMEM = matrix (gs, ts)
    MGSM = matrix(size_of _GSMEM)
    MGOM = matrix(length_of_GSMEM, 1)
    Update_PHA(TP) = Estimate_Transition_Probabilities(MGSM, MGOM, PHA)
    Return Model
End-Update-transition-probabilities

Estimate-transition-probabilities(MGSM, MGOM, PHA)
        Bin = matrix(size-of-matrix(MGSM))
                For i = 1 to number-of-rows(MGSM)
                        For j = 1 to number-of-columns(MGSM)
                                If MGOM(i) ≠ 0
                                Bin(i, j) = MGSM(i, j) / MGOM(i)
                End-For
        End-For
        Return Bin
End-Estimate-transition-probabilities
```

Figure 5.8: Function for estimating the Transition Probabilities with guards

In approach three, estimating transition probabilities of a PHA becomes more complex when the guards satisfied are taken into account, and there are multiple-path transitions. That is there are multiple ways to transition from a source mode to target mode(s). For example, in Figure 5.9, we assume that there are two ways to move from mode m_c to mode m_{po}. A mode transition is possible via transition τ_3 whenever guard C_{12} is satisfied or via transition τ_5 whenever guard C_{13} is satisfied. Since different

guards are satisfied when we transition from mode m_c to mode m_{po}, we can deduce that the transition probability estimated will be different from those calculated in the first two approaches. To date, we have not research this estimation approach in detail; however, we feel that such an approach may be an interesting topic for further study.

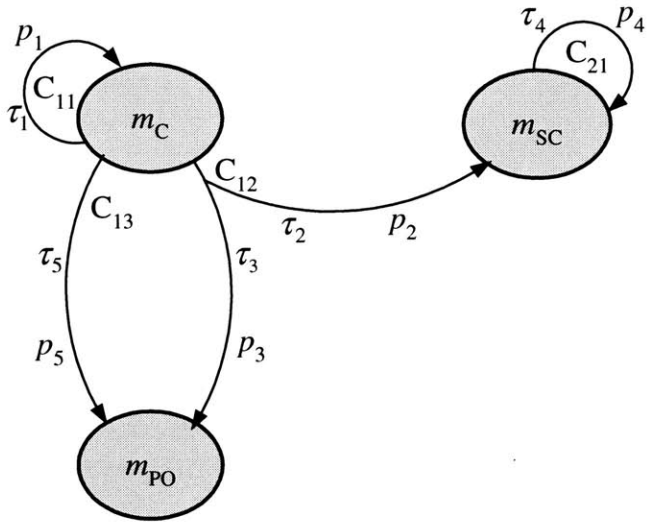


Figure 5.9: PHA with multiple paths from mode m_c to mode m_{po}

Our Hybrid Expectation Maximization algorithm then utilizes a PHA along with labeled_data to estimate the parameters of the PHA. This estimation problem we framed as the Hybrid Parameter Estimation problem.

5.5 Summary

In summary, we introduced a variant of the Expectation-Maximization algorithm for parameter estimation. EM is an iterative relaxation algorithm that repeatedly updates the model parameters. The Expectation step of the EM algorithm uses the current model to label each data point in the measured data set with one or more most likely modes that match that data. The Maximization step then uses the labeled data to estimate the parameters of the model for each mode. This process is repeated until convergence is achieved.

Recall that the Expectation step, which labels the measurement data with a set of modes, is exactly the task Hofbaur and Williams addressed under hybrid monitoring and diagnosis, through the development of HPA mode estimation. Hence hybrid learning may be view as a generalization of mode estimation that includes the additional maximization step for PHA.

Chapter 6 – Experiments

6.1 Overview

In this chapter, we assess the quality of the solution found by our Hybrid Learning system. To accomplish this, we test our learning system on two example applications: a (1) Linear Time-invariant system, and the (2) BIO-Plex Complex. We conclude the chapter by discussing limitation of the learning system and other issues that arise from our work.

6.2 Experiment 1: Linear Time-Invariant Systems

Consider a system that accepts an input u and produces an output y (see Figure 6.1). The behavior of the system can be approximated as having four nominal operational modes, modes 1, 2, 3 & 4. Each mode is assumed to be a linear time-invariant (*LTI*) system with an input u, a hidden state x, and an output y.

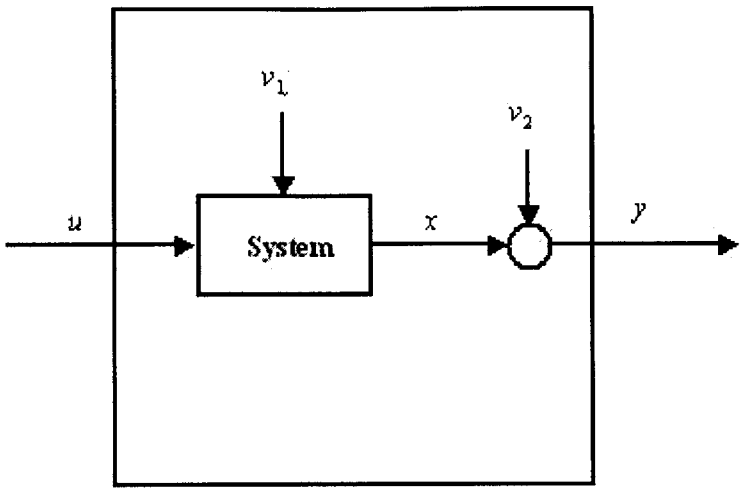


Figure 6.1: System's behavior of Linear Time-invariant System

The modes of the overall system can be represented by a set of linear time-invariant difference equations:

$$\begin{aligned} x_{k+1} &= A_j x_k + B_j u_k + v_{1,k} \\ y_k &= C_j x_k + D_j u_k + v_{2,k} \end{aligned} \tag{6.1}$$

where $j \in \{1,2,3,4\}$ and (A_j, B_j, C_j, D_j) represents the set of equation parameters for mode j of the system. x_{k+1} denotes the state variable at time step t_{k+1}. In addition, the input disturbance, v_1, can be modeled as a random uncorrelated sequence with zero-mean and Gaussian distribution specified by covariance Q. Similarly the measurement noise, v_2, is zero-mean Gaussian noise with covariance R. We assume that each mode of the

system is affected by the same v_1 and v_2. For simplicity, we assume that the system has no failure modes.

The transient behavior of the LTI system is assumed to be sufficiently fast in comparison to our sampling rate (see Figure 6.2). Our hybrid learning system monitors the LTI's behavior by taking sample measurements of the inputs u, the state variable estimates $\hat{x}$, the outputs y and the current system mode m at specific time points. In addition, with each data point we are able to predict the most likely mode of the LTI system.

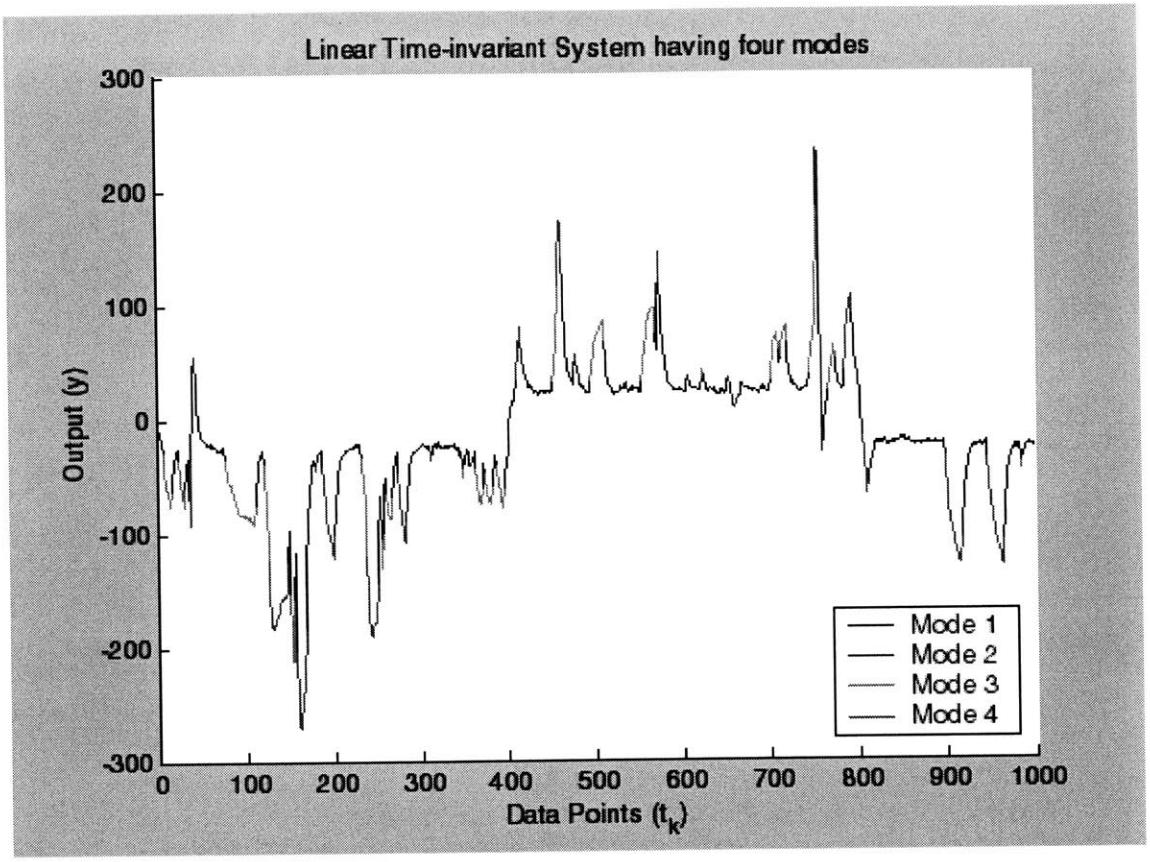


Figure 6.2: The behavior of the Linear Time-invariant system. With each time step, we track the behavior of the Linear Time-invariant system. We show its behavior for the first 1,000 data points out of the 100,000 data points in the labeled_data set.

The behavior of the LTI system is modeled by a probabilistic hybrid automaton, which is shown in Figure 6.3.

6.2.1 *PHA of the LTI system*

Recall that a PHA is used to model the modes of a system and its transitions between modes. To transition out of mode m_1, for example, either guard condition C_{11} or C_{12} must be satisfied. Whenever guard condition C_{12} is satisfied, the system transitions to either mode m_3 with probability p_3 or to mode m_4 with probability p_4.

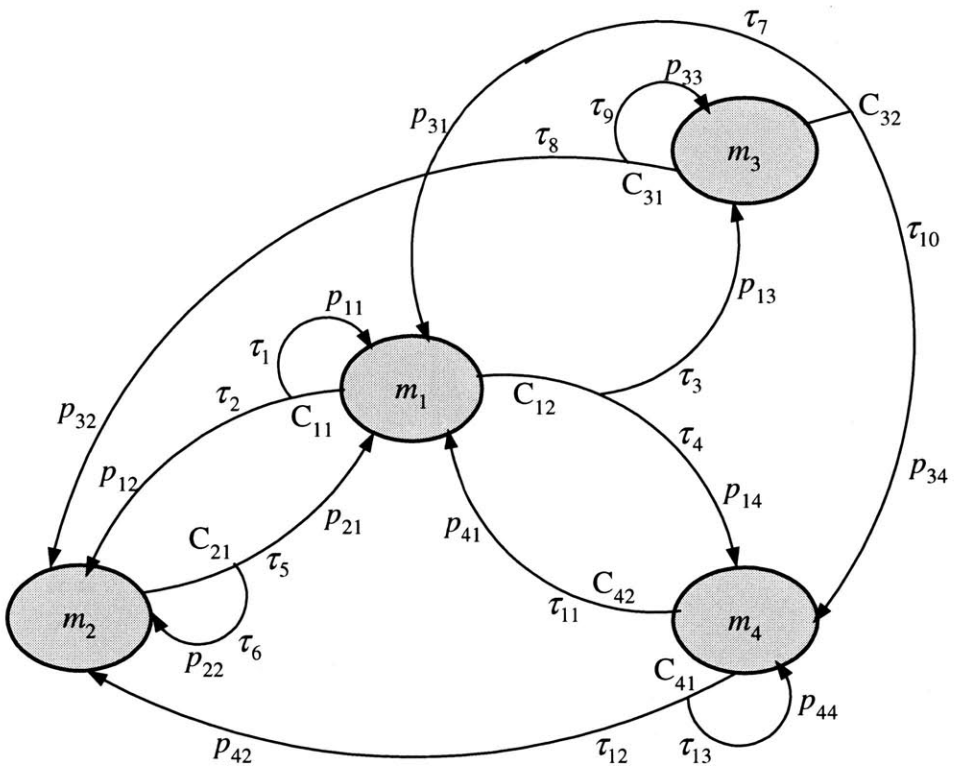


Figure 6.3: PHA of the linear Time-invariant system

6.2.2 *Hybrid Parameter Estimation of the LTI system*

Given the PHA of the LTI system and the labeled_data set, we can now estimate the mode parameters of the system. In section 5.2, we define the mode parameters of a system as the unification of the equation parameters of the system mode with the transition probabilities over the modes. According to the hybrid parameter estimation problem[1], we can estimate the parameter set, (A_j, B_j, C_j, D_j), and the transition probabilities of mode *j*, provided that we know the labeling of each data point in labeled_data.

6.2.3 *Simulation*

Results of the Hybrid-E step: We generated a labeled_data set of 100,000 data points for the above LTI system model. The labeled_data set is comprised of: a sequence of 100,000 observations $(y_{(0)},\ldots, y_{(99,999)})$, control inputs $(u_{(0)},\ldots,u_{(99,999)})$, state variable estimates $(\hat{x}_{(0)},\ldots,\hat{x}_{(99,999)})$, and most likely modes $(m_{(0)},\ldots,m_{(99,999)})$ of the LTI system.

The equation parameters for system modes 1, 2, 3, and 4 are given in equations (6.2), (6.3), (6.4), and (6.5), respectively.

$$A_1 = \begin{bmatrix} 0.8 & 0 \\ 0 & 0.7 \end{bmatrix} \quad B_1 = \begin{bmatrix} 1.5 \\ 2.5 \end{bmatrix} \quad C_1 = [1 \quad 1] \quad D_1 = 0 \tag{6.2}$$

$$A_2 = \begin{bmatrix} 0 & 1 \\ -0.8 & 1.6 \end{bmatrix} \quad B_2 = \begin{bmatrix} 0.5 \\ 1.5 \end{bmatrix} \quad C_2 = [1 \quad 1] \quad D_2 = 0 \tag{6.3}$$

[1] The hybrid parameter estimation problem is stated in section 5.2 of this thesis.

$$A_3 = \begin{bmatrix} 0.7 & 0 \\ 1 & 0.8 \end{bmatrix} \quad B_3 = \begin{bmatrix} 1.5 \\ 2.5 \end{bmatrix} \quad C_3 = [1 \quad 1] \quad D_3 = 0 \tag{6.4}$$

$$A_4 = \begin{bmatrix} 0.6 & 0 \\ 0 & 0.8 \end{bmatrix} \quad B_4 = \begin{bmatrix} 1.5 \\ 2.5 \end{bmatrix} \quad C_4 = [1 \quad 1] \quad D_4 = 0 \tag{6.5}$$

In addition, the actual transition probabilities of the system are shown in Figure 6.4:

Guards		
C_{11}	p_{11}	p_{12}
C_{12}	p_{13}	p_{14}
C_{21}	p_{21}	p_{22}
C_{31}	p_{32}	p_{33}
C_{32}	p_{31}	p_{34}
C_{41}	p_{41}	–
C_{42}	p_{41}	p_{44}

Guards		
C_{11}	0.990	0.010
C_{12}	0.640	0.360
C_{21}	0.100	0.990
C_{31}	0.010	0.990
C_{32}	0.650	0.350
C_{41}	1	–
C_{42}	0.010	0.989

Figure 6.4: The actual probabilities of the simulated LTI system. Whenever guard condition C_{12} is satisfied, for example, the system transition from mode m_1 to either mode m_3 with the probability $p_{13} = 0.64$ or mode m_4 with the probability $p_{14} = 0.36$.

The Hybrid EM algorithm is to estimate the aforementioned parameters of the LTI system, given the results of the Hybrid-E step. We initialized the algorithm with the PHA of the LTI system and the generated labeled_data set. The algorithm initially assumes that the PHA can be in any one of its four modes with equal probability.

6.2.4 Results

The Hybrid EM algorithm is able to estimate the parameters within 120 seconds on a Pentium III machine. The estimated equation parameters for modes 1, 2, 3, and 4 are given by equations (6.6), (6.7), (6.8), and (6.9), respectively.

$$A_1 = \begin{bmatrix} 0.7650 & 0.0200 \\ 0.0071 & 0.6796 \end{bmatrix} \quad B_1 = \begin{bmatrix} 1.5843 \\ 2.4758 \end{bmatrix} \quad C_1 = [1.0003\ 1.0004] \quad D_1 = -0.0058 \tag{6.6}$$

$$A_2 = \begin{bmatrix} 0 & 0.9269 \\ -0.7495 & 1.4930 \end{bmatrix} \quad B_2 = \begin{bmatrix} 0.3755 \\ 1.2845 \end{bmatrix} \quad C_2 = [0.9998\ 1.0003] \quad D_2 = -0.0006 \tag{6.7}$$

$$A_3 = \begin{bmatrix} 0.6893 & -0.0007 \\ 0.9201 & 0.7666 \end{bmatrix} \quad B_3 = \begin{bmatrix} 1.3554 \\ 2.9214 \end{bmatrix} \quad C_3 = [1.0033\ 0.9998] \quad D_3 = -0.0090 \tag{6.8}$$

$$A_4 = \begin{bmatrix} 0.5313 & 0.7596 \\ 0.0179 & 0.7038 \end{bmatrix} \quad B_4 = \begin{bmatrix} 3.7817 \\ 3.8444 \end{bmatrix} \quad C_4 = [0.9999\ 1.0004] \quad D_4 = -0.0044 \tag{6.9}$$

The Hybrid EM algorithm estimated the transition probabilities of the LTI system as shown in Figure 6.5:

Guards		
C_{11}	p_{11}	p_{12}
C_{12}	p_{13}	p_{14}
C_{21}	p_{21}	p_{22}
C_{31}	p_{32}	p_{33}
C_{32}	p_{31}	p_{34}
C_{41}	p_{41}	–
C_{42}	p_{41}	p_{44}

Guards		
C_{11}	0.9899	0.0101
C_{12}	0.6380	0.3620
C_{21}	0.1001	0.8999
C_{31}	0.010	0.990
C_{32}	0.6476	0.3524
C_{41}	1	–
C_{42}	0.0129	0.9871

Figure 6.5: The estimated probabilities of the simulated LTI system.

where the mean of the absolute value of the error is 2.235, the median error is 0.34559, and the standard deviation is 5.028. Finally, in Figure 6.6, we see that if the Hybrid algorithm runs 10 times longer, it produces 10 percent less errors. This is justified by the best-fit line with a slope of 1.0524 through these error estimates.

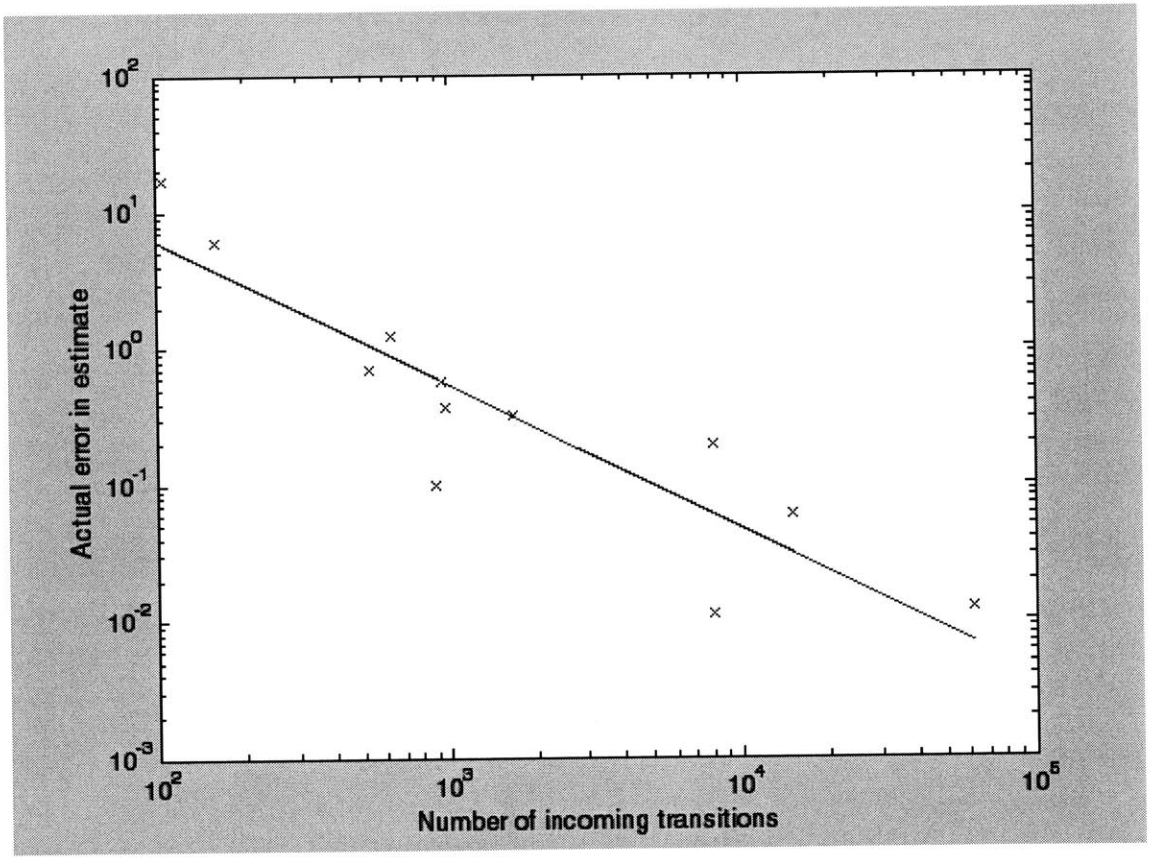


Figure 6.6: Error estimates of the transition probabilities. The number of incoming transitions and the actual error in calculating the transition probabilities. Running the HMLR system for 10 times longer produces 10 percent less errors.

These parameters are extremely good since these values are the results of one complete pass through the entire data. Recall that Hybrid-EM algorithm is an iterative procedure that iterates between a Hybrid-E step and a Hybrid-M step. The Hybrid-E step fixes the current parameters and computes posterior probabilities over the hidden states given the posterior distributions. On the other hand, the Hybrid-M step fixes current posterior probabilities and computes the parameters. These steps are done until the algorithm obtains the best set of parameter estimates that matches the model. To evaluate the performance of the Hybrid EM algorithm after multiple passes through the labeled_data is interesting. We should be considered this as future work in order to improve the efficiency of our algorithm.

6.3 Experiment 2: BIO-Plex Complex

Our next application is the NASA's Advance Life Support System, a five-chamber facility called BIO-Plex (see Figure 6.7). BIO-Plex is a simulated biosphere-type environment used to appraise technologies essential to life support and human habitation. It supplies all the necessary oxygen (O_2), water, and approximately eighty-five percent of food for a crew of two researchers on a continuous basis. Plants are grown in the plant growth chambers (PGCs), where they provide food for the researchers. Food selection includes peanut, potato, rice, soybean and wheat.

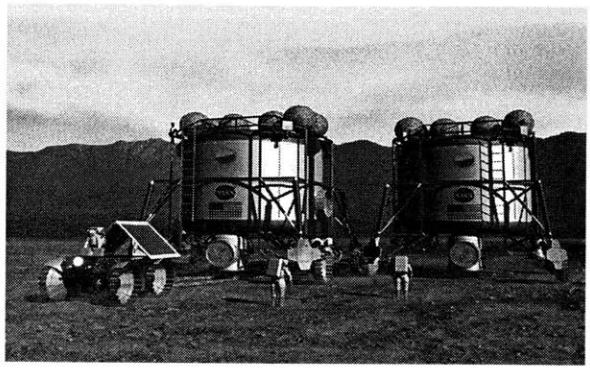


Figure 6.7: (BIO-Plex) Bioregenerative Planetary Life Support System Test Complex

In addition to the food provided by the plants in the PGC, the plants convert exhaled carbon dioxide (CO_2) into required O_2. The O_2 produced by the plants are captured and supplied to the crew. To effectively sustain a closed-loop system, it is essential to regulate the gas exchange between the plant growth chambers and the crew chambers. This regulation is performed by the chamber control subsystem. In this thesis, we confine our evaluation to four subsystems: the PGC, the lighting system, a CO_2 flow controller, and the chamber control (see Figure 6.8).

The chamber control subsystem maintains a simulated 20/4-hour day/night schedule. This schedule is maintained in order for the plants in the PGC to grow optimally and for effective gas exchange.

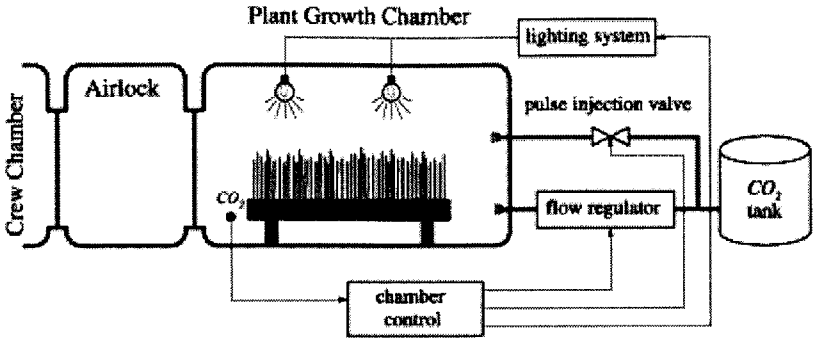


Figure 6.8: Selected schematic of BIO-Plex complex

The behavior of the system is approximated as having four modes. Three nominal operational modes, which are m_1, m_2 and m_3, and one light failure mode, m_4. Each mode of the system can be represented by the following nonlinear time-invariant difference equation:

$$\begin{aligned} fun1 &= -1.446\times10^{-2} * (72.0 - K_{4,j} * e^{-(x_k/400)}) \\ x_{k+1} &= x_k + K_{1,j} * (K_{2,j} * fun1(x_k) + K_{3,j} + u_k) \end{aligned} \tag{6.10}$$

where $j \in \{1,2,3,4\}$, $\left(K_{1,j},...,K_{4,j}\right)$ represents the equation parameter set for mode j of system, and x_k denotes the state variable at time step t_k.

6.3.1 Hybrid Modeling of the BIO-Plex

In Figure 6.9, we modeled the behavior of the system of the BIO-Plex as a PHA. The system can only be in m_1 during $0 \le x \le 240$ minutes of everyday, whereas it can be

in any "day" mode during $240 \leq x \leq 1440$ minutes in everyday. m_1 is also called the night mode. Due to extremely high CO_2 concentration in this mode, which is unsuitable for humans, the PGC's door remains shut (C_{11}) and the crew is denied access to enter the PGC (τ_{11}). These restrictions are enforced by the chamber control.

When the crew requests to enter the PGC, the CO_2 concentration must be lowered to 500 ppm (τ_{12}). To accomplish this, the flow regulator is turned off and the door is opened (C_{21}). Once these preconditions are achieved, we assume that the subsystem is in m_2.

Once a set point of 500 ppm is maintained, the door remains open and the crew is allowed access to the PGC (τ_{24}). Safety precaution requires the subsystem to inhibit CO_2 injection (C_{31}) while the crew of two is in the chamber. We assume that the subsystem is now in service mode, m_3. Once in m_3, the crew can perform a single task or a series of tasks, which include harvesting, re-planting, maintenance or other required services. Upon the completion of the aforementioned services, the crew quickly exits PGC (τ_{32}).

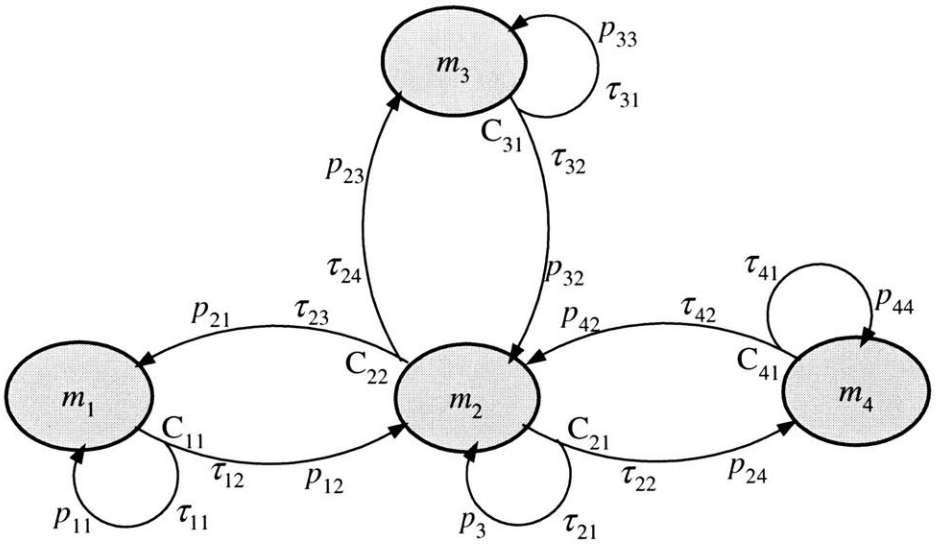


Figure 6.9: PHA for the subsystem of the BIO-Plex complex

Like other systems, the system can function improperly during its operations. There are many possible types of failure, but we model only one failure mode in this subsystem. The system can only enter this failure mode, m_4, whenever it is in m_2 at the previous time step. m_4 differs from m_2 only by a change in the lighting system (τ_{24}). In m_2, the lighting system works perfectly, whereas in m_4 the lighting system functions at an 80 percent level.

6.3.2 *Simulation*

Having modeled the subsystem as a PHA, our next task is to calculate the parameters of the PHA. We obtained a labeled_data set of 8,500 data points for the above subsystem from the Hybrid-E step. The actual equation parameters for modes 1, 2, 3, and 4 are given in equations (6.11), (6.12), (6.13), and (6.14) respectively.

$$K_{1,1} = 20.1625 \quad K_{2,1} = -1.4461e-2 \quad K_{3,1} = 0 \quad K_{4,1} = 78.89 \tag{6.11}$$

$$K_{1,2} = 11.8373 \quad K_{2,2} = -1.4461e-2 \quad K_{3,2} = 0 \quad K_{4,2} = 78.89 \tag{6.12}$$

$$K_{1,3} = 11.8373 \quad K_{2,3} = -1.4461e-2 \quad K_{3,3} = 0.3 \quad K_{4,3} = 78.89 \tag{6.13}$$

$$K_{1,4} = 11.8373 \quad K_{2,4} = -1.2094e-2 \quad K_{3,4} = 0 \quad K_{4,4} = 78.89 \tag{6.14}$$

The actual transition probabilities of the subsystem are shown in Figure 6.10.

Guards		
C_{11}	p_{11}	p_{12}
C_{21}	p_{22}	p_{24}
C_{22}	p_{21}	p_{23}
C_{31}	p_{33}	p_{32}
C_{41}	p_{44}	p_{42}

Guards		
C_{11}	0.990	0.010
C_{21}	0.990	0.010
C_{22}	0.55	0.450
C_{31}	0.980	0.020
C_{41}	0.980	0.020

Figure 6.10: The actual probabilities of the simulated control subsystem of the BIO-Plex complex.

We initialized Hybrid EM algorithm with the PHA of the system and the given labeled_data set.

6.3.3 Results

The Hybrid EM algorithm estimates the parameters within 190 seconds on a Pentium III machine. The estimated equation parameters for mode 1, 2, 3, and 4 are given by equations (6.15), (6.16), (6.17), and (6.18), respectively.

$$K_{1,1} = 20.2026 \quad K_{2,1} = -1.435e-2 \quad K_{3,1} = 0.0051 \quad K_{4,1} = 78.833 \tag{6.15}$$

$$K_{1,2} = 11.6912 \quad K_{2,2} = -1.5155e-2 \quad K_{3,2} = 0.0022 \quad K_{4,2} = 78.7097 \tag{6.16}$$

$$K_{1,3} = 11.8330 \quad K_{2,3} = -1.3136e-2 \quad K_{3,3} = 0.0096 \quad K_{4,3} = 78.8118 \tag{6.17}$$

$$K_{1,4} = 11.7539 \quad K_{2,4} = -1.2022e-2 \quad K_{3,4} = 0.0031 \quad K_{4,4} = 78.1219 \tag{6.18}$$

The estimated transition probabilities are shown in Figure 6.11.

Guards		
C_{11}	p_{11}	p_{12}
C_{21}	p_{22}	p_{24}
C_{22}	p_{21}	p_{23}
C_{31}	p_{33}	p_{32}
C_{41}	p_{44}	p_{42}

Guards		
C_{11}	0.9976	0.024
C_{21}	0.9980	0.0020
C_{22}	0.5000	0.5000
C_{31}	0.9900	0.0010
C_{41}	0.9900	0.0010

Figure 6.11: The estimated probabilities of the simulated control subsystem of the BIO-Plex complex.

The mean of the absolute value of the error is 0.2796, the median error is 0.1010, and the standard deviation is 0.3255. Finally, in Figure 6.11, we see that if the Hybrid algorithm

runs for 10 times longer, it produces 6 percent less errors. This is justified by the best-fit line with a slope of 0.6659 through these error estimates.

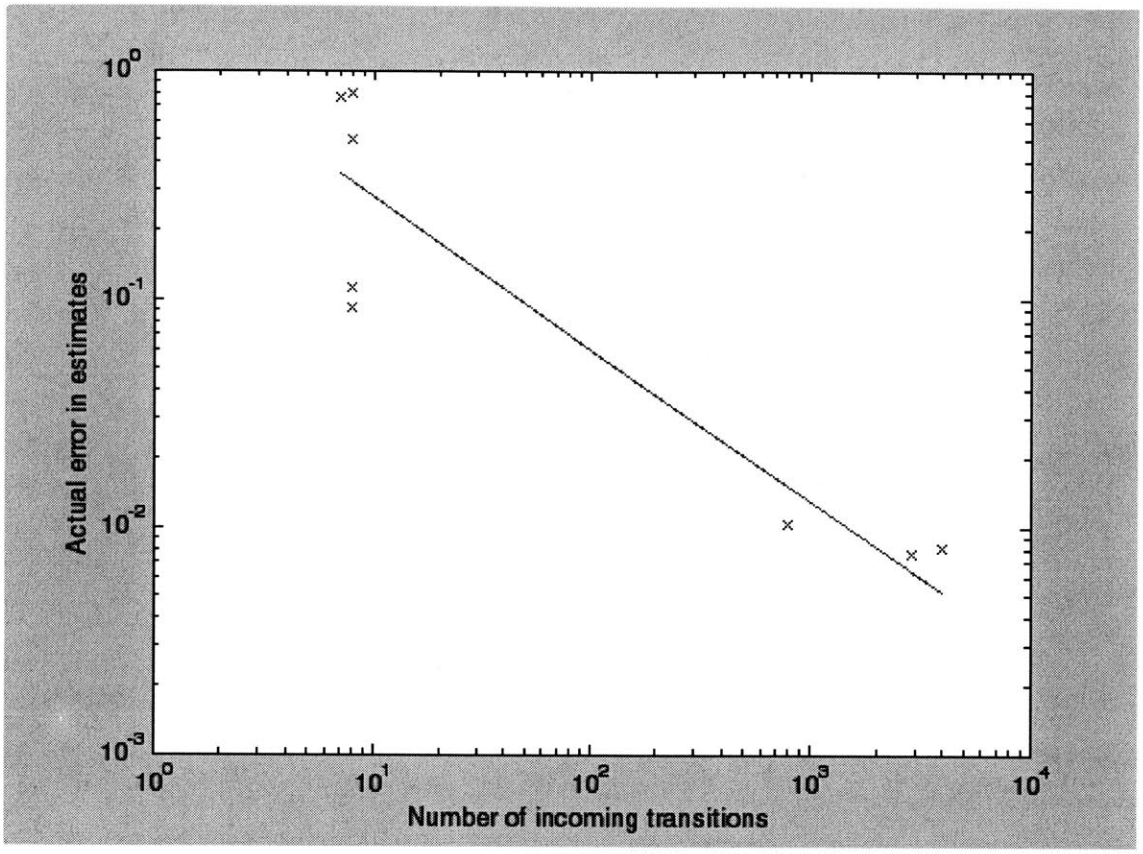


Figure 6.12. Error estimates of the transition probabilities. The number of incoming transitions and the actual error in calculating the transition probabilities. Running the HMLR system for 10 times longer produces 6 percent less errors.

These parameters are extremely good since these parameter estimates are the results of one complete pass through the entire data. Allowing the Hybrid EM algorithm to pass multiple times through the labeled_data will improve the parameter estimates

obtained over time. We currently are working on folding the Hybrid-E step and the Hybrid-M step together.

6.4 Limitations

The Hybrid EM algorithm proposed here possesses a range of limitations to the hybrid parameter estimation problem. These limitations are discussed in this section.

First and foremost, our capability uses a batch approach to learning. This approach requires a complete pass through the entire data set in order to determine a set of parameters. This approach makes no attempt to estimate the covariance of the parameters. In contrast, online approaches provide an incremental estimate of the parameters and the covariance at each time step. However online algorithms are less powerful than offline algorithms, since the parameter estimates are general less accurate than in the batch approach.

Second, our Hybrid EM algorithm is currently incapable of supporting large-scale modeling of complex physical systems. These physical systems generally exhibit sets of complex concurrent and sequential behaviors that are well beyond the modeling scope of our capability. This incapability is due to the modeling representation used by the Hybrid EM algorithm. Recall that the algorithm uses PHA. A PHA is incapable of capturing these complex concurrent and sequential behaviors. To make our algorithm capable of supporting these behaviors, we propose using Hierarchical Probabilistic Hybrid Automata (HPHA). We discuss HPHA in section 7.4.1 of this thesis.

6.5 Discussion

The main conclusion we can draw is that the Hybrid EM algorithm performed well in these examples, which consisted of four parameters. However, for large number of parameters performing parameter estimation is extremely difficult. Major difficulties during parameter estimation include getting stuck in local minima and the slow convergence rate of optimization algorithms when applied to complex non-linear problems. Many estimation techniques for example Levenberg-Marquart routine can by no means escape the curse of dimensionality. They frequently get stuck within local minima, make little progress or take an immoderate amount of time to converge. However, most techniques become more efficient when the variable and parameter spaces are decrease adequately. We therefore have to conduct more experiments to see how efficient our Hybrid EM algorithm when used on more complex non-linear systems.

Chapter 7 – Conclusions and Future Work

7.1 Overview

We begin by briefly mentioning other areas of inquiry related to parameter estimation. The interested reader should refer to the cited work for more details. This is followed by a summary of the contributions of the thesis. Finally, section 7.4 gives a range of research opportunities that arises from this work.

7.2 Related Work

Shumway and Stoffer [1991] address the problem of learning the parameters of state-space models with a single real-valued hidden state vector and switching output matrices. The probability of selecting a specific output matrix is a pre-determined time-varying function, independent of previous selections. Shumway and Stoffer derived a pseudo-expectation-maximization (pseudo-EM) algorithm in which the expectation step would require calculating a Gaussian mixture with M^T components and is approximated by a single Gaussian at each time step.

Kim [1994] extends the aforementioned work to the case where both the state dynamics and the output matrices switch, and where the switching succeed Markovian dynamics. Kim uses an approximation in which the exponential Gaussian mixture is reduced to M Gaussians at each time step. Other researchers have used Markov chain Monte Carlo techniques for state and parameter estimation in switching models [Carter

and Kohn, 1994; Athaide, 1995] and in other dynamic probabilistic networks [Dean and Kanazawa, 1989; Kanazawa et al., 1995].

Ghahramani and Hinton [1996a; 1996b; 1998] introduced a different approach to parameter estimation. They presented a learning algorithm for all of the parameters of the model, including the Markov switching parameters. Using a structures variational approximation [Saul and Jordan, 1996], Ghahramani and Hinton demonstrated that this algorithm maximizes a strict lower bound on the log likelihood of the data, rather than a heuristically motivated pseudo-likelihood. The resulting algorithm decouples into forward-backward recursions on a hidden Markov model, and Kalman smoothing recursions on each state-space model. The states of the HMM determine the soft assignment of each observation to a state-space model; and the predictions errors of the state-space models determine the observation probabilities for the HMM.

Another related proposal comes from Ghahramani and Roweis [1999]. Here they introduced a generalization of the EM algorithm for parameter Estimation in nonlinear dynamical systems. The Expectation step uses the Extended Kalman Smoothing approach to estimate the state, while the Maximization step re-estimates the parameters using these uncertain state estimates. The nonlinear Maximization step is generally difficult because it requires integrating out the uncertainty in the states. However, Ghahramani and Roweis claimed that if Gaussian radial basis function (RBF) approximators are used to model the nonlinearities, the integrals become tractable and the maximization step can be solved via systems of linear equations. Like Ghahramani and Roweis, we introduced a variant of the EM algorithm. Our E step also uses a Hybrid Observer to estimate the state variables and automaton modes [Hofbaur and Williams,

2000]. The Hybrid State Estimator of the E step, (see Chapter 2), generalizes the Kalman filer, and is responsible for estimating the continuous state variables. The Maximization step re-estimates the parameters using these uncertain state estimates. The M step simplifies to a linear regression problem.

7.3 Summary

In Chapter 4, we modeled the behavior of complex physical systems, which may be characterized by both discrete and continuous dynamics. For this purpose, we introduce a model called Probabilistic Hybrid Automaton (*PHA*). A PHA is a modeling formalism that merges Hidden Markov models (*HMM*) with continuous system models. Hybrid Automata allow us to represent both the discrete stochastic behavior and the continuous dynamics in an expressive way.

In Chapter 5, we addressed the challenges of learning Hybrid Automata by introducing a variant on the Expectation Maximization algorithm. This modified algorithm enables learning of a complex physical system as it moves between its distinctive behavioral modes. First, we introduced Hybrid Learning as the method, which folds Hybrid Mode/State Estimation and Hybrid Parameter Estimation together. Second, we briefly demonstrated how Hybrid Mode/State Estimation tracks and diagnoses a PHA by creating a Hybrid Observer that uses the results of the continuous state to estimate the system's modes. Third, we framed the Hybrid Parameter Estimation as a method that unifies standard nonlinear estimation techniques for estimating the equation parameters of the system's modes with classical probabilistic estimation techniques for estimating the transition probabilities of a PHA.

In Chapter 6, we assessed the quality of the solution found by our Hybrid EM algorithm. To accomplish this, we tested our learning system on two example applications: a (1) Linear Time-invariant system, and (2) BIO-Plex Complex. We produced results as promised for these systems. We concluded the chapter by discussing limitation of the Hybrid EM algorithm and other issues that arises from our work.

7.4 Future Work

Our overall goal is to provide a high quality Hybrid EM algorithm that supports large-scale modeling and learning of complex physical systems. Recall that physical systems, such as Rovers, exhibit a rich set of combined discrete/continuous behaviors. We introduced hierarchical probabilistic hybrid automata (*HPHA)* to capture the behavior of such systems (see section 7.4.1). Our next task is to improve the efficiency of our Hybrid EM algorithm. To improve efficiency, we introduce model-based decomposition. We address this issue in section 7.4.2. Finally, we recommend testing our improved Hybrid EM algorithm on Cooperative vehicles since they exhibit a rich set of continuous and discrete behavior. Such vehicles may serve as the major test-bed for evaluating and validating our proposed model learning and refinement methods in the future. We discuss learning of cooperative vehicles in section 7.4.4.

7.4.1 Extending Hybrid EM algorithm to handle HPHA

Hierarchical probabilistic hybrid automata (*HPHA*) support large-scale modeling of complex physical systems. HPHA are generalization of probabilistic hybrid automata

(PHA) that are assorted in a hierarchy. That is the mode of a PHA may itself be a PHA, which is activated by its parent. By introducing hierarchy, we empower the representation of complex concurrent and sequential behaviors that are well beyond the modeling range of single PHAs. In addition, each transition in the HPHA may have multiple targets, permitting a PHA in the HPHA to be in several modes simultaneously. This approach enables a compact representation of recursive behavior [Hofbaur and Williams, 2000; Hofbaur and Williams, 2002b].

7.4.2 Model-based Decomposition

Recall that the problem of model learning is for large parameter systems extremely difficult. Major difficulties during learning include being stuck in local minima and the slow convergence rate of optimization algorithms when applied to complex non-linear problems. Techniques that include reducing the variable and parameter spaces have proven to be efficient in limiting the aforementioned difficulties.

We propose to collapse down the search space visited by learning algorithms by decomposing the problem into smaller sub-problems. Our decomposition capability will extend previous work on decompositional model-based learning (*DML*) [Williams and Millar, 1998]. DML decomposes a model structure into a set of overlapping sub-models. Associated with these sub-models are subsets of the observations that are sufficient to perform learning on the sub-model. This technique is developed only for models consisting of systems of non-linear equations. For future work, we propose to extend DML to operate on HPHA models and to formulate the learning algorithms to operate on the sub-models identified by DML. This will involve, for instance, combining different

sets of decomposed estimators, based on which sets of modes likely match each data point.

7.4.3 Learning of the behavior of Single Robot

The demand for a single robot to carry out complex tasks with little or no supervision has motivated a great deal of research in the area of autonomy. These robots must be able to function robustly in unpredictable and dangerous environments. To support this, we address the problem of learning a robot's environment. This problem is closely related to the mapping and localization problem [Leonard and Feder, 1999].

Mapping is the problem of creating models of a robot's environment from sensor data. An interesting area of study is the problem of constructing detailed maps online, while the robot is moving. The online feature is relevant for a number of problems such as the Mars exploration problem, in which mapping is constantly interleaved with decision making as to where to move next [Burgard et. al, 2000; Simmons et. al., 2000]. To map an environment, a robot has to cope with range measurement noise and noise in odometery. This causes a problem of determining the location of the robot in relation to its own map, which is a localization problem.

Research conducted in the past on the mapping and localization problem has proposed many different techniques, such as *SLAM* algorithms [Castellanos et al. 1999; Leonard and Feder, 1999]. These techniques have advantages and shortcomings. An advantage of these techniques is they provide sound online solution to the mapping problem when applicable. One major shortcoming is that the correct associations

between measurements and features in the map must be known. In order for this to occur, features in the environment must have unique signatures.

In this research, we propose to convert our Hybrid model learning and refinement capability to an online algorithm that construct maps from sensor measurements, without the need for exact data association.

7.4.4 Learning of the behavior of Cooperative Vehicles

Robotics systems are now being created that must perform together to robustly achieve elaborate missions within unpredictable and sometimes dangerous environments. To achieve this robustness, we must go beyond current programming practice. Research in this field must address three important open issues. How do we program these teams of robots or vehicles to perform elaborate missions, while offering them a range of options for handling the unknown? How will these robots best handle uncertainty with communication as well as the uncertainty of the environment? How do we give these robots enough autonomy to perform these agile maneuvers?

The challenge is three folds. First, how do we develop model learning languages that can handle real-world problems in real-time? Second, how do we perform model learning and refinement in cooperative vehicles, while being robust to communication delays and communication lossage? Finally, how do we address the problem in which a team of robots builds a map online, while simultaneously accommodating errors in their odometry?

To date, our research has concentrated on a centralized approach to generate an efficient probabilistic algorithm for learning of complex physical systems. An interesting

application is to generalize our approach to support the learning of vehicles that are able to perform agile stunt maneuvers. To accomplish this, we generalize our approach to optimum learning of hybrid systems, in which robust hierarchical probabilistic hybrid automata are used to describe a range of possible agile maneuvers that each vehicle can perform.

The problem of mapping lends itself nicely to cooperative vehicle solutions, where such vehicles collaborate and jointly explore unpredictable and dangerous environments. The model learning and refinement capability will be tested in simulation on Mars exploration scenarios, where mapping is constantly interleaved with decision making as to where to move next [Burgard et. al, 2000; Simmons et. al., 2000]. In addition, our capability will be tested on hardware using a collection of four ATRV rovers. Hybrid Learning will involve determining the rover's mode parameters with respect to its surrounding world.

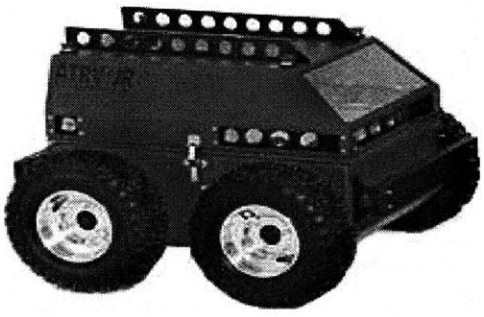


Figure 7.1: An example of an ATRV-Jr owned by our research group.

7.5 Conclusion

Although there remain issues to be considered by future work, the research described in this thesis takes several steps toward the goal of developing hybrid model-based estimation techniques to address the problem of learning the behavior of complex physical systems. Probabilistic hybrid automata address the challenge of describing the behavior of these systems. The Hybrid EM algorithm makes significant strides towards the goal of creating model-based estimation techniques for these PHA, by generalizing the previous work on hybrid mode estimation to a Hybrid EM algorithm. However, many open issues remain to be explored, offering a number of intriguing research opportunities.

References

[Arnold, 1973] V. I. Arnold. *Ordinary Differential Equations,* The MIT Press, Cambridge, MA.1973.

[Athaide, 1995] C. R. Athaide. *Likelihood Evaluation and State Estimation for Nonlinear State Space Models.* Ph. D. Thesis, Graduate Group in Managerial Science and Applied Economics, University of Pennsylvania, Philadelphia, PA.

[Burgard et. al, 2000] W. Burgard, D. Fox, M. Moors, R. Simmons, and S. Thrun. Collaborative multi-robots exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation, (ICRA),* San Franciso, CA, 2000. IEEE.

[Carter and Kohn, 1994] C. K. Carter and R. Kohn. On Gibbs sampling for state space models. *Biometrika*, 81:541-553.

[Castellanos et al. 1999] J. Castellanos, J. M. M. Montiel, J. Neira, and J.D. Tradós. The Spmap: A probabilistic framework for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation,* 15(5):948-952.

[Dean and Kanazawa, 1989] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142-150.

[Dempster et. al., 1977] A. P. Dempster, A. N. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1): 1-38, 1977.

[Finn, 1999] C. Finn. Documentation of the BIO-Plex Baseline Simulation Model (Draft). NASA Ames Research Center. 1999.

Ghahramani and Hinton [1996a] Z. Ghahramani and G. E. Hinton. Parameter estimation for linear dynamical systems. University of Toronto Technical Report CRG-TR-96-2, 6 pages. http://www.gatsby.ucl.ac.uk/~zoubin/papers.html

Ghahramani and Hinton [1996b] Z. Ghahramani and G. E. Hinton. The EM Algorithm for Mixtures of Factor Analyzers. University of Toronto Technical Report CRG-TR-96-1, 8 pages. http://www.gatsby.ucl.ac.uk/~zoubin/papers.html

[Ghahramani and Hinton, 1998] Z. Ghahramani and G. E. Hinton. Variational learning for switching state-space models. Neural Computation, 12(4):963-996

[Ghahramani and Roweis 1999] Z. Ghahramani, S. and Roweis. Learning nonlinear dynamical systems using an EM algorithm. In M. S. Kearns, S. A. Solla, D. A. Cohn, (eds.) Advances in Neural Information Processing Systems 11:599-605. MIT Press.

[Hirsch and Smale, 1974] M W. Hirsch and S. Smale. *Differential Equations, Dynamic Systems, and Linear Algebra,* Academic Press, San Diego, CA, 1974.

[Hofbaur and Williams, 2002] M. Hofbaur and B.C. Williams. Mode estimation of probabilistic hybrid systems, in *Hybrid Systems: Computation and Control, HSCC 2002,* eds., C. J. Tomlin and M. R. Greenstreet, volume 2289 of *lecture Notes in Computer Science,* 253-266, Springer Verlag, 2002.

[Hofbaur and Williams, 2002b] M. Hofbaur and B.C. Williams. Hybrid Diagnosis with Unknown Behavioral Modes, in *Proceedings of the 13th International Workshop on Principles of Diagnosis,* 2002.

[Kanazawa et al., 1995] K. Kanazawa, D. Koller, and S. J. Russell. Stochastic simulation algorithms for dynamic probabilistic networks. In P. Besnard and S. Hanks, editors, *Uncertainty in Artificial Intelligence. Proceedings of the Eleventh Conference,* pages 346-351. Morgan Kaufmann Publishers, San Francisco, CA.

[Kim, 1994] C. J. Kim. Dynamic linear models with Markov-switching. *J. Econometrics,* 60:1-22.

[Kinney, 1997] J. J. Kinney. *Probability: An Introduction with Statistical Applications.* pp. 21, 36, & 66. John Wiley & Sons, Inc. New York, 1997.

[Leonard and Feder] J. J. Leonard and H. J. S. Feder. A computationally efficient method for large-scale concurrent mapping and localization. In J. Hollerbach and D. Koditschek, editors, *Proceedings of the Ninth International Symposium on Robotics Research,* Salt Lake City, Utah, 1999.

[MIMS III, 1983] F. Mims, III. *Getting Started in Electronics.* Radio Shack, 1983.

[McIlrith et al., 2000] S. McIlrith, G. Biswas, D. Clancy, and V. Gupta. Hybrid Systems Diagnosis. pp 282-295. In Proceedings of Hybrid Systems: Computation and Control, 2000.

[McWhorter and Evans, 1994] G. McWhorter and A. J. Evans. *Basic Electronics.* Radio Shack, 1994.

[Nicholson, 1980] H. Nicholson. *Modelling of dynamical systems,* Vol. 1 (Vol. 2m 1981) Peter Peregrinus, Stevenage, U.K., 1980.

[Press, 1992] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. Numerical Recipes in C, 2nd Edition. Cambridge University Press. 1992.

[Saul and Jordan, 1996] L. Saul and M. I. Jordan. Exploiting tractable substructures in Intractable networks. In D. Touretzky, M. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems 8.* MIT Press.

[Savage, 2000] D. Savage. NASA Outlines Mars Exploration Program for the Next Two Decades. 2000. http://solarsystem.nasa.gov/whatsnew/pr/001026G.html
[Shumway and Stoffer, 1982] R. H. Shumway and D. S. Stoffer. An approach to time series smoothing and forecasting using the EM algorithm. *J. Time Series Analysis,* 3(4):253-264, 1982.
[Shumway and Stoffer, 1991] R. H. Shumway and D. S. Stoffer. Dynamic Linear models with switching. *J. of the Amer. Stat. Assoc.,* 86:763-769, 1991.

[Simmons et. al., 2000] R. Simmons D. Apfelbaum, W. Burgard, M. Fox, D. an Moors, S. Thrun, and H. Younes. Cordination for multi-robot exploration and mapping. In *Proceedings of the AAAI National Conference on Artificial Intelligence,* Austin, TX 2000. AAAI.

[Strang, 1986] G. Strang. *Introduction to Applied Mathematics.* pp 32 – 41 & 137 – 138. Wellesley-Cambride Press, Wellesley, MA, 1986.

[White, 1994] R. M. White. *Lecture notes in EECS 40 I & 41 I.* University of California at Berkley, 1994.

[Williams and Millar, 1998] B. Williams and W. Milliar. Deompositonal Model-based Learning and it Analogy to Diagnosis. In *Proceedings of AAAI-1998.* AAAI

[Williams and Nayak, 1996] B. Williams and P. Nayak. A Model-based Approach to Reactive Self-configuring Systems. In *Proceedings of AAAI-1996.*

[Young et. al., 2000] Report on the Loss of Mars Polar Lander and Deep Space 2, (2000) in The Mars Program Independent Assessment Team report, 2000. NASA Headquarters.

Appendix A – PHA description

We frame a single Probabilistic Hybrid Automaton as an automaton that consist of the PHA UID, a set of modes, a set of transitions, and a set of variables. Such variables' set is comprised of input(s), state variable(s), and output(s) of a PHA (see Figure A.1).

(**PHA**:
UID: Unique id of PHA
Variables: # input(s), # state(s), # output(s)
Modes: $\mathcal{M} = \{m_{(0)}, ..., m_{(k)}\}$ ⇔ finite set of modes in the automaton
Transitions: $\tau_i = \{\tau_{i1}, ..., \tau_{in}\}$ ⇔ finite set of transitions in the automaton)

Figure A.1: PHA Structure of a Hybrid System

Next, we frame a mode of a PHA (see Figure B.2) to comprise of: (1) a mode UID, (2) an associated set of ordinary differential (ODEs), difference, or algebraic equation(s), which describes the behavior of the system in the mode, (3) an associated set of guards that must be satisfied in order for a particular transition function to occur, (4) a set of transition functions that transition from the source modes to the target modes in the PHA whenever a set of particular guards are satisfied. Such mode transitions are a set of outgoing transitions from a source mode.

(**Mode:**
UID: $m_{(k)}$ ⇔ Unique id of mode
ODE: the set of ordinary differential/difference equations for this mode
Guards: $C_{ij}, \ldots, C_{in}$ ⇔ guards satisfied when exiting m_k
Transitions: $\tau_r, \ldots, \tau_s$) ⇔ the set of transitions for mode m_k

Figure B.2: Mode Structure of the PHA

Finally, we frame a transition function of a PHA (see Figure B.3) to consist of: (1) a transition UID, (2) a source mode that specifies the origin of a particular mode transition, (3) a guard that has to be satisfied, and (4) an associated thread which gives the probability distribution of transiting from the source mode(s) to particular target mode(s).

(**Transition**:
UID: τ_r
Source: $m_{(k)}$ ⇔ Source mode of transition τ_r
Guard: C_{ij} ⇔ guard that has to be satisfied whenever transition τ_r is taken
Thread: $\left[p_j \quad m_l \right]$) ⇔ [probability target mode]

Figure B.3: Transition Structure of τ_r

Appendix B – PHA Examples

PHA for the Servo valve

The Servo valve used in chapter 4 can be modeled as a Probabilistic Hybrid Automata (PHA). Below, we give the entire PHA description:

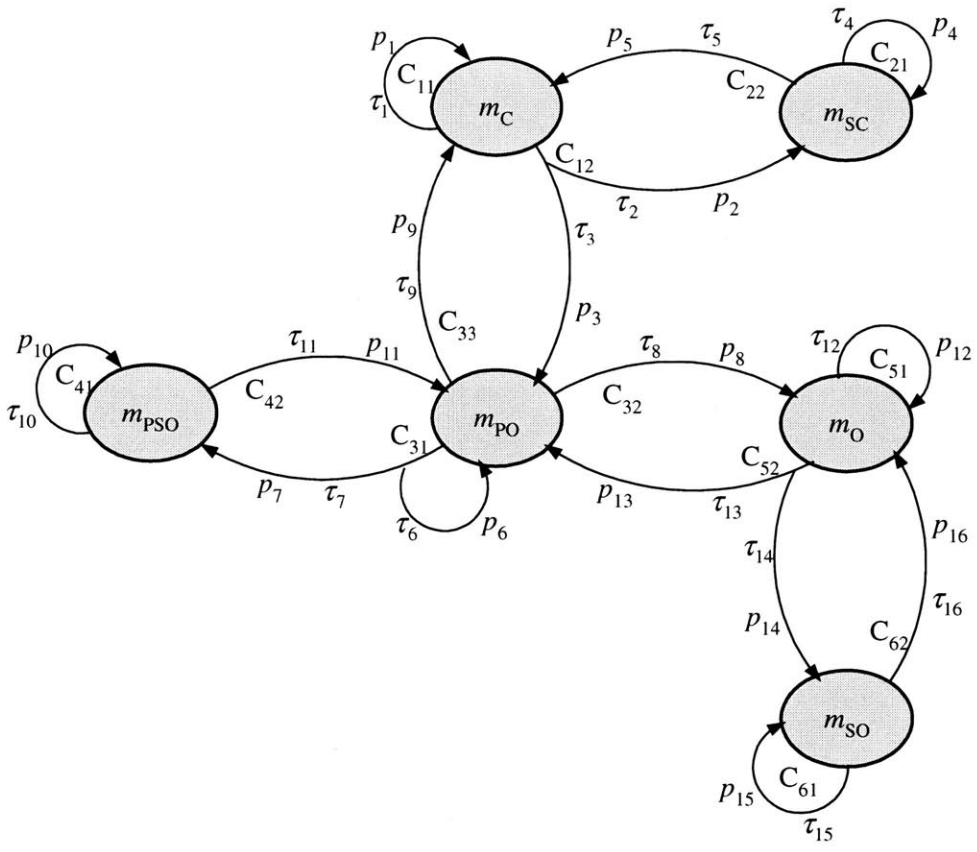


Figure B.1: PHA of the Servo valve

PHA: S_v

Variables: 1 input u, 1 state $\mathbf{x}$, 1 output Q, 6 modes

Modes: m_c, m_{po}, m_o, m_{sc}, m_{so}, m_{pso}

Transitions: τ_1, τ_2, τ_3, τ_4, τ_5, τ_6, τ_7, τ_8, τ_9, τ_{10}, τ_{11}, τ_{12}, τ_{13}, τ_{14}, τ_{15}, τ_{16}

Next, a mode of the PHA is comprised of: (1) a set of ODE, (2) an associated Kalman Filter, (3) the guards satisfied to exist a particular mode, (4) a set of transitions:

Mode: m_c
ODE: N/A
Filter: N/A
Guards: C_{11}, C_{12}
Transitions: τ_1, τ_2, τ_3

Mode: m_{sc}
ODE: N/A
Filter: N/A
Guards: C_{21}, C_{22}
Transitions: τ_4, τ_5

Mode: m_{po}
ODE: N/A
Filter: N/A
Guards: C_{31}, C_{32}, C_{33}
Transitions: τ_6, τ_7, τ_8, τ_9

Mode: m_{pso}
ODE: N/A
Filter: N/A
Guards: C_{41}, C_{42}
Transitions: τ_{10}, τ_{11}

Mode: m_o
ODE: N/A
Filter: N/A
Guards: C_{51}, C_{52}
Transitions: τ_{12}, τ_{13}, τ_{14}

Mode: m_{so}
ODE: N/A
Filter: N/A
Guards: C_{61}, C_{62}
Transitions: τ_{15}, τ_{16}

We frame a transition of a PHA to comprise of the following: (1) Source mode, (2) the guard Satisfied, and (3) the thread which consist of the probability of transitioning from the source mode to the new mode found in the tread:

Transition: τ_1
Source: m_c
Guard: C_{11}
Thread: $[\, p_1 \quad m_c \,]$

Transition: τ_2
Source: m_c
Guard: C_{12}
Thread: $[\, p_2 \quad m_{sc} \,]$

Transition: τ_3
Source: m_c
Guard: C_{12}
Thread: $[\, p_3 \quad m_{po} \,]$

Transition: τ_4
Source: m_{sc}
Guard: C_{21}
Thread: $[\, p_4 \quad m_{sc} \,]$

Transition: τ_5

Source: m_{sc}

Guard: C_{22}

Thread: $[\, p_5 \quad m_c \,]$

Transition: τ_6

Source: m_{po}

Guard: C_{31}

Thread: $[\, p_6 \quad m_{po} \,]$

Transition: τ_7

Source: m_{po}

Guard: C_{31}

Thread: $[\, p_7 \quad m_{pso} \,]$

Transition: τ_8

Source: m_{po}

Guard: C_{32}

Thread: $[\, p_8 \quad m_o \,]$

Transition: τ_9

Source: m_{po}

Guard: C_{33}

Thread: $[\, p_9 \quad m_c \,]$

Transition: τ_{10}

Source: m_{pso}

Guard: C_{41}

Thread: $[\, p_{10} \quad m_{pso} \,]$

Transition: τ_{11}

Source: m_{pso}

Guard: C_{42}

Thread: $[\, p_{11} \quad m_{po} \,]$

Transition: τ_{13}

Source: m_o

Guard: C_{52}

Thread: $[\, p_{13} \quad m_{po} \,]$

Transition: τ_{12}

Source: m_o

Guard: C_{51}

Thread: $[\, p_{12} \quad m_o \,]$

Transition: τ_{14}

Source: m_o

Guard: C_{52}

Thread: $[\, p_{14} \quad m_{so} \,]$

Transition: τ_{15}

Source: m_{so}

Guard: C_{61}

Thread: $[\, p_{15} \quad m_{so} \,]$

Transition: τ_{16}

Source: m_{so}

Guard: C_{62}

Thread: $[\, p_{16} \quad m_o \,]$

Linear Dynamic System modeled as a PHA

The following Linear Time-invariant System model is framed as a Probabilistic Hybrid Automaton. The system has 1 input, 2 state variables and 1 output. It has 4 modes and 13 transitions between the modes.

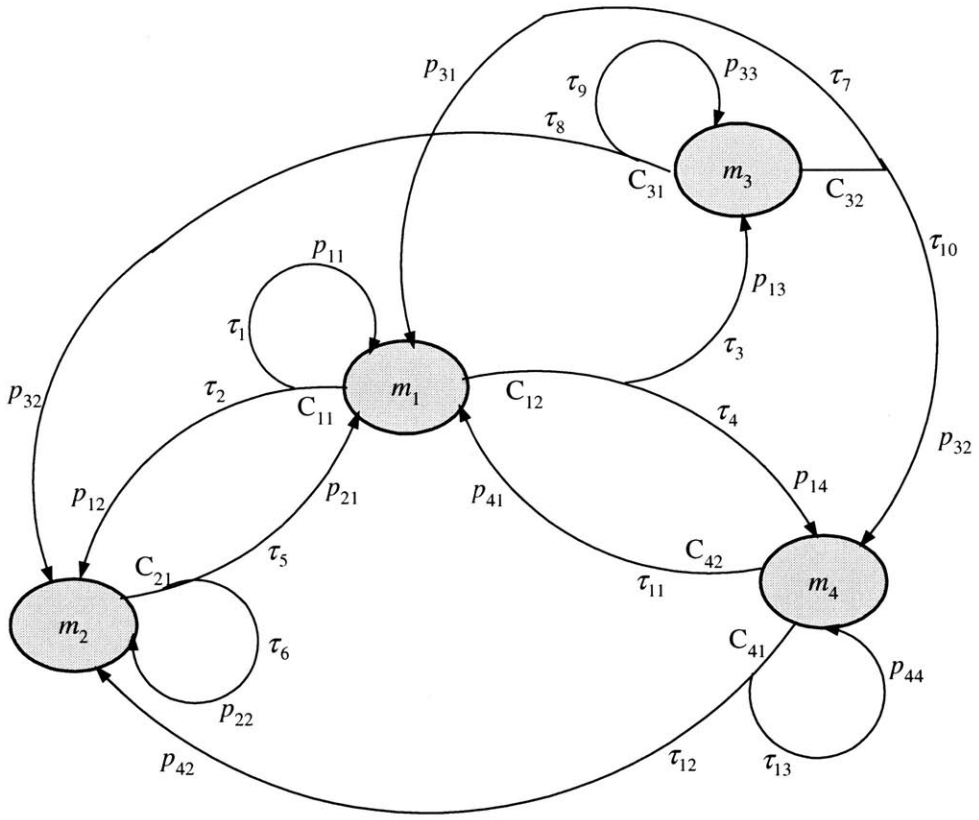


Figure B.2: PHA of Linear Time-invariant System

We frame each mode of the PHA to consist of: (1) a set of ODE, (2) an associated Kalman Filter, (3) the guards satisfied to exist a particular mode, and (4) a set of transitions. These are as follows:

Mode: m_1
ODE: N/A
Filter: N/A
Guards: C_{11}, C_{12}
Transitions: τ_1, τ_2, τ_3, τ_4

Mode: m_2
ODE: N/A
Filter: N/A
Guards: C_{21},
Transitions: τ_5, τ_6)

Mode: m_3

ODE: N/A
Filter: N/A
Guards: C_{31}, C_{32}
Transitions: τ_7, τ_8, τ_9, τ_{10}

Mode: m_4

ODE: N/A
Filter: N/A
Guards: C_{41}, C_{42}
Transitions: τ_{11}, τ_{12}, τ_{13}

We then frame a transition of a PHA to comprise of the following: (1) Source mode, (2) the guard Satisfied, and (3) the thread, which consist of the probability of transitioning from the source mode to the new mode found in the tread:

Transition: τ_1

Source: m_1
Guard: C_{11}
Thread: $\left[p_{11} \quad m_1 \right]$

Transition: τ_2

Source: m_1
Guard: C_{11}
Thread: $\left[p_{12} \quad m_2 \right]$

Transition: τ_3

Source: m_1
Guard: C_{12}
Thread: $\left[p_{13} \quad m_3 \right]$

Transition: τ_4

Source: m_1
Guard: C_{12}
Thread: $\left[p_{14} \quad m_4 \right]$

Transition: τ_5

Source: m_2
Guard: C_{21}
Thread: $\left[p_{21} \quad m_1 \right]$

Transition: τ_6

Source: m_2
Guard: C_{21}
Thread: $\left[p_{22} \quad m_2 \right]$

Transition: τ_7

Source: m_3
Guard: C_{32}
Thread: $\left[p_{31} \quad m_1 \right]$

Transition: τ_8

Source: m_3
Guard: C_{31}
Thread: $\left[p_{32} \quad m_2 \right]$

Transition: τ_9

Source: m_3
Guard: C_{31}
Thread: $\left[p_{33} \quad m_3 \right]$

Transition: τ_{10}

Source: m_3
Guard: C_{32}
Thread: $\left[p_{34} \quad m_4 \right]$

Transition: τ_{11}

Source: m_4

Guard: C_{42}

Thread: $\left[p_{41} \quad m_1 \right]$

Transition: τ_{12}

Source: m_4

Guard: C_{41}

Thread: $\left[p_{42} \quad m_2 \right]$

Transition: τ_{13}

Source: m_4

Guard: C_{41}

Thread: $\left[p_{44} \quad m_4 \right]$

Appendix C – Pseudo code: Hybrid EM algorithm

```
Hybrid_EM(Data, PHA)
        Begin loop
                labeled_data = Hybrid _E(Data, PHA);
                Update_PHA = Hybrid_M(labeled_data, PHA);
                If converged?(PHA, Update_PHA)
                        Return Update_PHA
                else
                        PHA = Update_PHA
                End-If
        End  loop
End-Hybrid_EM

Hybrid_E(Data,PHA)
        Detect mode changes
        Labels each data point in data set with the most likely mode of Model
        Return labeled_data
End-Hybrid_E

Hybrid-M(labeled-data,PHA)
        ; Given a set of labeled data, estimated the equation parameters and
        ; transition probabilities for PHA.

        Update_PHA = Update_Equation_Parameters(labeled_data, PHA);
        Update_PHA = Update_Transition_Probabilities(labeled_data, PHA);
        Return Update_PHA
End-Hybrid_M

Update_Equation_Parameters(labeled-data, PHA)
    ; Given a set of labeled data, estimates the parameters for the equations
    ; of every mode in PHA.

        For each m in modes(PHA)
                bucket(m) ={ }
        End-For
        For each <data-point, mode> in labeled-data
                add data-point to bucket(mode)
        End-For
        For each m in modes(PHA)
                m-eqn(ID(m)) = equations(m)
                m-data(ID(m)) = bucket(m)
        End-For
        Set_of_Parameters = Estimate_Parameters(m-eqn, m-data)
    return PHA
End-Update_Equation_Parameters
```

```
Update_Transition_Probabilities (labeled_data,PHA)
    ;   Given a set of labeled data
    ;   Estimate the transition probabilities for the model

    m = number of modes(PHA)
    ;     gs = guards satisfied whenever we are in mode m
    MGM = matrix(m, gs)
    ;     ts = transition taken when a particular mode is satisfied
    GSMEM = matrix (gs, ts)
    MGSM = matrix(size_of _GSMEM)
    MGOM = matrix(length_of_GSMEM, 1)
    Update_PHA(TP) = Estimate_Transition_Probabilities(MGSM, MGOM, PHA)
    Return Model
End-Update-Transition-Probabilities

Estimate-Transition-Probabilities(MGSM, MGOM, PHA)
        Bin = matrix(size-of-matrix(MGSM))
                For i = 1 to number-of-rows(MGSM)
                        For j = 1 to number-of-columns(MGSM)
                                If MGOM(i) ≠ 0
                                Bin(i, j) = MGSM(i, j) / MGOM(i)
                End-For
        End-For
        Return Bin
End-Estimate-Transition-Probabilities
```

Appendix D – Raw Data Dump

```
 To get started, select "MATLAB Help" from the Help menu.
 >> main
ans =
------------------------------Please wait------------------------------
ans =
-----------------------------Program Executing-----------------------------
Optimization terminated successfully:
Relative function value changing by less than OPTIONS.TolFun
Optimization terminated successfully:
Norm of the current step is less than OPTIONS.TolX
Optimization terminated successfully:
Norm of the current step is less than OPTIONS.TolX
Optimization terminated successfully:
 Relative function value changing by less than OPTIONS.TolFun
ans =
--------------------------------------------------------------------------
ans =
   (1) Mode parameters estimated
ans =
   (2) Transition probabilities calculated
ans =
   (3) PHA update
ans =
----------------------------Program Completed----------------------------
>> pha
pha =
        name: 1
    variable: [1x1 struct]
        mode: [1x4 struct]
  transition: [1x13 struct]
>> pha(1)
ans =
        name: 1
    variable: [1x1 struct]
        mode: [1x4 struct]
  transition: [1x13 struct]
>> pha(1).variable
ans =
    inputs: 1
    states: 2
   outputs: 1
     modes: 4
>> pha(1).mode
ans =
1x4 struct array with fields:
   id
   guards
```

```
    transitions
>> pha(1).mode(1)
ans =
          id: 1
      guards: [1 2]
 transitions: [1 2 3 4]
>> pha(1).mode(2)
ans =
          id: 2
      guards: 3
 transitions: [5 6]
>> pha(1).mode(3)
ans =
          id: 3
      guards: [4 5]
 transitions: [7 8 9 10]
>> pha(1).mode(4)
ans =
          id: 4
      guards: [6 7]
 transitions: [11 12 13]
>> pha(1).transition(1)
ans =
      id: 1
  thread: [0.9899 1]
>> pha(1).transition(2)
ans =
      id: 2
  source: 1
   guard: 1
  thread: [0.0101 2]
>> pha(1).transition(3)
ans =
      id: 3
  source: 1
   guard: 2
  thread: [0.6380 3]
>> pha(1).transition(4)
ans =
      id: 4
  source: 1
   guard: 2
  thread: [0.3620 4]
>> pha(1).transition(5)
ans =
      id: 5
  source: 2
   guard: 3
  thread: [0.1001 1]
>> pha(1).transition(6)
ans =
      id: 6
  source: 2
   guard: 3
```

```
   thread: [0.8999 2]
>> pha(1).transition(7)
ans =
       id: 7
   source: 3
    guard: 5
   thread: [0.6476 1]
>> pha(1).transition(8)
ans =
       id: 8
   source: 3
    guard: 4
   thread: [0.0106 2]
>> pha(1).transition(9)
ans =
       id: 9
   source: 3
    guard: 4
   thread: [0.9894 3]
>> pha(1).transition(10)
ans =
       id: 10
   source: 3
    guard: 5
   thread: [0.3524 4]
>> pha(1).transition(11)
ans =
       id: 11
   source: 4
    guard: 7
   thread: [1 1]
>> pha(1).transition(12)
ans =
       id: 12
   source: 4
    guard: 6
   thread: [0.0129 2]
>> pha(1).transition(13)
ans =
       id: 13
   source: 4
    guard: 6
   thread: [0.9871 4]
>> mode(1).parameters
ans =
   0.7650   0.0200   1.5843
   0.0071   0.6796   2.4758
   1.0003   1.0004  -0.0058
>> mode(2).parameters
ans =
  -0.0000   0.9269   0.3755
  -0.7495   1.4930   1.2845
   0.9998   1.0003  -0.0006
>> mode(3).parameters
```

```
ans =
   0.6893   -0.0007    1.3554
   0.9201    0.7666    2.9214
   1.0033    0.9998   -0.0090
>> mode(4).parameters
ans =
   0.5313    0.7596    3.7817
   0.0179    0.7038    3.8444
   0.9999    1.0004   -0.0044
>>
```

Appendix E – PHA Examples

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
%% Programmer: Melvin M. Henry                         %%
%% Topic: Mode Estimation of Probabilistics Hybrid Automata        %%
%% Date: Monday, August 20, 2001                       %%
%%                                                  %%
%% TASK:                                              %%
%% The main script that is used to invoke the different functions and %%
%% other scripts that are used to create the PHA model, load        %%
%% labeled_data, update Equation paramters, update the transition,    %%
%% and update the structure of the PHA                 %%
%%                                                  %%
%%                                                  %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%


'------------------------------Please wait------------------------------'
'---------------------------Program Executing---------------------------'
% Clear workspace
clear;
% Store PHA in the Hybrid System structure
MODEL;
% Loading the data set and creating "labeled_data"
load ldata8;
% Determine the length of labelel_data
n = length(labeled_data);
% Invoking the Hybrid EM algorithm
mode = Hybrid_EM(labeled_data,pha);
% Invoke of the Hybrid M function
% mode = Hybrid_M(labeled_data,pha);
% Invoking the Update Equation parameter function
% Calculating the parameters of each mode
% mode = Update_Eqn_Parameters(labeled_data,pha);
% Calculating the transition probabilies of the PHA
TPM = Update_Tran_Probabilities(labeled_data,pha);
% Updating the PHA with these probabilities of the PHA
Update_PHA;
storeResults;
'-------------------------------------------------------------------'
'  (1) Mode parameters estimated                    '
'  (2) Transition probabilities calculated          '
'  (3) PHA update                                   '
'--------------------------Program Completed--------------------------'
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                  %
% Name: Melvin M. Henry                                    %
% Type: Model                                        %
% Date: Thursday, July 26, 2001                           %
%                                                  %
% Description:                                        %
%  This file explains a particular 'MODEL' that is found in the Hybrid EM        %
% alogorithm function.  Each PHA has three parts: (1) Name (2) Variables (3) Modes.    %
```

```
% Each mode has an ID, ODEs, associated filters, and transitions.                %
%                                                                                %
% This is the new PHA format.                                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Defining the PHA
% Currently, there exist only one Model
% Name of the PHA
pha(1).name = 1;
% Variables of PHA  => 1 input, 2 States, 1 output
pha(1).variable.inputs = 1;
% State variable of the Model
pha(1).variable.states = 2;
% Output of the PHA
pha(1).variable.outputs = 1;
% Modes of the PHA
pha(1).variable.modes = 4;

% Counter for the number of transitions in the PHA
tn = 0;

% Number of mode in HPA 1
mc = pha(1).variable.modes;

% Define the mode structure
% Each mode has an id, associated modes, associated filters, and transitions
% Also each mode displays the "satified guards".
for i = 1:mc,  % Number of modes
  % Setting the guards satisfied whenever mode i exited
  if i == 1
    g = i;
  elseif i == 2 | i == 3
    g = i+1;
  else
    g = i+2;
  end

  if i == 2
    % Mode id
    pha(1).mode(i).id = i;
    % Mode ode
    pha(1).mode(i).ode = i;
    % Mode filter
    pha(1).mode(i).filter = i;
    % Satified guards
    pha(1).mode(i).guards = g;
    % Mode transition
    pha(1).mode(i).transitions = [tn+1, tn+2];
    tn = tn + 2;
  elseif i == 4
    pha(1).mode(i).id = i;        % Mode id
    pha(1).mode(i).ode = i;       % Mode ode
    pha(1).mode(i).filter = i;    % Mode filter
    pha(1).mode(i).guards = [g, g+1];  % Satisfied guards
    pha(1).mode(i).transitions = [tn+1, tn+2, tn+3];  % Mode transition
    tn = tn + 3;
  else    % mode 2 has only one transition
    pha(1).mode(i).id = i;        % Mode id
    pha(1).mode(i).ode = i;       % Mode ode
    pha(1).mode(i).filter = i;    % Mode filter
```

```
      pha(1).mode(i).guards = [g, g+1];    % Satisfied guards
      pha(1).mode(i).transitions = [tn+1, tn+2, tn+3, tn+4];  % Mode transition
      tn = tn + 4;
    end
  end

  % Define the transitions of the HPA
  % Each transition has an id, an associated guards, and associated threads
  for i = 1:mc,
    for h = 1:tn,
      % Each mode has at most 4 transitions
      if h >= 1 & h <= 4
        if h == 1 | h == 2
          C = 1;   % Setting the Guards for each Transition
        else
          C = 2;
        end
      elseif h >= 5 & h <= 6
        C = 3;
      elseif h  >= 7 & h <= 10
        if h == 7 | h == 10
          C = 5;   % Setting the Guards for each Transition
        else
          C = 4;
        end
      else
        if h == 11
          C = 7;   % Setting the Guards for each Transition
        else
          C = 6;
        end
      end

      % Setting the mode the system transition to
      if h == 1 | h == 5 | h == 7 | h == 11
        m = 1;
      elseif h == 2 | h == 6 | h == 8 | h == 12
        m = 2;
      elseif h == 3 | h == 9
        m =3;
      else
        m = 4;
      end

      % Transition counter for a mode
      tc = length(pha.mode(i).transitions);
      for j = 1:tc,
        % Intializing the transitions of the PHA
        if pha(1).mode(i).transitions(j) == h
          % transition id
          pha(1).transition(h).id = h;
          % source of transition
          pha(1).transition(h).source = i;
          % guard transition
          pha(1).transition(h).guard = C;
          % transition thread
          pha(1).transition(h).thread = [h, m;];
        end
      end
    end
  end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                          %%
%% Name: Melvin M. Henry                             %%
%% Date: Monday, May 7, 2001                          %%
%%                                          %%
%% Algorithm: Hybrid_EM                              %%
%%                                          %%
%% The Hybrid_EM algorithm is acroymn for Hybrid Expectation            %%
%% Maximization algorithm.  It consists of two steps:             %%
%% (1) Hybrid Expectation Step (E-Step)                    %%
%% (2) Hybrid Maximization Step (M-Step)                   %%
%%                                          %%
%% METHOD:                                       %%
%%     The function Hybrid_EM receives two arguments Data and Model.     %%
%% First, the Hybrid_E function is invoked with these two arguments.     %%
%% Within the Hybrid_E, each data point is assigned to the most likely     %%
%% mood it is belief it belongs to. However, inorder to perform this     %%
%% task, the assumption that the parameters of the model are known.     %%
%%     Then the function Hybrid_M is invoked.  Here, this function has     %%
%% two arguments: labeled_data and Model.  These arguments are used to     %%
%% estimate the parameters of the model and the transition probabilites.  %%
%% The 'new_Model' is returned which contains these estimates.  The      %%
%% new_Model is compared to the current 'Model' to check for convergence   %%
%%                                          %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function New_Model = Hybrid_EM(labeled_data,pha)

% Given a set of Data points and a Model, labeled the data according to
% the most likely mode it is belief to be in.
% labeled_data = Hybrid_E(Data,Model);

New_Model = Hybrid_M(labeled_data,pha);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  Programmer: Melvin M. Henry                        %%
%%  Topic: Mode Estimation of Probabilistics Hybrid Automata        %%
%%  Date: Friday, May 4, 2001                       %%
%%                                      %%
%%  TASK:                                    %%
%%  Given a set of labeled data, estimate the equation parameters      %%
%%      and transition probabilities for the Model.            %%
%%                                      %%
%%  METHOD:                                   %%
%%     The first function is invoked to estimate the equations       %%
%%  Parameters.  Then the second function is invoked to estimate       %%
%%  the Transition Probabilities.                      %%
%%                                      %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function Model = Hybrid_M(labeled_data,pha)

% Invokes functions
Model = Update_Eqn_Parameters(labeled_data,pha);
% Model1 = Update_Tran_Probabilities(labeled_data,pha);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
%% Programmer: Melvin M. Henry                    %%
%% Topic: Mode Estimation of Probabilistics Hybrid Automata       %%
%% Date: Friday, June 29, 2001                    %%
%%                                  %%
%% TASK:                              %%
%% Given a set of labeled data, estimate the parameters for the      %%
%%    equations of every mode in the model. Each labeled data is    %%
%%    a pair of (<data point, mode>).                %%
%%                                  %%
%% METHOD:                               %%
%%                                  %%
%%                                  %%
%%                                  %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%


function Model = Update_Eqn_Parameters(labeled_data,pha)

% Allocating memory for local variable called bucket. Each bucket is
% used to sort a data point according to the most likely mode that the
% data point belong to.

% Allocate memory for local variables

% l is equal to the number of columns in labeled_data
l = size(labeled_data);
nc = l(1,2);
ld = length(labeled_data);

% Each bucket is a local variable used to sort the data points
% There are three modes of this system
% Determine the number of modes in the Model
m = pha(1).variable.modes;

% Each mode is governed by a set of equations.
% m_eqn is a local variable used to stored these set of equations
m_eqn = zeros(m,1);

% Handles a variable set of modes
% Creating the same number of buckets as modes.
for i = 1:m,
  bucket(i,1).mode = i;
  % Buckets are filled with zeros
  bucket(i,1).values = zeros(ld,nc);
  % Initialize the bucket counter
  bucket(i,1).counter = 0;
end

for i = 1:m,
  % For mode i, check for transition out of mode i. If
  % transition occurs, the data point is not stored in bucket i.

  % Store the datapoints into buckets. Each bucket represents a mode of the PHA.
  % Data points are stored according to the most likely mode.
  for j = 1:ld-1,
    if (labeled_data(j,nc) == i)
      if (labeled_data(j+1,nc) == i)
        % Increment the bucket counter by 1
```

```
            bucket(i,1).counter = bucket(i,1).counter + 1;
            % store all remaining data points belonging to mode i into bucket i
            bucket(i,1).values(bucket(i,1).counter,:) = labeled_data(j,:);
         end
      end
   end
end

% Store datapoints found in buckets into m_data matrices
for i = 1:m,
  m_data(i,1).mode = i;
  m_data(i,1).values = bucket(i,1).values(1:bucket(i).counter,:);
  m_data(i,1).counter = bucket(i,1).counter;
end

Model = Estimate_Parameters(m_eqn, m_data);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Programmer: Melvin M. Henry                              %%
%% Topic: Mode Estimation of Probabilistics Hybrid Automata            %%
%% Date: Friday, June 29, 2001                              %%
%%                                                 %%
%% TASK:                                              %%
%% Given a set of labeled data, estimate the transition              %%
%%    probabilities for the model.  Each labeled data is a pair        %%
%%    (<data point, mode>).                                 %%
%% METHOD:                                            %%
%%                                                 %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Model = Estimate_Parameters(m_eqn,m_data)

% The different types of outputs
ny = 1;

% The different types of inputs
nu = 1;

% the different types of state-variables that exist.
nx = 2;

% Number of States and output combined
nyd = nx + ny;

% Number of types of States and inputs combined
nxd = nx + nu;

x0 = zeros(3);

% Determine the number of modes in the labeled data
m = m_data(length(m_data)).mode;   % Change to make it not fixed

% Creating number of input vectors of 1 row by mode counter cloumn
% filled with 1s.  one for each mode of the system.
 for i = 1:m,
   u(i,1).mode = i;
   u(i,1).values = m_data(i,1).values(1:m_data(i,1).counter,1);
   u(i,1).counter = m_data(i,1).counter;
end
```

```
% Creating the number of state vectors
for k = 1:nx,
   for i = 1:m,
      x(i,k).mode = i;
      x(i,k).values = m_data(i,1).values(1:m_data(i,1).counter,k+1);
      x(i,k).counter = m_data(i,1).counter;
   end
end

% Creating the number of output vectors (structures)
for i = 1:m,
    y(i,1).mode = i;
    y(i,1).values = m_data(i,1).values(1:m_data(i,1).counter,4);
    y(i,1).counter = m_data(i,1).counter;
end

for i = 1:m,
   % Creating storage area for XDATA and YDATA
   % Setting XDATA and YDATA to matrices of zeros
   ydata = zeros(length(x(i,1).values),nyd);
   xdata = zeros(length(x(i,1).values),nxd);
   % Storing the values into XDATA and YDATA
   for j = 1:nxd,
      c1 = length(x(i).values);
      for k = 2:c1,
         if j == nxd,
            ydata(k-1,j) = y(i,1).values(k-1,1);
            xdata(k-1,j) = u(i,1).values(k-1,1);
         else
            ydata(k-1,j) = x(i,j).values(k,1);
            xdata(k-1,j) = x(i,j).values(k-1,1);
         end
      end
   end
   % Transpose matrix so information can be representing in
   % the proper format for accessing.
   xdata = transpose(xdata);
   ydata = transpose(ydata);
   % Store the parameters of each mode in a data struture for
   % future access
   mode(i).parameters = lsqcurvefit(@cal_Param,x0,xdata,ydata);
   % Store the parameters of mode i into mat file resultsD
   % save resultsD store -MAT -APPEND
   % save resultsD2 resultsD -ASCII
end

% Return the structure that contains all the modes' parameters
% of the PHA structure
Model = mode;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Programmer: Melvin M. Henry                          %%
%% Topic: Mode Estimation of Probabilistics Hybrid Automata         %%
%% Date: Wednesday, August 1, 2001                      %%
%%                                             %%
%% TASK:                                          %%
%% Given a set of labeled data, estimate the transition          %%
%%    probabilities for the model.  Each labeled data is a pair   %%
%%    (<data point, mode>). This implementation takes into        %%
```

```
%%    consideration the guards that are satisfied whenever there  %%
%%    is a transition from mode i to mode j                %%
%%                                          %%
%% METHOD:                                          %%
%%    For each pair of modes (mode-1, mode-2), estimate the       %%
%%    fraction of times a transition is made from mode-1 to       %%
%%    mode-2 among all transtions out of mode-1.              %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

function Model = Update_Tran_Probabilities(labeled_data,pha)

% Allocate memory for local variables
% determine the number of modes in the PHA
m = pha(1).variable.modes;

% Mode entered after a guard is satified
me = 2;

% For each mode, determine the number of guards it has.
% Store largest number of guards found while looking at the modes.
mGlen = length(pha(1).mode(1).guards);
for i = 2:m,
   % If the next mode has more guards than the previous mode,
   % store the maximum number of guards always
   Glen = length(pha(1).mode(i).guards);
   if Glen > mGlen
      mGlen = Glen;
   end
end

% Create a matrix for guards of each mode
MGM = zeros(m,mGlen);

for i = 1:m,
   % Determine how many guards are there in each mode
   tgl = length(pha(1).mode(i).guards);
   for cl = 1:tgl,
      % Store the guards satisfied by each mode into the
      % Mode Guard Matrix (MGM)
      MGM(i,cl) = pha(1).mode(i).guards(cl);
   end
end

% For a particular guard c, iterate through the transition set
% to determine which modes are entered whenever mode i is existed.
for i = 1:m,
   % Store the length of each mode transition's set
   tranLen = length(pha.mode(i).transitions);
   % Store the first transition ID in the transition set
   tranLow = pha.mode(i).transitions(1);
   % Store the last transition ID in the mode transition set
   tranHigh = pha.mode(i).transitions(1) + tranLen - 1;
   for cl = 1:mGlen,
      Mn = 1;
      % For a particular mode i and a guard position
      % store the guard satisfied
      Gid = MGM(i,cl);
      for t = tranLow:tranHigh,
         if Gid == pha.transition(t).guard
            % Store modes entered from mode i into the
```

```
            % Guard satified Mode entered matrix (GSMEM)
            GSMEM(Gid,Mn) = pha.transition(t).thread(me);
            Mn = Mn+1;
         end
      end
   end
end

% Store the dimension of the GSMEM.  We really want the column size
% of the GSMEM
ML = size(GSMEM);
ML = ML(1,2);

% Creating a matrix the same size as GSMEM to store the number of
% times each mode is entered whenever a guard is satisfied
SM = size(GSMEM);

% Mode Guard Satisfication Matrix
% use to store the guard satisfied whenever we exit a particular mode
MGSM = zeros(SM(1,1), SM(1,2));
MGOM = zeros(SM(1,1), 1);

% Determine the column that contains the mode of each data point
mc = size(labeled_data);
mc = mc(1,2);

% Calculating the length of the labeled_data
ld = length(labeled_data);

for k = 2:ld-1,
   % Modes are located in the last column of Labeled_data

   % For each consecutive pair (<data-point1,mode1>,<data-point1,mode1>)
   % in the labeled_data, calculate the transition probabilities
   % whenever a particular guard is satisfied.
   u = labeled_data(k,mc);
   v = labeled_data(k+1,mc);

   for cl = 1:mGlen,
      % Finding all the Guard IDs whenever we exit a particular mode
      Gid = MGM(u,cl);
      for tl = 1:ML,
         if Gid ~= 0 & GSMEM(Gid,tl) ~= 0 & GSMEM(Gid,tl) == v
            % Increment the current value by one whenever a transition
            % occur which satisfied Guard 'Gid' and a particular mode
            MGSM(Gid,tl) = MGSM(Gid,tl) + 1;
            % Increment the current value by one whenever a transition
            % occur which satisfied Guard 'Gid'
            MGOM(Gid) = MGOM(Gid) + 1;
         end
      end
   end
end

% Invoke the function that estimate the Transition Probabilities
Model = Estimate_Tran_Probabilities(MGSM, MGOM, pha);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
%%  Programmer: Melvin M. Henry                    %%
%%  Topic: Mode Estimation of Probabilistics Hybrid Automata       %%
```

```
%% Date: Thursday, May 31, 2001                                    %%
%%                                                                 %%
%% TASK:                                                          %%
%% Given the Mode Transition Matrix (MTM) and Mode Origination     %%
%%    Matrix (MOM)of some unknown model, calculate the transition  %%
%%    Probabilities of the Model.                                  %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%

function Bin = Estimate_Tran_Probabilities(MGSM, MGOM, pha)

% Allocate memory for matrix 'Bin' the same size as MGSM
% Set all the values of Bin to zero
Bin = zeros(size(MGSM));

% Store the dimension of MGSM
n = size(MGSM);

% Calculate the Transition Probabilities of each mode
% over all rows of MGSM
for i = 1:n(1,1),
  % over all columns of MGSM
  for j = 1:n(1,2),
    Bin(i,j) = MGSM(i,j) / MGOM(i);
  end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
%% Programmer: Melvin M. Henry                                    %%
%% Topic: Mode Estimation of Probabilistics Hybrid Automata       %%
%% Date: Monday, August 20, 2001                                  %%
%%                                                                %%
%% TASK:                                                          %%
%% Given the old PHA and the Transition probabilities matrix, we can  %%
%% update the PHA to contain these estimated valves.              %%
%%    a pair of (<data point, mode>).                             %%
%%                                                                %%
%% METHOD:                                                        %%
%%                                                                %%
%%                                                                %%
%%                                                                %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%

% Allocate memory for local variables
% determine the number of modes in the PHA
m = pha(1).variable.modes;

% Mode entered after a guard is satified
% Represents the position where the mode entered is stored
me = 2;

% For each mode, determine the number of guards it has.
% Store largest number of guards found while looking at the modes.
mGlen = length(pha(1).mode(1).guards);
for i = 2:m,
  % If the next mode has more guards than the previous mode,
  % store the maximum number of guards always
  Glen = length(pha(1).mode(i).guards);
```

```
  if Glen > mGlen
    mGlen = Glen;
  end
end

% Create a matrix for guards of each mode
MGM = zeros(m,mGlen);

for i = 1:m,
  % Determine how many guards are there in each mode
  tgl = length(pha(1).mode(i).guards);
  for cl = 1:tgl,
    % Store the guards satisfied by each mode into the
    % Mode Guard Matrix (MGM)
    MGM(i,cl) = pha(1).mode(i).guards(cl);
  end
end

% For a particular guard c, iterate through the transition set
% to determine which modes are entered whenever mode i is existed.
for i = 1:m,
  % Store the length of each mode transition's set
  tranLen = length(pha.mode(i).transitions);
  % Store the first transition ID in the transition set
  tranLow = pha.mode(i).transitions(1);
  % Store the last transition ID in the mode transition set
  tranHigh = pha.mode(i).transitions(1) + tranLen - 1;
  for cl = 1:mGlen,
    Mn = 1;
    % For a particular mode i and a guard position
    % store the guard satisfied
    Gid = MGM(i,cl);
    for t = tranLow:tranHigh,
      if Gid == pha.transition(t).guard
        % Store modes entered from mode i into the
        % Guard satified Mode entered matrix (GSMEM)
        GSMEM(Gid,Mn) = pha.transition(t).thread(me);
        Mn = Mn+1;
      end
    end
  end
end

% Determining the number of transitions in the PHA
ln = length(pha.transition);

% Updating the transition probabilities of the PHA
for k = 1:ln,
  % Guard satisfied when the mode was exits
  Gid = pha(1).transition(k).guard;
  % destination of mode transition
  d = pha(1).transition(k).thread(me);

  % Search through GSMEM to find the destination mode of the
  % guarded transtion
  for cl = 1:mGlen,
    dm = GSMEM(Gid,cl);
    if dm == d
      % Storing probabilities for each transition
      % into the PHA's structure
      p = TPM(Gid,cl);
      pha(1).transition(k).thread = [p, d;];
```

```
    end
  end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
%% Programmer: Melvin M. Henry                          %%
%% Topic: Mode Estimation of Probabilistics Hybrid Automata        %%
%% Date: Friday, November 11, 2001                       %%
%%                                                       %%
%% TASK:                                                 %%
%% Once we have calculated the equation parameters for each mode and %%
%% we have estimated all the transition probabilities of the PHA, we %%
%% should store these results in a text file called data which is then %%
%% used by the Hybrid E step of the Hybrid EM algorithm.       %%
%%                                                       %%
%% Method:                                               %%
%%    The Parameters of the PHA are stored in a text file called data %%
%% in the standard Lisp format since the Hybrid E step is written in  %%
%% Lisp.                                                 %%
%%                                                       %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%

% Transition function Starting number
st = 1;
% Open the file data with write permission
fid = fopen('data.tex', 'w');
% Transition through the each mode in the PHA
for i = 1:mc,
  % intializing the row and column to begin at 1
  r = 1;
  c  = 1;
  if i ~= 1
     fprintf(fid,['\n\n']);
  end
  % Print the results estimated to file storeResults
  % Storing mode i of the PHA
  fprintf(fid,['((M',num2str(pha.mode(i).id)]);
  % Storing the parameter A in the file
  fprintf(fid,['\n(A((',num2str(mode(i).parameters(r,c)),' ']);
  fprintf(fid,[num2str(mode(i).parameters(r,c+1)),')']);
  fprintf(fid,[' (',num2str(mode(i).parameters(r+1,c)),' ']);
  fprintf(fid,[,num2str(mode(i).parameters(r+1,c+1)),')))']);
  % Storing the parameter B in the file
  fprintf(fid,['\n(B(',num2str(mode(i).parameters(r+2,c)),' ']);
  fprintf(fid,[num2str(mode(i).parameters(r+2,c+1)),'))']);
  % Storing the parameter C in the file
  fprintf(fid,['\n(C((',num2str(mode(i).parameters(r,c+2)),')(']);
  fprintf(fid,[num2str(mode(i).parameters(r+1,c+2)),')))']);
  % Storing the parameter D in the file
  fprintf(fid,['\n(D(',num2str(mode(i).parameters(r+2,c+2)),'))']);
  % Transition counter for a mode
  tc = length(pha.mode(i).transitions);
  en = st + tc - 1;
  n = 1;
  % Transitioning through all the transition fucntions for mode i
  for j = st:en,
    % Storing the transition probilities of each mode i
    % Storing the transition function (TF) number j
    fprintf(fid,['\n(T',num2str(n)]);
```

```
        % Storing the mode i and the transition goal of TF
        fprintf(fid,['((M',num2str(pha(1).transition(j).thread(2)),' ']);
        % Storing the transition probability of TF
        fprintf(fid,[num2str(pha(1).transition(j).thread(1)),')))']);
        if j == tc
           fprintf(fid,'))');
        end
        n = n + 1;
    end
    % The next transition function number is 1 greater than last TF number
    st = en + 1;
end
% Close the file with file identifier FID
st = fclose(fid);
```