Moment-Linear Stochastic Systems and their Applications

by

Sandip Roy

B.S., University of Illinois at Urbana-Champaign (1998)

S.M., Massachusetts Institute of Technology (2000)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2003

Author_____
Department of Electrical Engineering and Computer Science
August 6, 2003

Certified by_____
George C. Verghese
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Certified by_____
Bernard C. Lesieutre
Staff Scientist, Lawrence Berkeley National Laboratory
Thesis Supervisor

Accepted by_____
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Moment-Linear Stochastic Systems and their Applications

by

Sandip Roy

Submitted to the Department of Electrical Engineering and Computer Science
on August 6, 2003, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Our work is motivated by the need for tractable stochastic models for complex network and system dynamics. With this motivation in mind, we develop a class of discrete-time Markov models, called moment-linear stochastic systems (MLSS), which are structured so that moments and cross-moments of the state variables can be computed efficiently, using linear recursions. We show that MLSS provide a common framework for representing and characterizing several models that are common in the literature, such as jump-linear systems, Markov-modulated Poisson processes, and infinite server queues. We also consider MLSS models for network interactions, and hence introduce moment-linear stochastic network (MLSN) models. Several potential applications for MLSN—in such areas as traffic flow modeling, queueing, and stochastic automata modeling—are explored. Further, we exploit the quasi-linear structure of MLSS and MLSN to analyze their asymptotic dynamics, and to construct linear minimum mean-square-error estimators and minimum quadratic cost controllers. Finally, we study in detail two examples of MLSN, a stochastic automaton called the influence model and an aggregate model for air traffic flows.

Thesis Supervisor: George C. Verghese
Title: Professor of Electrical Engineering and Computer Science

Thesis Supervisor: Bernard C. Lesieutre
Title: Staff Scientist, Lawrence Berkeley National Laboratory

Dedication

To my parents, Dilip and Sikha
and to my brother, Kaushik

Acknowledgements

First and foremost, I would like to my two amazing supervisors, George Verghese and Bernie Lesieutre. George has guided me in many aspects of my development as a Ph.D. student, not only by challenging me to pursue exciting and relevant research, but also by teaching me how to present and discuss this work in a precise and rigorous way. His dedication and love for problem solving, and for academic exploration, have rubbed off on me. Most of all, he has been a friend and mentor whenever I have needed one.

I have been working with Bernie since my first day at MIT: he has been a mentor and a good friend throughout my Master's and Ph.D. work, during journeys to Germany and New York and California, and in so much else that I have done at MIT. Bernie has always taken a lot of time to teach me interesting, exciting, and just plain cool things, and has challenged me to pursue problems that are both difficult and practical.

I am grateful to the remaining members of my thesis committee, John Tsitsiklis and Muriel Medard, for their valuable suggestions and encouragement. I learned much about probability theory, and about teaching, while working with John as a teaching assistant and interacting with him regarding the thesis. Muriel has introduced me to areas of research that I knew nothing about, and has been a kind mentor also.

I would especially like to recognize three others with whom I have been lucky enough to work. Banavar Sridhar, my mentor at NASA's Ames Research Center during Summer 2003, taught me much about Air Traffic Control, and then gave me the freedom to pursue questions of my interest. Chalee Asavathiratham preceded me as George's student; his work on the influence model, and our conversations together on many topics, have been invaluable. Carlos Gomez-Uribe and I have worked together closely during the past year. I have really enjoyed my many conversations, technical and otherwise, with Carlos. There are many other teachers and colleagues who have helped and guided me during my time at MIT. Thank you all.

The many kindly souls at the the Laboratory for Electromagnetic and Electronic Systems, my home base at MIT, have enriched and enlivened my life here. Vivian has helped me throughout, and Tushar has been a dear friend. Thanks also to Ernst and Tim, and all the others, for many interesting and amusing conversations.

A graduate student, like a child, needs a village to support him. My good friends at MIT and beyond have been that village for me. Andy, Yabei, Chi and Hsiang-wei, Anna, Dave, Laura, I am so grateful!

My work has been generously supported by the Department of Defense, the Air Force

Office of Scientific Research, and the Electric Power Research Institute.

Most importantly, thanks to my parents and my brother for their love, friendship, and support. All that I have achieved is because of you!

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Contributions

Large networks with complex, stochastic dynamics have become increasingly common in recent years, and the need for applicable models for these networks has grown concurrently. In this thesis, we develop a tractable class of structured discrete-time Markov models for network and system dynamics, and explore several applications and examples of the models. Our aim in considering these models is to capture the essence of some of the stochastic interactions that occur in interconnected systems, within a framework that allows considerable analysis of network dynamics, state estimation, and control.

1.1 Motivation

In today's interconnected world, events of interest can have far-reaching impact on multiple systems, and involve dynamics across several temporal and spatial scales. For instance, consider the Western United States power outage of August 10th, 1996. The initiating event for this failure, which affected approximately 8 million customers, was predicated by environmental conditions: extreme heat caused high demand and hence high power dissipation on power lines; this high power dissipation generated heat, causing a power line to sag into a tree (which was not appropriately trimmed) and short circuit. The ensuing propagating power failure was exacerbated by the high demand. Communications failures among power transmission companies also contributed to the systemic failure, in that these companies were unaware of existing statuses in the network and so were improperly prepared for the initiating event. The power failure affected traffic signals, and so snarled road traffic. Air traffic and data communications networks were also indirectly impacted. Meanwhile, some essential systems, such as hospitals, airports, and (on a lighter note) casinos, continued to operate using emergency generators. While the August 10th power outage consituted a spectacular, and rare, failure of multiple networks, the underlying connections within and among these networks clearly also impact their day-to-day operations.

The example above clarifies that interactions within, and among, multi-component systems are significant in understanding their global behavior. It further highlights that transient events in such networks are of importance, and that stochastics play a significant role in their behavior. In short, we see the need for stochastic, dynamic models for networks from which we can compute aspects of their global behavior. This thesis seeks to contribute to the stochastic modeling of network dynamics.

Modeling, and in particular stochastic modeling, of network dynamics is by no means a new undertaking. For instance, queueing networks have been extensively studied, and have found wide applicability in the context of, e.g., manufacturing and telecommunications (see [83] for an introduction). Linear and non-linear dynamical systems that are driven by noisy inputs, or are subject to parameter uncertainties, are prevalent in several contexts, including as models for power systems (see, e.g., [113] for an introduction to linear systems with stochastic inputs). Stochastic automata models have been used to represent spin dynamics in ferromagnetic materials, and as computational tools for optimizing certain multivariate costs (e.g., [18]). These three, and many other, stochastic models have proven to appropriately represent particular engineered and natural systems, and to be amenable to analyses that can be used to characterize and improve these systems.

While particular network models have been useful in several application areas, the large scale and multi-faceted nature of modern network interactions—such as those described in the introductory example of the Western states' power failure—have motivated simplified, and more general, approaches to modeling in recent years (see, e.g., [136, 27, 15]). These approaches seek to extend, or modify, the traditional work on network modeling by exposing global features of network dynamics that are common to many networked systems. More specifically, these works postulate very simple stochastic or deterministic interactions and/or particular network structures, and determine features of the global dynamics of such networks. The features that they observe in these toy models (e.g., the presence of heavy-tailed distributions for uncertain measures, the emergence of certain types of coherent dynamics, or the fragility of the network to particular uncommon failures) are also observed in various natural and engineered systems, lending some credence to these models. In a general sense, these models have the benefit of being potentially broad in their applicability, and simple or structured enough to allow for significant analysis of some global dynamics. On the flip side, the models often cannot capture the details of the interactions in specific networks, and their dynamics (and their analysis) are sometimes highly sensitive to particular features of the network graph and/or interaction structure.

We also are motivated by the need for models for multi-faceted network dynamics, from which interesting global characteristics of these dynamics can be inferred. It is our belief, however, that models for large-scale networks should, as much as possible, be derived from or matched with the actual interactions that occur among components of particular systems. Thus, we see value in modeling approaches that incorporate and generalize existing network and component models (e.g., certain queueing network and linear systems models), while concurrently enforcing sufficient structure on the dynamics to allow for global analysis. With this general motivation in mind, we aim to develop a class of network models for which we can analyze aspects of the dynamics and study some relevant questions of estimation and control. At the same time, we aim to show the applicability, or potential applicability, of our models by relating them to existing models and by considering particular applications for them.

1.2 An Introductory Example of DNA Evolution

The essential feature of our models that leads to their tractability is a quasi-linear structure in the interactions among network components. This structure allows us to partially characterize their dynamics at low computational cost, and to develop optimal estimators and controllers for the models. By the same token, this structure in the interactions limits the scope of the systems that we can model. It is helpful to pursue one toy example to illustrate how our models are structured and how this structure affords some special tractabilities.

Our example concerns a stochastic model for "DNA evolution" (in quotes, because we do not, and cannot, claim that the model we describe is a good representation for evolution; it is only meant to serve as a motivational example). For our purposes, a single strand of "DNA" consists of a sequence of 100 nucleotide bases. Each position along the sequence is one of four bases, adenine (A), cytosine (C), guanine (G), or tyrosine (T). We are concerned with tracking the base sequence of this strand of DNA over time, measured in generations. At a simplified level, we might expect that this strand of DNA changes with time through a probabilistic mutation process, perhaps a Markov process—i.e., one in which probabilities for transition (mutation) between the current time (generation) and the next time are completely specified by the current base sequence.

Let's consider several discrete-time Markov process representations for "DNA evolution". For each representation, we assume that the state of the system (the entire 100-base se-

quence) is known at an initial time, and we then attempt to find the probabilities that each base is present at each position at future times.

1. The most general representation for the state update is one for which probabilities can be specified arbitrarily for each transition between any two 100-base sequences (see Figure 1.1). Note that there are a total of 4^{100} possible 100-base sequences. Thus, $4^{100} \times 4^{100}$ probabilities are needed to model all transitions in this general representation. Even if the transition probabilities could somehow be specified, the exact analysis of the model is daunting. To compute the probability that a particular position has a particular base at a time k, it turns out that we need to compute the probabilities that each of the 4^{100} sequences is present at each time between the initial time and k. This calculation requires us to multiply a matrix of dimension $4^{100} \times 4^{100}$ with a 4^{100}-component vector for k times! This calculation in infeasibile, suggesting that we should seek structured representations which simplify analysis.

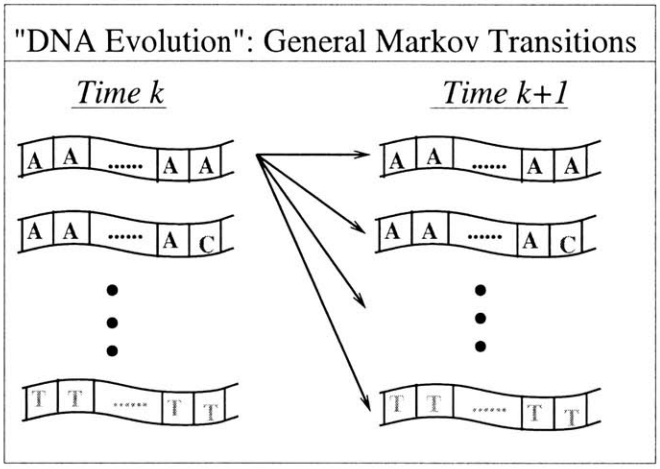


Figure 1.1: A general Markov description for the evolution of a 100-base DNA sequence is illustrated.

2. A simple Markov process representation for "DNA evolution" is one in which the base at each position evolves independently (see Figure 1.2). In this case, the transitions at each position are governed by independent Markov processes. Thus, the probability that a position has a particular base at a time k can be obtained from the initial probabilities that each base is present at this position, separately from the remainder of the sequence. This calculation requires multiplication of a 4×4 matrix with a 4-component

vector for k times. Even when this computation is repeated for all 100 positions on the chain, the required computation is small. Furthermore, because each position evolves independently, joint probabilities for seqence and subsequence configurations can be found immediately by multiplying the individual probabilities (assuming that the bases at each position are initially independent). While computation is significantly reduced by assuming each position evolves independently, the applicability of the model is also severely limited: no dependencies among the positions in the DNA chain can be modeled if the sites are to evolve independently. In fact, models for DNA in which each site evolves independently have been developed (e.g., [68]). As discussed in [8], however, such models cannot capture the dependencies among bases that are observed in real DNA sequences.

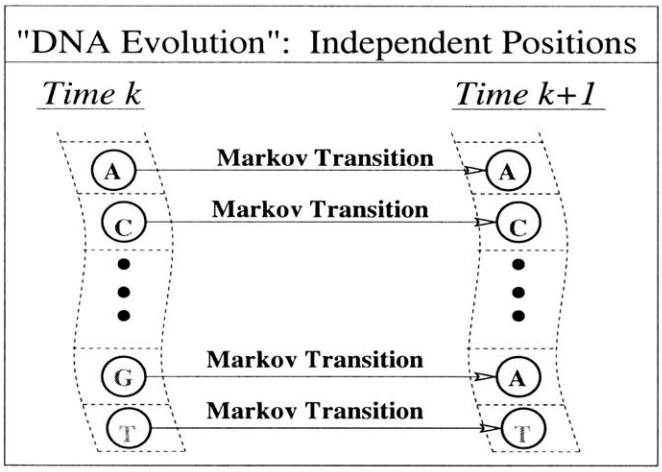


Figure 1.2: Independent evolution of each base is illustrated.

3. We wish to model DNA evolution in such a way that probabilities for all sequence configurations need not be tracked, yet each position is not constrained to evolve independently. We might be tempted to do so by constraining the probabilities for the base at a particular position at the next time to depend only on the current base at that position and at the (in general) two neighboring positions (see Figure 1.3). Surprisingly, such a *spatial structure* in the network interactions does not generally simplify the analysis of base probabilities of individual positions, because the base probabilities for a single position at a particular time depend upon the bases at more and more positions, as one looks further back in time. That is, to compute the probability that a particular position has a certain base a time k, we need to compute the joint probabilities for the bases

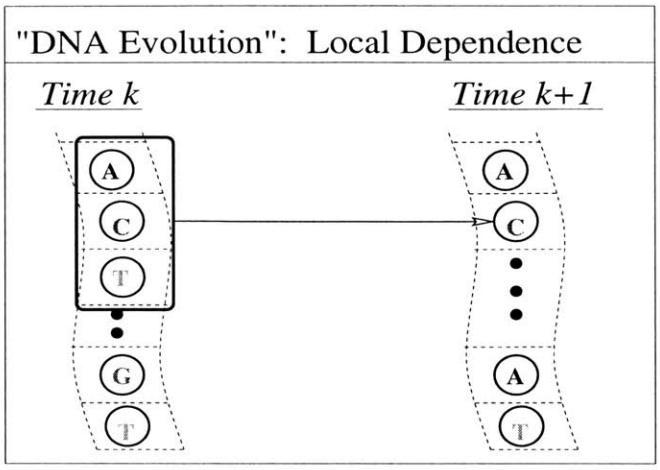


Figure 1.3: The evolution of each base position during the transition to the next generation is governed by neighboring bases at the current generation.

of three neighboring sites at time $k-1$, for five neighboring sites at time $k-2$, etc. (see Figure 1.4). Thus, while simplifying the network structure reduces the number of parameters in the model significantly, it may not simplify its analysis. Interestingly, a model for DNA evolution with localized interactions has been studied in [8]. To analyze this model, the researchers use an approximation scheme which exploits the (potential) weak dependence of spatially separated bases. While such approximations can be used to simplify analysis of spatially structured models, there is no general guarantee that close approximations can be found.

4. Our main conceptual insight is that tractability of stochastic network dynamics can be guaranteed by constraining the *form* of the stochastic interactions in a model, rather than the *graphical structure* or the *strength* of the interactions in the model. For instance, we can constrain neighbor-dependent base evolution so as to ensure tractability, as follows: each position can be modeled as selecting one of its neighbors (possibly including itself) with some probability at each time-step; the current base at the selected neighbor is then assumed to specify probabilities for the next base at the selecting position. With these constraints on the interactions, it turns out we can compute probabilities for the base at each position at a time k as a *linear function* of individual base probabilities (at that, and at neighboring, positions) at time $k-1$. Thus, we can compute these time-k probabilities from individual base probabilities at any past time using a linear recursion. Further, we find that the computational cost required for this analysis is low: it is quadratic, not exponential, in the length of the sequence. If joint probabilities of multiple bases—i.e.,

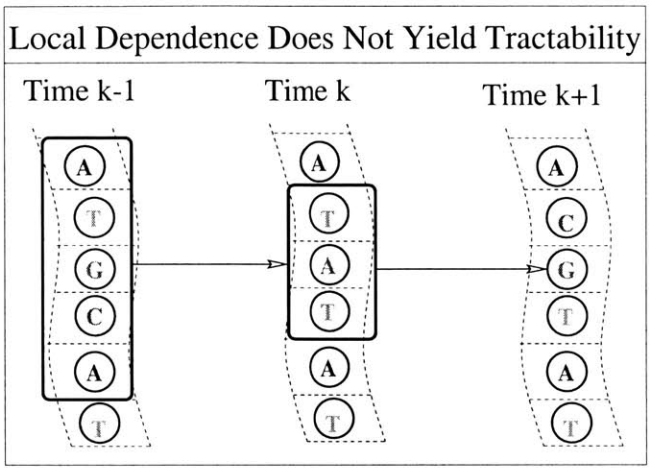


Figure 1.4: We depict that evolution with local Markov dependences is not necessarily tractable. The joint statistics of multiple base positions must be considered at previous times to analyze individual base position statistics at the current time.

sequence configuration probabilities—are required, it turns out that the computation grows gracefully with the number of bases whose joint probabilities are needed. We note that the special analysis of the described model has no essential connection with its graph structure; we can analyze the model for any interaction structure, as long as these interactions are structured as described above.

Like the example introduced above, the various models that we consider in the thesis are specially structured so that temporal dynamics can be partially characterized using linear recursions.

1.3 List of Contributions

1. We introduce a discrete-time Markov model that is structured so that statistics of state variables can be found efficiently using linear recursions. We show that this model, which we call a *moment-linear stochastic system* (MLSS) provides a common representation for such diverse models as jump-linear systems, Markov-modulated Poisson processes, and memoryless infinite-server queues, as well as variants of these with stochastic parameters.

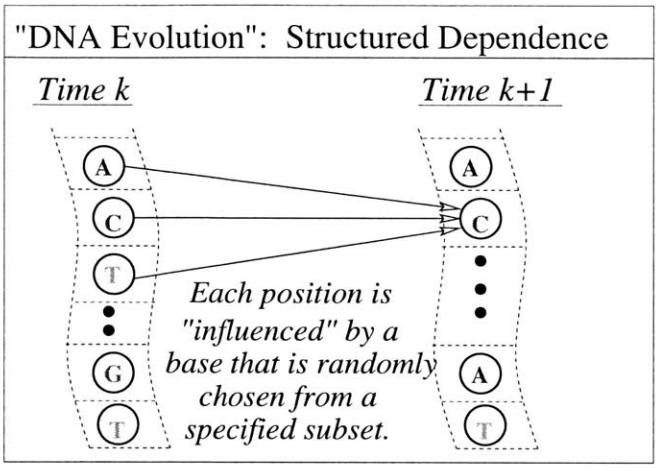


Figure 1.5: If the evolution probabilities are specially structured, then partial information about each base position can indeed be found with low computational cost. This figure depicts one such structure: in particular, each base position is updated by selecting one neighboring base randomly, which then specifies probabilities for the next base at the position.

2. Through examples, we expose a variety of network interactions that can be captured using MLSS and show that MLSS can be a keen tool for characterizing stochastic network dynamics. Our examples are drawn from a wide range of contexts, including queueing network theory, stochastic automata modeling, and traffic modeling.

3. We extensively analyze the dynamics of MLSS, and use these analyses to provide fresh insights into the dynamics of several examples. We also develop and apply methodologies for optimal linear estimation and quadratic control in the context of MLSS.

4. We formulate the influence model (originally introduced in [9]) as an MLSS, and use the MLSS formulation to develop some new results concerning this model.

5. We develop an MLSS model for aircraft counts in regions of an Air Traffic System, and evaluate the performance of the model using historical data.

1.4 Thesis Outline

The thesis is organized as follows:

- In Chapter 2, we introduce the general class of structured Markov models that is the central topic of the thesis. We call these models *moment-linear stochastic systems* (MLSS). We develop the basic recursive analysis of MLSS dynamics, and present several common models from the literature that can be represented as MLSS. We also briefly discuss why the developed analysis is, in some respect, unique to MLSS.

- In Chapter 3, we consider the use of MLSS as models for network dynamics, and hence informally introduce *moment-linear stochastic networks* (MLSN). We introduce several examples of MLSN, and discuss a block representation for MLSS dynamics that is relevant in the analysis of some networked systems. We also discuss the use of MLSN to represent flows, in particular.

- In Chapter 4, MLSS dynamics are analyzed further. We focus in particular on characterizing cross-time-step dynamics and asymptotics.

- In Chapter 5, we derive the *linear minimum mean square error* (LMMSE) estimate for the state of an MLSS from a temporal sequence of imperfect observations.

- In Chapter 6, we study minimum-quadratic-cost dynamic control of an MLSS.

- Chapter 7 contains a case study of a particular MLSN, the *influence model*. We summarize the basic development of the influence model, which was introduced in [9] as a network of interacting Markov chains, from the perspective of MLSN models. We also present some results that extend the analysis given in [9].

- In Chapter 8, we construct an MLSN representation for the aggregate dynamics of an *air traffic system* (ATS). Through this example, we aim to show how MLSN can be used to model certain real systems.

- In Chapter 9, we draw some general conclusions about our work, and suggest directions for future study.

Our development makes use of some basic probabilistic and linear-algebraic concepts. The reader is referred to Appendix A for a background summary.

Chapter 2

Moment-Linear Stochastic Systems: Introduction

In this chapter, we formulate the definition of a *moment-linear stochastic system* (MLSS), the quasi-linear discrete-time Markov model that is the primary topic of this thesis. We then describe the model's basic analysis—i.e., we exploit its special quasi-linear structure to construct linear recursions for statistics (moments and cross-moments) of state variables. We also briefly explore the scope of MLSS, by showing that the described analyses are unique to MLSS, in a certain respect. The remainder of the chapter is devoted to five examples of MLSS. These examples are drawn from prevalent models in the literature, and so they show the broad applicability of MLSS.

2.1 Formulation

An MLSS is a discrete-time Markov model in which the conditional distributions for the next state given the current state are specially constrained at each time-step. In particular, these conditional distributions are structured so that moments and cross-moments of state variables at each time-step can be found as linear functions of equal and lower moments and cross-moments of state variables at the previous time-step. In consequence, we can find these moments and cross-moments using linear recursions. In the following discussion, we precisely define an MLSS and highlight the special structure of the model that leads to its tractability.

Formally, consider a discrete-time Markov process with m-component real state vector. The state (i.e., state vector) of the process at a time k is denoted $\mathrm{s}[k]$, and the notation $\{\mathrm{s}[k]\}$ is used to represent the state sequence $\mathrm{s}[0]$, $\mathrm{s}[1]$, ... We use the notation $s_i[k]$ to denote the ith component of the state vector at time k.

Since $\{\mathbf{s}[k]\}$ is a Markov process, it is completely specified by the distribution for the initial state $\mathbf{s}[0]$, as well as the conditional distribution for the *next state* $\mathbf{s}[k+1]$ given the *current state* $\mathbf{s}[k]$ at each time k. An MLSS is a Markov process for which these conditional distributions are specially constrained.

Specifically, consider the conditional expectation $E(\mathbf{s}[k+1]^{\otimes r} \mid \mathbf{s}[k])$, for $r = 1, 2, \ldots$, where the notation $\mathbf{s}[k+1]^{\otimes r}$ refers to the Kronecker product of the vector $\mathbf{s}[k+1]$ with itself r times and is termed the rth-*order state vector* at time k. This expectation vector contains all rth moments and cross-moments of the state variables $s_1[k+1], \ldots, s_n[k+1]$ given $\mathbf{s}[k]$, and so we call the vector the *conditional* rth *(vector) moment* for $\mathbf{s}[k+1]$ given $\mathbf{s}[k]$. We say that the process $\{\mathbf{s}[k]\}$ is rth-*moment linear* at time k if the conditional rth moment for $\mathbf{s}[k+1]$ given $\mathbf{s}[k]$ can be written as follows:

$$E(\mathbf{s}[k+1]^{\otimes r} \mid \mathbf{s}[k]) = H_{r,0}[k] + \sum_{i=1}^{r} H_{r,i}[k]\mathbf{s}[k]^{\otimes i}, \tag{2.1}$$

for some set of matrices $H_{r,0}[k], \ldots, H_{r,r}[k]$.

The Markov process $\{\mathbf{s}[k]\}$ is called a *moment-linear stochastic system* (MLSS) of degree $\widehat{r}$ if it is rth-moment linear for all $r \leq \widehat{r}$, and for all times k. If a Markov model is moment linear for all r and k, we simply call the model an MLSS. We call the constraint (2.1) the rth-*moment linearity condition* at time k, and call the matrices $H_{r,0}[k], \ldots, H_{r,r}[k]$ the rth-*moment recursion matrices* at time k. These recursion matrices feature prominently in our analysis of the temporal evolution of MLSS.

Although it is natural to formulate an MLSS in the vector notation presented above, some insight into its structure can be gained by considering conditional moments and cross-moments of individual state variables separately rather than in vector notation. To do so, consider one of the elements in the vector $E(\mathbf{s}[k+1]^{\otimes r} \mid \mathbf{s}[k])$; from our definition for an MLSS, this conditional moment or cross-moment (of time $k+1$ state variables given the time k state) can be written as an affine function of the first r Kronecker powers of the conditioned state $\mathbf{s}[k]$. However, note that an affine function of the first r Kronecker products of the state $\mathbf{s}[k]$ is simply an rth-degree polynomial of the time k state variables. Thus, a Markov process is an MLSS of degree $\widehat{r}$ if all rth conditional moments and cross-moments of time $k+1$ state variables given the time k state can be written as rth-degree polynomials of the time k state variables, for all $r \leq \widehat{r}$ and for all k.

2.1.1 Time-Invariant MLSS

A time-invariant Markov process that is an MLSS of degree $\widehat{r}$ is called a *time-invariant MLSS* of degree $\widehat{r}$. For a time-invariant MLSS (of degree $\widehat{r}$), the conditional rth moment for $\mathbf{s}[k+1]$ given $\mathbf{s}[k]$ is identical at each time k (for $r \leq \widehat{r}$). As a consequence, the rth-moment linearity condition can be written in the form

$$E(\mathbf{s}[k+1]^{\otimes r} \mid \mathbf{s}[k]) = H_{r,0} + \sum_{i=1}^{r} H_{r,i}\mathbf{s}[k]^{\otimes i}, \tag{2.2}$$

for some set of matrices $H_{r,0}, \ldots, H_{r,r}$.

To illustrate our formulation of MLSS, we present two simple examples of (time-invariant) MLSS:

Example 2.1
Consider the discrete-time Markov model with state update

$$\begin{aligned} \mathbf{s}[k+1] &= A_1\mathbf{s}[k], \quad w.p.\ p_1 \\ \mathbf{s}[k+1] &= A_2\mathbf{s}[k], \quad w.p.\ p_2 = 1 - p_1. \end{aligned} \tag{2.3}$$

To compute the conditional vector moments for $\mathbf{s}[k+1]$ *given* $\mathbf{s}[k]$*, let's define a random variable* $\sigma[k]$ *that equals* 1 *if* $\mathbf{s}[k+1] = A_1\mathbf{s}[k]$ *and equals* 2 *if* $\mathbf{s}[k+1] = A_2\mathbf{s}[k]$*. Conditioning on* $\sigma[k]$*, we can write the* r*th conditional vector moment as*

$$\begin{aligned} E(\mathbf{s}[k+1]^{\otimes r} \mid \mathbf{s}[k]) &= E(\mathbf{s}[k+1]^{\otimes r} \mid \mathbf{s}[k], \sigma[k] = 1)p_1 + E(\mathbf{s}[k+1]^{\otimes r} \mid \mathbf{s}[k], \sigma[k] = 2)p_2 \\ &= p_1(A_1\mathbf{s}[k])^{\otimes r} + p_2(A_2\mathbf{s}[k])^{\otimes r} \\ &= (p_1A_1^{\otimes r} + p_2A_2^{\otimes r})\mathbf{s}[k]^{\otimes r}, \end{aligned} \tag{2.4}$$

where we have invoked the mixed-product property of the Kronecker product in the final step ([72]). Thus, this model satisfies the r*th-moment linearity condition for all* r*, and so the model is an MLSS.*

Example 2.2
Consider the scalar Markov model with state update

$$s[k+1] = x[k]g(s[k]), \tag{2.5}$$

where $x[k]$ is an independent Normal (Gaussian) random variable with mean 0 and variance 1, and $g()$ is an arbitrary non-linear function.

Then note that $E(s[k+1] \mid s[k]) = E(x[k])g(s[k]) = 0$, so the model satisfies the first moment-linearity condition (with $H_{1,1} = H_{1,0} = 0$). However, $E(s[k+1]^2 \mid s[k]) = E(x^2[k])g^2(s[k]) = g^2(s[k])$. Thus, for arbitrary $g()$, $E(s[k+1]^2 \mid s[k])$ is not a quadratic function of $s[k]$, and so the second moment-linearity condition is not satisfied. This system is therefore an MLSS of degree 1.

Incidentally, if $g(s) = s$ or $g(s) = |s|$, it is straightforward to check that the all moment-linearity conditions are satisfied.

2.2 Basic Analysis

MLSS are amenable to analysis, in that we can find statistics of the state $\mathbf{s}[k]$ (i.e., moments and cross-moments of state variables) using linear recursions. In particular, for an MLSS of degree $\widehat{r}$, $E(\mathbf{s}[k+1]^{\otimes r})$ (called the rth *moment* of $\mathbf{s}[k+1]$) can be found in terms of the first r moments of $\mathbf{s}[k]$ for any $r \leq \widehat{r}$. To find these rth moments, we use the law of iterated expectations and then invoke the rth-moment linearity condition:

$$\begin{aligned} E(\mathbf{s}[k+1]^{\otimes r}) &= E(E(\mathbf{s}[k+1]^{\otimes r} \mid \mathbf{s}[k])) \\ &= E(H_{r,0}[k] + \sum_{i=1}^{r} H_{r,i}[k]\mathbf{s}[k]^{\otimes i}) \\ &= H_{r,0}[k] + \sum_{i=1}^{r} H_{r,i}[k]E(\mathbf{s}[k]^{\otimes i}). \end{aligned} \tag{2.6}$$

We call Equation 2.6 the rth-*moment recursion* at time k. Considering equations of the form 2.6, we see that the first r moments of $\mathbf{s}[k+1]$ can be found as a linear function of the first r moments of $\mathbf{s}[k]$. Thus, by applying the moment recursions iteratively, the rth moment of $\mathbf{s}[k]$ can be written in terms of the first r moments of the initial state $\mathbf{s}[0]$.

2.2.1 A Concise Representation for the Moment Recursions

The recursions developed in equations of the form 2.6 can be rewritten in a more concise form, by stacking rth and lower moment vectors into a single vector. In particular, we

define the rth *extended state vector* to be $\mathbf{s}'_{(r)}[k] = \begin{bmatrix} \mathbf{s}'[k]^{\otimes r} & \dots & \mathbf{s}'[k]^{\otimes 1} & 1 \end{bmatrix}$. The rth *extended moment vector* at a time $k+1$, or $E(\mathbf{s}_{(r)}[k+1])$, can be written in terms of the extended moment vector at time k by assimilating the first r moment recursions at time k (given in Equation 2.6) into a single recursion. In particular, we find that

$$E(\mathbf{s}_{(r)}[k+1]) = \tilde{H}_{(r)}[k]E(\mathbf{s}_{(r)}[k]), \tag{2.7}$$

where

$$\tilde{H}_{(r)}[k] = \begin{bmatrix} H_{r,r}[k] & H_{r,r-1}[k] & \dots & H_{r,1}[k] & H_{r,0}[k] \\ 0 & H_{r-1,r-1}[k] & \dots & H_{r-1,1}[k] & H_{r-1,0}[k] \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & H_{1,1}[k] & H_{1,0}[k] \\ 0 & \dots & 0 & 0 & 1 \end{bmatrix}.$$

We call Equation 2.7 the rth *extended moment recursion* at time k.

The rth extended moment recursion allows us to explicitly specify the rth extended vector moments at time k in terms of the initial rth extended vector moments, as $E(\mathbf{s}_{(r)}[k]) = \prod_{j=0}^{k-1} \tilde{H}_{(r)}[j]\mathbf{s}_{(r)}[0]$. For a time-invariant MLSS, the explicit expression for the rth extended vector moment at time k becomes $E(\mathbf{s}_{(r)}[k]) = \tilde{H}^k_{(r)}\mathbf{s}_{(r)}[0]$.

2.2.2 A Note on Redundancy in the Vector Moments

In our analysis of MLSS so far, we have focused on characterizing moments (and conditional moments) of the state—i.e., expectations of self-Kronecker products of the state vector. These vector moments are redundant, in that the same cross-moments of state variables are repeated multiple times in a vector moment. This redundancy in the vector moments has two somewhat disconcerting effects:

- It enforces a certain structure in the moment recursion matrices, since the moment linearity condition relating the vector moments at one time-step with the vector moments at the previous time-step must generate the same values for the redundant cross-moments.
- By the same token, it introduces degeneracy in the moment recursions, since the moments and cross-moments of state variables at one time-step can be written equivalently in terms of multiple identical cross-moments at the previous time-step.

This redundancy typically does not hamper our efforts to find vector moments and cross-moments using the basic analysis of MLSS. In most examples, one valid set of recursion matrices can be constructed from the stochastic description of the state vector; these recursion matrices can then be used to determine moments and cross-moments. It is irrelevant that other sets of recursion matrices could be used to generate the same statistics, or that the recursion matrices are structured.

However, some analyses that we pursue later in the thesis (relating, for example, to asymptotics of MLSS) are impacted by the degeneracies caused by vector moment degeneracies. In these analyses, we reformulate the moment recursions so as to eliminate some or all of these redundancies. The mappings among various forms of the moment recursions are formalized in Appendix B.

We note that redundancies in the vector moments are not the only features of MLSS that can lead to degeneracy in the moment recursions. In particular MLSS, constraints on state variables can also result in degeneracy in the moment recursions. Such degeneracies are described for one of the example MLSS introduced in Section 2.4.

The reader may wonder why we have chosen to write the higher moment recursions in terms of extended moment vectors, which are redundant. We have used this formulation because many systems of interest to us can be conveniently and intuitively represented in this notation, as will become clear in the examples in Section 2.4.

2.2.3 Complexity of the Moment Recursions

Here, we discuss the order of the computation required to determine the rth moment of the state at time k from the first r moments at the initial time 0, for an MLSS with m-component state vector.

For general MLSS, we are required to find this moment vector using k iterations of the extended moment recursion. Each of these iterations requires multiplication of an rth extended moment vector with an extended moment recursion matrix. Since the rth extended moment vector has $\sum_{i=0}^{r} m^i$ entries, each vector-matrix multiply requires on the order of $(\sum_{i=0}^{r} m^i)^2$ additions and multiplications. (In fact, about one-half of the entries in the extended moment recursions are necessarily 0, but these zero entries do not change the order of the computation.) Thus, the numbers of multiplications and additions required to find

the rth moment at time k are on the order of $k(\sum_{i=0}^{r} m^i)^2$.

To gain a bit more intuition regarding the complexity of the rth moment computation, note that $\sum_{i=0}^{r} m^i$ is close to m^r for large m. (More precisely, it is upper bounded by $\frac{m}{m-1}m^r$.) Thus, the required number of additions and multiplications is on the order of km^{2r}. That is, the computation roughly grows linearly with k, polynomially with m, and exponentially with r.

We note that the complexity analysis described above is a worst case analysis. For some MLSS, structural features of the model (e.g., constraints on the form of the recursion matrices, time-invariance) can be exploited to reduce the computation required to find moments and cross-moments.

Example 2.3

This example illustrates the basic analysis of MLSS. Consider the scalar system with state update

$$s[k+1] = x[k] + 1, \tag{2.8}$$

where $x[k]$ is uniformly distributed between 0 and $s[k]$. We can straightforwardly check that the first two moment linearity conditions for this MLSS are $E(s[k+1] \mid s[k]) = \frac{1}{2}s[k]+1$ and $E(s[k+1]^2 \mid s[k]) = \frac{1}{3}s^2[k] + \frac{1}{2}s[k] + \frac{1}{4}$. (We can straightforwardly show that higher moment linearity conditions also hold, but omit these conditions from our discussion here.) Using the basic analysis of MLSS, we thus see that the first and second moment recursions are given by

$$\begin{aligned} E(s[k+1]) &= \frac{1}{2}E(s[k]) + 1 \\ E(s[k+1]^2) &= \frac{1}{3}E(s^2[k]) + \frac{1}{2}E(s[k]) + \frac{1}{4}. \end{aligned} \tag{2.9}$$

Assembling the first and second moment recursions into a single update, we find that the second extended moment recursion is given by

$$E(\mathbf{s}_{(2)}[k+1]) = \begin{bmatrix} \frac{1}{3} & \frac{1}{2} & \frac{1}{4} \\ 0 & \frac{1}{2} & 1 \\ 0 & 0 & 1 \end{bmatrix} E(\mathbf{s}_{(2)}[k]), \tag{2.10}$$

where $\mathbf{s}_{(2)}[k] = \begin{bmatrix} s^2[k] \\ s[k] \\ 1 \end{bmatrix}$

Example 2.4

This example illustrates how redundancies in a second-moment vector impact the formulation of the second-moment recursion. We consider an MLSS with 2-component state vector $\mathbf{s}[k]$. *For simplicity, we assume that the second-moment linearity condition for this MLSS has the form* $E(\mathbf{s}[k+1]^{\otimes 2} \mid \mathbf{s}[k]) = H_{2,2}\mathbf{s}[k]$. *(The MLSS in Example 2.1 has a second-moment linearity condition of this form.)*

Note that the second moment vector at time k *is*

$$E(s[k]^{\otimes 2}) = E\left(\begin{bmatrix} s_1^2[k] \\ s_1[k]s_2[k] \\ s_2[k]s_1[k] \\ s_2^2[k] \end{bmatrix}\right). \tag{2.11}$$

Thus, the second and third entries of the second moment vector are identical.

To see how this redundancy enforces a structure on the second moment vector, note that the product $[0,1,-1,0]E(\mathbf{s}[k+1]^{\otimes 2})$ *equals* 0. *Thus, by applying the second-moment linearity condition, we see that*

$$[0,1,-1,0]H_{2,2}\mathbf{s}[k]^{\otimes 2} = 0. \tag{2.12}$$

However, the product in Equation 2.12 is guaranteed to be 0 *in general*[1] *only if* $[0,1,-1,0]H_{2,2}$ *is a vector of the form* $[0,c,-c,0]$. *Thus, it is required that* $[0,1,-1,0]$ *is a left eigenvector of* $H_{2,2}$.

We can also check that the second-moment recursion matrix for this MLSS is degenerate. In particular, any matrix of the form $H_{2,2} + \mathbf{v} \otimes [0,1,-1,0]$, *where* $\mathbf{v}$ *is any length-4 column vector, is a valid second moment recursion matrix, since* $(H_{2,2} + \mathbf{v} \otimes [0,1,-1,0])\mathbf{s}[k]^{\otimes 2}$ *equals* $H_{2,2}\mathbf{s}[k]^{\otimes 2}$.

2.3 The Scope of MLSS

Throughout the thesis, we aim to show the significant tractability of MLSS, and to explore the insights into stochastic dynamics that are gained because of this tractability. Just as importantly, however, we need to understand what features of these models—and of the

[1] *We say "in general" because in particular examples the state vector* $\mathbf{s}[k]$ *may be constrained in such a way that the product in Equation 2.12 can be shown to equal zero without the condition described here.*

underlying physical processes that they may describe—allow for this tractability. One approach for understanding the scope of the moment-linear analysis is to consider many examples of the models, as we do at several points in the thesis. However, we believe it is useful also to look at the scope of MLSS from a theoretical viewpoint, by trying to characterize the set of models that are MLSS or are similarly tractable to MLSS in a general way.

To this end, we describe a rudimentary study that begins to explore the following question: what is the broadest class of Markov models for which rth moments and cross-moments of state variables at a given time can be found in terms of rth and lower moments and cross-moments at the previous time-step? Our discussion only scratches the surface of the range of questions that could be asked regarding the scope of MLSS, but we hope that it will give the reader a first indication of both the breadth and the limitations of our models.

2.3.1 A Rudimentary Study: Scope of the First-Moment Recursion

A special property of MLSS is that that expected value of $\mathbf{s}[k+1]$ can be computed from just the expected value of $\mathbf{s}[k]$—no further information about the distribution of $\mathbf{s}[k]$ is needed. It turns out that, for Markovian systems, this special tractability is unique to systems that satisfy a first-moment linearity condition. More precisely, say that $E(\mathbf{s}[k])$ is known, but the distribution $\mathbf{s}[k]$ is otherwise unknown. Then $E(\mathbf{s}[k+1])$ can be computed if and only if the conditional expectation $E(\mathbf{s}[k+1] \,|\, \mathbf{s}[k])$ is an affine function of $\mathbf{s}[k]$.

We have already proven the forward (if) direction of this statement, in developing the MLSS moment recursions. To prove the reverse (only if) direction, we consider computing $E(\mathbf{s}[k+1])$ for various distributions of $\mathbf{s}[k]$ with mean $E(\mathbf{s}[k])$. In order for $E(\mathbf{s}[k+1])$ to be computable from $E(\mathbf{s}[k])$, it is required that the mean of $\mathbf{s}[k+1]$ computed for each such distribution of $\mathbf{s}[k]$ is identical. It turns out that this requirement forces the conditional expectation $E(\mathbf{s}[k+1] \,|\, \mathbf{s}[k])$ to be affine with respect to $\mathbf{s}[k]$. For the sake of clarity, we only describe the proof for a scalar Markov process $s[k]$ here, and give an outline of the proof in the general case in Appendix C.

Outline of Proof for the Scalar Case

Consider a Markov process with scalar state $s[k]$, and assume that $E(s[k])$ is known for some k. For clarity, use the notation $\bar{s}$ for this expectation. Also, assume that $E(s[k+1])$ can be computed from $E(s[k])$. Then we can make the following deductions:

- Consider a plot of $E(s[k+1] \mid s[k])$ as a function of $s[k]$, and consider three points on this curve: $E(s[k+1] \mid s[k] = \bar{s} - 1)$, $E(s[k+1] \mid s[k] = \bar{s})$, and $E(s[k+1] \mid s[k] = \bar{s} + 1)$. These three points are collinear (Figure 2.1). To see why, consider two possible distributions for $s[k]$. First, say that $s[k]$ equals $\bar{s}$ with probability 1. Then $E(s[k]) = \bar{s}$ and, from the law of iterated expectations, $E(s[k+1]) = E(s[k+1] \mid s[k] = \bar{s})$. Second, say that $s[k]$ equals $\bar{s} - 1$, with probability $\frac{1}{2}$, or equals $\bar{s} + 1$, with probability $\frac{1}{2}$. Then $E(s[k]) = \bar{s}$ and, from the law of iterated expectations, $E(s[k+1]) = \frac{1}{2}E(s[k+1] \mid s[k] = \bar{s} - 1) + \frac{1}{2}E(s[k+1] \mid s[k] = \bar{s} + 1)$. Since $E(s[k+1])$ must be equal for both distributions of $s[k]$ in order for $E(s[k+1])$ to be computable from $E(s[k])$, we see that $E(s[k+1] \mid s[k] = \bar{s}) = \frac{1}{2}E(s[k+1] \mid s[k] = \bar{s} - 1) + \frac{1}{2}E(s[k+1] \mid s[k] = \bar{s} + 1)$. Hence, the three points are collinear: $E(s[k+1] \mid s[k]) = hs[k] + b$ for some h and b.

- We aim to show that for all $\beta > 0$, $E(s[k+1] \mid s[k] = \bar{s} + \beta) = h(\bar{s} + \beta) + b$—that is, that all points $E(s[k+1] \mid s[k] = \bar{s} + \beta)$ (to the right of $E(s[k+1] \mid s[k] = \bar{s})$) fall on the same line as three collinear points (Figure 2.2). To do so, let's consider another possible distribution for $s[k]$. Say that $s[k]$ equals $\bar{s} - 1$, with probability $\frac{\beta}{\beta+1}$, and equals $\bar{s} + \beta$ with probability $\frac{1}{\beta+1}$. It is straightforward to check that $E(s[k]) = \bar{s}$, and that $E(s[k+1]) = \frac{\beta}{\beta+1}E(s[k+1] \mid s[k] = \bar{s} - 1) + \frac{1}{\beta+1}E(s[k+1] \mid s[k] = \bar{s} + \beta)$. In order for $E(s[k+1])$ to be computable for $E(s[k])$, we thus require that $E(s[k+1] \mid s[k] = \bar{s}) = \frac{\beta}{\beta+1}E(s[k+1] \mid s[k] = \bar{s} - 1) + \frac{1}{\beta+1}E(s[k+1] \mid s[k] = \bar{s} + \beta)$. Finally, by substituting $h\bar{s} + b$ for $E(s[k+1] \mid s[k] = \bar{s})$ and $h(\bar{s} - 1) + b$ for $E(s[k+1] \mid s[k] = \bar{s} - 1)$, and then doing some algebra, we see that $E(s[k+1] \mid s[k] = \bar{s} + \beta) = h(\bar{s} + \beta) + b$.

- We can similarly prove that the points on the curve to the left of $E(s[k+1] \mid s[k] = \bar{s})$ must fall on the same line.

Hence, we have proven that $E(s[k+1] \mid s[k])$must take the form $hs[k] + b$, for $E(s[k+1])$ to be computable from $E(s[k])$.

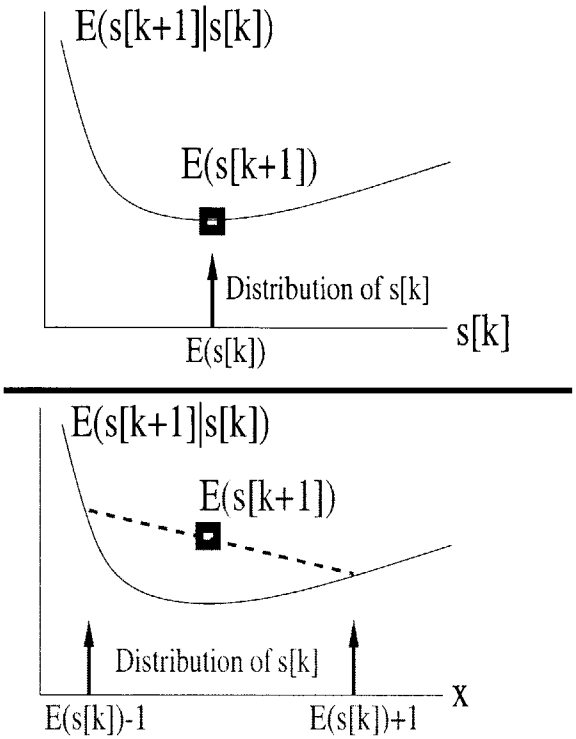


Figure 2.1: $E(s[k+1])$ cannot be computed from $E(s[k])$, unless $E(s[k+1] \mid s[k] = E(s[k]))$, $E(s[k+1] \mid s[k] = E(s[k]) - 1)$, and $E(s[k+1] \mid s[k] = E(s[k]) + 1)$ are collinear, because otherwise the mean of $s[k+1]$ will depend on the particular distribution for $s[k]$.

Discussion

Our result is interesting because it characterizes the class of discrete-time Markov models for which the expected state at each time $k+1$ can be calculated from the expected state at time k (in any manner, linear or not). We have shown that the characteristic of the update law necessary to generally allow this mean value calculation is that the conditional expectation for the state at each time $k+1$ given the state at time k is an affine function of the state at time k, *which is precisely the first-moment linearity condition of an MLSS.*

We caution that our result requires some refinement, and generalization, before it can provide insight into the tractability (or lack thereof) of various classes of models. First, we would like to be able to relax the strong assumption that the distribution of s[k] is completely unknown except its mean. For example, we would like a result that applies to models in which the state space is naturally constrained (e.g., queueing models, in which

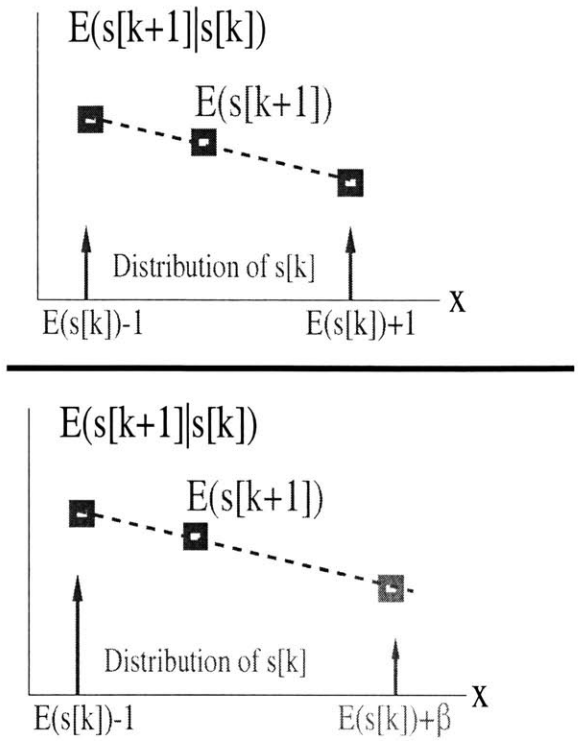


Figure 2.2: By choosing different distributions for $s[k]$, we can verify that the remaining points on $E(s[k+1]|s[k])$ must lie on the same line as the original three collinear points, if $E(s[k+1])$ is computable from $E(s[k])$.

the state variables must be integral). Second, we must understand better how higher moments play a role in the computability of next time-step statistics. Third, we must study how results on computability or required computational cost at one time-step translate to results on the analysis of the entire state sequence dynamics, or of the asympotics. (For instance, the steady-state probability mass function for *a dynamic Ising model*—a type of finite state stochastic automaton defined on a network—can be determined from pairwise interactions among sites, even though characterization of transient dynamics requires computation involving the joint configurations of all sites [137]). The following diabolical example also verifies that asymptotics can sometimes be determined even if one-step prediction of the mean is impossible:

Example 2.5
Consider the scalar system with the following update:

$$s[k+1] = 2, \quad \text{if } 3 < s[k] < 4 \tag{2.13}$$
$$s[k+1] = 1, \text{otherwise.}$$

Let's say that the mean of the initial state $s[0]$ is known, but its distribution is otherwise unknown. From the theorem above, we know that the mean of $s[1]$ cannot be uniquely determined. However, regardless of the distribution of $s[0]$, the state for $k \geq 2$ equals 1 with probability 1 for this model. Thus, though the mean of $s[0]$ is not sufficient to determine the mean of $s[1]$, it is sufficient to determine the mean of $s[k]$, $k \geq 2$.

In this section, we have focused on characterizing the scope of MLSS, by showing that the models are exclusive in some sense—i.e., that only models satisfying moment linearity conditions (in particular, the first-moment linearity condition) are amenable to the analyses that we describe for MLSS. Another important step in delineating the scope of our modeling framework is to describe the types of interactions that can be represented as MLSS updates. While the many examples of MLSS that we describe suggest the types of interactions that can be represented, it would be interesting to explore whether all MLSS interactions generally have some qualitative similarities. We leave this study for future work.

2.4 Examples of Moment-Linear Stochastic Systems

The remainder of this chapter is devoted to five examples of MLSS. These examples are directly drawn from the existing literature on linear and quasi-linear models, or are closely related to such models in the literature. These examples serve to illustrate the formulation and basic analysis of MLSS, to motivate MLSS models by highlighting their broad applicability, and to give an indication of the scope of MLSS. The analyses of the examples described in this section are largely not new, but we believe that these analyses are nevertheless important because they place the examples in the context of MLSS, and set the stage for novel methods of analysis that are described later in the article.

Because the models considered in this section are prevalent, a wealth of literature exists for each model, and a thorough analysis of this literature is beyond the scope of this thesis. Here, we present literature that relates to our basic analysis (i.e., literature that is concerned

with analyzing moments and cross-moments of state variables). We also present some literature regarding extensions of this basic analysis for the examples; in particular, we discuss literature concerning asymptotics of the models, and concerning linear estimation and quadratic cost control of these models. These results portend analyses of MLSS that are developed in Chapters 4, 5, and 6, but are given here so that the examples are presented in a thorough manner.

Other than presenting relevant literature, we seek to show that these five models are indeed MLSS and to introduce their basic analysis. For some of the models, we present the MLSS formulation generally; in other cases, the formulation of a particular example suffices to illustrate the general case, so only an example is given. We will revisit the models described here throughout the thesis.

2.4.1 Markovian Linear Systems Driven by I.I.D. Noise

Discrete-time Markovian linear systems driven by independent, identically distributed (i.i.d.) noise samples are commonly used in a wide range of engineering and scientific disciplines (see, e.g., [85] for a few examples). Markovian linear systems driven by noise are typically specified with a *state sequence* and an *output sequence* [85]. The state sequence, which is of interest to us in this section, constitutes a Markov process and is governed by an equation of the form $\mathbf{s}[k+1] = A_k\mathbf{s}[k] + B_k\mathbf{u}[k] + G_k\mathbf{w}[k]$, where $\mathbf{s}[k]$ is the state at time k, $\mathbf{u}[k]$ is a vector input at time k, $\{\mathbf{w}[k]\}$ is an i.i.d. noise process (i.e., a sequence of independent random vectors), and A_k, B_k, and G_k are appropriately-dimensioned matrices [85]. We require that the statistics (moments and cross-moments) of the noise samples are known and finite. (We note that our analysis does not strictly require that the noise samples at each time are independent, but only that all cross-moments across time-steps can be factorized into Kronecker products of moments at each time-step.)

It is well-known that the expected state $E(\mathbf{s}[k])$ can be found through a linear recursion (e.g., [85, 113]). It is also well-known that second moments of the state variables can also be found recursively; these second moments are typically computed through a linear matrix equation on the covariance matrix of the state [85]. If $\{\mathbf{w}[k]\}$ is a Gaussian white noise process, then the state $\mathbf{s}[k]$ at each time k is known to be Gaussian, and its distribution can be calculated from the first two moments (see, e.g., [85]).

Higher moments of linear systems are also known to satisfy recursions. Such recursions

have been developed given particular noise input processes [106, 61]. Kronecker product-based recursions for higher moments and/or cumulants, given more general input noise processes, have also been explored, under the general framework of *higher-order statistics* for linear systems (see [99] for a review).

Asymptotics of the state $\mathbf{s}[k]$ and of its first two moments have been extensively studied, and conditions for convergence of moments and of the state vector are well-known [85]. Steady-state values for second moments and cross-moments of state variables are sometimes found as the fixed point vector of a Kronecker product-based recursion, which is similar to a second moment recursion in our formulation (e.g., [26, 87]) In addition to characterization of the state $\mathbf{s}[k]$, estimation of the state sequence from the output sequence and feedback control of the state (using an input sequence) have been studied extensively. We are particularly interested in linear minimum mean square error (LMMSE) estimation; the seminal work of Kalman [77] in the 1960's introduced LMMSE estimation for Markovian linear systems (see [85] and [20] for good summaries). Also of interest to us is minimum quadratic cost control. Minimum quadratic cost controllers for linear systems are well known, as summarized in [85] and [20].

We illustrate that a linear system driven by i.i.d. noise can be formulated as an MLSS through an example. In particular, consider the system with state sequence defined by

$$\mathbf{s}[k+1] = \begin{bmatrix} 0.6 & 0.2 \\ -0.3 & 0.8 \end{bmatrix} \mathbf{s}[k] + \mathbf{v}[k], \tag{2.14}$$

where $\{\mathbf{v}[k]\}$ is a stationary white noise process, and $v_1[k]$ and $v_2[k]$ are independent random variables that are uniformly distributed over the intervals $[0, 1]$ and $[0, 2]$, respectively.

The conditional expectation for the next-state of this process given the current state is

$$\begin{aligned} E(\mathbf{s}[k+1] \,|\, \mathbf{s}[k]) &= E(\begin{bmatrix} 0.6 & 0.2 \\ -0.3 & 0.8 \end{bmatrix} \mathbf{s}[k] + \mathbf{v}[k] \,|\, \mathbf{s}[k]) \\ &= \begin{bmatrix} 0.6 & 0.2 \\ -0.3 & 0.8 \end{bmatrix} \mathbf{s}[k] + E(\mathbf{v}[k] \,|\, \mathbf{s}[k]) \\ &= \begin{bmatrix} 0.6 & 0.2 \\ -0.3 & 0.8 \end{bmatrix} \mathbf{s}[k] + E(\mathbf{v}[k]) \end{aligned} \tag{2.15}$$

Equation 2.15 shows that the state sequence is first-moment linear at each time k.

We can verify from Equation 2.14 that rth-Kronecker powers of the next-state are affine with respect to the first r Kronecker products of the current state. Using this observation, we can show that the state sequence satisfies all higher moment linearity conditions at each time k. For instance, a little algebra shows that the second conditional vector moment of $\mathbf{s}[k+1]$ given $\mathbf{s}[k]$ is given by

$$E(\mathbf{s}[k+1]^{\otimes 2} \,|\, \mathbf{s}[k]) = H_{2,2}\mathbf{s}[k]^{\otimes 2} + H_{2,1}\mathbf{s}[k] + H_{2,0}, \tag{2.16}$$

where $H_{2,2} = \begin{bmatrix} 0.6 & 0.2 \\ -0.3 & 0.8 \end{bmatrix}^{\otimes 2}$, $H_{2,1} = \begin{bmatrix} 0.6 & 0.2 \\ -0.3 & 0.8 \end{bmatrix} \otimes E(\mathbf{v}[k]) + E(\mathbf{v}[k]) \otimes \begin{bmatrix} 0.6 & 0.2 \\ -0.3 & 0.8 \end{bmatrix}$, and $H_{2,0} = E(\mathbf{v}[k]^{\otimes 2})$. Similar expressions can be derived for higher conditional moments, but these expressions are not shown here.

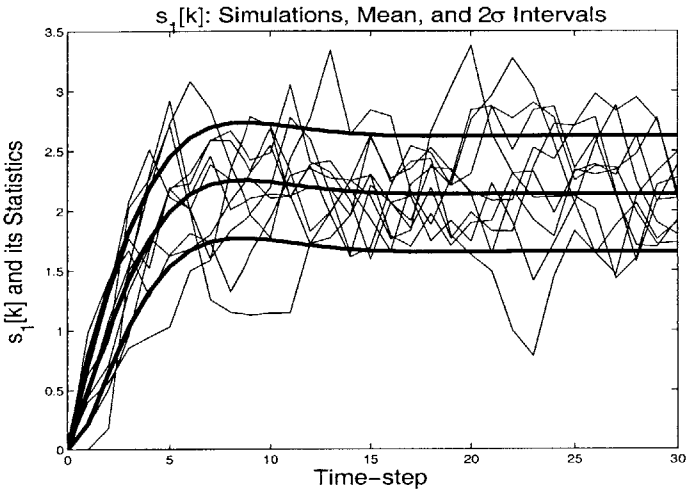


Figure 2.3: This figure plots the state variable $s_1[k]$ as a function of time in 10 simulations of the noise-driven linear system in Equation 2.14. Also, the expected value for $s_1[k]$, as well as 2σ intervals about the mean, are shown. We assume the state vector of the system is initially $\mathbf{0}$.

Since our example linear system is an MLSS, moments and cross-moments of state variables at each time-step can be found using moment recursions (Equation 2.6). We have used the moment recursions to find the expected value of $s_1[k]$, as well as 2σ (i.e., two standard deviation) intervals around this mean, for 20 time-steps. For these calculations, we assume an initial state $\mathbf{s}[0] = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$. Simulations of $s_1[k]$, along with the mean and two-standard deviation intervals, are shown in Figure 2.3. In the figure, there is good

agreement between the computed statistics and the simulations.

Although we have only considered an example, our discussion illustrates that discrete-time linear systems driven by i.i.d. noise (and additional deterministic input) can generally be formulated and analyzed as MLSS.

2.4.2 Finite-State Markov Chains

Markov chains with finite state-space are also prevalent in systems modeling (see [53] for a general introduction). In our development here, we describe a Markov chain as transitioning among a finite number of *statuses* or values along a discrete time axis[2]. Recall that the evolution of a finite-state Markov chain is specified by a probability distribution for its initial status, as well as *transition probabilities*—i.e., conditional distributions for the next-status of the chain, given the current status. These transition probabilities are often presented in a *transition matrix*, a stochastic matrix whose rows specify the next-status probabilities for each current status.

It is well-known that the (unconditioned) probabilities that the Markov chain is in each status at a particular time-step (henceforth called *status probabilities*) can be found through a linear recursion. In the following example, we illustrate that this recursion can in fact be viewed as a first moment recursion of a particular LSS whose state vector *indicates* the current status of the Markov chain. This reformulation exploits a simple (and well-known) equivalence between status probabilities and expectations of status indicators [110].

The further analyses of MLSS that we will pursue in later chapters—namely, asymptotics of state dynamics, estimation, and control—have been considered extensively for Markov chains. The main results concerning the asymptotics of Markov chains can be recovered from our MLSS formulation. Estimation of Markov chain statuses from noisy observations is of importance in speech processing, bioinformatics, and several other settings [116, 16, 48]. This need for estimation of statuses from observations has led to the development of *hidden Markov models* (HMMs) [116, 48]. In the context of HMMs, efficient non-linear algorithms for calculating the probability that a Markov chain is in a particular status given a sequence of observations have been developed, and thus estimation of the most probable status given the observations can be achieved [116, 13]. The linear esimator

[2]In the literature, the values assumed by a Markov chain are typically denoted as *states* [53]; we use the alternative terminology to distinguish from the LSS state.

that we develop for MLSS (presented in Chapter 5) turns out to be a suboptimal estimator for the HMM; it may prove valuable in bounding the probability of error for the optimal HMM estimator.

Consider an m-status Markov chain with transition matrix A. To formulate this Markov chain as an MLSS, we define a m-component state vector $\mathbf{s}[k]$ that *indicates* the time-k status of the Markov chain—i.e., $\mathbf{s}[k]$ has a single entry of 1 at the position corresponding to the time-k status, and is otherwise 0. The conditional expectation $E(\mathbf{s}[k+1] \mid \mathbf{s}[k])$ is therefore a vector of next-status probabilities, given the current state vector (and hence current status of the Markov chain). This next-status probability vector is specified by the row of the transition matrix A corresponding to the current status, and so can be written as $A'\mathbf{s}[k]$. Thus, we find that

$$E(\mathbf{s}[k+1] \mid \mathbf{s}[k]) = A'\mathbf{s}[k]. \tag{2.17}$$

The first conditional moment of $\mathbf{s}[k+1]$ given $\mathbf{s}[k]$ completely specifies the conditional distribution for the next state, given the current state. Thus, higher conditional vector moments do not characterize the dynamics of the model any further, and are not of any particular use. Nevertheless, for the sake of completeness, we briefly discuss why higher moment linearity conditions hold. To do so, consider the conditional rth moments. The entries in this vector, which are expectations of products of multiple different time-$k+1$ state variables, are always zero. Thus, the only non-zero entries have the form $E((s_i[k+1])^r \mid \mathbf{s}[k])$. Since $s_i[k+1]$ is either 0 or 1, we have that

$$E((s_i[k+1])^r \mid \mathbf{s}[k]) = E(s_i[k+1] \mid \mathbf{s}[k]). \tag{2.18}$$

Since $E(s_i[k+1] \mid \mathbf{s}[k])$ can be written as a linear function of the state variables, $E((s_i[k+1])^r \mid \mathbf{s}[k])$ can also be written as a linear function of the state variables, and the conditional expectation $E(\mathbf{s}[k+1]^{\otimes r} \mid \mathbf{s}[k])$ can be written as a linear function of $\mathbf{s}[k]$. Thus, the higher-moment linearity conditions for an MLSS hold.

As a specific example, consider a two-status Markov chain with transition matrix

$$A = \begin{bmatrix} 0.9 & 0.1 \\ 0.3 & 0.7 \end{bmatrix} \tag{2.19}$$

Defining the state vector $\mathbf{s}[k]$ to indicate the status of the Markov chain and applying the

MLSS reformulation described above, we see that the first-moment linearity condition is

$$E\left(\mathbf{s}[k+1] \,|\, \mathbf{s}[k]\right) = A'\mathbf{s}[k] = \begin{bmatrix} 0.9 & 0.3 \\ 0.1 & 0.7 \end{bmatrix} \begin{bmatrix} s_1[k] \\ s_2[k] \end{bmatrix} \tag{2.20}$$

We can also analyze the second conditional vector moment, as follows:

$$\begin{aligned} E\left(\mathbf{s}[k+1]^{\otimes 2} \,|\, \mathbf{s}[k]\right) &= E\left(\begin{bmatrix} (s_1[k+1])^2 \\ s_1[k]s_2[k] \\ s_2[k]s_1[k] \\ (s_2[k+1])^2 \end{bmatrix} |\, \mathbf{s}[k] \right) \\ &= E\left(\begin{bmatrix} s_1[k+1] \\ 0 \\ 0 \\ s_2[k+1] \end{bmatrix} |\, \mathbf{s}[k] \right) \\ &= \begin{bmatrix} 0.9 & 0.3 \\ 0 & 0 \\ 0 & 0 \\ 0.1 & 0.7 \end{bmatrix} \begin{bmatrix} s_1[k] \\ s_2[k] \end{bmatrix}. \end{aligned} \tag{2.21}$$

Interestingly, we can also write the second conditional vector moment as follows:

$$E\left(\mathbf{s}[k+1]^{\otimes 2} \,|\, \mathbf{s}[k]\right) = \begin{bmatrix} 0.9 & - & - & 0.3 \\ 0 & - & - & 0 \\ 0 & - & - & 0 \\ 0.1 & - & - & 0.7 \end{bmatrix} \begin{bmatrix} s_1[k] \\ 0 \\ 0 \\ s_2[k] \end{bmatrix} \tag{2.22}$$

$$= \begin{bmatrix} 0.9 & - & - & 0.3 \\ 0 & - & - & 0 \\ 0 & - & - & 0 \\ 0.1 & - & - & 0.7 \end{bmatrix} \mathbf{s}[k]^{\otimes 2}, \tag{2.23}$$

where the '-' entries in the matrix may be arbitrary (because the initialization of $\mathbf{s}[k]^{\otimes 2}$ is such that these entries are irrelevant). From Equation 2.22, we see that the second moment-linearity condition can be expressed in multiple (in fact, an infinite number) of different forms. In the context of the Markov chain, this degeneracy reflects inherent redundancy in the higher-moment vectors. In particular, all entries in the higher-moment vectors are either 0 or identical to entries in the first-moment vector.

2.4.3 Markovian Jump-Linear Systems

A *jump-linear system* is a stochastic *hybrid* model (i.e., combining continuous-valued and discrete-valued state processes) that has many applications, including as a model for leaks in an experimental heat-exchanger [94], more generally as a model for systems that are subject to sudden perturbations (e.g. [32]), and for control systems that are subject to communication delays [52]. We consider a discrete-time Markovian jump-linear system of the form

$$\mathbf{x}[k+1] = A(\mathbf{q}[k])\mathbf{x}[k] + B(\mathbf{q}[k])\mathbf{u}[k] \tag{2.24}$$

$$\mathbf{y}[k] = C(\mathbf{q}[k])\mathbf{x}[k] + D(\mathbf{q}[k])\mathbf{u}[k], \tag{2.25}$$

where $\{\mathbf{q}[k]\}$ is an indicator vector sequence representation for an underlying Markov chain with finite state-space, $\{\mathbf{u}[k]\}$ is a continuous-valued input process, $\{\mathbf{x}[k]\}$ is the *continuous-valued state process* of the system, and $\{\mathbf{y}[k]\}$ is an output process. In this section, we show that the *state update* (2.24) of the jump-linear system can be reformulated as an MLSS, for a known input $\{\mathbf{u}[k]\}$. (We will return to the output equation 2.25 and consider state-dependent inputs later, in the context of state estimation and control.)

Recursions for the mean and covariance matrix of $\mathbf{x}[k]$ are well-known in the literature (e.g [36]). Through our MLSS formulation, we rewrite these recursions in a Kronecker product-based notation. Asymptotics of the state and of state-variable statistics have been characterized, usually with the goal of determining sufficient conditions for stability (e.g., [51]). Later, we will show how the MLSS formulation can be used to derive conditions for moment convergence for MLSS. Both non-linear and linear estimators for the continuous-valued state of a jump-linear system, given the output, have been developed (see, e.g., [1], [36],[134])[3]. The linear estimator of MLSS that we will introduce in Section 5 can be applied to jump-linear systems; this estimator is identical to the linear estimator advanced by [36], except in that a slightly different quadratic cost is minimized by each. Control and stabilizability of jump-linear systems have also been explored (e.g., [47],[37],[51]).

We now discuss the MLSS reformulation for the state update. In particular, consider Equation 2.24 for a fixed input $\mathbf{u}[k]$. In this case, Equation 2.24 can be rewritten in the form

$$\mathbf{x}[k+1] = A(\mathbf{q}[k])\mathbf{x}[k] + \mathbf{b}_k(\mathbf{q}[k]). \tag{2.26}$$

[3]In these examples, the jump-linear system is typically subject to a stochastic input; for the sake of brevity, we do not explicitly consider stochastic inputs here, though our analysis carries over.

For convenience, we rewrite Equation 2.26 in an extended form as

$$\tilde{\mathbf{x}}[k+1] = \tilde{A}_k(\mathbf{q}[k])\tilde{\mathbf{x}}[k], \tag{2.27}$$

where $\tilde{\mathbf{x}}[k] = \begin{bmatrix} \mathbf{x}[k] \\ 1 \end{bmatrix}$ and $\tilde{A}_k(\mathbf{q}[k]) = \begin{bmatrix} A(\mathbf{q}[k]) & \mathbf{b}_k(\mathbf{q}[k]) \\ \mathbf{0} & 1 \end{bmatrix}$. We denote the transition matrix for the underlying Markov chain by Θ.

To reformulate the jump-linear system as an MLSS, we define a state vector that captures both the continuous state and underlying Markov dynamics of the jump-linear system. In particular, we define the state vector as $\mathbf{s}[k] = \mathbf{q}[k] \otimes \tilde{\mathbf{x}}[k]$, and consider the first conditional vector moment $E(\mathbf{s}[k+1] \,|\, \mathbf{s}[k])$. Since $\mathbf{s}[k]$ uniquely specifies $\mathbf{x}[k]$ and $\mathbf{q}[k]$, we can determine this first conditional vector moment as follows:

$$\begin{aligned} E(\mathbf{s}[k+1] \,|\, \mathbf{s}[k]) &= E(\mathbf{s}[k+1] \,|\, \mathbf{q}[k], \mathbf{x}[k]) \\ &= E(\mathbf{q}[k+1] \otimes \tilde{\mathbf{x}}[k+1] \,|\, \mathbf{x}[k], \mathbf{q}[k]) \\ &= E(\mathbf{q}[k+1] \,|\, \mathbf{q}[k]) \otimes \tilde{A}_k(\mathbf{q}[k])\tilde{\mathbf{x}}[k] \\ &= \Theta'\mathbf{q}[k] \otimes \tilde{A}_k(\mathbf{q}[k])\tilde{\mathbf{x}}[k] \end{aligned} \tag{2.28}$$

With a little bit of algebra, we can rewrite Equation 2.28 as

$$E(\mathbf{s}[k+1] \,|\, \mathbf{s}[k]) = \begin{bmatrix} \theta_{11}\tilde{A}_k(\mathbf{q}[k] = \mathbf{e}(1)) & \dots & \theta_{n1}\tilde{A}_k(\mathbf{q}[k] = \mathbf{e}(n)) \\ \vdots & \ddots & \vdots \\ \theta_{1n}\tilde{A}_k(\mathbf{q}[k] = \mathbf{e}(1)) & \dots & \theta_{nn}\tilde{A}_k(\mathbf{q}[k] = \mathbf{e}(n)) \end{bmatrix} \mathbf{s}[k], \tag{2.29}$$

where $\mathbf{e}(i)$ is an indicator vector with the ith entry equal to 1.

Equation 2.29 shows that the first-moment linearity condition holds for $\{\mathbf{s}[k]\}$. We can show that higher-moment linearity conditions hold in a similar fashion, though some additional bookkeeping is needed. We do not present these higher-moment linearity conditions here.

Since $\{\mathbf{s}[k]\}$ constitutes an MLSS, its moments can be found using linear moment recursions. We can manipulate these moments to find moments and cross-moments of the continuous-valued state variables of the jump linear system, as well as conditional moments and cross-moments of these continuous-valued variables given the current status of the underlying Markov chain.

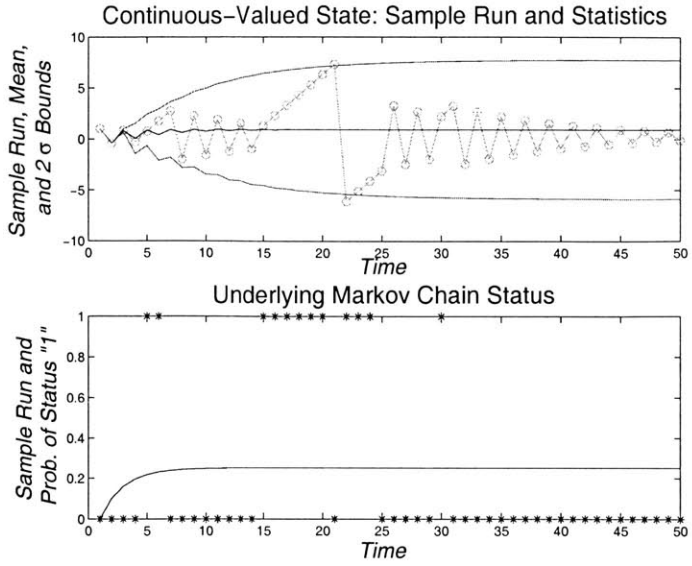


Figure 2.4: This figure shows a 50 time-step simulation of the example jump-linear system, along with statistics for the continuous-valued state and underlying Markov status. The upper plot in this figure specifies the continuous-valued state during the simulation, along with the computed mean value and two standard deviation intervals for this continuous-valued state. The lower plot indicates the status of the underlying Markov chain during the simulation and also shows the probability that the Markov chain is in status "1". The underlying Markov chain is in the status "0" initially, and the continuous state is initially $x[0] = 1$.

For illustration, we consider a jump-linear system with a two-status underlying Markov chain and a scalar continuous-valued state. The underlying Markov chain for this example has transition matrix $\Theta = \begin{bmatrix} 0.9 & 0.1 \\ 0.3 & 0.7 \end{bmatrix}$. The scalar continuous state is updated as follows: if the Markov chain is in the first status at time k, then the time-$(k+1)$ continuous state is $x[k+1] = -0.9x[k] + 0.5$; if the Markov chain is in the second status, the time-$(k+1)$ continuous state is $x[k+1] = x[k] + 1$. In this example, the expectation of the next state

$\mathbf{s}[k+1]$ given the current state $\mathbf{s}[k]$ is

$$E(\mathbf{s}[k+1] \,|\, \mathbf{s}[k]) = E\left(\begin{bmatrix}\sigma[k+1]\mathbf{x}[k+1]\\ \sigma[k+1]\end{bmatrix} \,|\, \mathbf{s}[k]\right) \tag{2.30}$$
$$= \begin{bmatrix} 0.9(-0.9) & 0.3(1) & 0.9(0.5) & 0.3(1) \\ 0.1(-0.9) & 0.7(1) & 0.1(0.5) & 0.7(1) \\ 0 & 0 & 0.9 & 0.3 \\ 0 & 0 & 0.1 & 0.7 \end{bmatrix} \begin{bmatrix}\mathbf{x}[k]\sigma[k]\\ \sigma[k]\end{bmatrix}.$$

The first- and second-moment recursions for this example are used to to determine the probabilities that the underlying Markov chain is in each status, the expected continuous-valued state, and 2-standard deviation intervals around this mean. These statistics are shown along with a 50 time-step simulation of the jump linear system in Figure 2.4. Based on the simulation, we might guess that the continuous-valued state remains bounded but does not settle to a constant. This intuition is borne out by the computed mean and 2-standard deviation bounds for the continuous-valued state, since the mean settles to a constant (0.94) and the 2-standard deviation intervals about the mean settle to a non-zero constant (6.8 on either side of the mean).

The first-moment recursion can also be used to compute expectations of the continuous-valued state at each time-step, given each possible status of the underlying Markov chain at that time-step, as shown in Figure 2.5. These conditional expectations allow us to compare the expected dynamics of the continuous-valued state at a particular time-step given that the underlying Markov chain is concurrently in each possible status. For instance, Figure 2.5 shows that, asymptotically, the expectation of the continuous-valued state is larger if the underlying Markov chain is in its second status.

2.4.4 Markov-Modulated Poisson Processes

A Markov-modulated Poisson process (MMPP) is an arrival process model that has commonly been used to represent sources in communications and manufacturing systems (e.g., [14], [107], [30]). In some applications ([14, 107]), data sources are modeled with a single MMPP; in other models, combined arrivals from multiple MMPPs are used to represent a data source [31, 98]. MMPP arrivals are typically envisioned as entering a queue or queueing network. Much work has focused on characterizing the steady-state dynamics

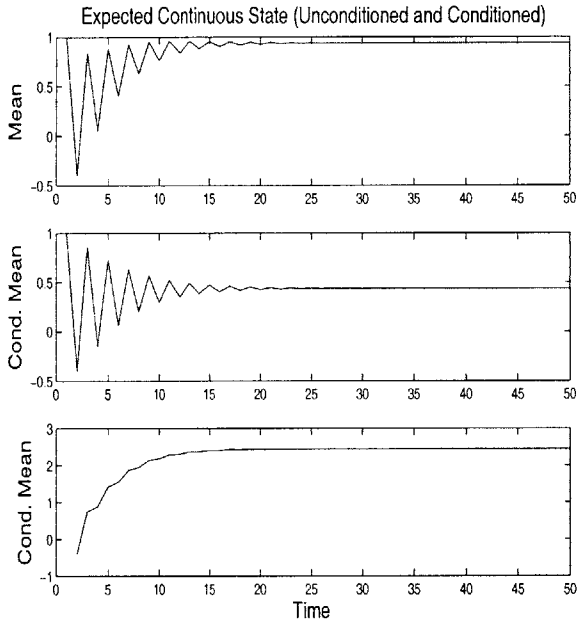


Figure 2.5: The top plot shows the expectation of the continuous-valued state of the example jump-linear system. The two lower plots show the conditional expectations of the continuous-valued state, given each possible status of the underlying Markov chain. In these calculations, we assume that the continuous-valued state is initially 1, and that the underlying Markov chain begins in its first status.

of queues and queueing networks with MMPP input (e.g. [107]). For queues with inputs from multiple MMPPs, in particular, intensive computation is required to determine steady-state dynamics [31, 98], motivating the need for reduced-order characterizations. Algorithms for state estimation from certain observations are discussed in [84],[127],[24]. Control of queueing system dynamics through design of MMPP inputs has been studied (e.g. [69]).

MMPPs are typically formulated in continuous time. A discrete-time formulation has been given in [124], and has been related to the continuous-time formulation. Here, we illustrate how a discrete-time MMPP can be represented as an MLSS through an example.

In our formulation of a discrete-time MMPP, we track the status $\{\sigma[k]\}$ of an *underlying Markov chain* as well as the number of arrivals $f[k]$ in a discrete-time arrival process. The underlying Markov chain evolves according to a transition matrix A, while the distribution for the number of arrivals at time $k+1$, or $f[k+1]$, is specified based on the time-k

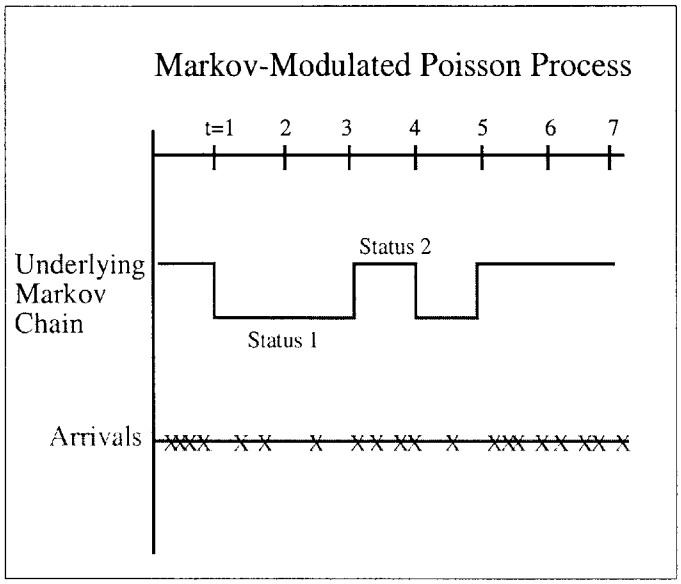


Figure 2.6: This figure illustrates the dynamics of an MMPP. An MMPP comprises an underlying Markov chain, as well as a Poisson arrival process which has a rate that is modulated by the underlying Markov chain.

status of the underlying Markov chain. In particular, the distribution for $f[k+1]$ given $\sigma[k]$ is a Poisson random variable, with mean specified by $\sigma[k]$. Note that $f[k+1]$ can be interpreted as the number of arrivals during a unit interval for a continuous-time Poisson process which has rate specified by $\sigma[k]$. An illustration of the MMPP that is based on this continuous-time interpretation is shown in Figure 2.6.

For instance, we consider an MMPP with a two-status underlying Markov chain that evolves according to the transition matrix $A = \begin{bmatrix} 0.9 & 0.1 \\ 0.3 & 0.7 \end{bmatrix}$. For convenience, we again define a length-2 status vector $\mathbf{q}[k]$ that indicates the status of the Markov chain. Given that $\mathbf{q}'[k] = \begin{bmatrix} 1 & 0 \end{bmatrix}$, the number of arrivals $f[k+1]$ is assumed to be a Poisson random variable with mean 1. Given that $\mathbf{q}'[k] = \begin{bmatrix} 0 & 1 \end{bmatrix}$, the number of arrivals is assumed to be a Poisson random variable with mean 10.

To formulate this example MMPP as an MLSS, we define a state vector $\mathbf{s}[k] = \begin{bmatrix} \mathbf{q}[k] \\ f[k] \end{bmatrix}$. To

verify the first-moment linearity condition for $\{\mathbf{s}[k]\}$, we note that

$$\begin{aligned} E(\mathbf{s}[k+1] \,|\, \mathbf{s}[k]) &= E(\begin{bmatrix} \mathbf{q}[k+1] \\ f[k+1] \end{bmatrix} \,|\, \mathbf{s}[k]) \\ &= \begin{bmatrix} E(\mathbf{q}[k+1] \,|\, \sigma[k]) \\ E(f[k+1] \,|\, \sigma[k]) \end{bmatrix} \\ &= \begin{bmatrix} \begin{bmatrix} 0.9 & 0.3 \\ 0.1 & 0.7 \end{bmatrix} \mathbf{q}[k] \\ \begin{bmatrix} 1 & 10 \end{bmatrix} \mathbf{q}[k] \end{bmatrix}. \\ &= \begin{bmatrix} 0.9 & 0.3 & 0 \\ 0.1 & 0.7 & 0 \\ 1 & 10 & 0 \end{bmatrix} \mathbf{s}[k]. \end{aligned} \tag{2.31}$$

We briefly discuss why higher-moment linearity conditions hold. To see why, note that the $E(\mathbf{s}[k+1]^{\otimes r} \,|\, \mathbf{s}[k]) = E(\mathbf{s}[k+1]^{\otimes r} \,|\, \mathbf{q}[k])$. This expectation can be further rewritten as $E(\mathbf{s}[k+1]^{\otimes r} \,|\, \mathbf{s}[k]) = q_1[k]E(\mathbf{s}[k+1]^{\otimes r} \,|\, \mathbf{q}'[k] = [1,0]) + q_2[k]E(\mathbf{s}[k+1]^{\otimes r} \,|\, \mathbf{q}'[k] = [0,1])$, since the expression is equal to $E(\mathbf{s}[k+1]^{\otimes r} \,|\, \mathbf{q}'[k] = [1,0])$ when $q_1 = 1$ and $q_2 = 0$, and similarly for $q_1 = 0$ and $q_2 = 1$. Thus, $E(\mathbf{s}[k+1]^{\otimes r} \,|\, \mathbf{s}[k])$ is a linear function of $\mathbf{s}[k]$, and the rth moment linearity condition holds for any r.

A simulation of the example MMPP is shown in Figure 2.7. Along with the simulation, the computed expectation and variance in the number of arrivals $f[k]$ are shown as functions of time.

2.4.5 A Discrete-Time Infinite Server Queue with Random Service Probabilities

Our final example is a particular discrete-time infinite-server queue. We pursue this particular example here because it simply captures several types of dynamics that can be represented using MLSS.

Although the queueing model that we discuss here is not common in the literature, it is closely related to some common queueing models. In particular, the queue that we describe here is a variation on a discrete-time $M/M/\infty$ queue—i.e., a queue with Poisson inputs and an infinite number of exponential servers (see, e.g., [83], [63], or [40] for intro-

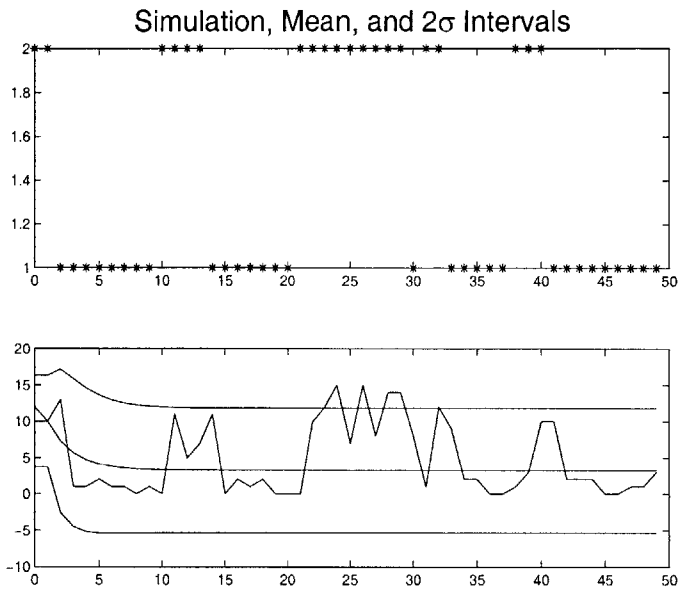


Figure 2.7: The upper plot in this figure specifies the status of the underlying Markov chain during a simulation of the example MMPP (the status "1" corresponds to $\mathbf{q}'[k] = [1, 0]$, while the status "2" corresponds to $\mathbf{q}'[k] = [0, 1]$). The lower plot shows the simulated number of arrivals $f[k]$, as well as the expected number of arrivals $E(f[k])$ and 2σ intervals about this mean. We have assumed that the underlying Markov chain is in the second status initially. Interestingly, the lower bound is negative, although $f[k]$ can never be negative; the large variance, which causes the negative lower bound, reflects the large positive deviations in $f[k]$ from its mean.

ductions to $M/M/\infty$ queues). Infinite-server queues, and in particular $M/M/\infty$ queues, are of interest as approximations for queueing systems with an abundance of servers [55]. They are particularly analyzable, in that the distribution for the number of jobs in the queue (i.e., the number of jobs being served) at each time-step can be found, given the initial number of jobs in the queue ([63], [40]). Of interest to us is the fact that these transient and steady-state distributions are indirectly obtained using the *moment-generating function* for the number of jobs, and in consequence the moments of the number of jobs in the queue can also be determined [63]. We have not found any work concerning estimation and control of $M/M/\infty$ queues, though estimation has been considered for other queueing models (e.g. [45]).

The discrete-time queueing example that we consider evolves as follows during each time-step:

- New jobs arrive at the queue. We assume that the number of new jobs is a Poisson random variable $a[k+1]$ with mean 10. Also, the number of jobs arriving during each time interval is assumed independent of the arrivals at other times, and of the number of jobs in the queue at previous times.
- Each job that is in the queue at time k is served and released (i.e., service is completed) independently at time $k+1$ with the same probability $p[k]$. The probability $p[k]$, which is itself stochastic, is assumed to be determined independently at each time-step according to the following p.m.f.:

$$p[k] = \begin{cases} 0.6, & w.p.\, 0.7 \\ 0.2, & w.p.\, 0.3 \end{cases} \tag{2.32}$$

We are interested in tracking the number of jobs $s_1[k]$ that are in the queue at each time k, as well as the number of jobs $s_2[k]$ that depart at time k.

The time-$(k+1)$ state variables $s_1[k+1]$ and $s_2[k+1]$ for this system can be expressed in terms of the time-k state variables by considering the jobs that enter and exit the system at time $k+1$:

$$\begin{aligned} s_1[k+1] &= s_1[k] + a[k+1] - b[k+1] \\ s_2[k+1] &= b[k+1], \end{aligned} \tag{2.33}$$

where $b[k+1]$ represents the number jobs that exit the queue at time $k+1$. Given $p[k]$, $b[k+1]$ is a binomial random variable with distribution $Binom(s_2[k], p[k])$. (Note that the time-$(k+1)$ state variables do not depend on $s_2[k]$, given $s_1[k]$, so we could equivalently define the system solely in terms of $s_1[k]$. We explicitly consider $s_2[k]$ because we later consider estimation of $s_1[k]$ given a corrupted measurement of $s_2[k]$.)

Consider the state vector $\mathbf{s}[k] = \begin{bmatrix} s_1[k] \\ s_2[k] \end{bmatrix}$. Note that $\{\mathbf{s}[k]\}$ constitutes a Markov process. In fact, $\{\mathbf{s}[k]\}$ is an MLSS. To see why, first consider the first-order conditional vector moment:

$$\begin{aligned} E(\mathbf{s}[k+1] \,|\, \mathbf{s}[k]) &= E(\begin{bmatrix} s_1[k] + a[k+1] - b[k+1] \\ b[k+1] \end{bmatrix} \,|\, \mathbf{s}[k]) \\ &= \begin{bmatrix} s_1[k] + 10 - E(b[k+1] \,|\, \mathbf{s}[k]) \\ E(b[k+1] \,|\, \mathbf{s}[k]) \end{bmatrix}. \end{aligned} \tag{2.34}$$

We can find the the expectation $E(b[k+1] \,|\, \mathbf{s}[k])$ by conditioning on the stochastic service

probability $p[k]$. We find that the expectation becomes $E(b[k+1] \,|\, \mathbf{s}[k]) = 0.7(0.6s_1[k]) + 0.2(0.3s_1[k]) = 0.48s_1[k]$. Substituting this expectation into Equation 2.34 and rewriting in matrix form yields

$$E(\mathbf{s}[k+1] \,|\, \mathbf{s}[k]) = \begin{bmatrix} 0.52 & 0 \\ 0.48 & 0 \end{bmatrix} \mathbf{s}[k] + \begin{bmatrix} 10 \\ 0 \end{bmatrix}. \tag{2.35}$$

Equation 2.35 shows that the state process satisfies the first-moment linearity condition of an MLSS.

In fact, it turns out that all moment linearity conditions hold. We do not specify the higher conditional moments in detail, but give a brief conceptual justification for why the higher-moment linearity conditions hold. A key observation needed for checking these higher-moment linearity conditions is that conditional rth moment for $b[k+1]$ given $\mathbf{s}[k]$ and $p[k]$, which has a binomial distribution with parameters $s_1[k]$ and $p[k]$, is an rth-degree polynomial with respect to $s_1[k]$ [75]. Next, since $p[k]$ is chosen independently of $\mathbf{s}[k]$, the rth moment for $b[k+1]$ given $\mathbf{s}[k]$ is also an rth-degree polynomial of $s_1[k]$. It is then straightforward to check that the rth moments of $s_1[k+1]$ and $s_2[k+1]$ given $\mathbf{s}[k]$ are rth-degree polynomials of $s_1[k]$.

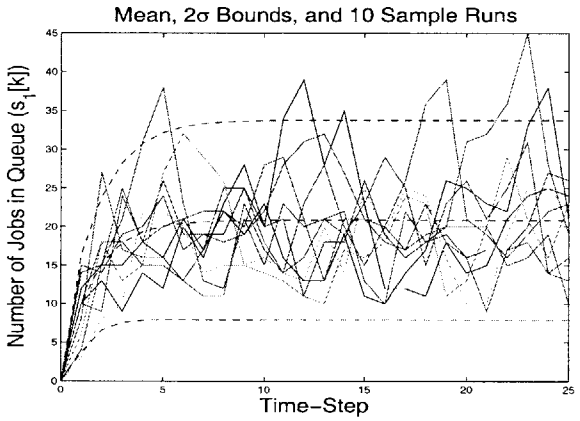


Figure 2.8: This figure shows the evolution of the number of jobs $s_1[k]$ in the queue. In particular, 10 simulations of the system are shown, and the moment recursions are used to find the mean number of jobs and two-standard deviation (2σ) intervals about the mean. The queue is assumed to be empty initially.

We have used the first and second moment recursions to find the expected number of jobs in the queue and 2σ intervals about this mean as a function of time (assuming an initially

empty queue), as shown in Figure 2.8. The second moment recursion can also be used to show that the steady-state correlation between $s_1[k]$ and $s_2[k]$ is negative (in particular, -0.28).

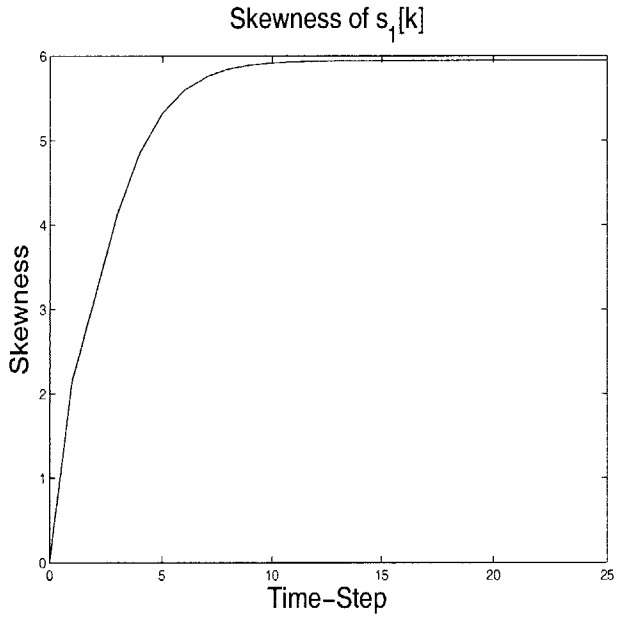


Figure 2.9: This figure shows the skewness of $s_1[k]$, the number of jobs in the queue. The skewness is positive, indicating a possibility for occasional abnormally-large numbers of jobs in the queue.

Also, we have used the third-moment recursion to find the skewness of $s_1[k]$ (defined as $[E((s_1[k] - E(s_1[k]))^3)]^{1/3}$). This skewness is shown in Figure 2.9. We find that the number of jobs in the queue is positively skewed. The positive skewness is reflected in the tendency for large positive anomalies in the simulated queue length, and in the contrasting hard lower bound (as seen in Figure 2.8).

It turns out that the steady-state distribution for the number of jobs in the queue can be determined numerically in a straightforward manner. Thus, we can compare the computational effort required to directly characterize the steady-state distribution with the computational effort required to determine the first three moments of this distribution using the MLSS formulation of the queue. In particular, we have compared the processor time required to determine the steady-state distribution with the processor time required to compute the first three moments (Figure 2.10). As shown in the figure, our MLSS-based computation of the first three moments is much quicker than our calculation of the distri-

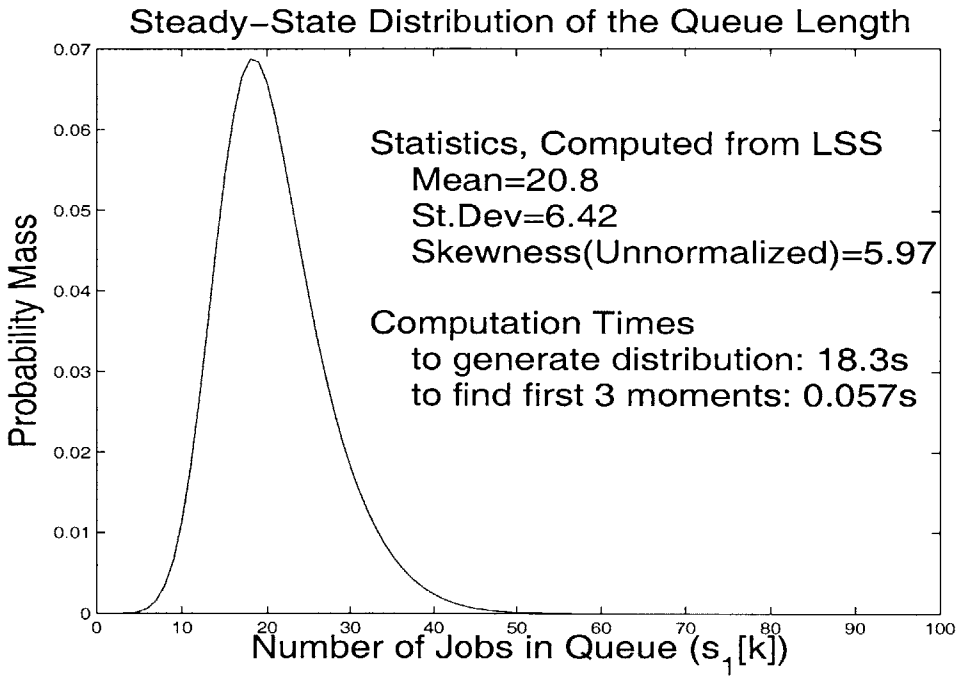


Figure 2.10: This figure shows the steady-state distribution for the number of jobs in the queue (found using numerical techniques), as well as the first three moments of this number (found using the MLSS model). The moments can be computed much more quickly than the distribution.

bution. This (admittedly rough) comparison suggests that MLSS can potentially be valuable for deducing state statistics for models in which computing the state distribution is infeasible or taxing.

It is worthwhile to mention that the random service probability in this example is indicative of a braoder tractability: MLSS can naturally capture variations of existing models that have randomly-varying parameters. As in this example, one only needs to condition on the random parameter to recover the moment-linearity conditions in this case.

Chapter 3

Using MLSS to Represent Network Dynamics

3.1 Introduction

This chapter focuses on four examples of MLSS that have network structure. While the dynamics represented in these four examples vary widely, each is a network in the sense that it comprises multiple distinguishable *sites* (components) with associated *statuses* (local states) that evolve in discrete time in an interrelated manner. More specifically, in each example the sites' statuses, collectively constituting the (global) state of the system, evolve according to a discrete-time Markov process. Moreover, the evolution is structured in such a manner that the state process—where the state is defined to be a concatenation of the *status vectors* that codify the sites' statuses—constitutes an MLSS. We informally use the term *moment-linear stochastic networks* (MLSN) to identify such examples of MLSS, i.e., those in which the dynamics can be viewed as occurring among groups of interacting or interrelated sites.

We formulate these network examples by first specifying a complete probabilistic description of the evolution of site statuses, with the aim of representing a particular physical process or developing an evocative example of certain types of network interactions. We then show that the model's state process constitutes an MLSS, and hence that state statistics can be found using linear recursions. In turn, the statistics obtained using the moment recursions allow us to characterize individual site statuses, as well as interdependencies among these statuses, at particular times.

In Section 2 we describe, in a general way, our motivations for seeking MLSS models for networks. In Section 3 we introduce the four examples of MLSN. In Section 4, we discuss a common theme of three of the four examples—namely, that they represent *flows*, or movements of items of material, in a network. In Section 5 we briefly discuss some possible approaches for generally specifying graphs for MLSN, and mention some difficulties in these approaches. Finally, in Section 6, we discuss a reordering of the MLSS moment

recursions, which facilitates representation of site interactions in cases for which a vector (rather than scalar) status is associated with each site.

3.2 Motivation

We are motivated to consider MLSS modeling of networks for several reasons:

- **MLSS can potentially provide computationally attractive representations for large systems (i.e., systems with many state variables), including networks.** Analysis of large stochastic systems can be daunting, because the required computation often grows exponentially in the number of state variables (see, e.g., [137, 56] for a discussion of the computation required to analyze stochastic network models). For instance, consider a network with n components, each of which is constrained to take two *statuses*, or values. Then the network can be in 2^n configurations at each time-step, and a Markovian description of the system's evolution would require a $2^n \times 2^n$ transition matrix. Representation of stochastic systems with continuous-valued state variables is similarly daunting, since joint configurations of state variables and mappings for transitions between such configurations are required for representation and analysis. By enforcing a moment-linear structure on the network dynamics, we can greatly reduce the required computations; at the same time, as our examples will show, a rich class of dynamics can nevertheless be captured using moment-linear interactions.

- **MLSS models can expose transient dynamics of networks.** Though considerable work has focused on determining steady-state characteristics of dynamic stochastic models (see, e.g. [90, 137]), and on developing dynamic models with particular steady-state characteristics (e.g., Markov chain Monte Carlo models [110]), we have seen less work on characterizing the transients of stochastic network models. MLSS models are specially structured so that statistics of both their transient and steady-state dynamics can be determined. We highlight the transient analysis of our network models in the examples in Section 3.3, and relate these models to other queueing and flow models that admit transient analysis.

- **The basic analysis of MLSS models can be used to expose dependencies among the dynamics at multiple nodes in a network.** In particular, say we let state variables in our network models represent properties or *statuses* of different components in the network. Then cross-statistics of these state variables, which can be found using the

moment recursions, partially characterize the dependencies among the components. Such characterizations can then potentially be used to explore the relationship between a network's structure and its dynamic evolution.

Example 3.1

This simple example of a number-passing game provides a first illustration of how the MLSS analysis can be used to study interesting transient dynamics of networked systems.

Consider a row of n people, labeled $1, \ldots, n$ from left to right. At the initial time, the leftmost person has a correct message—in this case, a paticular integer between 1 and 10—that he/she wishes to transmit to the other individuals. The remaining people take an initial guess (i.e., an integer chosen uniformly between 1 and 10) for the message. The people use the following (very inefficient) algorithm to transmit the message: at each discrete time-step, one of the first $n-1$ people (i.e., anyone except the rightmost person in the line) is chosen with probability $\frac{1}{n-1}$. The chosen person relays his/her message to the neighbor on the right, who assumes the received message as his/her guess for the message. The remaining people retain their previous messages. Since each person's message at any time depends only on his/her previous message and his/her left-neighbor's message, the row of people can be thought of as linear (string) network in which each node is affected by itself and (at most) one neighbor.

To formulate this model as an MLSS, we define each state variable, $1lei \leq n$, $s_i[k]$ as an indicator that the person has the correct message. Thus, note that $\mathbf{s}_1[0] = 1$, and $\mathbf{s}_i[0]$ is 0 with probability $\frac{9}{10}$ and 1 with probability $\frac{1}{10}$ for $2 \leq i \leq n$. Now consider the conditional expectation (first moment) for each time-$k+1$ state variable given the current state $\mathbf{s}[k]$. Note that $E(\mathbf{s}_1[k+1] \,|\, \mathbf{s}[k]) = 1 = s_1[k]$. For $2 \leq i \leq n$, this expectation can be computed as follows:

$$\begin{aligned} E(\mathbf{s}_i[k+1] \,|\, \mathbf{s}[k]) &= Pr(\mathbf{s}_i[k+1] = 1 \,|\, \mathbf{s}[k]) \\ &= \frac{1}{n-1} s_{i-1}[k] + \frac{n-2}{n-1} s_i[k]. \end{aligned} \tag{3.1}$$

Thus, we see that the model satisfies the first moment linearity condition. Higher moment-linearity conditions can be verified in similar fashion, but are not pursued further here.

We have applied the first moment recursion to determine the probability that each person in a row of $n = 5$ people has the correct message at each time-step k. These probabilities are shown in Figure 3.1. The probability plots show the gradual propagation of the correct message along the network of people. The higher-moment recursions can be used to determine conditional probabilities that one person has received the message given another person's status, and hence to illustrate the role of network structure in determining dependencies among the people's messages; these higher-moment

recursions are not discussed any further here.

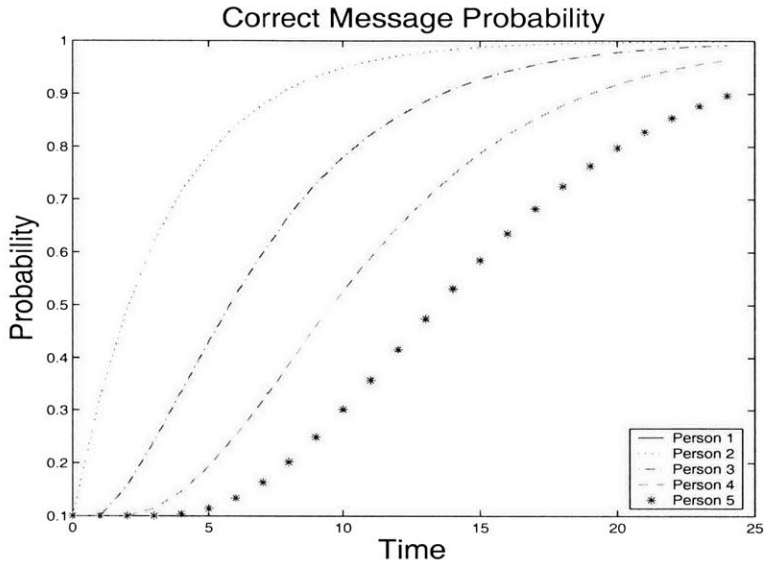


Figure 3.1: The probability that each person in the network has the correct message at each time is shown.

3.3 Examples of MLSN

We introduce four network models that can be formulated as MLSS. Our discussion will give some indication of the variety of network dynamics that can be represented using our models. Unlike the MLSS models, our MLSN models are not directly drawn from the literature, but are closely connected with various network models in the literature.

Two of these network models—the influence model and an aggregate flow model for an air traffic system—are pursued in further detail as case studies in Chapters 7 – 9. The other two models, a queueing network model and a bulk flow model for a road traffic network, do not warrant separate case studies and so are considered in somewhat more detail in this section.

3.3.1 Example: The Influence Model

The influence model, originally introduced in the thesis [9] as a network of interacting finite-state Markov chains, is an attempt at representing, in an analytically tractable way, some of the essentials of stochastic interactions that may occur among components of a network.[10]. The influence model falls in the general class of *stochastic cellular automata* models—in which network components or *cells* evolve through a discrete set of statuses according to various interaction rules [137]. The influence model is an automata model for which the probability distributions of site (cell) statuses can be determined using linear recursions. A specific form of our influence model is closely related with the *voter model* (also called the *invasion process*), another model that is also amenable to linear analysis [71, 90]. The circuit analog model given in [79] is yet another automaton that is amenable to linear analysis. The analysis of state statistics using linear recursions has been pursued in some detail for the influence model [9, 10], and to a lesser extent for the models introduced in [71], [90], and [79]. Asymptotics of the influence model have been characterized in [9]. One approach for influence model state and parameter estimation is discussed in [17], and ML state estimation in influence models can also be related to estimation in other stochastic models such as Hidden Markov Models (HMMs) ([116]) and dynamic Bayesian networks ([56]). An algorithm for control/design of a particular influence model is given in [122], but control of influence models has not been considered in a general setting.

The influence model comprises a network of n sites. Site i assumes one of a finite number m_i of possible *statuses* at each discrete-time instant. The status of site i at any discrete time k is represented by a m_i-component *status vector* $\mathbf{s}_i[k]$, which is an indicator vector containing a single 1 in the position corresponding to the present status, and 0 everywhere else. In keeping with the notation used in [9], we express the update of the influence model in terms of the row vectors $\mathbf{s}_i'[k]$ (i.e., the transposes of the status vectors).

We are concerned with the temporal evolution of the statuses of the sites. The update of the statuses constitutes a Markov process, in that their joint probability distribution at the next time step is independent of past statuses, given the current statuses of all sites. Updating the status of the ith site in the influence model can be thought of as involving the following three stages:

- Stage 1: The site i randomly selects one of its neighboring sites in the network—or selects itself—to be its *determining site* for Stage 2; site j is selected with probability d_{ij} (so $d_{ij} \geq 0$ and $\sum_j d_{ij} = 1$). We call the probabilities d_{ij} the *determining-site probabilities*.

- Stage 2: The present status $\mathbf{s}'_j[k]$ of the determining site j fixes the probability vector $\mathbf{p}'_i[k+1]$ that will be used in Stage 3 to randomly choose (or "realize") the next status of site i (so $\mathbf{p}'_i[k+1]$ is a row vector with nonnegative entries that sum to 1). Specifically, if site j is selected as the determining site, then $\mathbf{p}'_i[k+1] = \mathbf{s}'_j[k]A_{ji}$, where A_{ji} is a fixed *row-stochastic* $m_j \times m_i$ matrix, i.e., a matrix whose rows are probability vectors, with nonnegative entries that sum to 1. We call the matrices A_{ji} the *local status-evolution matrices* of the influence model.

- Stage 3: The next status $\mathbf{s}'_i[k+1]$ is realized in accordance with the $\mathbf{p}'_i[k+1]$ computed at Stage 2.

All sites are updated simultaneously, and independently, in this way. The *state vector* of the influence model is defined as $\mathbf{s}[k] = \begin{bmatrix} \mathbf{s}_1[k] \\ \vdots \\ \mathbf{s}_n[k] \end{bmatrix}$.

The influence model, as defined above, constitutes an MLSN. Let us briefly discuss why the first-moment linearity condition holds for this model. In particular, consider the conditional expectation for a site's next-status vector, given the model's current state: $E(\mathbf{s}_i[k+1] \,|\, \mathbf{s}[k])$. Since $\mathbf{s}_i[k+1]$ is an indicator vector for the status of site i at time $k+1$, the conditional expectation $E(\mathbf{s}_i[k+1] \,|\, \mathbf{s}[k])$ lists the probabilities for the time-$(k+1)$ status of site i given the time k state of the model. By conditioning on site i's choice of determining site choice for time $k+1$, we can immediately show that the vector of probabilities $E(\mathbf{s}_i[k+1] \,|\, \mathbf{s}[k])$ is given by $\sum_{j=1}^{n} d_{ij} A'_{ji} \mathbf{s}_j[k]$. Stacking the conditional expectations $E(\mathbf{s}_i[k+1] \,|\, \mathbf{s}[k])$ into a single vector shows that the influence model satisfies the first-moment linearity condition. Expressions for conditional expectations for higher joint-status vectors, and consequent expressions for moments of the state (in permuted form), are given in [9][1]. These expressions, which will be discussed in detail in Chapter 7, show that the model is an MLSN.

The rth MLSN moment recursions can be used to find expectations of joint status vectors at each time k. For the influence model, these expectations of rth joint status vectors specify probabilities for the joint statuses of groups of r sites (i.e., for the pattern of statuses of groups of r sites). Of particular interest, the nth (permuted) moment recursion for the

[1]Because status vectors in the influence model are constrained to be indicator vectors, it turns out that the conditional permuted vector moments can be written is several forms, one of which is given in [9]. These different forms for the moment recursions are sometimes amenable to different analyses.

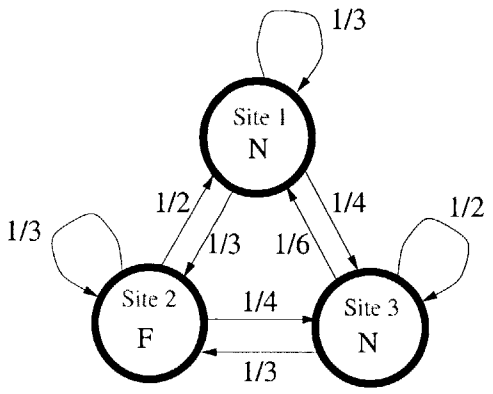


Figure 3.2: The determining site probabilities of the three-site example are depicted. In this diagram, the weights on the arrows *into* each site i specify the probabilities for the determining site choice of i. Sample statuses are also shown at each site.

influence model can be used to calculate probabilities for the joint statuses of all n sites in the influence model at each time-step, and so to completely characterize the probability distribution for the influence model's state at each time.

A three-site example is used to illustrate the influence model and its analysis as an MLSN. In this example, each site can be in one of two possible statuses, Normal (N) or Failed (F). Figure 3.2 shows the probabilities d_{ij} with which each site in this example chooses its determining site. Also, all local status transition matrices for this example are identically chosen to be $A_{ji} = A = \begin{bmatrix} 0.99 & 0.01 \\ 0.05 & 0.95 \end{bmatrix}$. Thus, in this *homogeneous influence model*, the rows of a single fixed stochastic matrix A constitute the set of probability vectors that govern the realization of the next statuses of sites. Which of these probability vectors is chosen depends on the status of the selected determining site.

We have used the first-moment recursion for the influence model to determine the probability that Site 2 is failed at each time-step, given that all sites are initially healthy. This probability is displayed along with a simulation of the status of Site 2 in Figure 3.3. We have also used the second-moment recursion to determine correlations among site statuses. For instance, we find that Site 2 is failed 76% of the time when Site 1 is failed (though Site 2 is in general only failed 17% of the time). Thus, we see that information about the status of Site 1 can be used to estimate the concurrent status of Site 2. In Chapter 7, we will discuss methods for estimating the status of a site given information about the history of

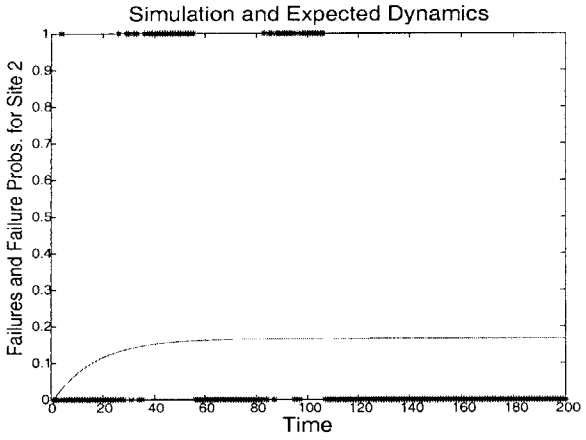


Figure 3.3: This figure shows the failure probability for site 2 during 200 time-steps, given that all sites are initially normal. A simulation of the status of site 2 is also shown. (A "*" at the top of the graph indicates a failed site, and a "*" at the bottom indicates a normal site.)

statuses at another site.

3.3.2 Example: A Heavy-Traffic Single-Server Queueing Network with State-Dependent Input Rates

The queueing network described here is the first of three examples of *flow* models—i.e., models in which items or material are envisioned as moving among sites in a network. In all three examples, our state variables are the numbers of items or the amounts of material that are present at the sites at discrete time-steps. The flows in these three examples are structured in such a way that their state processes constitute, or can be approximated as, MLSN.

There is an extensive literature on queueing networks (e.g., [83], [79]). Our queueing model builds on the *Jackson network*, a standard queueing model comprising a network of $M/M/1$ queues [53, 79]. Our model differs from the standard Jackson network in that the rates at which *jobs* (items) enter sites in our model at a particular time are allowed to depend (in a particular way) on the queue lengths at that time. Jackson-type models in which input rates depend upon queue lengths have been studied in [79], but the specifics of the dependence as well as the analysis techniques for these models differ from those of

our model. A broad class of single-server, memoryless queueing networks with state-dependent service and routing probabilities has also been studied in [96]; our model is a special case of the model described in [96], for which moments and cross-moments of state variables are amenable to analysis using linear recursions.

A heavy-traffic assumption (i.e., an assumption that queues in the network are busy with high probability at any time) is required for our model to be formulated and analyzed (approximately) as an MLSS. A heavy-traffic assumption is often germane in modeling real systems, because the behavior of the network under heavy-load conditions is of special interest. Quite general queueing network models (e.g., models with general service distributions and generally distributed but state-independent input processes) have been analyzed approximately under an assumption of heavy traffic (e.g., [43],[86], [67]). In particular, deterministic flow approximations for queueing network dynamics can be constructed under a heavy-traffic assumption. These deterministic flow approximations are capable of modeling first- and second-order statistics of the queueing network, and can be shown to capture these statistics exactly in certain limiting cases.

Deterministic flow approximations for queueing networks have served as a context for studying models with state-dependent input rates (e.g. [43, 86]). In some communications applications of these models, the queues themselves are modeled deterministically, but aspects of the input processes into the queues are modeled stochastically [5, 3, 4]. In [4], first- and second-order statistics have been determined for a linear approximation to such a model, and methods for control are also considered. Like [4], we also characterize state variable moments and consider linear estimation and control, but our work differs from [4] in that we explicitly consider the stochastics of service at the queues.

Our model comprises a network of n sites. The dynamics of the model—i.e., the flows of jobs among sites in this network—are specified in continuous time $t \geq 0$, as follows. Each site in the model contains a single-server first-in-first-out (FIFO) queue. The server at each site i is exponential, with mean service time denoted λ_i. Once a job has been served at site i, it is routed to the queue at the site j, where $j \neq i$, with probability p_{ij}, and leaves the system with probability p_{i0}. (The service times and routing choices of each job are assumed independent of all past history of the system, including the previous service times and route of that job.)

Additionally, jobs enter the queue at each site from outside the network according to a time-varying and state-dependent Poisson process. The rate of this *input* at queue i at a

particular time t is assumed to be $u_{0i}(t) \triangleq max(0, \overline{u}_{0i} + g_i(s_i(t) - \overline{s}_i))$, where $s_i(t)$ is the length of queue i (i.e., the number of jobs at queue i) at time t, the constant $\overline{u}_{0i}$ is a "target" rate, the constant $\overline{s}$ is a "target" queue length, and g_i is a negative constant. Thus, the input rate into each queue at any time is a linear (actually, affine) function of the queue length at that time, as long as the queue length is small enough that the resulting rate is non-negative. If the queue length becomes sufficiently large, the input rate is set to 0.

We are concerned with tracking the queue lengths at each site in the model, at discrete time-steps. In particular, our goal is to characterize the state variables $s_i[k] \triangleq s_i(k\Delta T)$, where ΔT specifies the interval at which we wish to track the queue lengths. If the following three assumptions hold, the state process $\{\mathbf{s}[k]\}$ of this queueing network is well-approximated by an MLSN:

1. *Heavy-traffic assumption*: with high probability, all queues in the network are busy (i.e., have at least one job) at each time t.
2. *Positive input rate assumption*: with high probability, the input rate $u_{0i}(t)$ into each queue i is positive.
3. *Short time-interval assumption*: the time interval ΔT is chosen so that, with high probability, the input rate $u_{0i}(t)$ into each queue i remains constant or nearly constant over intervals of duration ΔT.

We briefly give an imprecise justification for why these assumptions are sufficient for $\{\mathbf{s}[k]\}$ to be represented as an MLSN. To do so, consider $s_i[k+1]$. Given $\mathbf{s}[k]$, $s_i[k+1]$ can be expressed in terms of the time-k state variables by considering the flows into and out of site i between times $k\Delta T$ and $(k+1)\Delta T$:

$$s_i[k+1] = s_i[k] - \sum_{j=0, j\neq i}^{n} U_{ij}[k] + \sum_{j=0, j\neq i}^{n} U_{ji}[k], \tag{3.2}$$

where $U_{ij}[k]$, $1 \leq i \leq n$, $0 \leq j \leq n$, represents the number of jobs that flow from site i to site j (or out of the system, for $j = 0$) between times $k\Delta T$ and $(k+1)\Delta T$ and $U_{0i}[k]$ represents the flow into site i from outside the system. First, consider the flows among sites, and out of the system. Since all servers are assumed to be busy from the heavy traffic assumption, the process of jobs that flows out of each site i (either to other sites or out of the system) is a Poisson process with rate λ_i, independent of the past history of the system and of the other time-k flows. In consequence of the splitting law for Poisson processes,

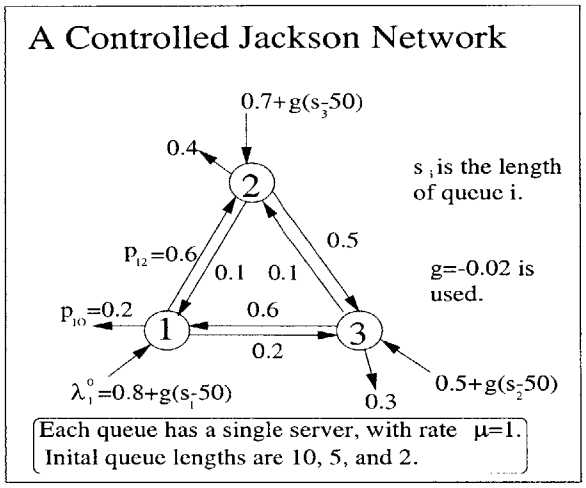


Figure 3.4: A three-site network of single-server queues with state-dependent input rates is shown. We track the lengths of the queues in this network at unit intervals (i.e., for $\Delta T = 1$). This example satisfies the three conditions required for analysis as an MLSN.

each process of the jobs that flow from a site i to another site j (or to the exterior of the system, for $j = 0$) is an independent Poisson process with rate $\lambda_i p_{ij}$. Thus, we know that the random variables $U_{ij}[k]$, $1 \leq i \leq n$, $0 \leq j \leq n$ are each independent Poisson random variables with means $\lambda_i p_{ij} \Delta T$.

Also, consider the distributions for the input flows $U_{0i}[k]$, given $\mathbf{s}[k]$. Since we have assumed that the rate of the input process remains essentially constant over intervals of duration ΔT, we see that, given $\mathbf{s}[k]$, each $U_{0i}[k]$ is an independent Poisson random variable with mean $u_{0i}[k] \triangleq u_{0i}(k\Delta T)\Delta T$. From the positive rate assumption, we further see that $u_{0i}[k] = \overline{u}_{0i} + g_i(s_i[k] - \overline{s}_i)$.

Thus, given $\mathbf{s}[k]$, each queue length $s_i[k+1]$ is found from the previous queue length $s_i[k]$ by adding and subtracting independent Poisson random variables with means that are either constants or linear functions of $s_i[k]$. Based on our analysis of the infinite-server queue in Section 2, it is clear that $\{\mathbf{s}[k]\}$ satisfies all moment-linearity conditions and so constitutes an MLSN. The first and second assumptions given above are central to this MLSN reformulation, since they are required to exclude non-linear features of the model and also ensure that the state update is Markovian.

If the state vector can be represented as an MLSN (i.e., if the three required assumptions

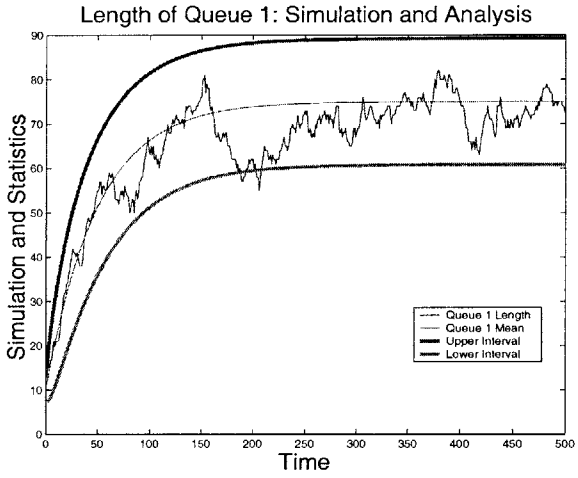


Figure 3.5: A simulation of the length of queue 1 in the example three-site network is shown. Also, the MLSN-based analysis of the model is used to approximate the mean queue length, and to construct 2σ intervals about the mean.

hold), then queue-length statistics can be found as a function of time using the moment recursions. We use the three-site model depicted in Figure 3.4 to illustrate this analysis. In particular, the MLSN formulation of the model is used to determine the expectation and variance of the length of the queue at Site 1, given an initially empty network. Mean and 2σ-intervals on the length of Queue 1 are plotted along with the simulation of the queue length in Figure 3.5. The simulation shown in Figure 3.5 verifies that the heavy-traffic assumption holds for Queue 1, and the heavy-traffic assumption can similarly be checked for the other two queues. The other two assumptions are also easy to check for this example. We are currently working to develop constraints on the parameters that guarantee the three conditions hold.

We note that the model described here can be generalized to allow certain state-dependencies in server rates and routing probabilities, as well as more general input-rate dependencies, while maintaining its tractability. Because these generalizations do not significantly alter the analysis of the model, we do not develop them in detail in this thesis.

Another interesting observation about the model is that, as long as the average queue lengths reach steady-state, average input rates and waiting times also can be characterized. The average input rates into the queue can be directly found from the average queue lengths, since these input rates are linear with respect to the queue lengths. Subsequently,

the average waiting times can be determined from the average average queue lengths and average input rates, using *Little's Theorem* (see, e.g., [83]).

3.3.3 Example: A Flow Model for an Air Traffic System

We have developed a dynamic stochastic model for aircraft counts in regions of an Air Traffic System (ATS) in [123]. This model is yet another flow network that is amenable to analysis as an MLSN.

In our model for an air traffic system, the state variables are the numbers of aircraft in n *Centers* or regions of the airspace, tracked at discrete times [123]. We denote the number of aircraft in Center i at a discrete time k by $s_i[k]$. These state variables change with time because of flows between sites (regions of the airspace), as well as flows into and out of the airspace.

First, between time-steps k and $k+1$, the state variables can change because of aircraft entering each Center upon departure from airports within the Center. The number of aircraft that depart from airports in Center i, $1 \leq i \leq n$, between times k and $k+1$ is modeled as a Poisson random variable $U_{0i}[k]$, with mean denoted by $\lambda_{0i}[k]$.

In addition to the flows into the ATS due to departures at airports, aircraft may change Centers, or leave a Center through arrival at an airport within the Center. In our model, we assume that each aircraft in Center i independently travels to Center j (or leaves the airspace for $j = 0$) between time-steps k and $k+1$ with probability $p_{ij}[k]$. We denote the total number of aircraft that flow from Center i to Center j between times k and $k+1$ by $U_{ij}[k]$. We denote the total number of aircraft that flow from Center i to Center j between times k and $k+1$ by $U_{ij}[k]$. If $\sum_{\substack{j=0 \\ j \neq i}}^{n} p_{ij}[k] << 1$, it can be shown the conditional distribution for the flow $U_{ij}[k]$ given the Center count $s_i[k]$ is well-approximated by a Poisson distribution, with mean flow $p_{ij}[k]s_i[k]$ [119]; the results that are presented here use this assumption, though the analysis is only slightly more complicated without this asumption (i.e., the system can still be analyzed as an MLSN, though the computation of moments is a bit more complicated). The dynamics of our model are depicted in Figure 3.6.

This model for an Air Traffic System is Markovian. In particular, each state variable $s_i[k+1]$ can be specified in terms of $s_i[k]$ and the flows $U_{ij}[k]$, which have specified distributions

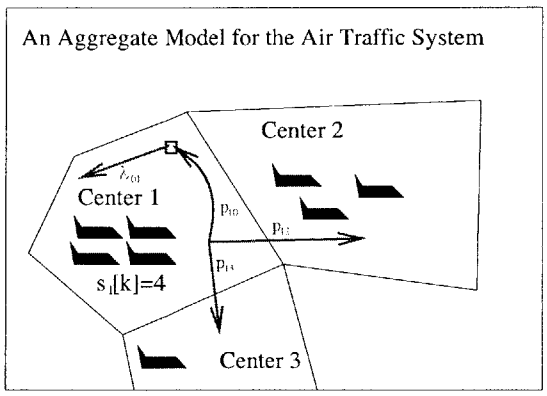


Figure 3.6: This figure shows the dynamics of our aggregate stochastic model for the ATS. Aircraft enter Centers according to Poisson processes. Also, during an interval of time, each aircraft in a Center may move to another Center or leave the system, with some probability. We are interested in tracking the number of aircraft in each Center in this model.

given $\mathbf{s}[k]$, as

$$s_i[k+1] = s_i[k] - \sum_{j=0, j\neq i}^{n} U_{ij}[k] + \sum_{j=0, j\neq i}^{n} U_{ji}[k], \tag{3.3}$$

It can be shown, using reasoning that is quite similar to that used in the analysis of the $M/M/\infty$ queue, that this model is an MLSN. An outline of the justification is given in [123]; we do not pursue it further here.

In the context of air traffic modeling, the MLSN moment recursions can be used to characterize the time-evolution of individual Center counts, and to indicate the nature of correlations among Center counts at particular time-steps. In [123], we have applied this model in representing Center counts in the U.S. ATS. Figure 3.7 compares model predictions for the numbers of aircraft in a particular Center, ZSE (which covers the Pacific Northwest region of the U.S.), with actual aircraft counts in ZSE during a span of approximately 12 hours[2]. The figure suggests that our model can capture both transient and steady-state dynamics of Center counts fairly well, though the model response tends to be sluggish compared to the actual transients.

It turns out that our model for an ATS can be viewed as a discrete-time network of infi-

[2]Some processing is required to select model parameters (i.e., $p_{ij}[k]$ and $\lambda_{0i}[k]$) based on historical data of flows in the ATS; this procedure is described in [123].

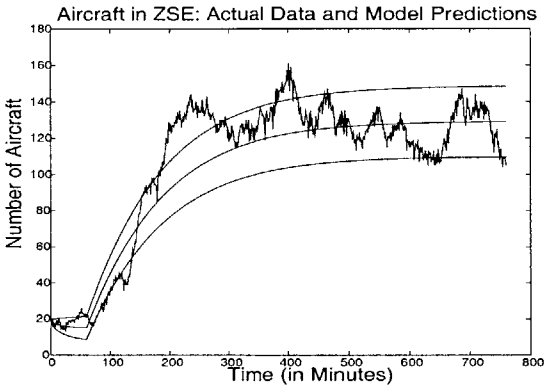


Figure 3.7: The actual number of aircraft in ZSE at 760 consecutive one-minute time steps is compared with the mean number of aircraft predicted by our model. This plot also includes the 2σ bounds on the aircraft count in ZSE predicted by our model. The actual data is largely contained within two standard deviations of the predicted mean, suggesting that the model predictions for the mean and standard deviation are both reasonable. The one noticeable difference between the actual data and the model prediction is the sluggishness in the model's transient as compared to the data.

nite server queues, by reasoning as follows. From a queueing theory viewpoint, we can describe each aircraft in a Center as being served independently at each time-step, with a probability $\sum_{\substack{j=0 \\ j\neq i}}^{n} p_{ij}[k]$ that service is completed at time-step k. In this interpretation, an aircraft that has completed service in Center i during time k then enters Center j (or leaves the system, for $j = 0$) by time $k+1$ with probability $\frac{p_{ij}[k]}{\sum_{\substack{j=0 \\ j\neq i}}^{n} p_{ij}[k]}$. Thus, we see that this model indeed can be envisioned as an open network of infinite server queues, in which served jobs (aircraft) are independently stochastically routed to other servers (regions), or out of the system.

Since our model can be viewed as a network of infinite-server queues, we briefly review relevant literature on such models. An introduction to discrete-time infinite-server queueing networks can be found in [40]. Infinite-server queueing networks with memoryless servers and Poisson input streams, which constitute a type of Jackson network with state dependent service rates, are particularly amenable to analysis [79]. For instance, the steady-state and transient joint distribution for the queue lengths in the network can be deduced [79]. Methods for characterizing (both transient and steady-state) first- and second-order statistics of queue lengths are well-known, and a characteristic-function method for inferring higher moments as a function of time has also been developed [88]. Our ap-

proach serves to reformulate these higher-moment computations as temporal linear recursions, and to provide a framework for further analysis (e.g., estimation and control) of these models.

3.3.4 Example: The Linear Routing Model

Our third example of an MLSN flow model differs from the first two models in that the state variables are not constrained to be integral. We call this model the *linear routing model* because flows in the network between times k and $k+1$ are determined by choosing among several linear combinations of the state variables at time k. We introduce and motivate this model through a toy example, in which we consider bulk traffic flow in a network of road segments.

A variety of stochastic models for traffic flows on single roads, and on networks of roads, have been developed (see, e.g., [70, 91, 41, 118, 108] for some of the more popular, and representative, models). These models span a variety of spatial and temporal scales, may be deterministic or stochastic, and may use continuous or discrete representations for flow densities. Our linear routing model, and associated traffic network model, are most closely related to macroscopic (bulk) dynamic models for flows (e.g., [70, 112, 103, 81]). In these models, as in ours, vehicle counts or densities in multiple interconnected road segments are tracked with time. Stochastics in these models are typically envisioned as orginating from congestion effects and routing of vehicles at junctions. Our model differs from these models in that routing is explicitly controlled by randomized traffic signalling, and in that congestion effects are not explicitly modeled. Although our traffic flow model is coarser in some respects than those found in the literature, we believe that it is valuable because network-level dynamics can be characterized (as discussed below), and relevant questions of estimation and control can be answered (as discussed in later chapters).

Consider the network of $n = 8$ one-way *road segments* and $z = 5$ *traffic lights* shown in Figure 3.8. Our goal is to model the numbers of vehicles on the n road segments, tracked at discrete intervals. In our model, these vehicle counts are allowed to be continuous variables, based on the motivation that we are modeling this network at a bulk level at which a fluid approximation for vehicle counts is reasonable. We denote the vehicle count on road segment i at time k by $s_i[k]$, and denote the state vector by $\mathbf{s}[k] \triangleq \begin{bmatrix} s_1[k] \\ \vdots \\ s_n[k] \end{bmatrix}$. These vehicle

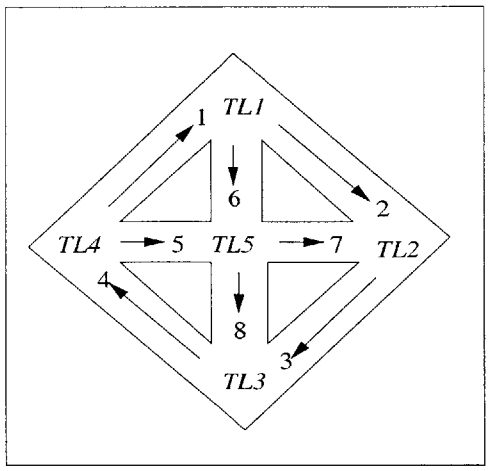


Figure 3.8: A traffic network comprising 8 one-way road segments and 5 traffic lights is shown.

counts change with time because of flows among road segments, as well as flows into and out of the traffic system. The flows are governed by the actions of the traffic lights in the model.

At each time-step k (i.e., during the interval between times k and $k+1$), each traffic light j is modeled as engaging in one of w_j *actions*. (For example, a traffic light may be envisioned as engaging in two actions, which correspond to the light being either red or green.) We assume that traffic light j chooses action $x \in 1, \ldots, w_j$ at time k with probability d_{jx}, independently of the actions of the other traffic lights at time k, and independently of all past history of the network. The action taken by traffic light j at time k is denoted $x_j[k]$.

Now consider the change in state variables between time-steps k and $k+1$ due to flows in the network. The flows leaving each road segment (either for other road segments, or out of the system) are governed by one of the $z = 5$ traffic lights. For instance, consider the vehicles on road segment 1, which are governed by traffic light 1. In this example, we assume that that this traffic light may either be green (denoted $x_1[k] = 1$) or red (denoted $x_1[k] = 2$). Whether the traffic light is green or red, we assume that 10% of the vehicles in the road segment 1 leave the network. If the traffic light is red, the remaining vehicles do not move from road segment 1. If the traffic light is green, we assume that 70% of the vehicles remaining in the system move to road segment 2, and the remaining vehicles move to road segment 6.

For this example, the flows in the network governed by traffic light 1 at time k (which, in this case, are the flows that originate from road segment 1, and go to road segments 2 and 6) can be written as a linear function of the state variables at time k for each possible traffic light action. That is, the change in the n state variables due to the actions of traffic light 1 take the form $H_1(x_1[k])\mathbf{s}[k]$, where $H_1(x_1[k] = 1)$ and $H_1(x_1[k] = 2)$ are two different $n \times n$ matrices. Similarly, we assume that the changes in state variables due to the actions of traffic light j can be written in the form $H_j(x_j[k])\mathbf{s}[k]$.

In this particular example, we assume that the number of vehicles entering each road segment i at time k is an independent Poisson random variable, with mean λ_i. The vector λ is defined as a n-component vector of these means.

Finally, we are ready to specify the state update of this model. Since each road segment is governed by one traffic light, the different flows due to the actions of different traffic lights are additive. Thus, $\mathbf{s}[k+1]$ can be found from $\mathbf{s}[k]$ as

$$\mathbf{s}[k+1] = \left(\sum_{j=1}^{z} H_j(x_j[k]) \right) \mathbf{s}[k] + H_0[k], \tag{3.4}$$

where H_0 is a vector containing the numbers of vehicles entering the road segments from outside the system at time k. We refer to systems with state updates in the form of Equation 3.4 as *linear routing models*. We note that a linear routing model is an instance of a Markovian jump-linear system, for which the underlying Markov chain has large state space but specially structured evolution.

The linear routing model can be shown to be an MLSN, with sites in the MLSN corresponding to road segments in our description of the model. This analysis is not pursued in detail here, but we do present the first-moment linearity condition as an example. The conditional expectation for $\mathbf{s}[k+1]$ given $\mathbf{s}[k]$ can be written as

$$\begin{aligned} E(\mathbf{s}[k+1] \,|\, \mathbf{s}[k]) &= \left(\sum_{j=1}^{z} E(H_j(x_j[k])) \right) \mathbf{s}[k] + E(H_0[k]) \\ &= \left(\sum_{j=1}^{z} \sum_{x=1}^{w_j} d_{xj} H_j(x) \right) \mathbf{s}[k] + \lambda. \end{aligned} \tag{3.5}$$

Using the MLSN formulation, we can compute statistics of vehicle counts in the traffic

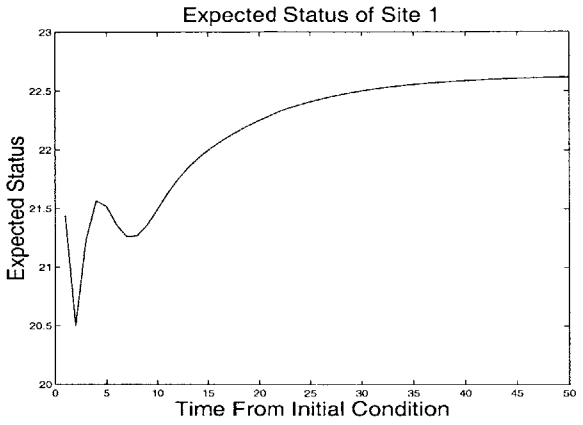


Figure 3.9: The first moment recursion is used to calculate the expected vehicle count in road segment 1 as a function of time. The expected number of vehicles in road segment 1 reaches a steady-state, even though the actual number continues to fluctuate.

network example. In Figure 3.9, we use the first-moment recursion to find the transient response of the expected number of vehicles in road segment 1 from a non-steady-state initial condition. The transient response of the mean depends on the initial counts in each road segment, but the mean does eventually reach a steady-state. In Figure 3.10, we plot steady-state correlations between the number of vehicles on road segment 1 and the numbers of vehicles on the other road segments.

3.3.4.1 Designing Traffic Flow: A First Experiment

We believe that MLSN models can serve as a framework for designing networks, because statistics of the network state can be easily computed and can possibly be related to network parameters. Although we have not yet studied MLSN design in a general sense, we describe a simple design experiment for the 8 road-segment traffic flow example, which highlights the possible value of using MLSN to study network design.

Our design experiment is motivated by an observation about the traffic flow on road segment 1 in the example network (Figure 3.8). In simulations of the model, we notice that the number of vehicles in road segment 1 occasionally becomes quite large, because traffic light 1 remains red for several time-steps while traffic light 4 remains green and so admits vehicles onto the road segment. By the same token, road segment 1 is often nearly empty,

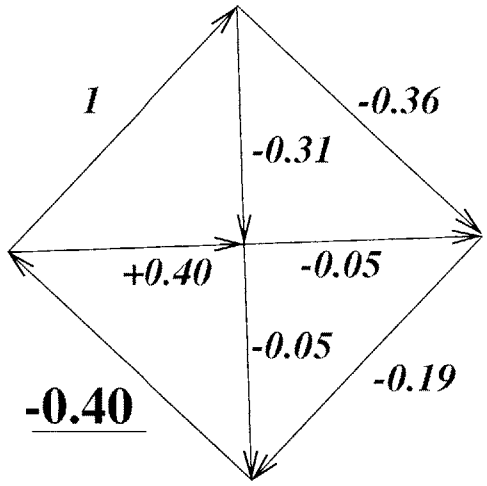


Figure 3.10: This plot shows steady-state correlations between the number of vehicles in road segment 1 and the number of vehicles in each of the other road segments at a particular time. The strong negative correlation between the vehicle counts in road segments 1 and 4 because a build-up of vehicles in road segment 1 corresponds to flow out of road segment 4 (because traffic light 4 is green).

since traffic light 1 may be green while traffic light 4 is red. This large variability in the flow in road segment 1 is indicated by the large steady-state standard deviation of the status $s_1[k]$, which can be found using the second moment recursion; for the set of network parameters that we have chosen, this standard deviation is 22.5, which is roughly equal to the steady-state mean number of vehicles in the road segment! The effect of the unsynchronized behavior of traffic lights 1 and 4 is also indicated by the strong negative correlation between the numbers of vehicles in road segments 1 and 4 at a particular time-step in steady-state (see Figure 3.10).

Because the mismatched timings of traffic lights 1 and 4 contribute to the variability in the flow on road segment 1, redesign of the network through synchronization of the traffic lights is compelling. As a first experiment toward such a design, we modify the traffic network as follows: we maintain the probabilities that each traffic light engages in each of its possible control actions from the original network. However, we no longer assume that the traffic lights operate independently. In particular, we assume that traffic lights 1 and 4 are perfectly synchronized. (Note that perfect synchronization suggests that the two traffic lights have the same probabilities of engaging in each action; the probabilities in the original traffic model were chosen to allow synchronization.)

It turns out that the network with synchronized traffic lights can also be formulated as an MLSN. We can show that the first moment recursion for the synchronized network is identical to the first moment recursion for the original model, so that the steady-state mean number of vehicles in road segment 1 remains the same as in the original model. However, the second moment recursion shows that the steady-state standard deviation in the traffic flow on road segment 1 drops to 16.4. A decrease in the magnitude of the negative correlation coefficient between the numbers of vehicles in road segments 1 and 4, from -0.4 to -0.16, is also observed. Thus, we expect that the number of vehicles on road segment 1 is less likely to be anomalously large or small for the synchronized model, as compared to the unsynchronized one. A histogram of the number of vehicles on road segment 1 verifies this advantage of synchronization (Figure 3.11).

3.4 Modeling Flow Networks

The last three examples that we introduced in Section 3.3 have in common that statuses represent quantities (i.e., numbers of items or amounts of material) present at sites, and these statuses change between successive time-steps due to the movement, or $flow$, of these quantities between adjoining sites in the network. In general, flow networks such as these are often good candidates for representation as MLSS, because the changes in nodal quantities are inherently linear with respect to the flows. For this reason, and because the notion of flows is common to three of our examples, we feel that is worthwhile to examine the dynamics of flow networks with moment-linear structure (henceforth referred to as *moment-linear flow networks*) in a general way.

The state update of a moment-linear flow network can be viewed as a two-stage process.

- First, *flow variables* at a time k, which describe the quantities that move between each pair of sites and move into and out of the network, are determined from the time-k state variables (statuses) in a moment-linear fashion. The moment-linear structure of this stage is dependent on the on the particulars of the represented system. For instance, in the air traffic model, the flows are Poisson random variables whose means are linear functions of the state variables, and hence the flow variables are moment linear with respect to the state variables. We use the notation $U_{ij}[k]$, $1 \leq i \leq n$, $1 \leq j \leq n$, $i \neq j$, to represent the flow from site i to site j at time k. We also denote the flow into site i from outside the system at time k by $U_{0i}[k]$, and the flow exiting the system from site i

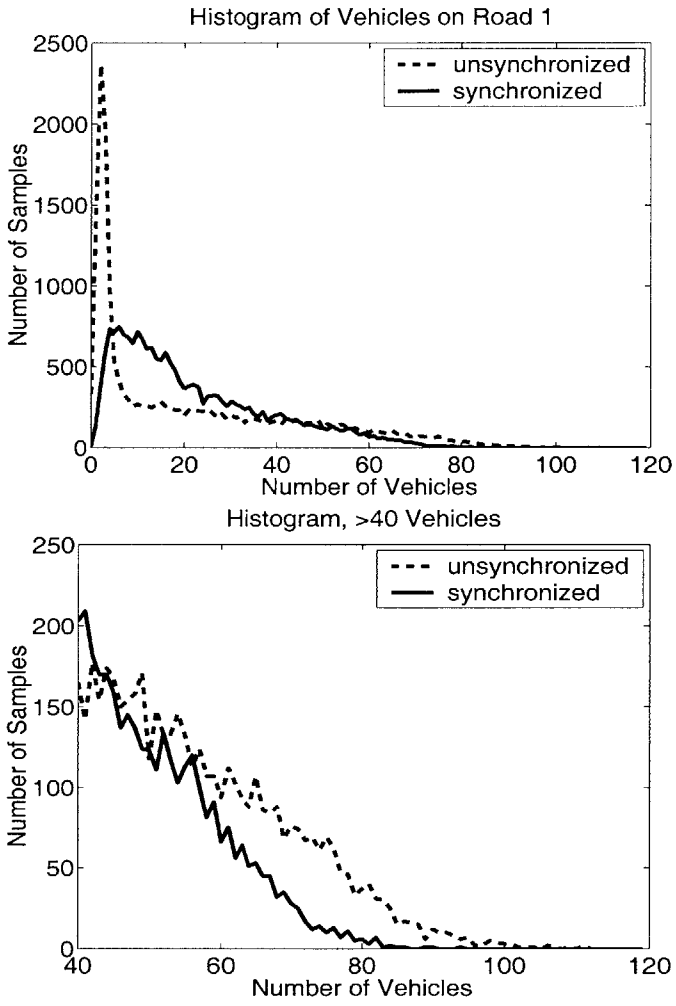


Figure 3.11: The plots compare histograms of the number of vehicles on road segment 1 in the unsynchronized and synchronized traffic flow networks. The histograms were generated based on single 10000 time-step simulations of each network. The upper plot shows the entire histogram, while the lower plot expands the histogram for large vehicle counts ($\geq$ 40). The histogram shows that the synchronized system is less prone to anamolously large and anomalously small flows than the unsynchronized system.

by $U_{i0}[k]$.

- Second, the time-$k+1$ state variables are computed from the time-k flow variables, by adding and subtracting the appropriate flow variables. That is, we compute the time-$(k+1)$ state variables from the time-k state variables and time-k flow variables, as follows:

$$s_i[k+1] = s_i[k] + \sum_{\substack{j=0 \\ j \neq i}}^{n} U_{ji}[k] - \sum_{\substack{j=0 \\ j \neq i}}^{n} U_{ij}[k], \tag{3.6}$$

  for $1 \leq i \leq n$. This stage of the update is identical for all flow models. It is structured so that the time-$(k+1)$ state variables are moment linear (in fact, purely linear) with respect to the time-k state and flow variables.

We can show that that state update of this model is moment linear, in the following way. We first define an augmented state vector that lists all time-k state and flow variables, and verify that this augmented state vector is moment linear with respect to the time-k state vector. Next, we note that the time-$(k+1)$ state vector is moment linear with respect to the augmented time-k state vector. Finally, by invoking the law of conditional expectations, we can show that the time-$(k+1)$ state vector is moment linear with respect to the time-k state vector. In this way, we can express the moment recursion matrices in terms of products of matrices that relate the moments of the time-$k+1$ state to the time-k augmented state and matrices that relate the time-k augmented state to the time-k state. We note that that the matrices relating the the moments of the time-$k+1$ state and the time-k augmented state depend only on the dimension of the state vector, not the particulars of the network stochastics.

We believe that this two-step formulation is useful, for a couple reasons. First, it clarifies that the three flow network examples can be viewed in a common framework, in which flows add to and subtract from site quantities, but the specifics of these flows depend on the particulars of the modeled systems. Second, the special structure of the update can potentially be used to develop results that apply to moment-linear flow networks generally.

3.5 Graphs for MLSN: Ideas and Difficulties

Since we view the four examples introduced in the last section as networks, it is natural to ask whether the interactions or interdependencies among their sites can be illustrated by a graph. For some of the examples, graphs play a central role in specifying the updating of the sites' statuses. For instance, in the influence model, a directed *network graph* is used to specify the weights of the influences between pairs of sites. Similarly, in the model for an ATS, aircraft are constrained to flow between contiguous regions in the airspace, so that a map of connections between regions is relevant in defining the flows. In the road traffic network, however, a graph with directed edges between "connected" sites (i.e., road segment pairs in which vehicles can move from one road segment to the other during a single time-step) does not aptly describe the state update. This is because the flows from a particular road segment to other road segments are strongly correlated, based on the action of the governing traffic lights. Thus, an apt graphical illustration for the state update would require representations for groups of sites, or alternately explicit representation of the traffic lights. The road traffic example highlights why a graph description for general MLSN is complicated—pairwise interaction among sites may not capture all relevant interactions in the network. Nevertheless, it is reasonable to consider graphs of the network that illustrate update interactions or dependencies in the first conditional moment expression, as long as we keep in mind that these graphs only partially capture update interactions. In similar fashion, graphs that illustrate update interactions among groups of sites or dependencies in higher conditional moments can be developed. Some caution is needed in developing graphs for illustrating higher moment conditions, because redundancies in the higher moment vectors can lead to non-unique specifications for the graph. We leave it to future work to precisely define general graphs for MLSN, and to determine the relationships among various graphical representations. An abstract graphical illustration of an MLSN is shown in Figure 3.12.

3.6 A Block Representation for Structured MLSS Dynamics

As we briefly mentioned before, for network examples with vector (rather than scalar) status representations, we find it useful to re-order the moment recursions so as to clarify interactions and status configurations of groups of sites. This section introduces a *block representation* for MLSS (and, in particular, MLSN) dynamics, which allows us to naturally

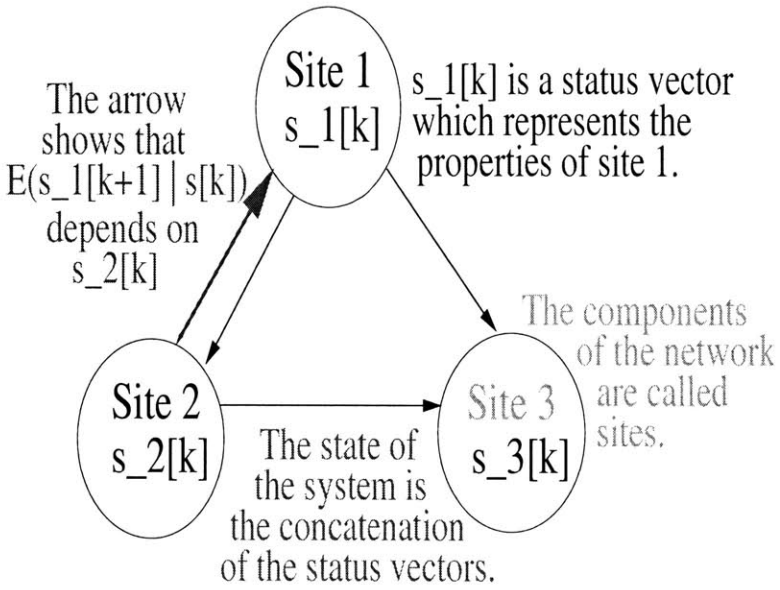


Figure 3.12: This figure illustrates the features of an MLSN. An MLSN is an informal term for an MLSS which seeks to model the interdependent evolution of parts of a network. Our picture shows one possible graphical representation for MLSS: here, an arrow indicates dependence of the conditional mean of the next-status at one site on the current status of a neighbor (possibly including itself).

represent interactions among vector site statuses.

3.6.1 A Block Kronecker Product

To construct a block representation for structured MLSS dynamics, it is useful to define a block Kronecker product notation. In particular, consider two vectors $\mathbf{x}$ and $\mathbf{y}$ that are partitioned into $n(\mathbf{x})$ and $n(\mathbf{y})$ subvectors, respectively, as follows: $\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_{n(\mathbf{x})} \end{bmatrix}$ and $\mathbf{y} =$

$\begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_{n(\mathbf{y})} \end{bmatrix}$. Then we define the *block Kronecker product* of $\mathbf{x}$ and $\mathbf{y}$, denoted $\mathbf{x} \boxtimes \mathbf{y}$, to be

$$\mathbf{x} \boxtimes \mathbf{y} = \left[\begin{array}{c} \mathbf{x}_1 \otimes \mathbf{y}_1 \\ \vdots \\ \mathbf{x}_1 \otimes \mathbf{y}_{n(\mathbf{y})} \\ \hline \vdots \\ \hline \mathbf{x}_{n(\mathbf{x})} \otimes \mathbf{y}_1 \\ \vdots \\ \mathbf{x}_{n(\mathbf{x})} \otimes \mathbf{y}_{n(\mathbf{y})} \end{array}\right]. \tag{3.7}$$

In words, the block Kronecker product is a list of the Kronecker products of each pair of subvectors of $\mathbf{x}$ and $\mathbf{y}$ (rather than a list of the scalar products of each pair of entries of $\mathbf{x}$ and $\mathbf{y}$, as in a standard Kronecker product).

We note that the entries of $\mathbf{x} \boxtimes \mathbf{y}$ are obtained by a simple permutation of the entries of $\mathbf{x} \otimes \mathbf{y}$. We specify the permutation matrix relating $\mathbf{x}$ and $\mathbf{y}$ in Appendix B.

3.6.2 Block Representations for MLSS

In the examples of network dynamics that we have discussed, we have associated statuses or status vectors with each site in the network. We have then shown that the state process of the system—where the state is defined to be the concatenation of the status vectors—is an MLSS. Thus, we see that it may be natural to view the state $\mathbf{s}[k]$ of an MLSS as being partitioned into subvectors.

In particular, say that the state $\mathbf{s}[k]$ of an MLSS is partitioned into n subvectors $\mathbf{s}_1[k], \ldots, \mathbf{s}_n[k]$ (i.e., $\mathbf{s}[k] = \begin{bmatrix} \mathbf{s}_1[k] \\ \vdots \\ \mathbf{s}_n[k] \end{bmatrix}$). In keeping with our network perspective, let's call these n subvectors *status vectors*, and think of them as being associated with n *sites*.

In MLSS with partitioned states of this sort, it may be natural to specify the dynamics in

terms of interactions between sites or groups of sites, and to characterize moments and cross-moments of status vectors. For this reason, it is useful to re-order the rth-order state vector in such a way that Kronecker products of status vectors form contiguous parts of it. To this end, we define the *permuted* rth-*order* state vector, denoted $\mathbf{s}_{[r]}[k]$ to be

$$\mathbf{s}_{[r]}[k] = \mathbf{s}[k]^{\boxtimes r}. \tag{3.8}$$

That is, the permuted rth-order state vector is the block Kronecker product of the vector $\mathbf{s}[k]$ with itself r times. We can straightforwardly show that the entries of $\mathbf{s}_{[r]}[k]$ are obtained by permuting those of $\mathbf{s}[k]^{\otimes r}$. Details of this permutation can be found in Appendix B.

3.6.3 Further Notation

In Chapter 7, we will use the block representation for MLSS in specifying higher-moment recursions for the influence model. In this context, the moment recursion matrices are developed by considering interactions among groups of sites. In constructing these recursion matrices, we often find it useful to work with Kronecker products of status vectors of groups of sites. It becomes tedious to present expressions in terms of such Kronecker products, so we define a little further notation to simplify presentation of Kronecker products of multiple status vectors.

First, we define an rth-order *grouping* $\mathbf{v}$ as an ordered list of r (not necessarily distinct) sites: $\mathbf{v} = \{v_1, \ldots, v_r\}$, where each $v_i \in 1, \ldots, n$. We define the *joint status vector* of the rth-order grouping $\mathbf{v}$ at time k as

$$\mathbf{s}_{\mathbf{v}}[k] = \mathbf{s}_{v_1}[k] \otimes \mathbf{s}_{v_2}[k] \otimes \ldots \otimes \mathbf{s}_{v_r}[k]. \tag{3.9}$$

The joint status vector of a grouping is simply a concise notation for the Kronecker product of the status vectors of the sites in that grouping.

The permuted rth-order state vector $\mathbf{s}_{[r]}[k]$ is a list of all joint status vectors of rth-order groupings, arranged in *lexicographic order* according to the grouping (see Appendix A) for a definition of lexicographic order).

Chapter 4

Further Analysis of MLSS and MLSN Dynamics

In this chapter, we extend the basic analysis of MLSS and MLSN dynamics in several directions, and also develop some methodologies that are required for these analyses.

The chapter is organized as follows.

- In Section 1, we outline an approach for computing statistics *across* time-steps.
- In Section 2, we discuss reduced formulations for state and moment vectors, which eliminate redundancies in these vectors. These reduced representations are required for the developments in the remainder of the chapter.
- In Section 3, we characterize aspects of the asymptotic dynamics of MLSS and MLSN. We discuss several notions of convergence, and also mention non-convergent dynamics that can occur in MLSS and MLSN.

4.1 Statistics Across Time-Steps

The moment-linear structure of our model also facilitates calculation of statistics *across* time-steps (i.e., cross-moments between state variables at multiple time-steps). Such statistics are valuable in that they can identify aspects of the stochastic dynamics that persist through time. In the literature, correlations among state variables at two time-steps are commonly studied for stochastic systems (see, e.g., [113], [114], [54] for a few applications). Such correlations have been found for some network models; for instance, correlations across time-steps have been derived for a particular cellular automaton [117], and for a model for computer network traffic [128]. In the context of signal processing, these correlations play a role in several important theoretical developments, including the analysis

of spectra of stochastic systems and the development of filters for noise-reduction in linear systems (see, e.g., [113]). Some (comparatively) recent work has focused on *higher-order statistics* (i.e., third and higher moments/cross-moments and cumulants/cross-cumulants) of linear and nonlinear systems, and their application in signal identification and processing (see, e.g., [99],[100],[109]). Of particular interest to us is the development of Kronecker-product expressions for higher-order statistics of linear systems ([133],[99]). Like these articles, we also compute higher-order statistics across time-steps in a Kronecker product representation, but for MLSS rather than linear systems.

4.1.1 Formulation

We describe a method for finding expectations of Kronecker powers of extended state vectors (of various degrees) at multiple time-steps for an MLSS (or MLSN). That is, we consider the expectation

$$E\left(\otimes_{i=1}^{T}\mathbf{s}_{(r_i)}[k_i]\right), \tag{4.1}$$

where we have assumed without loss of generality that $k_1 > k_2 > \ldots > k_T \geq 0$. The expectation 4.1 contains statistics of state variables across times $k_1, \ldots, k_T$. Specifically, Equation 4.1 contains all moments and cross-moments of the form $E\left(\prod_{i=1}^{T}\prod_{j=1}^{m} s_j[k_i]^{\epsilon_{ij}}\right)$, where each α_{ij} is a non-negative integer and $\sum_{j=1}^{m}\alpha_{ij} \leq r_i$ for each i.

Example 4.1
To find cross-correlations between state variables at two time-steps, k_1 and k_2, with $k_1 > k_2$ (i.e., expectations of the form $E(s_i[k_1]s_j[k_2])$), we evaluate the expectation $E(\mathbf{s}_{(1)}[k_1] \otimes \mathbf{s}_{(1)}[k_2])$ according to the procedure developed in the following subsection. Note that the expectation that we evaluate contains not only cross-correlations between the two time-steps but also expectations of state variables at each time-step.

4.1.2 Evaluation of Equation 4.1

In this subsection, we show how to evaluate the expectation 4.1. For clarity, we limit our analysis to time-invariant MLSS.

Our approach for determining the expectation 4.1 is to rewrite it in terms of an expectation

of Kronecker products of extended state vectors at times $k_2, \ldots, k_T$. The process used to rewrite the expectation can then be applied iteratively to find the Expectation 4.1 solely in terms of an extended moment vector at time k_T. Finally, a moment recursion can be used to find the extended moment vector at time k_T, as described in Chapter 2 .

We rewrite the expectation 4.1 by conditioning on $\mathbf{s}[k_2], \ldots, \mathbf{s}[k_T]$, as follows:

$$\begin{aligned} & E\left(\otimes_{i=1}^{T} \mathbf{s}_{(r_i)}[k_i]\right) \\ &= E\left(E\left(\otimes_{i=1}^{T} \mathbf{s}_{(r_i)}[k_i] \,|\, \mathbf{s}[k_2], \ldots, \mathbf{s}[k_T]\right)\right) \\ &= E\left(E\left(\mathbf{s}_{(r_1)}[k_1] \,|\, \mathbf{s}[k_2], \ldots, \mathbf{s}[k_T]\right) \otimes \left(\otimes_{i=2}^{T} \mathbf{s}_{(r_i)}[k_i]\right)\right) \\ &= E\left(E\left(\mathbf{s}_{(r_1)}[k_1] \,|\, \mathbf{s}[k_2]\right) \otimes \left(\otimes_{i=2}^{T} \mathbf{s}_{(r_i)}[k_i]\right)\right) \\ &= E\left(\tilde{H}_{(r_1)}^{k_1-k_2} \mathbf{s}_{(r_1)}[k_2] \otimes \left(\otimes_{i=2}^{T} \mathbf{s}_{(r_i)}[k_i]\right)\right). \end{aligned} \tag{4.2}$$

The *mixed-product property* of the Kronecker product (see, e.g., [72]) can then be used to rewrite (4.2) as

$$\begin{aligned} & E\left(\otimes_{i=1}^{T} \mathbf{s}_{(r_i)}[k_i]\right) \\ &= \left(\tilde{H}_{(r_1)}^{k_1-k_2} \otimes \mathbf{I}_{\alpha(1)}\right) E\left(\left(\mathbf{s}_{(r_1)}[k_2] \otimes \mathbf{s}_{(r_2)}[k_2]\right) \otimes \left(\otimes_{i=3}^{T} \mathbf{s}_{(r_i)}[k_i]\right)\right), \end{aligned} \tag{4.3}$$

where $\mathbf{I}_{\alpha(1)}$ is an identity matrix with dimension $\alpha(1)$ equal to the total length of the vector $\otimes_{i=2}^{T} \mathbf{s}_{(r_i)}[k_i]$, or $\alpha(1) = \prod_{j=2}^{T} \frac{m^{r_j+1}-1}{m-1}$.

To continue the analysis, note that $\mathbf{s}_{(r_1)}[k_2] \otimes \mathbf{s}_{(r_2)}[k_2]$ contains only monomials of time k_2 state variables with total degree less than or equal to $r_1 + r_2$. Thus, all entries of $\mathbf{s}_{(r_1)}[k_2] \otimes \mathbf{s}_{(r_2)}[k_2]$ are also contained in the vector $\mathbf{s}_{(r_1+r_2)}[k_2]$, and so $\mathbf{s}_{(r_1)}[k_2] \otimes \mathbf{s}_{(r_2)}[k_2]$ can be written as a linear function of $\mathbf{s}_{(r_1+r_2)}[k_2]$. That is, there is a matrix L_{r_1,r_2} such that

$$\mathbf{s}_{(r_1)}[k_2] \otimes \mathbf{s}_{(r_2)}[k_2] = L_{r_1,r_2} \mathbf{s}_{(r_1+r_2)}[k_2]. \tag{4.4}$$

Some algebra is required to construct the mapping matrix L_{r_1,r_2}; this algebra is discussed in Appendix B[1].

[1]Note that the form of the matrix L_{r_1,r_2} depends only on the dimension m of the state vector, and on r_1 and r_2; it does not depend on the time-step k_2 or the value of the state vector. In other words, given m, we can construct a set of matrices L_{r_1,r_2} that specify maps between Kronecker products of extended state vectors and higher extended state vectors, regardless of the specifics of the MLSS or of the computation of interest.

Substituting Equation 4.4 into Equation 4.3, we find that

$$E\left(\otimes_{i=1}^{T}\mathbf{s}_{(r_i)}[k_i]\right) \tag{4.5}$$
$$= \left(\tilde{H}_{(r_1)}^{k_1-k_2} \otimes \mathbf{I}_{\alpha(1)}\right) E\left(L_{r_1,r_2}\mathbf{s}_{(r_1+r_2)}[k_2] \otimes \left(\otimes_{i=3}^{T}\mathbf{s}_{(r_i)}[k_i]\right)\right).$$

We again use the mixed-product property to rewrite Equation 4.5 as

$$E\left(\otimes_{i=1}^{T}\mathbf{s}_{(r_i)}[k_i]\right) \tag{4.6}$$
$$= \left(\tilde{H}_{(r_1)}^{k_1-k_2} \otimes \mathbf{I}_{\alpha(1)}\right)\left(L_{r_1,r_2} \otimes \mathbf{I}_{\beta(1)}\right) E\left(\otimes_{i=2}^{T}\mathbf{s}_{(\widehat{r}_i)}[k_i]\right),$$

where $\widehat{r}_2 = r_1 + r_2$, $\widehat{r}_j = r_j$ for $j \geq 3$, and $\beta(1)$ is equal to the length of the vector $\otimes_{i=3}^{T}\mathbf{s}_{(r_i)}[k_i]$—i.e. $\beta(1) = \prod_{j=3}^{T}\frac{m^{r_j+1}-1}{m-1}$.

Equation 4.6 shows that $E\left(\otimes_{i=1}^{T}\mathbf{s}_{(r_i)}[k_i]\right)$ can be written as a linear function of the expectation of Kronecker products of extended state vectors at times $k_2, \ldots, k_T$. By iteratively applying the analysis described above $T-1$ times in total, we can find $E\left(\otimes_{i=1}^{T}\mathbf{s}_{(r_i)}[k_i]\right)$ in terms of an extended moment vector at time k_T, and so (by applying an extended moment recursion) in terms of an extended moment vector at the initial time 0. This iterative procedure yields the following expression for $E\left(\otimes_{i=1}^{T}\mathbf{s}_{(r_i)}[k_i]\right)$:

$$E\left(\otimes_{i=1}^{T}\mathbf{s}_{(r_i)}[k_i]\right) \tag{4.7}$$
$$= \left(\prod_{i=1}^{T-1}\left(\tilde{H}_{(\tilde{r}_i)}^{k_i-k_{i+1}} \otimes \mathbf{I}_{\alpha(i)}\right)\left(L_{\tilde{r}_i,r_{i+1}} \otimes \mathbf{I}_{\beta(i)}\right)\right) \tilde{H}_{(\tilde{r}_T)}^{k_T} E(\mathbf{s}_{(\tilde{r}_T)}[0]),$$

where $\tilde{r}_i = \sum_{j=1}^{i} r_i$, $\alpha(i) = \prod_{j=i+1}^{T}\frac{m^{r_j+1}-1}{m-1}$, $\beta(i) = \prod_{j=i+2}^{T}\frac{m^{r_j+1}-1}{m-1}$ for $i \leq T-2$, and $\beta(T-1) = 1$, $\mathbf{I}_{\alpha(i)}$ and $\mathbf{I}_{\beta(i)}$ are identity matrices of dimension $\alpha(i)$ and $\beta(i)$ respectively, and $(L_{\tilde{r}_i,r_{i+1}}$ specifies the linear map $\mathbf{s}_{(\tilde{r}_i)}[k_{i+1}] \otimes \mathbf{s}_{(r_{i+1})}[k_{i+1}] = L_{\widehat{r}_1,r_2}\mathbf{s}_{(\widehat{r}_1+r_2)}[k_2]$.

Though the notation in Equation 4.7 is complicated, the basic concept behind the equation can be simply understood, by considering the terms in the equation from right to left. In this way, we see that Equation 4.7 iteratively computes statistics across groups of time-steps of the form $k_T, \ldots, k_i$ (i.e., across some of the time-steps in the set $k_T, \ldots, k_1$, starting from the earliest time-step). Statistics of sufficiently high order are determined at each iteration to eventually allow calculation of $E\left(\otimes_{i=1}^{T}\mathbf{s}_{(r_i)}[k_i]\right)$. It's interesting to note that the computational complexity of Equation 4.7 is similar to that of the recursion used to find statistics of degree $\tilde{r}_T = \sum_{j=1}^{T} r_j$ at a single time-step.

The procedure described here can also straightforwardly be applied to a time-varying

model: the resulting expression for the expectation 4.1 then contains products of extended recursion matrices at multiple time-steps, rather than powers of a single extended recursion matrix. In determining statistics across time-steps for MLSN in particular, it may be useful to rearrange the entries in the expectation 4.1 to reflect interactions among sites. These rearrangements are similar in spirit to the permuted state vectors developed in Section 3.6; we do not consider the rearrangements any further here.

Example 4.2

In this example, we apply the procedure above to find the expectation $E(\mathbf{s}_{(1)}[k_1] \otimes \mathbf{s}_{(1)}[k_2])$, $k_1 > k_2$, *for an MLSS with* $m = 2$ *state variables. Our discussion of this special case serves to highlight some of the details of the general analysis above.*

By following the procedure described in Equation 4.2 and then applying the mixed-product property of Kronecker products, we can rewrite $E(\mathbf{s}_{(1)}[k_1] \otimes \mathbf{s}_{(1)}[k_2])$ *as*

$$E\left(\mathbf{s}_{(1)}[k_1] \otimes \mathbf{s}_{(1)}[k_2]\right) = (\tilde{H}_{(1)}^{k_1-k_2} \otimes I_3) E(\mathbf{s}_{(1)}[k_2] \otimes \mathbf{s}_{(1)}[k_2]), \tag{4.8}$$

where I_3 *is an* 3×3 *identity matrix. Note that* $E(\mathbf{s}_{(1)}[k_2] \otimes \mathbf{s}_{(1)}[k_2])$ *is a vector of length* $(m+1)^2 = 9$, *and that* $\tilde{H}_{(1)}^{k_1-k_2} \otimes I_{m+1}$ *is a matrix of dimension* $(m+1)^2 \times (m+1)^2 = 9 \times 9$, *so the matrix-vector multiply in Equation 4.8 is properly dimensioned.*

The next step in the analysis is to rewrite $E(\mathbf{s}_{(1)}[k_2] \otimes \mathbf{s}_{(1)}[k_2])$ *as a linear function of* $E(\mathbf{s}_{(1+1)}[k_2]) = E(\mathbf{s}_{(2)}[k_2])$. *For this simple example, this linear map can be constructed by inspection, as follows:*

$$\begin{aligned} E(\mathbf{s}_{(1)}[k_2] \otimes \mathbf{s}_{(1)}[k_2]) &= E\left(\begin{bmatrix} \mathbf{s}[k] \\ 1 \end{bmatrix}^{\otimes 2}\right) \\ &= E\left(\begin{bmatrix} s_1^2[k_2] \\ s_1[k_2]s_2[k_2] \\ s_1[k_2] \\ s_1[k_2]s_2[k_2] \\ s_2^2[k_2] \\ s_2[k_2] \\ s_1[k_2] \\ s_2[k_2] \\ 1 \end{bmatrix}\right) \end{aligned} \tag{4.9}$$

$$
\begin{aligned}
&= \begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & & & 1 & & & \\ & & & 1 & & & & \\ & & & & 1 & & & \\ & & & & & & 1 & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{bmatrix} E\left(\begin{bmatrix} s_1^2[k_2] \\ s_1[k_2]s_2[k_2] \\ s_1[k_2]s_2[k_2] \\ s_2^2[k_2] \\ - - - - \\ s_1[k_2] \\ s_2[k_2] \\ - - - - \\ 1 \end{bmatrix} \right) \\
&= L_{1,1} E(\mathbf{s}_{(2)}[k_2]).
\end{aligned}
$$

Substituting into Equation 4.8 gives

$$E\left(\mathbf{s}_{(1)}[k_1] \otimes \mathbf{s}_{(1)}[k_2]\right) = (\tilde{H}_{(1)}^{k_1-k_2} \otimes I_3) L_{1,1} E(\mathbf{s}_{(2)}[k_2]). \tag{4.10}$$

Finally, by invoking the second extended moment recursion, we can write Equation 4.10 as

$$E\left(\mathbf{s}_{(1)}[k_1] \otimes \mathbf{s}_{(1)}[k_2]\right) = (\tilde{H}_{(1)}^{k_1-k_2} \otimes I_3) L_{1,1} \tilde{H}_{(2)}[k]. \tag{4.11}$$

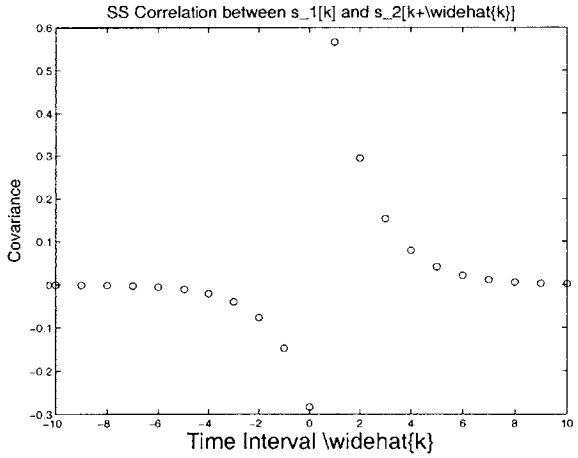


Figure 4.1: The steady-state normalized cross-correlation function between the stochastic process of the number of jobs in the queue and the stochastic process of the number of exiting jobs is shown. That is, we plot the steady-state value for $E(s_1[k]s_2[k+\widehat{k}]) - E(s_1[k])E(s_2[k+\widehat{k}])$ as a function of the time interval $\widehat{k}$.

Example 4.3

Let's revisit the infinite server queue with random service probabilities introduced in Section 2.4.5. Using the techniques described in this section, we have found the steady-state normalized cross-correlation function between the number of jobs in the queue and the number of jobs exiting the queue, as shown in Figure 4.1. As one might expect, the number of jobs currently in the queue is positively correlated with the number of jobs released during future time intervals: if there is currently a large number of jobs in the queue, it is likely that a larger-than-average number of jobs will be released during future time intervals. The value of this positive correlation decreases with time, as the average behavior of the system returns to steady-state. Also, Figure 4.1 shows that the number of jobs in the queue at a certain time is negatively correlated with the number of released jobs at previous times: the number of jobs in the queue at a certain time reflects the number of jobs that have not been released at previous times, explaining the negative correlation. Interestingly, this negative correlation is less pronounced than the positive correlation at future times, because a larger number of jobs in the queue at the current time-step can also indicate a larger number of jobs, and larger service rate, at previous time-steps.

4.2 Reduced Representations for MLSS

4.2.1 Motivation and Approach

Our development so far has been concerned with representing example systems as MLSS and MLSN, and then computing state statistics for these systems using linear moment recursions. Though we have noted the our representations for state statistics contain redundancies (both because of repeated cross-moments in the vector moments and because of intrinsic contraints on state variables in some examples), these redundancies have been tangential to our analysis so far. In subsequent sections of this chapter, we will seek to relate qualitative features of MLSS dynamics (e.g., asymptotic and settling characteristics) with MLSS parameters (e.g., eigenvalues of recursion matrices). In these analyses, the degeneracies in recursion matrices resulting from redundant moment vectors are significant, since the MLSS parameters of interest may depend on the form of the recursion matrices considered. Thus, it is important for us to structure moment representations in such a way that meaningful characteristics of model dynamics can be gleaned from parameters. Further, appropriate forms for recursion matrices are valuable because they can simplify analysis of parameters such as dominant or subdominant eigenvalues.

With this motivation in mind, we discuss reduced representations for moment vectors and the corresponding moment recursions. For the sake of convenience, we constrain our discussion to time-invariant or stationary MLSS, though similar representations for time-varying MLSS can be constructed.

4.2.2 Reduced Representation: Definitions

We construct (as done implictly in [9], in developing higher moment recursions for the influence model) reduced representations for MLSS state and moment vectors by identifying subsets of the entries in these vectors that are sufficient for determining the entire vector.

In particular, consider an MLSS of degree $\widehat{r}$. For any $r \leq \widehat{r}$, a set of vectors $\widehat{\mathbf{s}}_{[1]}[k], \ldots, \widehat{\mathbf{s}}_{[r]}[k]$ is called an *rth reduced state vector set* if all entries of each $\widehat{\mathbf{s}}_{[i]}[k]$, $1 \leq i \leq r$, are entries of $\mathbf{s}[k]^{\otimes i}$, and further each $\mathbf{s}[k]^{\otimes i}$ can be written in the form $\sum_{j=1}^{i} M_{ij}\widehat{\mathbf{s}}_{[j]}[k] + N_i$, for some $M_{i,1}, \ldots, M_{i,i}$ and N_i, and for all k. In words, $\widehat{\mathbf{s}}_{[1]}[k], \ldots, \widehat{\mathbf{s}}_{[r]}[k]$ is an rth reduced state vector set if the entries of $\widehat{\mathbf{s}}_{[i]}[k]$ are drawn from $\mathbf{s}[k]^{\otimes i}$, and if each $\mathbf{s}[k]^{\otimes i}$ can be written as a linear function of $\widehat{\mathbf{s}}_{[1]}[k], \ldots, \widehat{\mathbf{s}}_{[i]}[k]$. As a direct consequence of the definition, the vector $\widehat{\mathbf{s}}_{[r]}[k]$ can always be written in the form $L_r\mathbf{s}[k]^{\otimes r}$ for some matrix L_r, since each element of $\widehat{\mathbf{s}}_{[r]}[k]$ is an element of $\mathbf{s}[k]^{\otimes r}$.

Note that we have defined the rth reduced state vector set so that each subset $\widehat{\mathbf{s}}_{[1]}[k], \ldots, \widehat{\mathbf{s}}_{[i]}[k]$ is a sufficient statistic for $\mathbf{s}[k]^{\otimes i}$, and further $\mathbf{s}[k]^{\otimes i}$ is a sufficient statistic for $\widehat{\mathbf{s}}_{[i]}[k]$. Thus, we might hope to develop moment recursions for *reduced moment vectors* (i.e., expectations of reduced state vectors), by exploiting the mappings between the reduced and original state vectors. The ensuing discussion shows that such reduced moment recursions can indeed be constructed.

Example 4.4

Consider the MLSS formulation for a three-status Markov chain (see Section 2.4.2). Recall that the state vector $\mathbf{s}[k] = \begin{bmatrix} s_1[k] \\ s_2[k] \\ s_3[k] \end{bmatrix}$ *at each time k in the MLSS formulation is a length-3 indicator vector for the status of the Markov chain. We claim that a 1st reduced state vector set for this example is given by (the single vector)* $\widehat{\mathbf{s}}_{[1]}[k] = \begin{bmatrix} s_1[k] \\ s_2[k] \end{bmatrix}$. *To check, note the following:*

- *Each entry of* $\widehat{\mathbf{s}}_{[1]}[k]$ *is an entry of* $\mathbf{s}[k]$.

- *Since the state vector is an indicator, the final entry of* $\mathbf{s}[k]$ *is exactly specified by the first two entries, as* $s_3[k] = 1 - s_1[k] - s_2[k]$. *Thus,* $\mathbf{s}[k]$ *can be written in terms of* $\widehat{\mathbf{s}}_{[1]}[k]$ *as*

$$\mathbf{s}[k] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & -1 \end{bmatrix} \widehat{\mathbf{s}}_{[1]}[k] + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{4.12}$$

Thus, the conditions required for $\widehat{\mathbf{s}}_{[1]}[k]$ *to be a* 1*st reduced state vector set are satisfied. Also note that* $\widehat{\mathbf{s}}_{[1]}[k]$ *can be written as a linear function of* $\mathbf{s}[k]$, *as*

$$\widehat{\mathbf{s}}_{[1]}[k] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{s}[k]. \tag{4.13}$$

In order to reformulate the extended moment recursions in the framework of the reduced state vectors, it is useful for us to assemble the first r reduced state vectors into a single vector. Thus, we define the *extended reduced state vector* of degree r to be

$$\widehat{\mathbf{s}}_{(r)}[k] = \begin{bmatrix} \widehat{\mathbf{s}}_{[r]}[k] \\ \widehat{\mathbf{s}}_{[r-1]}[k] \\ \vdots \\ \widehat{\mathbf{s}}_{[1]}[k] \end{bmatrix} \tag{4.14}$$

(Note that, unlike the extended state vector, we do not include a unity entry at the bottom of the extended reduced state vector.)

4.2.3 Relationships between Extended State Vectors and Extended Reduced State Vectors

Assembling the linear relations between higher state vectors (i.e., Kronecker products of the state vector) and reduced state vectors, we find that the extended state vector can be expressed in terms of the extended reduced state vector as

$$\mathbf{s}_{(r)}[k] = M_{(r)} \widehat{\mathbf{s}}_{(r)}[k] + N_{(r)}, \tag{4.15}$$

where

$$M_{(r)} = \begin{bmatrix} M_{r,r} & M_{r,r-1} & \cdots & M_{r,1} \\ & M_{r-1,r-1} & \cdots & M_{r,1} \\ & & \ddots & \\ & & & M_1 \\ & & & 0 \end{bmatrix}$$

and

$$N_{(r)} = \begin{bmatrix} N_r \\ N_{r-1} \\ \vdots \\ N_1 \\ 1 \end{bmatrix}.$$

Similarly, the extended reduced state vector can be written in terms of the extended state vector as

$$\widehat{\mathbf{s}}_{(r)}[k] = L_{(r)}\mathbf{s}_{(r)}[k], \tag{4.16}$$

where

$$L_{(r)} = \begin{bmatrix} L_r & & & \\ & L_{r-1} & & \\ & & \ddots & \\ & & & L_1 \quad 0 \end{bmatrix}$$

Example 4.5

Let's again consider the three-status Markov chain from Example 4.4. Note that the 1st extended state vector for this example is the length-4 vector $\mathbf{s}_{(1)}[k] = \begin{bmatrix} \mathbf{s}[k] \\ 1 \end{bmatrix}$*, while the first extended reduced state vector is the length-2 vector* $\widehat{\mathbf{s}}_{(1)}[k] = \widehat{\mathbf{s}}_{[1]}[k]$*. From the above discussion relating extended state vectors with extended reduced state vectors, we see that*

$$\mathbf{s}_{(1)}[k] = M_{(1)}\widehat{\mathbf{s}}_{(1)}[k] + N_{(1)}, \tag{4.17}$$

where

$$M_{(1)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & -1 \\ 0 & 0 \end{bmatrix} \tag{4.18}$$

and

$$N_{(1)} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}. \tag{4.19}$$

Also, we can write the extended reduced state vector as

$$\widehat{\mathbf{s}}_{(1)}[k] = L_{(1)}\mathbf{s}_{(1)}[k], \tag{4.20}$$

where

$$L_{(1)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.21}$$

4.2.4 Recursions for Reduced Moment Vectors

Because of the linear mappings between the extended state vectors and the extended reduced state vectors, we can develop recursions for extended reduced moment vectors (i.e., expectations of extended reduced state vectors), as follows:

$$\begin{aligned} E(\widehat{\mathbf{s}}_{(r)}[k+1]) &= L_{(r)}E(\mathbf{s}_{(r)}[k+1]) \\ &= L_{(r)}\tilde{H}_{(r)}E(\mathbf{s}_{(r)}[k]) \\ &= L_{(r)}\tilde{H}_{(r)}M_{(r)}E(\widehat{\mathbf{s}}_{(r)}[k]) + L_{(r)}\tilde{H}_{(r)}N_{(r)} \end{aligned} \tag{4.22}$$

From here on, we use the term *extended reduced recursion matrix* for the matrix $\widehat{H}_{(r)} \triangleq L_{(r)}\tilde{H}_{(r)}M_{(r)}$, and use the term *extended reduced recursion vector* for the vector $\widehat{B}_{(r)} \triangleq L_{(r)}\tilde{H}_{(r)}N_{(r)}$. In this notation, the recursion in Equation 4.22 is given by

$$E(\widehat{\mathbf{s}}_{(r)}[k+1]) = \widehat{H}_{(r)}E(\widehat{\mathbf{s}}_{(r)}[k]) + \widehat{B}_{(r)}. \tag{4.23}$$

Example 4.6

Again consider the three-status Markov chain from examples 4.4 and 4.5. Say that the transition matrix for the Markov chain is $D = \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.1 & 0.6 & 0.3 \\ 0 & 0.3 & 0.7 \end{bmatrix}$. *Using the procedure above, we can construct the recursion for the 1st reduced vector moment. This recursion turns out to be*

$$E(\widehat{\mathbf{s}}_{(1)}[k+1]) = \begin{bmatrix} 0.7 & 0.1 \\ -0.1 & 0.3 \end{bmatrix} E(\widehat{\mathbf{s}}_{(1)}[k]) + \begin{bmatrix} 0.1 \\ 0.3 \end{bmatrix} \tag{4.24}$$

4.2.5 More on the Extended Reduced Recursion Matrix and Vector

Because of the way reduced state vectors have been defined, the extended reduced recursion matrix and vector are specially structured. In particular, the matrix $\widehat{H}_{(r)}$, like $\check{H}_{(r)}$, is block upper-triangular:

$$\widehat{H}_{(r)} = \begin{bmatrix} \widehat{H}_{r,r} & \widehat{H}_{r,r-1} & \dots & \widehat{H}_{r,1} \\ 0 & \widehat{H}_{r-1,r-1} & \dots & \widehat{H}_{r-1,1} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \widehat{H}_{1,1} \end{bmatrix}, \tag{4.25}$$

where $\widehat{H}_{i,j} = L_i \sum_{t=j}^{i} H_{i,t} M_{t,j}$, for $i \geq j$.

Like $\widehat{H}_r$, $\widehat{B}_{(r)}$ can be related to the original recursion matrices of the MLSS, as

$$\widehat{B}_{(r)} = \begin{bmatrix} \widehat{B}_1 \\ \vdots \\ \widehat{B}_r \end{bmatrix}, \tag{4.26}$$

where $\widehat{B}_i = L_i(\sum_{t=1}^{i} H_{i,t} N_t + H_{i,0})$.

4.2.6 Eliminating Cross-Moment Repetitions

As we have already discussed, state and moment vectors can be redundant both because of system-specific constraints on the dynamics, and because of repetitions of cross-moments in the vector moments. Cross-moment repetitions are identical in all MLSS, so they can be

eliminated in a systematic fashion. We discuss one approach for constructing a reduced state vector set that eliminates repeated cross-moments.

In order to construct this reduced representation, it is useful to develop a notation for products of groups of state variables. To do so, define an ith *state variable grouping* to be an ordered list of i indices of state variables. That is, an ith state variable grouping $\mathbf{v}$ is a list of the form $\mathbf{v} = \{v_1, \ldots, v_i\}$, where each $v_j \in 1, \ldots, m$. Next, define the *state variable product* at time k for the state variable grouping $\mathbf{v}$ as

$$s_{\mathbf{v}}[k] = \prod_{j=1}^{i} s_{v_j}[k]. \tag{4.27}$$

In total, there are m^i distinct ith state variable groupings. The vector $\mathbf{s}[k]^{\otimes i}$ is simply a list of the corresponding m^i state variable products, arranged in lexicographic order according to the grouping.

Redundancies in moment vectors come about because some entries in $\mathbf{s}[k]^{\otimes i}$ are identical. In particular, consider any two entries in $\mathbf{s}[k]^{\otimes i}$ corresponding to state variable groupings that are *rearrangements* of each other (by rearrangement, we mean that the one state variable grouping can be constructed by reordering the other grouping). Based on the definition of the state variable product, these entries are identical (i.e., they are defined by the same function of the state variables). Thus, their expectations must be identical, and so the entries introduce redundancies in the extended moment vectors.

To construct an rth reduced state vector set which eliminates thes redundancies, let's define an ith *primary state variable grouping*, $1 \leq i \leq r$, to be a state variable grouping $\mathbf{v}$ for which $v_1 \leq \ldots \leq v_i$. Note that there are $\binom{m+i-1}{m-1}$ distinct ith primary state variable groupings. Define the vector $\widehat{\mathbf{s}}_{[i]}[k]$ to be a list of the state variable products corresponding to these state variable groupings, arranged in lexicographic order. Note the following:

- Each entry in $\mathbf{s}[k]^{\otimes i}$ is identical to an entry in $\widehat{\mathbf{s}}_{[i]}[k]$, and vice versa. Thus, $\mathbf{s}[k]^{\otimes i}$ can be written in the form $M_{i,i}\widehat{\mathbf{s}}_{[i]}[k]$, and $\widehat{\mathbf{s}}_{[i]}[k]$ can be written as $L_i\mathbf{s}[k]^{\otimes i}$. Thus, $\widehat{\mathbf{s}}_{[1]}[k], \ldots, \widehat{\mathbf{s}}_{[r]}[k]$ constitute an rth valid reduced state vector set.
- In general, two state variable products are guaranteed to be equal only if their state variable groupings are rearrangements of each other. Note that the state variable groupings corresponding to any two entries in $\widehat{\mathbf{s}}_{[i]}[k]$ cannot be rearrangements of each other, so the $\widehat{\mathbf{s}}_{[i]}[k]$ generally do not contain identical entries. Thus, expectations of the vectors

$\widehat{\mathbf{s}}_{[i]}[k]$ do not generally have redundant moments and cross-moments.

We have shown how to construct reduced state vector sets so as to eliminate redundant cross-moments. In our discussion, we have not explicitly shown how to construct the matrices $M_{i,i}$ and L_i. These mapping matrices are constructed in in Appendix B.

Example 4.7
Consider an MLSS with $m = 2$ state variables, and consider the second-moment vector. In order to eliminate redundant cross-moments in the second-moment vector, we must construct a reduced representation that eliminates repeated entries in the vector $\mathbf{s}[k]^{\otimes 2} = \begin{bmatrix} s_1^2[k] \\ s_1[k]s_2[k] \\ s_2[k]s_1[k] \\ s_2^2[k] \end{bmatrix}$. *For this example, we can use the vector* $\widehat{\mathbf{s}}_{[2]}[k] = \begin{bmatrix} s_1^2[k] \\ s_1[k]s_2[k] \\ s_2^2[k] \end{bmatrix}$ *as the reduced representation. Note that* $\mathbf{s}[k]^{\otimes 2}$ *and* $\widehat{\mathbf{s}}_{[2]}[k]$ *are related by*

$$\mathbf{s}[k]^{\otimes 2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \widehat{\mathbf{s}}_{[2]}[k], \tag{4.28}$$

and

$$\widehat{\mathbf{s}}_{[2]}[k] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{s}[k]^{\otimes 2}. \tag{4.29}$$

4.2.7 Minimal Representations

It is useful for us to define *minimality* of reduced representations. That is, we would like to develop a condition for checking whether or not any further reduction in a particular state representation is possible. The way in which we define a minimal state vector set is motivated by the convergence analyses that we pursue in Section 4.3; for now, we simply present the condition for minimality.

Consider an rth reduced state vector set for a particular MLSS. We define the minimality of this set in terms of its extended reduced moment vector $E(\widehat{\mathbf{s}}_{(r)}[k])$. First, we define the concept of a *valid initial condition* for the extended reduced moment vector. In particular, we say that $\tilde{\mathbf{s}}$ is a valid initial condition for the extended reduced moment vector if we can specify a valid probability distribution on the initial state $\mathbf{s}[0]$ (which meets the constraints of the particular MLSS) such that $E(\widehat{\mathbf{s}}_{(r)}[k]) = \tilde{s}$.

Next, we define minimality as follows. We say that the rth reduced state vector set is an rth minimal state vector set if there exists a vector $\bar{\mathbf{s}}$, positive constant ϵ, and norm $||.||$, such that $\bar{\mathbf{s}} + \delta$ is a valid initial condition for the corresponding extended reduced moment vector, for all δ such that $||\delta|| \leq \epsilon$. In words, a reduced state vector set is defined to be minimal if there is a valid initial moment vector that can be perturbed in any way by a small amount and still remain a valid initial moment vector.

Example 4.8

We again consider the three-status Markov chain of Examples 4.4, 4.5, and 4.6. In particular, consider the 1*st reduced state vector set* $\widehat{\mathbf{s}}_{[1]}[k] = \begin{bmatrix} s_1[k] \\ s_2[k] \end{bmatrix}$*, and consider the initial extended reduced moment vector,* $E(\widehat{\mathbf{s}}_{[1]}[0]) = E\left(\begin{bmatrix} s_1[0] \\ s_2[0] \end{bmatrix}\right)$*. In this example, the two entries of this vector,* $E(s_1[0])$ *and* $E(s_2[0])$*, are the probabilities that the Markov chain are in status* 1 *and* 2*, respectively, at the initial time-step. These initial status probabilities are constrained to be positive, and* $E(s_1[0]) + E(s_2[0])$ *must be less than* 1*, but they can otherwise be arbitrary. Thus, we can prove that the reduced state vector set is minimal, as follows. Choose* $\bar{\mathbf{s}} = \begin{bmatrix} 0.6 \\ 0.2 \end{bmatrix}$*, and* $\epsilon = 0.1$*. Then we can clearly construct an initial state distribution such that* $E(\widehat{\mathbf{s}}_{[1]}[0]) = \bar{\mathbf{s}} + \delta$ *for* $||\delta||_1 < \epsilon$*, so the state vector set is minimal.*

4.2.8 A Subspace Interpretation to Our Development

In this section, we have developed a formalism for eliminating redundancies in the vector moments of an MLSS by using reduced state vector representations for the MLSS. In concluding the section, we briefly discuss a second approach for accounting for redundancies in the vector moments, without explicitly constructing reduced state vector sets.

To do so, consider an rth extended vector moment $\mathbf{s}_{(r)}[k]$ for an MLSS of dimension m.

Note that $\mathbf{s}_{(r)}[k]$ is a real vector of dimension $\widehat{m}_r = \sum_{i=0}^{r} m^i$—i.e., it is defined in the space $R^{\widehat{m}_r}$. However, any redundancies in the vector $\mathbf{s}_{(r)}[k]$ correspond to linear constraints on its entries (e.g., they force some of the entries to be identical); in this case, $\mathbf{s}_{(r)}[k]$ is constrained to a subspace of $R^{\widehat{m}_r}$, which we call the *relevant subspace*.

The reduced representations that we have developed eliminate these constraints by reformulating the extended vector moments in a lower-dimensional subspace. Alternately, we can maintain the dimension of the extended vector moment but explicitly identify and account for the constraints placed by the redundancies. In particular, we can identify subspaces within the span of the recursion matrices that the vector moment is confined to when excited by initial vectors in the relevant subspace. This approach has been used in [9] to identify *irrelevant* modes (eigenvector directions) in the influence model's recursion matrices. We will use a similar approach in studying the influence model.

4.3 Asymptotics of MLSS

Our aim here is to outline methods for characterizing the asymptotic dynamics of MLSS and MLSN, and to study the asymptotics of several examples. We caution that our approach is rudimentary in many respects: because we study methods that are general to MLSS and MLSN, our methods overlook some features of the asymptotics in particular models that are related to the specifics of these models' dynamics. Nevertheless, our approach provides some fresh insights into the asymptotic dynamics of several examples and identifies some key characteristics of models that define their asymptotic behavior.

We focus primarily on studying moment convergence of MLSS in the following discussion, but also touch on several other types of asymptotics, including mean-square convergence, distributional convergence, and some non-convergent dynamics.

4.3.1 Moment Convergence

The moments of random vectors strongly characterize their distribution, and so the asymptotic dynamics of moments of random processes are of some interest. For instance, moment asymptotes can sometimes be used to prove convergence of the state distribution [62, 2, 92], or to bound the probability that the state strays far from its mean at large time (see, e.g.,

[62]). The large-deviations theory also uses moment information to bound the probability that a random variable strays far from its mean, and has been applied in studying asymptotics of, e.g., queueing networks ([89]). The asymptotics of the moments are also valuable in proving, or disproving, some notions regarding convergence of the state of a random process, such as *convergence in mean-square* and *convergence in probability* (see, e.g., [62]). We will discuss these notions of convergence and their connection with moment convergence in later parts of this section.

Techniques for checking moment convergence (and for the corresponding steady-state moment analysis) are well-known for discrete-time linear systems driven by stochastic inputs. The moment dynamics, and asymptotics, of MLSS are in general richer than those of purely linear systems, as highlighted in our analysis and examples.

Definitions

We define the notion of *moment convergence* in MLSS and MLSN in terms of the extended moment vectors, as follows. Consider an MLSS (or MLSN) of degree $\widehat{r}$. For $r \leq \widehat{r}$, we say that the MLSS is *strictly rth-moment convergent* or rth-moment convergent to the vector $\overline{s}_{(r)}$ if $\lim_{k\to\infty} E(\mathbf{s}_{(r)}[k]) = \overline{s}_{(r)}$, for any (possibly random) initial condition $\mathbf{s}_{(r)}[0]$ that has its first r moments finite.

Often, a weaker notion of moment convergence which allows for the asymptotics of the extended moment vector to depend on the initial state distribution is useful. Thus, we say that an MLSS is *marginally* rth-moment convergent if $\lim_{k\to\infty} E(\mathbf{s}_{(r)}[k])$ exists for any $\mathbf{s}_{(r)}[0]$ that has first r moments finite. Note that the value of the limit may in general depend on $E(\mathbf{s}_{(r)}[0])$ for marginally moment convergent MLSS.

An MLSS that is not marginally rth-moment convergent is said to be rth-moment *nonconvergent*. For an rth-moment nonconvergent MLSS, there is at least one $E(\mathbf{s}_{(r)}[0])$ for which $\lim_{k\to\infty} E(\mathbf{s}_{(r)}[k])$ does not exist.

Our objective is to develop methods for relating the moment convergence properties of an MLSS to its parameters.

The Direct Method Fails

The extended moment vectors of an MLSS satisfy linear recursions of the form $E(\mathbf{s}_{(r)}[k+1]) = \tilde{H}_r E(\mathbf{s}_{(r)}[k])$. Thus, we can directly develop conditions for rth moment convergence of MLSS in terms of the eigenvalues and eigenvectors of $\tilde{H}_r$, using standard linear systems results (e.g., [42]). However, because the extended moment vectors may in general be redundant, this direct approach may fail. In particular, $\tilde{H}_{(r)}$ may have eigenvalues that seem to indicate the rth extended moment vector does not converge, while in fact these eigenvalues are irrelevant in the sense that the corresponding eigenvector directions cannot be excited by a valid extended moment vector. Thus, we often cannot develop necessary conditions for convergence, nor tight sufficient conditions, solely in terms of the eigenvalues and eigenvectors of $\tilde{H}_{(r)}$.

By eliminating the redundancies in the extended moment vectors, good sufficient conditions (and in some cases necessary conditions) for moment convergence can be developed. This development is a primary motivation for the reduced representations of MLSS developed in Section 4.2.

Example 4.9

Consider a Markovian jump-linear system with a two-status underlying Markov chain, specified by the transition matrix $D = \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix}$. *Define* $\mathbf{q}[k]$ *to indicate the status of the underlying Markov chain. Assume that the scalar continuous-valued state is updated as follows:*

$$\begin{aligned} x[k+1] &= a_1 x[k] + b_1, \quad \text{if } \mathbf{q}'[k] = [1,0] \\ x[k+1] &= a_2 x[k] + b_2, \quad \text{if } \mathbf{q}'[k] = [0,1]. \end{aligned} \tag{4.30}$$

Recall from Section 2.4.3 that this jump-linear system can be reformulated as an MLSS with state $\mathbf{s}[k] = \mathbf{q}[k] \otimes \begin{bmatrix} x[k] \\ 1 \end{bmatrix}$. *We would like to determine sufficient conditions on* D, a_1, b_1, a_2, *and* b_2, *for strict* 1*st-moment convergence of* $\mathbf{s}[k]$ *(which in turn proves* 1*st-moment convergence of the continuous-valued state).*

From Equation 2.29 in Section 2.4.3, we see that

$$\tilde{H}_{(1)} = \begin{bmatrix} d_{11}\begin{bmatrix} a_1 & b_1 \\ 0 & 1 \end{bmatrix} & d_{21}\begin{bmatrix} a_2 & b_2 \\ 0 & 1 \end{bmatrix} & \mathbf{0} \\ d_{12}\begin{bmatrix} a_1 & b_1 \\ 0 & 1 \end{bmatrix} & d_{22}\begin{bmatrix} a_2 & b_2 \\ 0 & 1 \end{bmatrix} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 1 \end{bmatrix} \tag{4.31}$$

It is easy to check that $\tilde{H}_{(1)}$ has two unity eigenvalues regardless of the parameters D, a_1, b_1, a_2, and b_2. One of these unity eigenvalues corresponds to the unity entry at the bottom right of $\tilde{H}_{(1)}$, and is clearly insignificant to the dynamics of the state $\mathbf{s}[k]$. However, the other unity eigenvalue is associated with the dynamics of the underlying Markov chain and cannot be seen to be irrelevant by inspection. Thus, we cannot straightforwardly prove strict 1st-moment convergence of this MLSS by considering the eigenstructure of $\tilde{H}_{(1)}$. More generally, rth-moment convergence cannot straightforwardly be understood in terms of the eigenvalues of $\tilde{H}_{(r)}$.

Moment Convergence and Reduced Representations: Sufficiency

Tight sufficient conditions for rth-moment convergence of an MLSS can sometimes be developed by considering rth reduced state vector sets for the MLSS. In particular, say that an rth reduced state vector set has been developed for an MLSS, and consider the extended reduced moment vector $E(\widehat{\mathbf{s}}_{(r)}[k])$ for this representation.

Equation 4.22 (in Section 4.2.4) specifies the linear recursion satisfied by $E(\widehat{\mathbf{s}}_{(r)}[k])$. Let us consider convergence of this linear recursion. From standard linear systems theory, we see that $E(\widehat{\mathbf{s}}_{(r)}[k])$ is strictly convergent if the extended reduced recursion matrix $\widehat{H}_{(r)}$ has eigenvalues that are strictly less than 1 in magnitude.

Next, note that convergence of $E(\widehat{\mathbf{s}}_{(r)}[k])$ implies convergence of the extended moment vector $E(\mathbf{s}_{(r)}[k])$, since $E(\mathbf{s}_{(r)}[k])$ is a linear function of $E(\widehat{\mathbf{s}}_{(r)}[k])$. In consequence, we have proven that strict rth moment convergence of an MLSS is guaranteed if the eigenvalues of its extended reduced moment recursion $\widehat{H}_{(r)}$ are strictly less than 1.

In a similar fashion, we can show that an MLSS is marginally rth moment convergent if the magnitudes of all eigenvalues of $\widehat{H}_{(r)}$ are less than or equal to 1, all unity magnitude eigenvalues are actually 1, each unity eigenvalue corresponds to a different *Jordan block*

(see, e.g., [42] for definition) of $\widehat{H}_{(r)}$, and the left eigenvectors corresponding to the unity eigenvalues are all orthogonal to $\widehat{B}_{(r)}$ (recall that $\widehat{B}_{(r)}$ was defined in Equation 4.23.

Recall that an extended reduced recursion matrix $\widehat{H}_{(r)}$ is necessarily block upper-triangular (see Equation 4.25). Thus, the eigenvalues of $\widehat{H}_{(r)}$ can be determined by computing the eigenvalues of each of its diagonal blocks. The block upper-triangular structure can also be used to simplify computation of the eigenvectors of $\widehat{H}_{(r)}$, as described in [131]. These simplifications in the eigenanalysis of $\widehat{H}_{(r)}$ are especially valuable for developing sufficient conditions for convergence that are phrased explicitly in terms of the parameters of a particular MLSS, and for analyzing large systems with potentially high computational cost.

Example 4.10

We develop meaningful conditions for strict rth-moment convergence of the jump-linear system considered in Example 4.9, using a reduced representation for this system.

To decide how to construct a useful reduced state vector set for this example, let's first explore the structures of the state vectors $\mathbf{s}[k]^{\otimes i}$*, to better understand their degeneracies. With a little thought (and some algebra), we see that the entries of* $\mathbf{s}[k]^{\otimes i}$ *all have one of the following three forms:*

$$\begin{aligned} q_1[k](x[k])^j, \quad & 0 \leq j \leq i \\ q_2[k](x[k])^j, \quad & 0 \leq j \leq i \\ 0. & \end{aligned} \tag{4.32}$$

Thus, it is clear that $\mathbf{s}[k]^{\otimes i}$ *is redundant in several senses:*

- *Entries in the vector* $\mathbf{s}[k]^{\otimes i}$ *are repeated several times.*
- *Entries in the vector* $\mathbf{s}[k]^{\otimes i}$ *are contained in* $\mathbf{s}[k]^{\otimes j}$*,* $j < i$*. For instance,* $q_1[k](x[k])^j$*,* $j < i$ *is also contained in the vector* $\mathbf{s}[k]^{\otimes j}$*.*
- *The two entries* $q_1[k]$ *and* $q_2[k]$ *are redundant, since* $q_2[k] = 1 - q_1[k]$*.*

To eliminate these redundancies, we can construct an rth reduced state vector set $\widehat{\mathbf{s}}_{[1]}[k], \ldots, \widehat{\mathbf{s}}_{[r]}[k]$

as follows:

$$\widehat{\mathbf{s}}_{[1]}[k] = \begin{bmatrix} q_1[k]x[k] \\ q_2[k]x[k] \\ q_1[k] \end{bmatrix} \tag{4.33}$$

$$\widehat{\mathbf{s}}_{[i]}[k] = \begin{bmatrix} q_1[k](x[k])^i \\ q_2[k](x[k])^i \end{bmatrix}, \quad 2 \leq i \leq r.$$

It is clear from Equation 4.32 that the entries of each $\widehat{\mathbf{s}}_{[i]}[k]$ in Equation 4.33 are contained in $\mathbf{s}[k]^{\otimes i}$, and that each entry in of $\mathbf{s}[k]^{\otimes i}$ can be written as a linear function of the entries in $\widehat{\mathbf{s}}_{[1]}[k], \ldots, \widehat{\mathbf{s}}_{[i]}[k]$. Thus, the vectors specified in Equation 4.33 are indeed a valid reduced state vector set.

Since $\widehat{\mathbf{s}}_{[1]}[k], \ldots, \widehat{\mathbf{s}}_{[r]}[k]$ is a reduced state vector set, we can test for rth-moment convergence by determining the eigenvalues of the extended reduced moment recursion matrix $\widehat{H}_{(r)}$ corresponding to this reduced state vector set. Since we are only concerned with determining the eigenvalues of $\widehat{H}_{(r)}$, we need only construct the diagonal blocks of $\widehat{H}_{(r)}$. These diagonal blocks turn out to be simple to construct from the system's stochastic description, so we compute the diagonal blocks in this direct way rather than by relating $\widehat{H}_{(r)}$ to $\tilde{H}_{(r)}$. To do so, note that $E(q_1[k+1] \,|\, \mathbf{s}[k])$ can be rewritten as $E(q_1[k+1] \,|\, q_1[k]) = d_{11}q_1[k] + d_{21}(1 - q_1[k]) = (d_{11} - d_{21})q_1[k] + d_{21}$. Also, consider $E(q_1[k+1](x[k+1])^i \,|\, \mathbf{s}[k])$. It is easy to check that this expectation can be written as $d_{11}q_1[k](a_1x[k] + b_1)^i + d_{21}q_2[k](a_2x[k] + b_2)^i$. Similarly, $E(q_2[k+1](x[k+1])^i \,|\, \mathbf{s}[k])$ can be written as $d_{12}q_1[k](a_1x[k] + b_1)^i + d_{22}q_2[k](a_2x[k] + b_2)^i$. Assembling the appropriate terms from these expectations, we see that the diagonal blocks of $\widehat{H}_{(r)}$ are given by

$$\widehat{H}_{1,1} = \begin{bmatrix} d_{11}a_1 & d_{21}a_2 & -- \\ d_{12}a_1 & d_{22}a_2 & -- \\ 0 & 0 & d_{11} - d_{21} \end{bmatrix} \tag{4.34}$$

$$\widehat{H}_{i,i} = \begin{bmatrix} d_{11}a_1^i & d_{21}a_2^i \\ d_{12}a_1^i & d_{22}a_2^i \end{bmatrix}, \quad i > 1,$$

where the symbol $--$ has been used to represent entries that are irrelevant to the eigenanalysis and so need not be specified.

By determining the eigenvalues of the $\widehat{H}_{i,i}$, $1 \leq i \leq r$, we can specify a sufficient condition for rth moment convergence In particular, we find that this jump-linear system is guaranteed to be

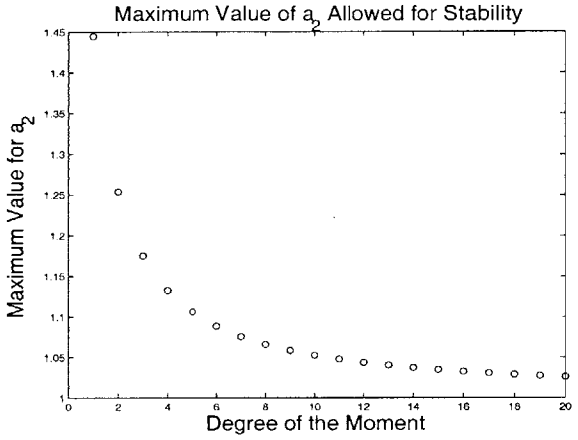


Figure 4.2: In this plot, we show that the parameter values required for moment convergence of a jump linear system are a function of the degree of moment convergence. In particular, we consider the system with $d_{11} = 0.7$, $d_{12} = 0.3$, $d_{21} = 0.4$, $d_{22} = 0.6$, and $a_1 = 0.5$. We use the condition 4.35 to plot as a function of r the maximum a_2 for which the system is guaranteed to be rth-moment convergent. Note that an a_2 guaranteeing rth-moment convergence can be greater than 1 (so that the system $x[k+1] = a_2 x[k] + b_2$ would be unstable) for each r. However, a_2 must be at most 1 if all moments of the jump linear system are to converge.

rth-moment convergent if the following $r + 1$ conditions hold:

$$\begin{aligned} &|d_{11} - d_{21}| < 1 \\ &\rho\left(\begin{bmatrix} d_{11}a_1^i & d_{21}a_2^i \\ d_{12}a_1^i & d_{22}a_2^i \end{bmatrix}\right) < 1, \quad 1 \leq i \leq r, \end{aligned} \tag{4.35}$$

where $\rho([H])$ is the spectral radius *of the matrix H (i.e., the maximum among the magnitudes of the eigenvalues of H). In Figure 4.10, we use these conditions to explore how the parameter values needed for moment convergence can change with the moment of interest.*

Example 4.11

We also show strict rth-moment convergence for the MLSN model for bulk road traffic flow (Section 3.3.4), given some general conditions on flows in the network. Recall that the state update for the road traffic network has the form $\mathbf{s}[k+1] = \left(\sum_{j=1}^{z} H_j(x_j[k])\right)\mathbf{s}[k] + H_0[k]$, where the matrices $H_j(x_j[k])$ describe the movements of vehicles among (and out of) the road segments due to the probabilistic actions $x_j[k]$ of the z traffic lights, and the random vector $H_0[k]$ describes flows into each road segment from outside the system.

To show moment convergence, we specify a reduced state vector set $\widehat{\mathbf{s}}_{[1]}[k], \ldots, \widehat{\mathbf{s}}_{[r]}[k]$. *In this example, we choose the reduced state vectors in the obvious way, as* $\widehat{\mathbf{s}}_{[i]}[k] = s[k]^{\otimes i}$ *(so that the extended reduced state vector is identical to the extended state vector, except for an extra unity entry at the end of the extended state vector). Although the reduced state vectors have some degeneracy, since entries are repeated multiple times, they are sufficient for proving moment convergence in this case.*

Let's consider the diagonal blocks of the extended reduced recursion matrix $\widehat{H}_{(r)}$ *corresponding to this reduced state vector set. From the state update, we see that these diagonal blocks are given by*

$$\widehat{H}_{i,i} = E\left(\left[\sum_{j=1}^{z} H_j(x_j[k])\right]^{\otimes i}\right), \tag{4.36}$$

where the expectation is taken with respect to the actions $x_j[k]$ *of the* z *traffic lights. To prove* r*th moment convergence, we must show that all eigenvalues of* $\widehat{H}_{i,i}$, $1 \leq i \leq r$, *are strictly less than* 1 *in magnitude.*

To bound the eigenvalues of $\widehat{H}_{i,i}$, *let's first consider the columns of the matrix* $\sum_{j=1}^{z} H_j(x_j[k])$, *for a particular set of actions* $x_1[k] = \tilde{x}_1, \ldots, x_z[k] = \tilde{x}_z$. *The* i*th column of this matrix specifies the fractions of the vehicles in the road segment* i *at time* k *that are present in each road segment at time-step* $k+1$, *due to the flows resulting from the actions of the traffic lights. Regardless of the particular actions taken by the traffic lights, these fractions should be non-negative for any realistic traffic model; also, we would expect that the sum of the entries in each column is less than or equal to* 1. *From here on, let's assume that a non-zero fraction of the vehicles in each road segment exits at each time-step, so that the sum of the entries in each column is strictly less than unity*[2].

Now consider the matrices $(\sum_{j=1}^{z} H_j(x_j[k]))^{\otimes i}$, $1 \leq i \leq r$. *We can easily check that the entries in each column are non-negative, and that their sum is strictly less than* 1. *Since the matrix* $\widehat{H}_{i,i}$ *is a weighted average of matrices of the form* $(\sum_{j=1}^{z} H_j(x_j[k]))^{\otimes i}$, $1 \leq x_j[k] \leq w_j$, $1 \leq j \leq z$, *the entries in each column of* $\widehat{H}_{i,i}$ *are non-negative, and their sum is less than* 1. *Thus, from Gersğorin's Theorem, we see that all eigenvalues of* $\widehat{H}_{i,i}$ *are strictly less than* 1 *in magnitude, and so this MLSN is* r*th moment convergent for all* r.

[2]*Moment convergence can also be proved with weaker assumptions on the flows out of the network, but we use this assumption to simplify our presentation.*

Moment Convergence and Reduced Representations: Necessity

Necessary conditions for rth-moment convergence can be developed using minimal representations for MLSS. In particular, say that a minimal state vector set has been constructed for a particular MLSS. Call the valid initial condition, interval, and norm used to prove minimality (see Section 4.2.7) $\bar{s}$, ϵ, and $||.||$, respectively. Consider the *extended minimal recursion matrix* $\widehat{H}_{(r)}$ (i.e., the extended reduced recursion matrix corresponding to that minimal state vector set). We claim the following: it is necessary that the eigenvalues of $\widehat{H}_{(r)}$ are strictly less than 1 in magnitude for rth-moment convergence. That is, if the eigenvalues of $\widehat{H}_{(r)}$ are not strictly less than 1 in magnitude, then the MLSS is not rth moment convergent.

To prove the claim, consider the rth *extended minimal moment vector* $E(\widehat{\mathbf{s}}_{(r)}[k])$ (i.e., the extended reduced moment vector corresponding to that minimal state vector set). Let's prove that $E(\widehat{\mathbf{s}}_{(r)}[k])$ does not converge strictly (i.e., to a single value, from all *valid* initial conditions) if the eigenvalues of $\widehat{H}_{(r)}$ are not strictly less than 1 in magnitude. To do so, consider Equation 4.22, which specifies the recursion satisfied by $E(\widehat{\mathbf{s}}_{(r)}[k])$. It is well known from linear systems theory that this recursion is not strictly convergent if any eigenvalue of $\widehat{H}_{(r)}$ has magnitude greater than or equal to 1—i.e., the limiting value of the recursion for all (valid and invalid) initial conditions is not identical. Invoking linearity, we can see that the limiting value of $\widehat{H}_{(r)}^k E(\widehat{\mathbf{s}}_{(r)}[0])$ also cannot be identical for all $E(\widehat{\mathbf{s}}_{(r)}[0])$. Thus, there is a vector $\bar{\mathbf{s}}_b$ such that $\lim_{k\to\infty} \widehat{H}_{(r)}^k \bar{\mathbf{s}} \neq \lim_{k\to\infty} \widehat{H}_{(r)}^k \bar{\mathbf{s}}_b$. Thus, $\lim_{k\to\infty} \widehat{H}_{(r)}^k (\bar{\mathbf{s}} - \bar{\mathbf{s}}_b) \neq 0$, and so $\lim_{k\to\infty} \widehat{H}_{(r)}^k \alpha(\bar{\mathbf{s}} - \bar{\mathbf{s}}_b) \neq 0$ for any $\alpha \neq 0$. Thus, $\lim_{k\to\infty} \widehat{H}_{(r)}^k (\bar{\mathbf{s}} + \alpha(\bar{\mathbf{s}} - \bar{\mathbf{s}}_b)) \neq \lim_{k\to\infty} \widehat{H}_{(r)}^k \bar{\mathbf{s}}$ for any $\alpha > 0$. Based on the way that we have defined the minimal state vector set, there is necessarily some α such that $\bar{\mathbf{s}} + \alpha(\bar{\mathbf{s}} - \bar{\mathbf{s}}_b)$ is a valid initial condition for the MLSS. Thus, we see that there are two valid initial conditions such that the limiting value of $\widehat{H}_{(r)}^k E(\widehat{\mathbf{s}}_{(r)}[0])$ is not identical. Again invoking linearity, we see that $E(\widehat{\mathbf{s}}_{(r)}[k])$ does not converge to the same vector from all valid initial conditions.

Next, note that $E(\widehat{\mathbf{s}}_{(r)}[k])$ is contained in the extended moment vector $E(\mathbf{s}_{(r)}[k])$. Thus, if $E(\widehat{\mathbf{s}}_{(r)}[k])$ does not converge strictly, $E(\mathbf{s}_{(r)}[k])$ also does not converge strictly, and the MLSS is not rth moment convergent. Thus, we have proven the claim.

Example 4.12

Again consider the jump-linear system studied in Examples 4.9 and 4.10. It can be shown that the reduced state representation considered in Example 4.10 is in fact minimal.

To see why, consider the initial condition of the rth reduced moment vector set for this example. A valid initial condition can be chosen by choosing the probability $E(q_1[0])$ *that the underlying Markov chain is in Status* 1*, as well as the conditional moments* $E((x[0])^i \mid q_1[0] = 1)$ *and* $E((x[0])^i \mid q_2[0] = 1)$ *for* $1 \leq i \leq r$*. Note that* $E(q_1[0])$ *can be chosen in the continuous range* $[0, 1]$*—let's choose an initial* $E(q_1[0])$ *that is strictly in the interior of this space (e.g., in* $(0, 1)$*). Also, note that there is a closed ball in* r*-dimensional space such that any point in this ball consitutes a valid set of initial conditions* $E((x[0])^i \mid q_1[0] = 1)$*,* $1 \leq i \leq r$ *(this is just a reflection of the fact that moments of a random variable can lie in a continuous range)—let's say that we choose* $E((x[0])^i \mid q_1[0] = 1)$*,* $1 \leq i \leq r$ *that are strictly within this ball. In a similar fashion, let's say that we choose* $E((x[0])^i \mid q_1[0] = 2)$*,* $1 \leq i \leq r$*, that are strictly in the interior of their valid space.*

From these initial conditions, we can compute all entries in the reduced moment vector set: $E(q_1[k])$ *is already specified, the entries* $E(q_1[0](x[0])^i)$ *can be computed as* $E(q_1[0])E((x[0])^i \mid q_1[0] = 1)$*, and the entries* $E(q_2[0](x[0])^i)$ *can be computed as* $(1 - E(q_1[0]))E((x[0])^i \mid q_2[0] = 1)$*. These entries constitute a valid* r*th reduced moment set.*

Now say that we modify each entry in the reduced moment vectors by at most ϵ*. Then the new probability that the underlying Markov chain is in Status* 1 *differs from* $E(q_1[0])$ *by at most* ϵ*, so this new probability is valid for small enough* ϵ*. Changing the reduced moment vectors by a magnitude of* ϵ *changes* $E((x[0])^i \mid q_1[0] = 1)$ *from* $\frac{E(q_1[0](x[0])^i)}{E(q_1[0])}$ *to* $\frac{E(q_1[0](x[0])^i) \pm \epsilon}{E(q_1[0]) \pm \epsilon}$*. By reducing the magnitude of* ϵ*, the second quantity can be made arbitrarily close to the first, and hence the modified values for* $E((x[0])^i \mid q_1[0] = 1)$ *are valid conditional moments for small enough* ϵ*. In turn, the corresponding entries of the modified reduced moment vectors are valid. Using a similar argument, we see that the modified values of the entries* $E(q_2[0](x[0])^i)$ *are also valid for small enough* ϵ*. Hence, we have shown that the reduced state representation is in fact minimal.*

Since the reduced state representation is minimal, the sufficient conditions for r*th moment convergence given in Example 4.10 are in fact also necessary for* r*th moment convergence. This example suggests minimal state representations can be a powerful approach for developing necessary and sufficient conditions for convergence.*

4.3.2 Convergence of the MLSS State

In this section, we consider MLSS for which the state process $\{\mathbf{s}[k]\}$ itself converges to a constant or to a random variable. In particular, we discuss two notions for the convergence of the state of a random process, namely *convergence in mean square* and *convergence in prob-*

ability (see [62] for an introduction), in the context of MLSS. Convergence in mean-square is related to the first and second moments and cross-moments of the state variables of a stochastic process, so we can straightforwardly develop sufficient conditions for convergence in mean square in terms of the parameters (i.e., recursion matrices) of the first- and second-moment recursions. In turn, convergence in the mean square guarantees convergence in probability [62], so we can also check for convergence in probability.

A stochastic process $\{\mathbf{s}[k]\}$ is said to converge to the *constant vector* $\bar{\mathbf{s}}$ *in mean-square* if

$$\lim_{k\to\infty} E\left((\mathbf{s}[k]-\bar{s})'(\mathbf{s}[k]-\bar{s})\right) = 0. \tag{4.37}$$

It is well-known (see, e.g., [62]) that $\{\mathbf{s}[k]\}$ converges to $\bar{\mathbf{s}}$ in mean-square if and only if the following two conditions hold:

- The sequence $E(\mathbf{s}[k])$ converges, and $\lim_{k\to\infty} E(\mathbf{s}[k]) = \bar{\mathbf{s}}$.
- The sequence $E(\mathbf{s}[k]^{\otimes 2})$ converges, and $\lim_{k\to\infty} E(\mathbf{s}[k]^{\otimes 2}) = \bar{\mathbf{s}}^{\otimes 2}$

(Typically, the second condition is specified in terms of the limiting value of the correlation matrix $E(\mathbf{s}[k]\mathbf{s}'[k])$, but our formulation in Kronecker product notation is equivalent.)

Thus, we see that the state vector of an MLSS necessarily converges to the constant $\bar{\mathbf{s}}$ if 1) the MLSS is second-moment convergent, 2) $\lim_{k\to\infty} E(\mathbf{s}[k]) = \bar{\mathbf{s}}$, and 3)$\lim_{k\to\infty} E(\mathbf{s}[k]^{\otimes 2}) = \bar{\mathbf{s}}^{\otimes 2}$. Our discussion of moment convergence in the previous section shows one approach for checking these three conditions, and for explicitly specifying the conditions in terms of the MLSS recursion matrices.

Example 4.13

As a particular example of mean-square convergence to a constant, consider the scalar MLSS

$$\begin{aligned} s[k+1] &= 0.4s[k], \quad \text{w.p. } 0.5 \\ s[k+1] &= 1.3s[k], \quad \text{w.p. } 0.5. \end{aligned} \tag{4.38}$$

We can straightforwardly verify that the state vector is second-moment convergent, and that the first and second moments of the state both converge to 0*. Thus, the state of the MLSS is seen to converge to* 0 *in mean square. We therefore know that the state of the MLSS converges to* 0 in probability*—i.e., $lim_{k\to\infty} Pr(|s[k]-0| > \epsilon) = 0$ for any ϵ. Interestingly, although the system is mean-square convergent, higher moments of the state do not converge. The divergence of higher*

moments signals that the state can become anomalously large at large time, but not with high enough probability to prevent convergence of the second moment to 0. *Simulations of the system 4.38 are shown in Figure 4.3. The mean response of the system, as well as* 2σ *intervals around the mean, are also shown.*

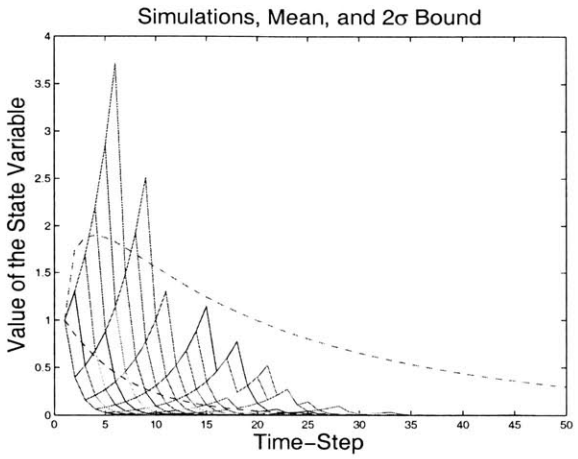


Figure 4.3: Simulations of a scalar MLSS in which the state of the MLSS converges to the constant 0 *in mean square. The mean and two-standard deviation intervals about the mean are also shown. Note that the third moment of this MLSS does not converge.*

Another possible asymptotic behavior of an MLSS is convergence in mean square to a *random vector*. Specifically, a random process $\{\mathbf{s}[k]\}$ is said to converge to the random vector $\bar{\mathbf{s}}$ in mean square if $\lim_{k\to\infty} E\left((\mathbf{s}[k]-\bar{\mathbf{s}})'(\mathbf{s}[k]-\bar{\mathbf{s}})\right) = 0$. That is, a random process converges to a random vector if for every possible sample path of the process a (possibly different) limiting value can be assigned such that the expected squared deviation of the process from its limits approaches 0 with time. Note that convergence to a constant is a special case of convergence to a random vector, in which the limiting vector takes only one value, with probability 1.

Example 4.14

We can design an MLSS that converges to a random vector in mean square, but does not converge

to a constant. For instance, consider an MLSS with two state variables, updated as follows:

$$\mathbf{s}[k+1] = \begin{bmatrix} 0.3 & 0.7 \\ 0 & 1 \end{bmatrix} \mathbf{s}[k], \quad \text{w.p. } 0.5 \tag{4.39}$$
$$\mathbf{s}[k+1] = \begin{bmatrix} 1 & 0 \\ 0.7 & 0.3 \end{bmatrix} \mathbf{s}[k], \quad \text{w.p. } 0.5.$$

Assume that the initial state is $\mathbf{s}'[0] = [1, 0]$. *It is straightforward to check that, in the limit, the two state variables converge to the subspace* $s_1[k] = s_2[k]$. *If* $s_1[k] = s_2[k]$, *then the next state* $\mathbf{s}[k+1]$ *equals* $\mathbf{s}[k]$, *regardless of which of the two possibilities in Equation 4.39 is exercised. Thus, since each sample path of this process converges (in general to a different value), we expect that the sequence converges to a random variable in mean square. (Convergence in mean square can be justified rigorously for this example using theorems for convergence discussed in, e.g., [62].) Simulations of the system 4.39 verify convergence (Figure 4.4).*

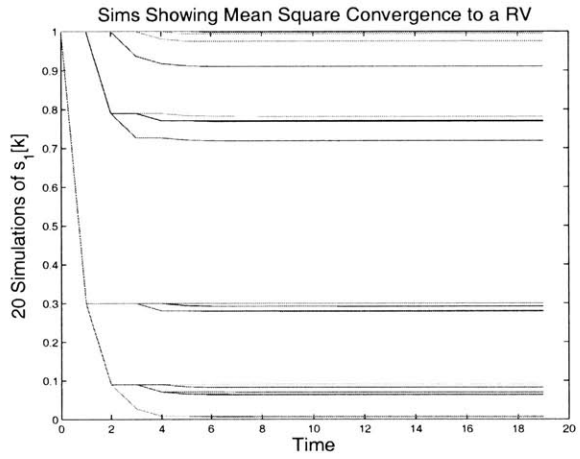


Figure 4.4: We show $s_1[k]$ *for* 20 *simulations of the system 4.39. Each simulated sample path converges to a different value, reflecting that the system converges in mean square to a random vector.*

This simple example indicates how MLSS generally converge to a random variable in the mean square: they converge to a subspace of the state space, in which they are constrained to remain constant. This insight can be translated into specific conditions on the first and second moment recursion matrices that guarantee convergence to a random vector, but these conditions are not developed any further here.

4.3.3 Toward Further Analysis: Distributional Convergence, Non-Convergent Dynamics, and Settling

Here, we list three further directions for study regarding the asymptotics of MLSS. First, we are interested in checking for *distributional convergence*—i.e., convergence of the probability (or cumulative) distribution of the state vector. In several examples of MLSS and MLSN (including the queueing model with random service probabilities, particular influence models, the air traffic model, and the road traffic network model), we observe that all moments of the state vectors converge, possibly indicating distributional convergence. We are motivated to check for distributional convergence in these examples, in order to develop a stronger characterization for the models' asymptotics. We anticipate that proofs for distributional convergence of MLSS will be based on the following two approaches:

- We can show distributional convergence by showing moment convergence, as long as the asymptotics for the moments satisfy certain conditions, which guarantee that the distribution of the state is uniquely specified by its moments (see [2], [62], [130]).
- We can use distributional convergence results for *general Markov chains* (i.e., discrete-time Markov processes with generally defined state spaces) [104]. For MLSS with finite state space (e.g., the influence model), distributional convergence results for finite-state Markov chains can be used. Convergence analysis of the influence model is discussed in [9].

Example 4.15

Consider the infinite server queue with random service probabilities that was introduced in Section 2.4.5. We can prove that the (cumulative) distribution of the number of jobs in the queue converges with time, using the approach of [104]. The details of this proof are tangential to our analysis and so are not pursued here. However, we briefly outline the proof and hope that the reader can fill in the details if he/she is interested.

The result is proved by using a result for distributional convergence of general Markov chains that was developed by Kendall, Numellin, and Tweedie ([104]). The details of the proof are concerned with showing that the assumptions of this theorem hold. The following are assumptions that must be verified:

- *Recall that the number of jobs in the queue, $s_1[k]$, is a Markov chain (in the terminology of [104]) with state space comprising the non-negative integers. We must show that this Markov chain*

is irreducible *and* aperiodic *(see [104] for definitions of these concepts for general Markov chains).*

- *It is required that we can construct a function $V(s_1[k])$ that maps from the state space $s_1[k]$ to the interval $[1, \infty]$, such that $E(V(s_1[k+1]) \mid s_1[k]) \leq \beta V(s_1[k])$ for some $\beta < 1$ and for all $s_1[k]$ except those in a* small set *(see [104]). It is straightforward to show that the function $V(s) = s + 1$ satisfies the condition, for any $\beta > 0.52$.*

Once these assumptions have been shown, the theorem of Kendall, Numellin, and Tweedie can be used to prove that the cumulative distribution function of $s_1[k]$ converges. Furthermore, the 1-norm of the difference between the cumulative distribution at time k and the steady-state distribution is upper bounded by a multiple of β^k. In other words, the transient cumulative distribution of the number of jobs in the queue approaches the steady-state in a geometric fashion. This approach to steady-state is shown in Figure 4.5.

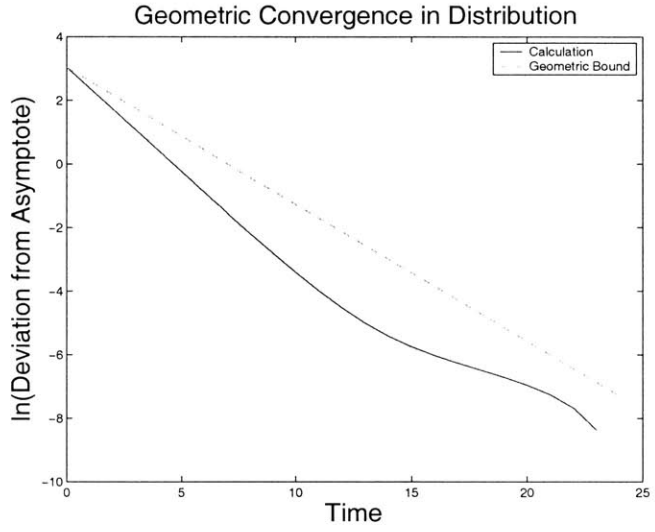


Figure 4.5: The one-norm of the difference between the transient distribution and the asymptotic distribution of number of jobs in the queue is plotted as function of time, on a semi-log scale. The plot verifies the geometric convergence of this deviation for the asymptote.

As a second direction of study, we are interested in characterizing some non-convergent dynamics of MLSS. In particular, some MLSS are not moment convergent, but their moments remain bounded. Such MLSS can display periodic or other dynamics, and may be of interest. We are also interested in characterizing other non-convergent MLSS, by specifying the rates of growth of state variables with time. One interesting approach for studying

non-convergent MLSS is to construct other MLSS with update rules that are scaled versions of the update rules for the original MLSS.

Third, we are interested in exploring the *settling* of MLSS— i.e., the process by which MLSS state and moment dynamics approach their asymptotic dynamics. Questions regarding the settling properties of influence models were introduced in [9], and we will pursue some of these questions in detail later in the thesis. We intend to consider settling more generally in the context of MLSS in future work.

Example 4.16

Consider the jump-linear system discussed in Examples 4.9, 4.10, and 4.12. If the system is rth-moment convergent, the settling times of the first r moments are closely related to the dominant eigenvalue (i.e., the eigenvalue of maximum magnitude) of the extended minimal recursion matrix $\widehat{H}_{(r)}$*. It is interesting to ask whether or not the settling times of higher moments are slower than those of lower moments. To check, note that the eigenvalues of* $\widehat{H}_{(r)}$ *are the eigenvalues of the matrices* $\widehat{H}_{1,1}, \ldots, \widehat{H}_{r,r}$*. Thus, by comparing the eigenvalues of the matrices* $H_{i,i}$*, we can check whether or not higher moments will settle more slowly than lower moments.*

For instance, consider an example with the following parameters: $D = \begin{bmatrix} 0.7 & 0.3 \\ 0.2 & 0.8 \end{bmatrix}$, $a_1 = 0.9$, *and* $a_2 = 0.5$ *(the values of* b_1 *and* b_2 *are not relevant in computing the eigenvalues of* $\widehat{H}_{(r)}$*). In Figure 4.6, we plot the maximum eigenvalue of each* $\widehat{H}_{i,i}$ *for this example. The plot shows that the maximum eigenvalue of* $\widehat{H}_{i,i}$ *decreases with i, so we would expect settling times of higher moments to be no larger than settling times of lower moments in this example.*

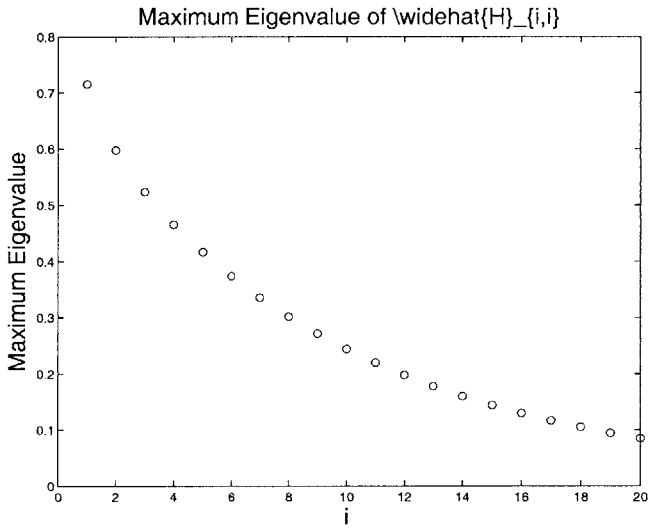


Figure 4.6: The maximum magnitudes of the eigenvalues of each $\widehat{H}_{i,i}$ are shown as a function of i for the example jump-linear system. Thse eigenvalues decrease with i in this example, suggesting that the settling rates of higher moments is not slower than the settling rates of lower moments in this example.

Chapter 5

Linear Minimum Mean Square Error Estimation in MLSS

In this chapter, we develop a recursive algorithm for linear minimum mean square error (LMMSE) filtering of MLSS and MLSN. That is, we show how to recursively find the best linear estimator, in a mean-squared sense, for the state of an MLSS, given a sequence of imperfect observations of the state. To construct this recursive estimator, we exploit the special quasi-linear structure of MLSS, and of the observations in our model (specified below). For the sake of notational clarity, we describe the filtering algorithm for time-invariant MLSS (and MLSN), though the analysis for time-varying MLSS is identical.

Although we focus on *filtering* (estimation of the state a particular time-step using observations up to that time-step), the techniques developed in this chapter can be extended to construct optimal *predictors*, *smoothers*, and *state sequence estimators*.

5.1 Some Relevant Literature

LMMSE estimation of linear systems was introduced by Wiener, and for state-space systems, in the seminal work of Kalman [77]. See the review in [76] for a history of LMMSE estimation. A thorough and easy-to-read description of the discrete-time *Kalman filter* (i.e., the LMMSE filter of a discrete-time linear state-space system) is given in [28]. Our derivation of the LMMSE filter of an MLSS closely follows the derivation of the discrete time Kalman filter given in [20].

Our LMMSE filter for an MLSS can be viewed as a generalization of the Kalman filter. Our filter reduces to the standard Kalman filter if the state update is governed by a linear state space equation driven by additive, state-independent white noise, and observations are linear functions of the state with added independent white noise [28]. The book [28]

describes some generalizations of the Kalman filter, including filters for linear systems with correlated state and output noise and for linear systems with non-white noise processes. The generalization introduced in [141] is more similar to our filter. In [141], the Kalman filter is generalized to allow for observation noise that has a state-dependent variance. MLSS can be viewed as linear systems that have observations and state updates with state-dependent statistics, so our filter has some similarities with the filter given in [141]. The state update and observations of MLSS can be viewed as linear systems driven by state-dependent noise that is specially structured to maintain moment-linearity. The article [111] specifies another model with state-dependent parameters that is amenable to Kalman filter-type analysis. The well-known extended Kalman filter (see, e.g., [93]) for approximate estimation in non-linear systems may prove to be of use in our context, either as a tool for parameter estimation in our models or as a source of tractable generalizations for our models and filter. More research is needed in this area.

The methods of [141] and [111] have been applied in, e.g., financial times series modeling and characterization of fish populations [132, 101]. Estimation in a state-space model for HIV infection can also be done using the method of [141], though the authors use a simplified version of the method [65]. We envision that filtering in MLSS will be particularly valuable for study of the networked systems that we have described, because other estimation techniques (such as ML estimation) are computationally infeasible or taxing for these systems.

Several researchers have developed linear estimation techniques for arrival processes whose underlying rate processes are themselves random, and sometimes arrival-dependent (e.g., [126],[125]). Segall and Kailath describe a broad class of arrival processes of this type, for which a martingale formulation facilitates linear estimation using a Wiener filtering approach [125]. It may be that the approach of [125] motivates martingale-based analyses of MLSS, or suggests (possibly non-state-space) generalizations for which linear estimation is possible.

5.2 Observations in MLSS

To study estimation in MLSS, we must first specify observations for our model. We consider observations of MLSS that are structured to facilitate estimation, and yet are general enough to represent realistic measurements in our examples of MLSS and MLSN. In our

model, we assume that a real vector $\mathbf{z}[k]$ is observed at each time k, and is independent of the past history of the system (i.e., $\mathbf{s}[0], \ldots, \mathbf{s}[k-1]$ and $\mathbf{z}[0], \ldots, \mathbf{z}[k-1]$), given the current state $\mathbf{s}[k]$. Furthermore, we assume that the observation $\mathbf{z}[k]$ is first- and second-moment linear, given $\mathbf{s}[k]$. That is, $\mathbf{z}[k]$ is generated from $\mathbf{s}[k]$ in such a way that the first (vector) moment for $\mathbf{z}[k]$ given $\mathbf{s}[k]$ can be written as an affine function of $\mathbf{s}[k]$,

$$E(\mathbf{z}[\mathbf{k}] \mid \mathbf{s}[k]) = C_{1,1}\mathbf{s}[k] + C_{1,0}, \tag{5.1}$$

for some $C_{1,1}$ and $C_{1,0}$, and the second vector moment for $\mathbf{z}[k]$ given $\mathbf{s}[k]$ can be written as an affine function of $\mathbf{s}[k]$ and $\mathbf{s}[k]^{\otimes 2}$,

$$E(\mathbf{z}[\mathbf{k}]^{\otimes \mathbf{2}} \mid \mathbf{s}[k]) = C_{2,2}\mathbf{s}[k]^{\otimes 2} + C_{2,1}\mathbf{s}[k] + C_{2,0}. \tag{5.2}$$

for some $C_{2,2}$, $C_{2,1}$, and $C_{2,0}$.

It is worthwhile for us to discuss types of observations that can be represented in our framework, in order to better motivate our approach. Our aim in considering observations that are first- and second-moment linear with respect to the state is to provide a generalization of observations that are linear with respect to the state (as is assumed in the development of the Kalman filter). The following list specifies some types of observations that are first- and second-moment linear with respect to the state.

- Observations that are linear with respect to the state vector can be represented in our framework. For instance, an observation $\mathbf{z}[k] = C\mathbf{s}[k] + \mathbf{w}[k]$, where $\{\mathbf{w}[k]\}$ is an independent white-noise process, is first- and second-moment linear with respect to $\mathbf{s}[k]$.

- Observations that are determined by choosing randomly among multiple different linear functions of the state satisfy first and second moment linearity conditions.

  Example 5.1
  We can model a system in which linear observations may or may not, with some probability, be taken at each time-step. In this case, the observation $\mathbf{z}[k]$ at each time-step k may be a nontrivial linear function of the state with some probability, or 0 with some probability.

- Consider a state update that is structured so that state variables are always positive. Then observations that are Poisson random variables with means given by state variables (or by positive combinations of state variables) are first- and second- moment linear with respect to the state. Such observations may occur, for example, in systems in which hidden underlying state variables modulate an observed Poisson arrival process.

More generally, observations that are random vectors with distributions parametrized by state variables can possibly satisfy the moment-linearity conditions, depending on the manner in which the distribution is parametrized by the state.

Example 5.2

Consider a scalar observation $z[k]$ that is generated from a scalar, positive state $s[k]$ as a Poisson random variable with mean $\frac{1}{2}s[k]+1$. Then $E(z[k] \mid s[k]) = \frac{1}{2}s[k]+1$ and $E(z[k]^2 \mid s[k]) = (\frac{1}{2}s[k]+1)^2 + (\frac{1}{2}s[k]+1) = \frac{1}{4}s^2[k] + \frac{3}{2}s[k]+1$, so $z[k]$ is first- and second-moment linear with respect to $s[k]$. Figure 5.2 shows a scatter plot of $z[k]$, along with its computed mean and 2σ intervals. The scatter plot is generated assuming that each $s[k]$ is uniformly distributed between 1 and 5.

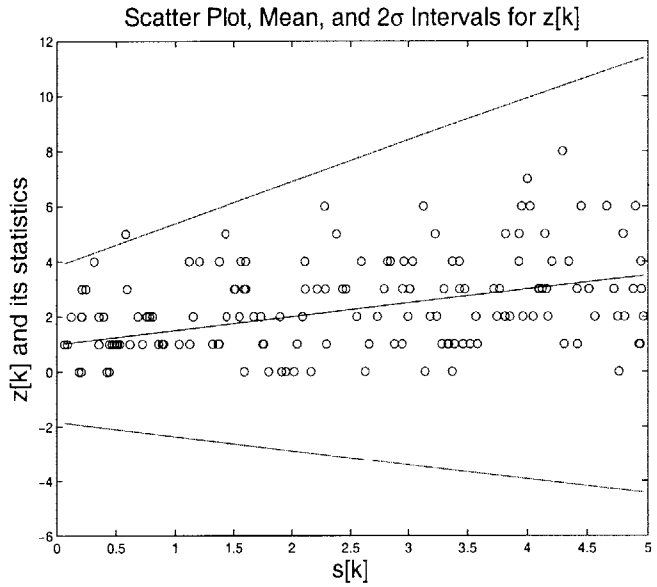


Figure 5.1: A scatter plot of 150 observations is shown. Each observation $z[k]$ is generated from the state $s[k]$ according to a Poisson distribution with mean $\frac{1}{2}s[k]+1$. Mean and 2σ intervals of $z[k]$ are shown along with the data. The state values were chosen to uniform in the interval $[0, 5]$.

- Consider an MLSS with a state vector comprising one or more status indicator vectors (e.g., the influence model state vector). In such models, it may be reasonable that observations can also be represented with indicators. If these indicator observations are generated based on a probability vector specified by a current status, or as a linear combination of such probability vectors, then the observations are first- and second-

moment linear with respect to the state.

Example 5.3

Hidden Markov Model (HMM) observations can be formulated as MLSS observations. The HMM that we consider comprises an underlying m-status Markov chain, along with an observation drawn from a finite alphabet $1, \ldots, q$. Given that the status of the underlying Markov chain at time k is i, the probability that each observation is generated at time k is specified by the length-q probability vector $\mathbf{p}_i$.

Assume that the status of the underlying Markov chain at time k is indicated by a length-m vector $\mathbf{s}[k]$, and that the observation is indicated by the length-q vector $\mathbf{z}[k]$. Note that $E(\mathbf{z}[k] \mid \mathbf{s}[k])$ is a list of the probabilities that each observation is generated, given $\mathbf{s}[k]$, and so can be written as

$$E(\mathbf{z}[k] \mid \mathbf{s}[k]) = \begin{bmatrix} \mathbf{p}_1 & \ldots & \mathbf{p}_m \end{bmatrix} \mathbf{s}[k]. \tag{5.3}$$

Also, since $\mathbf{z}[k]$ is an indicator vector, the entries in $\mathbf{z}[k]^{\otimes 2}$ are either 0 or are entries in $\mathbf{z}[k]$. Thus, we see that $E(\mathbf{z}[k]^{\otimes 2} \mid \mathbf{s}[k])$ can be written as a linear function of $\mathbf{s}[k]$, and so the observations are first- and second-moment linear with respect to the state $\mathbf{s}[k]$.

Generalizations of the HMM in which multiple observations are generated at each time-step can also sometimes be represented in our framework. For instance, the book [16] discusses a generalization in which probability vectors $\mathbf{p}_i$ for observations are specified by the state just as in a standard HMM, but multiple observations are generated independently according to these probabilities. These generalized HMM observations can be captured as MLSS observations, in a couple ways. First, an extended observation vector which is a concatenation of indicator vectors for each observation constitutes an MLSS observation. Alternately, an observation vector that counts the number of occurrences of each type of observation can be shown to be an MLSS observation.

5.3 Moment Linearity and LMMSE Estimation: A Useful Lemma

A key step in the derivation of the Kalman filter is the development of a relation between estimates (given data) for two random vectors that are themselves related by a linear transform (e.g., [20]). Here, we relate estimates for two random variables that are related by first- and second-moment linearity conditions. This relation is applied in developing the LMMSE estimator in the next section.

Consider two random vectors $\mathbf{x}$ and $\mathbf{y}$, for which the LMMSE estimate of $\mathbf{x}$ given $\mathbf{y}$,

denoted $\widehat{\mathbf{x}}(\mathbf{y})$, and the corresponding error covariance matrix $E\left((\mathbf{x}-\widehat{\mathbf{x}}(\mathbf{y}))(\mathbf{x}-\widehat{\mathbf{x}}(\mathbf{y}))'\right)$ are known. Now consider a random variable $\mathbf{z}$ that is independent of $\mathbf{y}$, given $\mathbf{x}$. Further, assume that $\mathbf{z}$ is first- and second-moment linear with respect to $\mathbf{x}$, with vector moments given by

$$E(\mathbf{z}\,|\,\mathbf{x}) = C_{1,1}\mathbf{x} + C_{1,0} \tag{5.4}$$

$$E(\mathbf{z}\otimes\mathbf{z}\,|\,\mathbf{x}) = C_{2,2}(\mathbf{x}\otimes\mathbf{x}) + C_{2,1}\mathbf{x} + C_{2,0}. \tag{5.5}$$

We write the LMMSE estimate of $\mathbf{z}$ given $\mathbf{y}$, denoted $\widehat{\mathbf{z}}(\mathbf{y})$, as well as the corresponding error covariance matrix $E\left((\mathbf{z}-\widehat{\mathbf{z}}(\mathbf{y}))(\mathbf{z}-\widehat{\mathbf{z}}(\mathbf{y}))'\right)$, in terms of the LMMSE estimate of $\mathbf{x}$ given $\mathbf{y}$, the corresponding error covariance matrix, and the first two moments of $\mathbf{x}$.

5.3.1 The Estimate of z, Given y

First consider $\widehat{\mathbf{z}}(\mathbf{y})$. The standard formula for calculating $\widehat{\mathbf{z}}(y)$ from the observation $\mathbf{y}$ and the statistics of $\mathbf{y}$ and $\mathbf{z}$ is

$$\widehat{\mathbf{z}}(\mathbf{y}) = \overline{\mathbf{z}} + \Sigma_{\mathbf{zy}}\Sigma_{\mathbf{yy}}^{-1}(\mathbf{y}-\overline{\mathbf{y}}), \tag{5.6}$$

where $\overline{\mathbf{z}} = E(\mathbf{z})$, $\overline{\mathbf{y}} = E(\mathbf{y})$, $\Sigma_{\mathbf{zy}} = E((\mathbf{z}-\overline{\mathbf{z}})(\mathbf{y}-\overline{\mathbf{y}})')$, and $\Sigma_{\mathbf{yy}} = E((\mathbf{y}-\overline{\mathbf{y}})(\mathbf{y}-\overline{\mathbf{y}})')$ (see, e.g., [20]). Using the first moment-linearity condition for $\mathbf{z}$ given $\mathbf{x}$, we can express the mean of $\mathbf{z}$ in terms of the mean of $\mathbf{x}$, as

$$\overline{\mathbf{z}} = E(\mathbf{z}) = C_{1,1}E(\mathbf{x}) + C_{1,0}. \tag{5.7}$$

The first moment linearity condition can also be applied to express the covariance matrix $\Sigma_{\mathbf{zy}}$ in terms of $\Sigma_{\mathbf{xy}} \triangleq E\left((\mathbf{x}-\overline{\mathbf{x}})(\mathbf{y}-\overline{\mathbf{y}})'\right)$ (where $\overline{\mathbf{x}} = E(x)$), as follows:

$$\begin{aligned}
\Sigma_{\mathbf{zy}} &= E((\mathbf{z}-\overline{\mathbf{z}})(\mathbf{y}-\overline{\mathbf{y}})') \\
&= E\left(E((\mathbf{z}-\overline{\mathbf{z}})(\mathbf{y}-\overline{\mathbf{y}})'\,|\,\mathbf{x})\right) \\
&= E\left(E(\mathbf{z}-\overline{\mathbf{z}}\,|\,\mathbf{x})E(\mathbf{y}-\overline{\mathbf{y}}\,|\,\mathbf{x})\right) \\
&= E\left((C_{1,1}\mathbf{x}+C_{1,0}-C_{1,1}\overline{\mathbf{x}}-C_{1,0})E(\mathbf{y}-\overline{\mathbf{y}}\,|\,\mathbf{x})\right) \\
&= C_{1,1}E\left((\mathbf{x}-\overline{\mathbf{x}})E(\mathbf{y}-\overline{\mathbf{y}}\,|\,\mathbf{x})\right) \\
&= C_{1,1}E\left(E((\mathbf{x}-\overline{\mathbf{x}})(\mathbf{y}-\overline{\mathbf{y}})\,|\,\mathbf{x})\right) \\
&= C_{1,1}E((\mathbf{x}-\overline{\mathbf{x}})(\mathbf{y}-\overline{\mathbf{y}})) \\
&= C_{1,1}\Sigma_{\mathbf{xy}}
\end{aligned} \tag{5.8}$$

Substituting the expressions for $\overline{\mathbf{z}}$ and $\Sigma_{\mathbf{zy}}$ into Equation 5.6 and doing a bit of algebra yields

$$\widehat{\mathbf{z}}(\mathbf{y}) \;=\; C_{1,1}\widehat{\mathbf{x}}(\mathbf{y}) + C_{1,0}. \tag{5.9}$$

5.3.2 The Error Covariance of the Estimate of z, Given y

Next, we compute the error covariance matrix for the estimate $\widehat{\mathbf{z}}(\mathbf{y})$ in terms of the statistics of $\mathbf{x}$ and $\widehat{\mathbf{x}}(\mathbf{y})$. It is well known (see, e.g., [20]) that this error covariance matrix is given by

$$E\left((\mathbf{z}-\widehat{\mathbf{z}}(\mathbf{y}))(\mathbf{z}-\widehat{\mathbf{z}}(\mathbf{y}))'\right) = \Sigma_{\mathbf{zz}} - \Sigma_{\mathbf{zy}}\Sigma_{\mathbf{yy}}^{-1}\Sigma_{\mathbf{yz}}. \tag{5.10}$$

Let us first compute the covariance matrix $\Sigma_{\mathbf{zz}} = E(\mathbf{z}\mathbf{z}') - E(\mathbf{z})E(\mathbf{z}')$. The mean value for $\mathbf{z}$ has already been computed in terms of the statistics of $\mathbf{x}$ in Equation 5.7. Next consider $E(\mathbf{z}\mathbf{z}')$. Next, note that the entries in the vector $E(\mathbf{z}\mathbf{z}')$ can be determined from the entries of the vector $E(\mathbf{z}\otimes\mathbf{z})$. In turn, $E(\mathbf{z}\otimes\mathbf{z})$ can be expressed in terms of of the vectors $E(\mathbf{x}\otimes\mathbf{x})$ and $E(\mathbf{x})$ by applying the second-moment linearity condition relating $\mathbf{x}$ and $\mathbf{z}$. In particular, we find that

$$E(\mathbf{z}\mathbf{z}') = \text{UNVEC}(C_{2,2}E(\mathbf{x}\otimes\mathbf{x})) + C_{2,1}E(\mathbf{x}) + C_{2,0}), \tag{5.11}$$

where the operation UNVEC is defined as follows:

- For a length-n^2 column vector Y, X=UNVEC(Y) is the $n\times n$ matrix X that has ith column given by $\begin{bmatrix} Y_{n(i-1)+1} \\ \vdots \\ Y_{ni} \end{bmatrix}$, for $1\le i\le n$. That is, UNVEC(Y) is constructed by assembling blocks of n entries of Y side-by-side to form an $n\times n$ matrix.

Next, substituting into Equation 5.10, we find that the error covariance for the estimate of $\mathbf{z}$ is

$$\begin{aligned} &E\left((\mathbf{z}-\widehat{\mathbf{z}}(\mathbf{y}))(\mathbf{z}-\widehat{\mathbf{z}}(\mathbf{y}))'\right) \\ &= \text{UNVEC}(C_{2,2}E(\mathbf{x}\otimes\mathbf{x}) + C_{2,1}E(\mathbf{x}) + C_{2,0}) \\ &-(C_{1,1}\overline{\mathbf{x}} + C_{1,0})(C_{1,1}\overline{\mathbf{x}} + C_{1,0})' - C_{1,1}\Sigma_{\mathbf{xy}}\Sigma_{\mathbf{yy}}^{-1}\Sigma_{\mathbf{yx}}C_{1,1}'. \end{aligned} \tag{5.12}$$

Finally, by adding and subtracting $C_{1,1}\Sigma_{\mathbf{xx}}C_{1,1}'$, we can rewrite Equation 5.13 explicitly in terms of the error covariance of the estimate of $\mathbf{x}$, as well as the first two vector

moments of $\mathbf{x}$, as

$$\begin{aligned} & E\left((\mathbf{z}-\widehat{\mathbf{z}}(\mathbf{y}))(\mathbf{z}-\widehat{\mathbf{z}}(\mathbf{y}))'\right) \\ & = \text{UNVEC}(C_{2,2}E(\mathbf{x}\otimes\mathbf{x}) + C_{2,1}E(\mathbf{x}) + C_{2,0}) \\ & -(C_{1,1}\overline{\mathbf{x}} + C_{1,0})(C_{1,1}\overline{\mathbf{x}} + C_{1,0})' - C_{1,1}\Sigma_{\mathbf{xx}}C_{1,1}' \\ & +C_{1,1}E((\mathbf{x}-\widehat{\mathbf{x}}(\mathbf{y}))(\mathbf{x}-\widehat{\mathbf{x}}(\mathbf{y}))')C_{1,1}'. \end{aligned} \tag{5.13}$$

5.3.3 Discussion and Comparison

We have specified the estimate of $\mathbf{z}$ and its corresponding error covariance in terms of the estimate of $\mathbf{x}$, the error covariance of this estimate, and the statistics of $\mathbf{x}$. It is interesting to note that the estimate $\widehat{\mathbf{z}}(\mathbf{y})$ is linear with respect to $\widehat{\mathbf{x}}(\mathbf{y})$. If we considered only this relation between the estimates, we could view $\mathbf{z}$ as being related to $\mathbf{x}$ through a purely linear transform (i.e., $\mathbf{z} = C_{1,1}\mathbf{x} + C_{1,0}$, or $\mathbf{z} = C_{1,1}\mathbf{x} + w$, where w is an independent noise signal with mean $C_{1,0}$).

However, the error covariance for the estimate of $\mathbf{z}$ depends on the first and second moments of $\mathbf{x}$, as well as the error covariance of $\widehat{\mathbf{x}}(\mathbf{y})$, in our case; for purely linear systems, it turns out that the error covariance for $\mathbf{z}$ would depend explicitly only on the error covariance of $\mathbf{x}$ (see, e.g., [20]). The more complex expression for error covariance in our context captures the more general second-moment relationships specified by moment-linearity conditions as compared to purely linear interactions.

In fact, we can make a stronger statement about the error covariance for the estimate of $\mathbf{z}$. In particular, let's compare the error covariance for the estimate of $\mathbf{z}$ with the error covariance for the estimate of a second random variable $\tilde{\mathbf{z}} = C_{1,1}\mathbf{x} + C_{1,0}$. The error covariance for the LMMSE estimate $\widehat{\tilde{\mathbf{z}}}(\mathbf{y})$ of $\tilde{\mathbf{z}}$ is

$$E\left((\tilde{\mathbf{z}}-\widehat{\tilde{\mathbf{z}}}(\mathbf{y}))(\tilde{\mathbf{z}}-\widehat{\tilde{\mathbf{z}}}(\mathbf{y}))'\right) = \Sigma_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}} - \Sigma_{\tilde{\mathbf{z}}\mathbf{y}}\Sigma_{\mathbf{yy}}^{-1}\Sigma_{\mathbf{y}\tilde{\mathbf{z}}}, \tag{5.14}$$

where $\Sigma_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}$, $\Sigma_{\tilde{\mathbf{z}}\mathbf{y}}$, and $\Sigma_{\mathbf{y}\tilde{\mathbf{z}}}$ are defined in the usual way. Note that $\Sigma_{\tilde{\mathbf{z}}\mathbf{y}} = \Sigma_{\mathbf{zy}}$, so $\Sigma_{\tilde{\mathbf{z}}\mathbf{y}}\Sigma_{\mathbf{yy}}^{-1}\Sigma_{\mathbf{y}\tilde{\mathbf{z}}} = \Sigma_{\mathbf{zy}}\Sigma_{\mathbf{yy}}^{-1}\Sigma_{\mathbf{yz}}$.

To compare $\Sigma_{\mathbf{zz}}$ and $\Sigma_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}$, we rewrite the two quantities as

$$\begin{aligned} \Sigma_{\mathbf{zz}} &= E\left[E((\mathbf{z}-E(\mathbf{z}\,|\,\mathbf{x}))(\mathbf{z}-E(\mathbf{z}\,|\,\mathbf{x}))'\,|\,\mathbf{x})\right] + E\left[(E(\mathbf{z}\,|\,\mathbf{x})-E(\mathbf{z}))(E(\mathbf{z}\,|\,\mathbf{x})-E(\mathbf{z}))'\right] \\ \Sigma_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}} &= E\left[E((\tilde{\mathbf{z}}-E(\tilde{\mathbf{z}}\,|\,\mathbf{x}))(\tilde{\mathbf{z}}-E(\tilde{\mathbf{z}}\,|\,\mathbf{x}))'\,|\,\mathbf{x})\right]. + E\left[(E(\tilde{\mathbf{z}}\,|\,\mathbf{x})-E(\tilde{\mathbf{z}}))(E(\tilde{\mathbf{z}}\,|\,\mathbf{x})-E(\tilde{\mathbf{z}}))'\right] \end{aligned} \tag{5.15}$$

(see, e.g., [21] for justification). Since $E(\tilde{\mathbf{z}}\,|\,\mathbf{x}) = E(\mathbf{z}\,|\,\mathbf{x})$, we see that

$E\left[(E(\mathbf{z}\,|\,\mathbf{x}) - E(\mathbf{z}))(E(\mathbf{z}\,|\,\mathbf{x}) - E(\mathbf{z}))'\right] = E\left[(E(\tilde{\mathbf{z}}\,|\,\mathbf{x}) - E(\tilde{\mathbf{z}}))(E(\tilde{\mathbf{z}}\,|\,\mathbf{x}) - E(\tilde{\mathbf{z}}))'\right]$. Thus, subtracting $\Sigma_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}$ from $\Sigma_{\mathbf{zz}}$ yields

$$\Sigma_{\mathbf{zz}} - \Sigma_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}} = E\left[E((\mathbf{z} - E(\mathbf{z}\,|\,\mathbf{x}))(\mathbf{z} - E(\mathbf{z}\,|\,\mathbf{x}))'\,|\,\mathbf{x})\right] - E\left[E((\tilde{\mathbf{z}} - E(\tilde{\mathbf{z}}\,|\,\mathbf{x}))(\tilde{\mathbf{z}} - E(\tilde{\mathbf{z}}\,|\,\mathbf{x}))'\,|\,\mathbf{x})\right]. \tag{5.16}$$

However, since $\tilde{\mathbf{z}} = C_{1,1}\mathbf{x} + C_{1,0} = E(\tilde{\mathbf{z}}\,|\,\mathbf{x})$, we see that $E\left[E((\tilde{\mathbf{z}} - E(\tilde{\mathbf{z}}\,|\,\mathbf{x}))(\tilde{\mathbf{z}} - E(\tilde{\mathbf{z}}\,|\,\mathbf{x}))'\,|\,\mathbf{x})\right]$ is 0, and Equation 5.16 reduces to

$$\Sigma_{\mathbf{zz}} - \Sigma_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}} = E\left[E((\mathbf{z} - E(\mathbf{z}\,|\,\mathbf{x}))(\mathbf{z} - E(\mathbf{z}\,|\,\mathbf{x}))'\,|\,\mathbf{x})\right]. \tag{5.17}$$

In turn, the difference between the error covariances of the estimates for $\mathbf{z}$ and $\tilde{\mathbf{z}}$ is seen to be

$$E\left((\mathbf{z} - \widehat{\mathbf{z}}(\mathbf{y}))(\mathbf{z} - \widehat{\mathbf{z}}(\mathbf{y}))'\right) - E\left((\tilde{\mathbf{z}} - \widehat{\tilde{\mathbf{z}}}(\mathbf{y}))(\tilde{\mathbf{z}} - \widehat{\tilde{\mathbf{z}}}(\mathbf{y}))'\right) = E\left[E((\mathbf{z} - E(\mathbf{z}\,|\,\mathbf{x}))(\mathbf{z} - E(\mathbf{z}\,|\,\mathbf{x}))'\,|\,\mathbf{x})\right]. \tag{5.18}$$

Note that the matrix $E\left[E((\mathbf{z} - E(\mathbf{z}\,|\,\mathbf{x}))(\mathbf{z} - E(\mathbf{z}\,|\,\mathbf{x}))'\,|\,\mathbf{x})\right]$ is positive semidefinite, since $E((\mathbf{z} - E(\mathbf{z}\,|\,\mathbf{x}))(\mathbf{z} - E(\mathbf{z}\,|\,\mathbf{x}))'\,|\,\mathbf{x})$ is positive semidefinite for any $\mathbf{x}$. Thus, we finally get that

$$E\left((\mathbf{z} - \widehat{\mathbf{z}}(\mathbf{y}))(\mathbf{z} - \widehat{\mathbf{z}}(\mathbf{y}))'\right) - E\left((\tilde{\mathbf{z}} - \widehat{\tilde{\mathbf{z}}}(\mathbf{y}))(\tilde{\mathbf{z}} - \widehat{\tilde{\mathbf{z}}}(\mathbf{y}))'\right) \succeq \mathbf{0}, \tag{5.19}$$

where the notation $A \succeq \mathbf{0}$ means that A is positive semidefinite. Conceptually, this result shows the estimate for $\mathbf{z}$ is at least as noisy as the estimate for $\tilde{\mathbf{z}}$—that is, the estimate is most accurate if the observation is purely linear with respect to the state.

Example 5.4

Consider a scalar random variable x. Let's say that the LMMSE estimate for x given an observation y, denoted $\widehat{x}(y)$, has been computed. Now consider a random variable z that is uniformly distributed between 0 and $2x$. Note that $E(z\,|\,x) = x$, and $E(z^2\,|\,x) = \frac{4x^2}{3}$, so the observations are first- and second-moment linear with respect to the state. Equation 5.9 shows that the best linear estimate for z given y is $\widehat{z}(y) = \widehat{x}(y)$. However, the error covariance for the estimate of $\mathbf{z}$ is larger than the error covariance for the estimate of $\mathbf{x}$, by the amount $E(E((z-x)^2\,|\,x))$ (Equation 5.18). A little algebra shows that this difference in the error covariances is $\frac{E(x^2)}{3}$.

5.4 The Recursive Algorithm for LMMSE State Estimation in MLSS

An efficient recursive algorithm for LMMSE estimation of the state $\mathbf{s}[k]$ at each time k from the sequence of observations $\mathbf{z}[0], \ldots, \mathbf{z}[k]$ is developed. We use the following notation to represent state estimates and their corresponding error covariance matrices:

- $\widehat{\mathbf{s}}_{k|k}$ denotes the LMMSE estimate for $\mathbf{s}[k]$ given $\mathbf{z}[0], \ldots, \mathbf{z}[k]$. That is, $\widehat{\mathbf{s}}_{k|k}$ is the linear combination of the vectors $\mathbf{z}[0], \ldots, \mathbf{z}[k]$ that minimizes the expectation $E\left((\mathbf{s}[k] - \widehat{\mathbf{s}}_{k|k})'(\mathbf{s}[k] - \widehat{\mathbf{s}}_{k|k})\right)$.
- $\Sigma_{k|k} \triangleq E\left((\mathbf{s}[k] - \widehat{\mathbf{s}}_{k|k})(\mathbf{s}[k] - \widehat{\mathbf{s}}_{k|k})'\right)$ denotes the error covariance of the estimate $\widehat{\mathbf{s}}_{k|k}$.
- $\widehat{\mathbf{s}}_{k+1|k}$ denotes the LMMSE estimate for $\mathbf{s}[k+1]$ given $\mathbf{z}[0], \ldots, \mathbf{z}[k]$. This one-step prediction or next-state update is required in the recursive estimation procedure.
- $\Sigma_{k+1|k} \triangleq E\left((\mathbf{s}[k+1] - \widehat{\mathbf{s}}_{k+1|k})(\mathbf{s}[k+1] - \widehat{\mathbf{s}}_{k|k})'\right)$ denotes the error covariance of the estimate $\widehat{\mathbf{s}}_{k+1|k}$.

The state estimates and error covariances at each time k are found iteratively, in two steps. First, a *measurement update* is used to determine $\widehat{\mathbf{s}}_{k|k}$ in terms of $\mathbf{z}[k]$, $\widehat{\mathbf{s}}_{k|k-1}$, $\Sigma_{k|k-1}$, and the *a priori* statistics of $\mathbf{s}[k]$. That is, the measurement update recalculates the estimate for $\mathbf{s}[k]$ given the new observation $\mathbf{z}[k]$. We also simultaneously compute the corresponding error covariance matrix $\Sigma_{k|k}$ in terms of $\Sigma_{k|k-1}$ and the *a priori* statistics of $\mathbf{s}[k]$.

Second, a *next-state update* is used to determine $\widehat{\mathbf{s}}_{k+1|k}$ in terms of $\widehat{\mathbf{s}}_{k|k}$, $\Sigma_{k|k}$, and the *a priori* statistics of $\mathbf{s}[k+1]$. That is, the next-state update is used to estimate the next state, given the observations up to the current time-step and the estimate of the current state given these observations. The corresponding error covariance matrix $\Sigma_{k+1|k}$ is simultaneously computed.

Our derivation of the LMMSE estimator closely follows the derivation of the Kalman filter given in [20].

5.4.1 The Measurement Update

It is well known (see, e.g., [20]) that the LMMSE estimate for $\mathbf{s}[k]$ given observations up to time k can be written as

$$\widehat{\mathbf{s}}_{k|k} = \widehat{\mathbf{s}}_{k|k-1} + \widehat{\mathbf{s}}_k\left(\mathbf{z}[k] - \widehat{\mathbf{z}}_k(\mathbf{Z}[k-1])\right) - E(\mathbf{s}[k]), \tag{5.20}$$

where $\mathbf{Z}[k-1]$ is a convenient notation for the vector of observations $\begin{bmatrix} \mathbf{z}[0] \\ \vdots \\ \mathbf{z}[k-1] \end{bmatrix}$, $\widehat{\mathbf{z}}_k(\mathbf{Z}[k-1])$ denotes the LMMSE estimate for $\mathbf{z}[k]$ given $\mathbf{Z}[k-1]$, and $\widehat{\mathbf{s}}_k\left(\mathbf{z}[k] - \widehat{\mathbf{z}}_k(\mathbf{Z}[k-1])\right)$ denotes the LMMSE estimate of $\mathbf{s}[k]$ given $\mathbf{z}[k] - \widehat{\mathbf{z}}_k(\mathbf{Z}[k-1])$.

First consider $\widehat{\mathbf{z}}_k(\mathbf{Z}[k-1])$. Since $\mathbf{z}[k]$ is independent of $\mathbf{Z}[k-1]$ given $\mathbf{s}[k]$ and $\mathbf{z}[k]$ is moment linear with respect to $\mathbf{s}[k]$, the lemma developed in Secion 5.3 applies. Thus, the estimate can be written as follows:

$$\widehat{\mathbf{z}}_k(\mathbf{Z}[k-1]) = C_{1,1}\widehat{\mathbf{s}}_{k|k-1} + C_{1,0}. \tag{5.21}$$

Also, by applying the lemma, the corresponding error covariance matrix can be written in terms of $\Sigma_{k|k-1}$ and the statistics of $\mathbf{s}[k]$:

$$\begin{aligned} &E\left((\mathbf{z}[k] - \widehat{\mathbf{z}}_k(\mathbf{Z}[k-1]))(\mathbf{z}[k] - \widehat{\mathbf{z}}_k(\mathbf{Z}[k-1]))'\right) \\ &= C_{1,1}\Sigma_{k|k-1}C_{1,1}' + N_k(E(\mathbf{s}[k]), E(\mathbf{s}[k]^{\otimes 2})), \end{aligned} \tag{5.22}$$

where

$$\begin{aligned} &N_k(E(\mathbf{s}[k]), E(\mathbf{s}[k]^{\otimes 2})) \\ &\triangleq \text{UNVEC}\left[C_{2,2}E(\mathbf{s}[k]^{\otimes 2}) + C_{2,1}E(\mathbf{s}[k]) + C_{2,0}\right] \\ &-(C_{1,1}E(\mathbf{s}[k]) + C_{1,0})(C_{1,1}E(\mathbf{s}[k]) + C_{1,0})' \\ &-C_{1,1}\text{UNVEC}\left[E(\mathbf{s}[k]^{\otimes 2}) - E(\mathbf{s}[k])^{\otimes 2}\right]C_{1,1}'. \end{aligned}$$

Next, we compute $\widehat{\mathbf{s}}_k\left(\mathbf{z}[k] - \widehat{\mathbf{z}}_k(\mathbf{Z}[k-1])\right)$. Using the standard formula for the LMMSE estimate (as in Equation 5.6), and invoking the unbiasedness of the LMMSE estimate

$\widehat{\mathbf{z}}_k(\mathbf{Z}[k-1])$, we find that

$$\begin{aligned}&\widehat{\mathbf{s}}_k\left(\mathbf{z}[k]-\widehat{\mathbf{z}}_k(\mathbf{Z}[k-1])\right)\\&=E(\mathbf{s}[k])+\widehat{\Sigma}_{\mathbf{s}[k],\mathbf{z}[k]}\widehat{\Sigma}^{-1}_{\mathbf{z}[k],\mathbf{z}[k]}\widehat{\Sigma}'_{\mathbf{s}[k],\mathbf{z}[k]}\left[\mathbf{z}[k]-\widehat{\mathbf{z}}_k(\mathbf{Z}[k-1])\right],\end{aligned} \tag{5.23}$$

where

$$\begin{aligned}\widehat{\Sigma}_{\mathbf{s}[k],\mathbf{z}[k]}&=E\left[\mathbf{s}[k](\mathbf{z}[k]-\mathbf{z}_k(\mathbf{Z}[k-1]))'\right]\\\widehat{\Sigma}_{\mathbf{z}[k],\mathbf{z}[k]}&=E\left[(\mathbf{z}[k]-\widehat{\mathbf{z}}_k(\mathbf{Z}[k-1]))(\mathbf{z}[k]-\widehat{\mathbf{z}}_k(\mathbf{Z}[k-1]))'\right].\end{aligned} \tag{5.24}$$

Our goal is to rewrite the estimate in Equation 5.23 in terms of the estimate $\widehat{\mathbf{s}}_{k|k-1}$, the corresponding error covariance $\Sigma_{k|k-1}$, the statistics of $\mathbf{s}[k]$, and the new observation $\mathbf{z}[k]$. Most of the terms in the equation have already been computed: $E(\mathbf{s}[k])$ can be calculated from the first-moment linearity conditions (and in fact eventually cancels out of our estimator), $E\left[(\mathbf{z}[k]-\widehat{\mathbf{z}}_k(\mathbf{Z}[k-1]))(\mathbf{z}[k]-\widehat{\mathbf{z}}_k(\mathbf{Z}[k-1]))'\right]$ is given in Equation 5.22, and $\widehat{\mathbf{z}}_k(\mathbf{Z}[k-1])$ is given in Equation 5.21. Thus, it only remains to determine $E\left[\mathbf{s}[k](\mathbf{z}[k]-\mathbf{z}_k(\mathbf{Z}[k-1]))'\right]$ in terms of the appropriate quantities. To do so, we first rewrite the expectation as follows:

$$\begin{aligned}&E\left[\mathbf{s}[k](\mathbf{z}[k]-\mathbf{z}_k(\mathbf{Z}[k-1]))'\right]\\&=E\left[\mathbf{s}[k](\mathbf{z}[k]-C_{1,1}\widehat{\mathbf{s}}_{k|k-1}-C_{1,0})'\right]\\&=E(\mathbf{s}[k]\mathbf{z}'[k])-E(\mathbf{s}[k](C_{1,1}\widehat{\mathbf{s}}_{k|k-1}+C_{1,0})')\\&=E(\mathbf{s}[k]E(\mathbf{z}'[k]\,|\,\mathbf{s}[k]))-E(\mathbf{s}[k](C_{1,1}\widehat{\mathbf{s}}_{k|k-1}+C_{1,0})')\\&=E(\mathbf{s}[k](C_{1,1}\widehat{\mathbf{s}}[k]+C_{1,0})')-E(\mathbf{s}[k](C_{1,1}\widehat{\mathbf{s}}_{k|k-1}+C_{1,0})')\\&=E(\mathbf{s}[k](\mathbf{s}[k]-\widehat{\mathbf{s}}_{k|k-1})')C'_{1,1}.\end{aligned} \tag{5.25}$$

Next, it is well known from standard orthogonality results on LMMSE estimates (see, e.g., [20]) that $E(\widehat{\mathbf{s}}_{k|k-1}(\mathbf{s}[k]-\widehat{\mathbf{s}}_{k|k-1}))$ equals 0, so $E(\widehat{\mathbf{s}}_{k|k-1}(\mathbf{s}[k]-\widehat{\mathbf{s}}_{k|k-1}))C'_{1,1}$ also equals zero. Subtracting this (zero) quantity from the right side of Equation 5.25, we find that

$$\begin{aligned}&E\left[\mathbf{s}[k](\mathbf{z}[k]-\mathbf{z}_k(\mathbf{Z}[k-1]))'\right]\\&=E((\mathbf{s}[k]-\widehat{\mathbf{s}}_{k|k-1})(\mathbf{s}[k]-\widehat{\mathbf{s}}_{k|k-1})')C'_{1,1}\\&=\Sigma_{k|k-1}C'_{1,1}\end{aligned} \tag{5.26}$$

Substituting Equations 5.26, 5.22, and 5.21 into Equation 5.23, we find that

$$\begin{aligned}&\widehat{\mathbf{s}}_k\left(\mathbf{z}[k]-\widehat{\mathbf{z}}_k(\mathbf{Z}[k-1])\right)=E(\mathbf{s}[k])+\\&\Sigma_{k|k-1}C_{1,1}'\left(C_{1,1}\Sigma_{k|k-1}C_{1,1}'+N_k(E(\mathbf{s}[k],E(\mathbf{s}[k]^{\otimes 2}))\right)^{-1}\\&\left(\mathbf{z}[k]-C_{1,1}\widehat{\mathbf{s}}_{k|k-1}-C_{1,0}\right),\end{aligned} \tag{5.27}$$

Finally, substituting into Equation 5.20 leads to

$$\begin{aligned}&\widehat{\mathbf{s}}_{k|k}=\widehat{\mathbf{s}}_{k|k-1}+\\&\Sigma_{k|k-1}C_{1,1}'\left(C_{1,1}\Sigma_{k|k-1}C_{1,1}'+N_k(E(\mathbf{s}[k],E(\mathbf{s}[k]^{\otimes 2}))\right)^{-1}\\&\left(\mathbf{z}[k]-C_{1,1}\widehat{\mathbf{s}}_{k|k-1}-C_{1,0}\right).\end{aligned}$$

Also, we find the error covariance matrix $\Sigma_{k|k}$ in terms of $\Sigma_{k|k-1}$ and the statistics of $\mathbf{s}[k]$. It is well known (see, e.g., [20]) that this error covariance can be written as

$$\Sigma_{k|k}=\Sigma_{k|k-1}-\widehat{\Sigma}_{\mathbf{s}[k],\mathbf{z}[k]}\widehat{\Sigma}^{-1}_{\mathbf{z}[k],\mathbf{z}[k]}\widehat{\Sigma}'_{\mathbf{s}[k],\mathbf{z}[k]}. \tag{5.28}$$

By substituting Equations 5.26 and 5.22 into Equation 5.28, this error covariance is immediately found to be

$$\Sigma_{k|k}=\Sigma_{k|k-1}- \tag{5.29}$$

$$\Sigma_{k|k-1}C_{1,1}'\left(C_{1,1}\Sigma_{k|k-1}C_{1,1}'+N_k\left(E(\mathbf{s}[k]),E(\mathbf{s}[k]^{\otimes 2})\right)\right)^{-1}C_{1,1}\Sigma_{k|k-1}, \tag{5.30}$$

This completes the derivation of the measurement update for recursive LMMSE estimation algorithm.

One note about the measurement update is required. We have implicitly assumed in our derivation that the matrix $C_{1,1}\Sigma_{k|k-1}C_{1,1}'+N_k\left(E(\mathbf{s}[k]),E(\mathbf{s}[k]^{\otimes 2})\right)$ is invertible. Since this matrix is an error covariance (corresponding to the estimate $\widehat{\mathbf{z}}_k(\mathbf{Z}[k-1])$), it is necessarily positive semi-definite. Roughly speaking, we would expect the matrix to be strictly positive definite, and so invertible, as long as the entries in $\mathbf{z}[k]$ cannot be exactly inferred from $\mathbf{Z}[k-1]$). We leave it to future work to state precise condition on an MLSS that guarantee invertibility of this matrix, and to specify the measurement update if the matrix is not invertible.

5.4.2 Next-State Update

Next, we compute the next-state estimate $\widehat{\mathbf{s}}_{k+1|k}$ and error covariance matrix $\Sigma_{k+1|k}$ in terms of $\widehat{\mathbf{s}}_{k|k}$, $\Sigma_{k|k}$, and the statistics of $\mathbf{s}[k]$. To do so, note that we have assumed that $\mathbf{s}[k+1]$ is moment-linear with respect to $\mathbf{s}[k]$, and is independent of $\mathbf{z}[0], \ldots, \mathbf{z}[k]$, given $\mathbf{s}[k]$. Thus, the lemma developed in 5.3 applies, and $\widehat{\mathbf{s}}_{k+1|k}$ can be found as

$$\widehat{\mathbf{s}}_{k+1|k} = H_{1,1}\widehat{\mathbf{s}}_{k|k} + H_{1,0}. \tag{5.31}$$

Also, using the lemma, we find that the corresponding error covariance matrix is given by

$$\Sigma_{k+1|k} = H_{1,1}\Sigma_{k|k}H'_{1,1} + M_k(E(\mathbf{s}[k], E(\mathbf{s}[k]^{\otimes 2})), \tag{5.32}$$

where

$$\begin{aligned}
&M_k(E(\mathbf{s}[k], E(\mathbf{s}[k]^{\otimes 2})) \\
&\triangleq \text{UNVEC}\left[H_{2,2}E(\mathbf{s}[k]^{\otimes 2}) + H_{2,1}E(\mathbf{s}[k]) + H_{2,0}\right] \\
&-(H_{1,1}E(\mathbf{s}[k]) + H_{1,0})(H_{1,1}E(\mathbf{s}[k]) + H_{1,0})' \\
&-H_{1,1}\text{UNVEC}\left[E(\mathbf{s}[k]^{\otimes 2}) - E(\mathbf{s}[k])^{\otimes 2}\right]H'_{1,1}.
\end{aligned}$$

This completes the next-state estimation step in the update procedure.

5.4.3 Putting the Pieces Together

The measurement update and next-state estimate together show how to determine the estimate $\widehat{\mathbf{s}}_{k+1|k}$ and corresponding covariance matrix $\Sigma_{k+1|k}$ from the new observation $\mathbf{z}[k]$, along with $\widehat{\mathbf{s}}_{k|k-1}$ and $\Sigma_{k|k-1}$. Thus, a method for recursively determining estimates of the state at time $k+1$ from observations up to time k (and, by applying another measurement update, up to time $k+1$) has been developed. The only step remaining is to specify the initial estimate $\widehat{\mathbf{s}}_{0|-1}$ and $\Sigma_{0|-1}$, which are defined to be the estimate and covariance matrix for the initial state given no observations. Note that the LMMSE estimate for the initial state given no observations is simply $E(\mathbf{s}[0])$, and the corresponding error covariance matrix is $\text{UNVEC}(E(\mathbf{s}[0]^{\otimes 2})) - E(\mathbf{s}[0])E(\mathbf{s}[0])'$. Thus, the initial conditions for the estimator can be constructed from the initial conditions for the first and second moments of the state.

5.5 Three Examples

In this section, we consider LMMSE filtering in the context of three example MLSS. The first two examples, which are concerned with state estimation for the infinite-server queue and for a jump-linear system, serve to illustrate the basic application of the filter developed above. In the third example, we indirectly use the LMMSE estimator to bound the probability of error of a Maximum Likelihood estimator in the context of HMMs. We intend for these examples to elucidate some basic uses of our estimator. We will consider LMMSE estimation further in the three case studies of MLSN at the end of the thesis.

5.5.1 Filtering: Infinite Server Queue

Here, we consider the infinite-server queue with random service probabilities that was introduced in Section 2.4.5. We attempt to estimate the number of jobs in the queue, based on an imperfect measurement of the number of jobs exiting the queue at each time-step.

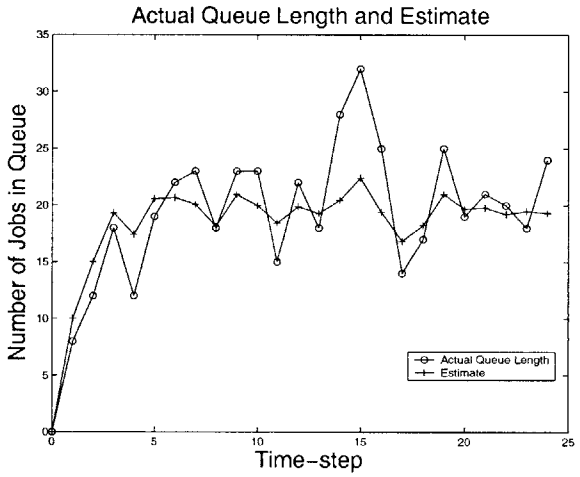


Figure 5.2: The actual number of jobs in the queue is compared with the LMMSE estimate for the number of jobs. Here, we assume that the state vector is initially $\mathbf{0}$, and that the initial estimate is also $\mathbf{0}$.

Recall that the MLSS formulation for this model has two state variables: $s_1[k]$ represents the number of jobs in the queue at time k, and $s_2[k]$ represents the number of jobs exiting the queue at time k. At each time k, we observe the number of jobs exiting the queue. We

assume that our observation $z[k]$ is noisy; specifically, we assume that $z[k]$ is generated from $s_2[k]$ according to the distribution $Binom(s_2[k], 0.9)$. (Such an observation may come about, for example, if jobs exiting the queue are counted, but each exiting job is accidentally missed with probability 0.1.) The observation $z[k]$ is first- and second-moment linear with respect to $\mathbf{s}[k]$, since $E(z[k] \mid \mathbf{s}[k]) = 0.9s_2[k]$, and $E(z[k]^2 \mid \mathbf{s}[k]) = 0.81s_2[k]^2 + 0.09s_2[k]$.

Using the techniques developed above, we construct the LMMSE estimate for the state $\mathbf{s}[k]$, given the observations $z[0], \ldots, z[k]$. Figure 5.2 shows the estimate for the number of jobs in the queue ($s_1[k]$) at each time-step during a 25 time-step simulation.

5.5.2 Filtering: Jump Linear System

We consider the jump-linear system example with scalar state and two-status underlying Markov chain introduced in Section 2.4.3. The transition matrix for the underlying Markov chain is $D = \begin{bmatrix} 0.9 & 0.1 \\ 0.3 & 0.7 \end{bmatrix}$. The scalar continuous-valued state is updated as follows: if the Markov chain is in the first status at time k, then the time-$(k+1)$ continuous-valued state is $x[k+1] = -0.9x[k] + 0.5$; if the Markov chain is in the second status, the time-$(k+1)$ continuous-valued state is $x[k+1] = x[k] + 1$. Recall that this jump-linear system can be formulated as an MLSS by choosing a state vector $\mathbf{s}[k] = \mathbf{q}[k] \otimes \begin{bmatrix} x[k] \\ 1 \end{bmatrix}$, where $\mathbf{q}[k]$ indicates the status of the underlying Markov chain.

We assume that a corrupted version of the continuous-valued state is observed. In particular, the observation at time k is assumed to be $z[k] = x[k] + N[k]$, where $N[k]$ is Gaussian random variable with mean 0 and variance 9.

We are concerned with estimating the continuous-valued state and underlying Markov status for this jump-linear system at time k, given the observations up to time k. We do so by finding the LMMSE estimate $\widehat{\mathbf{s}}_{k|k}$ for $\mathbf{s}[k]$, given $z[0], \ldots, z[k]$. This LMMSE estimate is an approximation for the MMSE estimate for $\mathbf{s}[k]$, given $z[0], \ldots, z[k]$. The MMSE estimate, which we denote $\underline{\mathbf{s}}_{k|k}$, is well-known (e.g., [21]) to be the conditional expectation for $\mathbf{s}[k]$,

given the observations $z[0], \ldots, z[k]$:

$$\underline{\mathbf{s}}_{k|k} = E(\mathbf{s}[k] \mid z[0], \ldots, z[k]) = E\left(\begin{bmatrix} q_1[k]x[k] \\ q_1[k] \\ q_2[k]x[k] \\ q_2[k] \end{bmatrix} \mid z[0], \ldots, z[k] \right). \tag{5.33}$$

By adding the first and third entries of this vector, we can get the MMSE estimate for $x[k]$, given $z[0], \ldots, z[k]$. Also, the second and fourth entries of the vector can be interpreted as probabilities that the underlying Markov chain is in each status given the observations, and so can be used to find the most likely underlying state. Since the LMMSE estimate is the best linear approximation for the MMSE, we can construct the best linear estimate for $x[k]$, and for the conditional probabilities of the underlying Markov status.

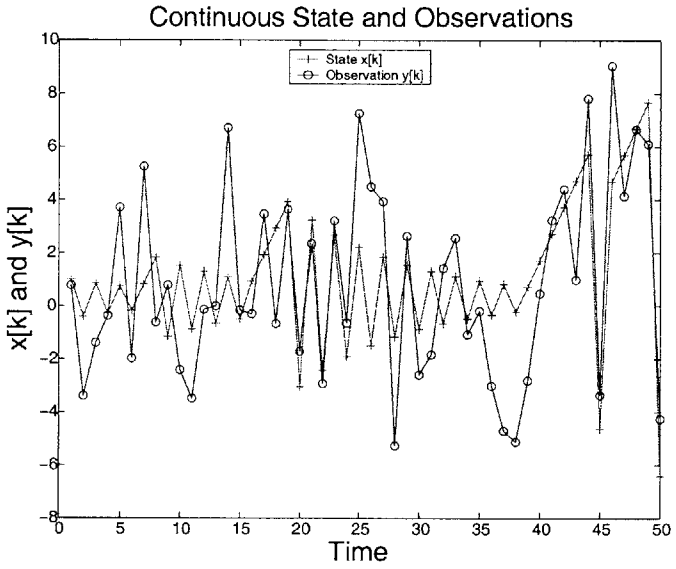


Figure 5.3: The continuous-valued state and observations during a 50 time-step simulation of the example jump linear system are shown.

Figure 5.3 shows the actual continuous-valued state and the corrupted observations in a 50 time-step simulation of the jump linear system. Figure 5.4 plots the actual continuous-valued state along with our LMMSE estimate for this continuous-valued state. The plot suggests that the LMMSE estimate is indeed a better estimate for the continuous-valued state than the observation sequence. This improvement is borne out in our analysis, since the error covariance of the estimate is 4.5, while the average squared error between the

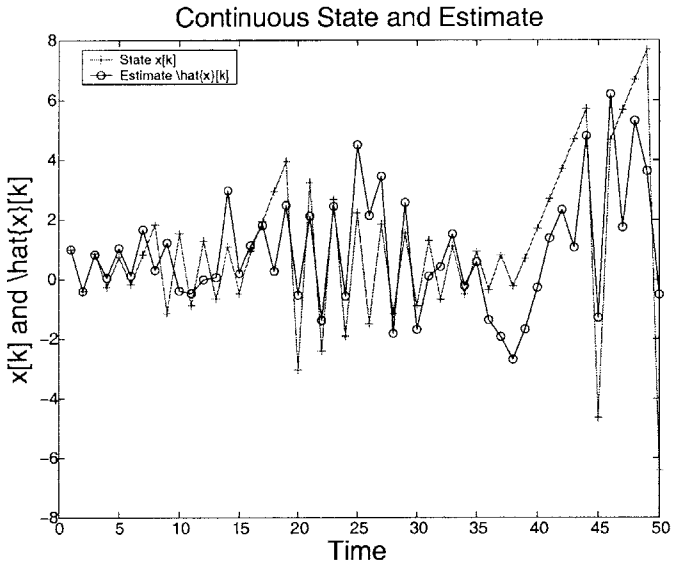


Figure 5.4: The LMMSE estimate for the continuous-valued state is compared with the actual continuous-valued state. The LMMSE estimate is a better approximation for the continuous-valued state than the observation sequence.

data and observation is 9.

We have also plotted in Figure 5.5 an LMMSE estimate for the probability that the underlying Markov chain is in status 2. The figure shows that the estimator has some ability to track the underlying Markov status, but is not particularly accurate in doing so. Our simulations also suggest that the probability estimate tends to remain below 0.5 disproportionately, so that an estimator based on these approximate probabilities would tend to be biased. It remains to be seen whether the shortcomings in the estimator are a consequence of the linearity constraint of our estimator, or are inherent to the dynamics of the system.

5.5.3 Filtering: Hidden Markov Model

This example mentions a perhaps surprising use for LMMSE estimation in developing error bounds for MAP estimators of HMMs. Please refer to example 5.3 for our MLSS formulation of a HMM.

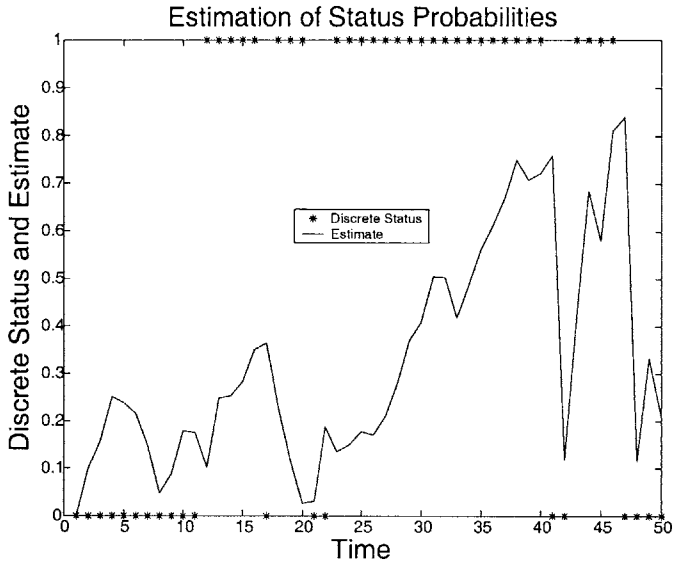


Figure 5.5: LMMSE estimation is used to approximate the conditional probability that the underlying Markov chain of the jump linear system is in the second status at each time, given the observations up to that time.

To understand how an LMMSE estimator can be used to bound the error probability, first consider the minimum mean-square-error (MMSE) estimate for the (indicator) state vector of a HMM at a time k, given the (indicator) observations from time 0 to k. With some algebra, we can show that the squared error minimized by MMSE estimate, which is the sum of the diagonal entries of the error covariance matrix, is in fact the probability of error for the MAP estimate of the state! Since the LMMSE estimator is restricted to be linear with respect to the observations, its mean squared error is at least as large at the mean squared error of the MMSE estimator. Thus, the sum of the diagonal entries of the error covariance matrix for the LMMSE estimator upper bounds the probability of error of the ML estimate. It remains to be seen whether this bound is ever sufficiently tight to be useful in characterizing the ML estimator for an HMM.

5.6 More on Estimation: Introduction

In this section, we mention one interesting observation about the LMMSE estimates for MLSS, which may allow us to extended our analysis. We also list some other directions for

study.

5.6.1 Including the Estimate in the State Vector

Let's consider the LMMSE estimate for the next state, given observations up to the current time-step, or $\widehat{\mathbf{s}}_{k+1|k}$. By assimilating the measurement update and next-state update, we see that $\widehat{\mathbf{s}}_{k+1|k}$ is a time-varying deterministic linear function of the estimate $\widehat{\mathbf{s}}_{k|k-1}$ and the observation $\mathbf{z}[k]$, say

$$\widehat{\mathbf{s}}_{k+1|k} = \zeta[k]\widehat{\mathbf{s}}_{k|k-1} + \gamma[k]\mathbf{z}[k] + \beta[k] \tag{5.34}$$

By exploiting Equation 5.34, we can show that an extended state representation which incorporates the next-state estimate is also an MLSS. In particular, consider the appended state vector $\tilde{\mathbf{s}}[k] = \begin{bmatrix} \mathbf{s}[k] \\ \widehat{\mathbf{s}}_{k|k-1} \end{bmatrix}$. It turns out that, if $\{\mathbf{s}[k]\}$ is an MLSS of degree $\widehat{r}$, then $\{\tilde{\mathbf{s}}[k]\}$ is also an MLSS of degree $\widehat{r}$. As part of the justification, we can check the first-moment linearity condition for the appended state vector, as follows:

$$\begin{aligned}
E\left(\tilde{\mathbf{s}}[k+1] \,|\, \tilde{\mathbf{s}}[k]\right) &= E\left(\begin{bmatrix} \mathbf{s}[k+1] \\ \widehat{\mathbf{s}}_{k+1|k} \end{bmatrix} \,|\, \mathbf{s}[k], \widehat{\mathbf{s}}_{k|k-1}\right) \\
&= \begin{bmatrix} E(\mathbf{s}[k+1] \,|\, \mathbf{s}[k]) \\ E(\zeta[k]\widehat{\mathbf{s}}_{k|k-1} + \gamma[k]\mathbf{z}[k] + \beta[k] \,|\, \mathbf{s}[k], \widehat{\mathbf{s}}_{k|k-1}) \end{bmatrix} \\
&= \begin{bmatrix} H_{1,1}\mathbf{s}[k] + H_{1,0} \\ \zeta[k]\widehat{\mathbf{s}}_{k|k-1} + \gamma[k]E(\mathbf{z}[k] \,|\, \mathbf{s}[k]) + \beta[k] \end{bmatrix} \\
&= \begin{bmatrix} H_{1,1}\mathbf{s}[k] + H_{1,0} \\ \zeta[k]\widehat{\mathbf{s}}_{k|k-1} + \gamma[k]C_{1,1}\mathbf{s}[k] + \gamma[k]C_{1,0} + \beta[k] \end{bmatrix} \\
&= \begin{bmatrix} H_{1,1} & 0 \\ \gamma[k]C_{1,1} & \zeta[k] \end{bmatrix} \begin{bmatrix} \mathbf{s}[k] \\ \widehat{\mathbf{s}}_{k|k-1} \end{bmatrix} + \begin{bmatrix} H_{1,0} \\ \gamma[k]C_{1,0} + \beta[k] \end{bmatrix}.
\end{aligned} \tag{5.35}$$

Similarly, by invoking the linear structure of the next-state update, as well as the conditional independence of the next-state and current observation given the current state, we can verify the higher-moment linearity conditions.

Since the appended state process is an MLSS, statistics of the next-state estimate at each time-step, as well as cross-statistics between the next-state estimate and the state, can be

computed. With a bit of algebraic manipulation, we can also determine statistics of the estimate $\widehat{\mathbf{s}}_{k|k}$, since this estimate is linear with respect to $\widehat{\mathbf{s}}_{k|k-1}$ and $z[k]$.

We note that, by considering the LMMSE update procedure, $\zeta[k]$, $\gamma[k]$, and $\beta[k]$ can be viewed as non-linear functions (which are constructed in a recursive manner) of the first and second moments of the state at times up to k and of the initial estimate. Hence, the moment recursions for the appended state vector can be thought of as non-linear difference equation describing the moments of the state and the estimate. Such an explicit representation of the dependence of the moment recursion matrices on previous moments may be valuable, e.g., in characterizing the steady-state dynamics of the estimator.

We believe that, in addition to allowing computation of estimator statistics, the appended state representation may be useful for developing partial-state-information controllers for MLSS. Given partial state information, estimation of the complete state is often a requisite step in the controller design. By using an appended state vector, we can represent controllers that use linear feedback of the state estimate as a control input as MLSS. Hence, the dynamics of the controlled systems can be characterized.

5.6.2 Other Directions for Study

- The form of the LMMSE estimator suggests some possibilities for accurate non-linear estimation. For instance, the *gain* or weighting given to the previous estimate and the observation in the LMMSE estimator depends on expected variance of the noise in the observation and state update procedures; it is tempting to replace this expected variance with an estimated variance, which uses the observed data to approximate the actual variance in the observation and state-update procedures. (Note that, in contrast to purely linear systems, the spreads in the observation and state-update equations of an MLSS may depend on the current state, and hence an estimate for these variances based on the data may be valuable for state estimation.) We hope to compare non-linear estimators of this form with the LMMSE estimator in the future.
- Much remains to be done in comparing our LMMSE estimator with other estimators applicable to systems that can be represented as MLSS. For instance, several estimators have been developed for jump-linear systems; a comparison of our estimator with these would be valuable.
- We have mentioned that LMMSE smoothing for MLSS can be achieved. We can also generalize the estimator to minimize a weighted mean-square-error. These variations

on the LMMSE estimator are straightforward to derive, but their explicit construction may be useful in particular applications. We leave this to future work.

Chapter 6

Minimum Expected Quadratic Cost Dynamic Control of MLSS

In this chapter, we discuss finite time-horizon minimum quadratic cost control of an MLSS with full state information. We first introduce the concept of inputs, and of stochastic quadratic costs, in the context of MLSS. We then derive the optimal input policy (i.e., the rule for choosing the input at each time-step from the observed state in order to minimize the expected cost over the time horizon). The derivation shows that, in analogy with quadratic cost control of linear systems, the optimal input at each time-step is linear with repect to the observed state. Finally, we illustrate quadratic control in a couple of examples.

For the sake of clarity, we limit our analysis to time-invariant MLSS and MLSN. The analysis of time-varying models is completely analogous.

6.1 Relevant Literature

Minimum expected quadratic cost control of noisy linear systems has been extensively studied. A description of several variants of this control problem—including control with and without perfect state information, control in discrete- and continuous-time systems, and finite and infinite horizon control—can be found in [20].

Of particular interest to us is the control of linear systems with stochastic parameters (e.g., [44, 78, 11, 140]), and of systems with state- and control- dependent noise (e.g., [82, 97]). Systems of these sorts have interactions that are similar to some interactions that can be modeled using MLSS (e.g., updates based on choosing randomly among several linear systems, and noise dynamics with state-dependent spread). Early work on systems with stochastic parameters and/or state-dependent noise [78, 82] focused on specifying the optimal controller, by (simply) generalizing the analysis of purely linear systems. Other

research (e.g., [44, 11]) has been concerned with characterizing the qualitative nature of the optimal controller, and the dynamics of the controlled system. For example, [11] shows that infinite-horizon optimal controllers do not exist for certain linear systems with stochastic parameters, reflecting the fact that optimal control of future dynamics is impossible for sufficiently noisy systems. In the examples at the end of this chapter, we occasionally point out qualitative features of MLSN control that match with (or differ from) those discussed in the literature.

More recently, quadratic cost control algorithms have been developed for jump-linear systems (see, e.g., [33, 35] for examples of discrete-time control). It turns out that many of the particular quadratic cost control problems that have been considered in the literature fit within our modeling framework.

We believe that our study of optimal control of MLSS is valuable in that it allows us to develop controllers for several generalizations of purely linear systems, and for systems with seemingly non-linear dynamics (such as influence and queueing networks).

6.2 MLSS with Inputs

To develop algorithms for dynamically controlling MLSS, we must introduce the concept of an input signal in our model. The inputs in our model are structured in such a way that the next state is first- and second-moment linear with respect to both the current state and the current input, as specified in the following definition[1]. *A system is a 2nd-degree MLSS with state sequence* $\mathbf{s}[k]$ *and input sequence* $\mathbf{u}[k]$ *if the conditional distribution for the next state is independent of the past history of the system, given the current state and input, and if there exist matrices* $H_{1,1}$, $D_{1,1}$, $H_{1,0}$, $H_{2,2}$, $G_{2,2}$, $D_{2,2}$, $H_{2,1}$, $D_{2,1}$, *and* $H_{2,0}$ *such that*

$$\begin{aligned}
&E(\mathbf{s}[k+1] \,|\, \mathbf{s}[k], \mathbf{u}[k]) = H_{1,1}\mathbf{s}[k] + D_{1,1}\mathbf{u}[k] + H_{1,0} \\
&E(\mathbf{s}[k+1]^{\otimes 2} \,|\, \mathbf{s}[k], u[k]) \\
&\quad = H_{2,2}\mathbf{s}[k]^{\otimes 2} + G_{2,2}(\mathbf{s}[k] \otimes \mathbf{u}[k]) + D_{2,2}\mathbf{u}[k]^{\otimes 2} \\
&\quad + H_{2,1}\mathbf{s}[k] + D_{2,1}\mathbf{u}[k] + H_{2,0}.
\end{aligned} \tag{6.1}$$

[1] We consider only first- and second-order statistics in the definition of an MLSS with input because only the first and second moments and cross-moments are needed for designing an optimal quadratic cost controller.

That is, a Markov process is a 2nd-degree MLSS with input $\mathbf{u}$ *if the first and second moments and cross-moments of the next state, given the current state and input, are first- and second-degree polynomials, respectively, of current state and input variables.*

6.2.1 Examples of MLSS with Inputs

We have restricted MLSS inputs to a form for which analytical expressions for optimal quadratic controllers can be found, yet several relevant types of input interactions can nevertheless be represented. To motivate our formulation of MLSS with inputs, we highlight a few types of input interactions that fall within our framework.

- State updates that are linear with respect to the previous state and input can be represented.

  Example 6.1
  Consider the scalar system with state updated as

  $$s[k+1] = 0.9s[k] + u[k] + N[k], \tag{6.2}$$

  where $N[k]$ *is a zero mean, unit variance independent Gaussian white noise process. Then the first two conditional moments for* $s[k+1]$*, given* $s[k]$ *and* $u[k]$*, are*

  $$\begin{aligned} E(s[k+1] \mid s[k], u[k]) &= 0.9s[k] + u[k] \\ E(s[k+1]^2 \mid s[k], u[k]) &= 0.81s^2[k] + 1.8s[k]u[k] + u^2[k] + 1 \end{aligned} \tag{6.3}$$

  Equation 6.3 verifies that the described state update specifies a 2nd-degree MLSS.

- State updates in which one of several linear functions of the state and input is chosen randomly can be represented.

  Example 6.2
  Consider the scalar system with state updated as

  $$s[k+1] = 0.9s[k] + N[k]u[k], \tag{6.4}$$

  where $N[k]$ *is a zero mean, unit variance independent Gaussian white noise process. Then the*

first two moments for $s[k+1]$, given $s[k]$ and $u[k]$, are

$$\begin{aligned} E(s[k+1] \mid s[k]) &= 0.9s[k] \\ E(s[k+1]^2 \mid s[k]) &= 0.81s^2[k] + u^2[k]. \end{aligned} \tag{6.5}$$

Thus, the update 6.5 specifies a 2nd-degree MLSS with input.

- Some updates with state- and input-dependent noise statistics can be represented.

Example 6.3

Consider a scalar system in which the next state $s[k+1]$ is generated from $s[k]$ and $u[k]$ as a Poisson random variable with mean $s[k]+u[k]$. (Assume here that $u[k]$ is chosen in such a way that $s[k]+u[k]$ is positive at each time-step.) Note that the conditional distribution of the "noise" in $\mathbf{s}[k+1]$, or the deviation from its conditional mean, depends on $s[k]$ and $u[k]$. Nevertheless, the model can be represented as a 2nd-degree MLSS, since $E(s[k+1] \mid s[k], u[k]) = s[k]+u[k]$, and $E(s[k+1]^2 \mid s[k], u[k]) = (s[k]+u[k])^2+s[k]+u[k] = s^2[k]+2s[k]u[k]+u^2[k]+s[k]+u[k]$.

- Some systems with indicator states and inputs can be modeled.

Example 6.4

Consider a controlled m-status Markov chain that operates as follows. At each time-step, the state $\mathbf{s}[k]$ of the Markov chain, defined to indicate the status of the chain, is updated based on either the previous state or the external input:

- *With probability p, the next state is determined from the current state according to the $m \times m$ transition matrix D.*
- *With probability $1-p$, the next state is generated according to the probabilities specified in a length-n input vector $\mathbf{u}[k]$. Note that the input $\mathbf{u}[k]$ must be a probability vector—i.e., its entries must be non-negative and sum to 1.*

The conditional expectation for $\mathbf{s}[k+1]$, given $\mathbf{s}[k]$ and $\mathbf{u}[k]$, is linear with respect to $\mathbf{s}[k]$ and $\mathbf{u}[k]$:

$$E(\mathbf{s}[k+1] \mid \mathbf{s}[k], \mathbf{u}[k]) = pD'\mathbf{s}[k] + (1-p)\mathbf{u}[k]. \tag{6.6}$$

Also, since the elements of $E(\mathbf{s}[k+1]^{\otimes 2} \mid \mathbf{s}[k], \mathbf{u}[k])$ are either entries of $\mathbf{s}[k]$ or are 0, this expectation can also be written as a linear function of $\mathbf{s}[k]$ and $\mathbf{u}[k]$.

As a particular example, consider a controlled Markov chain with $p = 0.9$, $D = \begin{bmatrix} 0.8 & 0.2 & 0 \\ 0 & 0.7 & 0.3 \\ 0 & 0 & 1 \end{bmatrix}$,

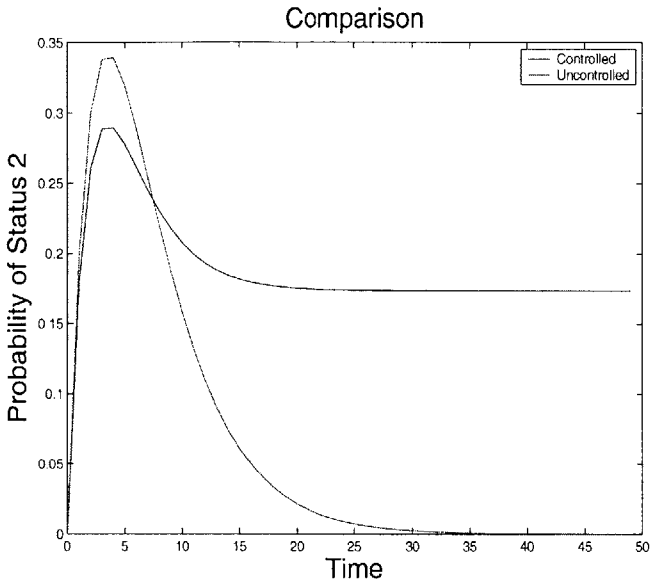


Figure 6.1: We compare the probability that the example controlled Markov chain is in status 2 (the curve with the lower peak that settles to a non-zero value) with the probability that the corresponding uncontrolled Markov chain in in status 2. The uncontrolled Markov chain eventually settles in status 3, so the probability that it is in status 2 approaches 0. At small time-steps, however, the uncontrolled chain does not get reset to the first status like the controlled chain, and so is more likely to be in status 2.

and $\mathbf{u}[k] = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$. *This system naturally evolves from Status 1 to Status 3, but is reset to Status 1 periodically by the external input. The first-moment recursion for this controlled Markov chain allows us to determine the probabilities that the chain is in each status at each time step. The probability that this controlled Markov chain is in Status 2 is compared with the probability that an uncontrolled Markov chain with transition matrix D is in Status 2 (Figure 6.1).*

6.3 The Problem of Interest

Now that an MLSS with input has been defined, we are ready to pose a minimum-quadratic-cost dynamic control problem. The cost that we minimize is a sum of incremental costs over a total of $T + 1$ time-steps, $k = 0, 1, \ldots, T$. The (scalar) incremental cost $C[k]$ at

time k is allowed to be stochastic, but its statistics are assumed to be determined solely by $\mathbf{s}[k]$ and $\mathbf{u}[k]$ (i.e., $C[k]$ is assumed independent of the past history of the system, given $\mathbf{s}[k]$ amd $\mathbf{u}[k]$). We assume that the expected cost $E(C[k] \mid \mathbf{s}[k], \mathbf{u}[k])$ has the following quadratic form:

$$\begin{aligned} & E(C[k] \mid \mathbf{s}[k], \mathbf{u}[k]) \\ & = \mathbf{s}'[k] P_{ss} \mathbf{s}[k] + \mathbf{s}'[k] P_{su} \mathbf{u}[k] + \mathbf{u}'[k] P_{uu} \mathbf{u}[k] \\ & + Q_s \mathbf{s}[k] + Q_u \mathbf{u}[k] + R, \end{aligned} \tag{6.7}$$

where the matrix $\begin{bmatrix} P_{ss} & \frac{1}{2}P_{su} \\ \frac{1}{2}P'_{su} & P_{uu} \end{bmatrix}$ is assumed to be (strictly) positive definite.

Our goal is to minimize the expected total cost over T time-steps, or $E(\sum_{k=0}^{T} C[k])$, by designing the inputs $\mathbf{u}[0], \ldots, \mathbf{u}[T]$ to dynamically control the state of the MLSS. We assume that the inputs at each time k are chosen based on *perfect state information*— i.e., the controller has available the exact current state, as well as past states and inputs, for use in choosing the current input $\mathbf{u}[k]$. In the ensuing discussion, we design the optimal *policy*, or rule for determining the input from the available state and past input information, at each time step. In mathematical notation, our goal is to determine the sequence of inputs $\mathbf{u}^*[0], \ldots, \mathbf{u}^*[T]$ such that

$$\mathbf{u}^*[0], \ldots, \mathbf{u}^*[T] = \arg \min_{\mathbf{u}[0], \ldots, \mathbf{u}[T]} E(\sum_{k=0}^{T} C[k]), \tag{6.8}$$

where each optimal input $u^*[k]$ must be determined solely from the current state, and past states and inputs.

6.3.1 Motivation for Allowing Stochastic Costs

We explicitly model costs as being stochastic (i.e., we specify costs in terms of conditional expectations given the state and input, rather than as explicit functions of the state and input). Our subsequent development of the control algorithm will show that this stochasticity has no bearing on the optimal control policy—the same control is derived if the stochastic cost is replaced by its mean[2]. Nevertheless, we feel that it is useful to model costs as

[2]The equivalence of the control algorithm for certain and uncertain costs is one example of *certainty equivalence* in quadratic control theory. Certaintly equivalence, or the equivalence of a control policy when a stochas-

stochastic given the state and output, because costs in problems of interest to us may in fact have a random component. The following example suggests one simple setting where stochastic quadratic costs may be reasonable.

Example 6.5

There has been some interest in recent years in analyzing the occurrence of cascading failures *(i.e., large and/or propagating failures) in networks. It has been found that real networks are often robust to single component failures, but can be prone to cascading failures if two or more components fail at the same time. As a very simple representation for costly cascading failures, let's say that the network has a state variable $s[k]$ that indicates the probability a single component will fail at a given time-step (e.g., during a day). (The value of this state variable will likely be modulated by design or control inputs, as well as system conditions such as loading.) Assuming that individual component failures are essentially independent, the probability that a cascading failure occurs at time-step k is $s^2[k]$. It is reasonable that a cascading failure would incur a large fixed cost, say C, while no cost would be incurred if a cascading failure did not occur. Thus, the cost $C[k]$ at time k is stochastic, but the expected cost is $E(C[k] \mid s[k]) = Cs^2[k]$, which is quadratic in $s[k]$.*

6.4 Deriving the Optimal Policy

It is well-known that optimizations of the form of Equation 6.8 can be solved iteratively using the dynamic programming (DP) algorithm (see, e.g, [20] for an introduction to DP). In a dynamic programming approach, the optimal policy at the final stage (e.g., at the final time-step contained in the cost expression) is determined first. Subsequently, policies are determined recursively backward-in-time, with the policy at each time determined assuming that optimal policies will be applied at future time steps.

For our system, the optimal policy at the final time-step is

$$\mathbf{u}^*[T](\mathbf{s}[T]) = \arg\min_{\mathbf{u}[T]} E(C[T] \mid \mathbf{s}[T], \mathbf{u}[T]), \tag{6.9}$$

and the corresponding *terminal cost*, or time T incremental cost assuming that the optimal

tic parameter is replaced by its mean, is an interesting feature of several optimal control procedures [20].

policy is used, is

$$J_T(\mathbf{s}[T]) = \min_{\mathbf{u}[T]} E(C[T] \mid \mathbf{s}[T], \mathbf{u}[T]). \tag{6.10}$$

(Our notation suggests that the optimal policy, and corresponding optimal cost, depend only on $\mathbf{s}[T]$ and not on previous states and inputs; this is generally true for Markovian systems such as ours [20].) By substituting the quadratic cost at time T (Equation 6.7 evaluated at $k = T$) in Equation 6.9, doing a bit of calculus, and invoking the positive definiteness of $\begin{bmatrix} P_{ss} & \frac{1}{2}P_{su} \\ \frac{1}{2}P'_{su} & P_{uu} \end{bmatrix}$, we find that the optimal policy at the final stage is

$$\mathbf{u}^*[T](\mathbf{s}[T]) = \frac{1}{2}P_{uu}^{-1}(-P'_{su}\mathbf{s}[T] - Q'_u) \tag{6.11}$$

The corresponding terminal cost is

$$J_T(\mathbf{s}[T]) = \mathbf{s}'[T]Y_2[T]\mathbf{s}[T] + Y_1[T]\mathbf{s}[T] + Y_0[T], \tag{6.12}$$

where

$$\begin{aligned} Y_2[T] &= P_{ss} - \frac{1}{4}P_{su}P_{uu}^{-1}P'_{su} \\ Y_1[T] &= Q_s - \frac{1}{2}Q_uP_{uu}^{-1}P'_{su} \\ Y_0[T] &= -\frac{1}{4}Q_uP_{uu}^{-1}Q'_u + R \end{aligned}$$

We can straightforwardly check that $Y_2[T]$ is positive definite, by invoking the positive definiteness of $\begin{bmatrix} P_{ss} & \frac{1}{2}P_{su} \\ \frac{1}{2}P'_{su} & P_{uu} \end{bmatrix}$ to check the positivity of $\mathbf{s}'[T]Y_2[T]\mathbf{s}[T]$ for all $\mathbf{s}'[T]$. It is also easy to see that $Y_2[T]$ is symmetric.

Next, assume that the optimal policy from time $k+1$ to time T, or $\mathbf{u}^*[k+1](\mathbf{s}[k+1]), \ldots, \mathbf{u}^*[T](\mathbf{s}[T])$, has been found, and that the corresponding optimal cost is given by the quadratic form

$$\begin{aligned} J_{k+1}(\mathbf{s}[k+1]) &\stackrel{\triangle}{=} E(\sum_{i=k+1}^{T} C[i] \mid \mathbf{s}[k+1], \mathbf{u}[k+1]) \\ &= \mathbf{s}'[k+1]Y_2[k+1]s'[k+1] + Y_1[k+1]\mathbf{s}[k+1] + Y_0[k+1], \end{aligned} \tag{6.13}$$

for some symmetric positive definite $Y_2[k+1]$, and some $Y_1[k+1]$ and $Y_0[k+1]$. Applying

the DP algorithm (see [20]), we find that the optimal cost at time k is given by

$$J_k(\mathbf{s}[k]) = \min_{\mathbf{u}[k]} \{E(C[k] \,|\, \mathbf{s}[k], \mathbf{u}[k]) + E(J_{k+1}(\mathbf{s}[k+1]) \,|\, \mathbf{s}[k], \mathbf{u}[k])\}, \tag{6.14}$$

and that the corresponding optimal policy at time k is

$$\mathbf{u}^*[k](\mathbf{s}[k]) = \arg \min_{\mathbf{u}[k]} \{E(C[k] \,|\, \mathbf{s}[k], \mathbf{u}[k]) + E(J_{k+1}(\mathbf{s}[k+1]) \,|\, \mathbf{s}[k], \mathbf{u}[k])\}. \tag{6.15}$$

Let's consider $E(J_{k+1}(\mathbf{s}[k+1]) \,|\, \mathbf{s}[k], \mathbf{u}[k])$. Substituting Equation 6.13 for $J_{k+1}(\mathbf{s}[k+1]$, we find that

$$\begin{aligned} &E(J_{k+1}(\mathbf{s}[k+1]) \,|\, \mathbf{s}[k], \mathbf{u}[k]) \\ &= E(\mathbf{s}'[k+1]Y_2[k+1]s'[k+1] \\ &+Y_1[k+1]\mathbf{s}[k+1] + Y_0[k+1] \,|\, \mathbf{s}[k], \mathbf{u}[k]). \end{aligned} \tag{6.16}$$

Since the expectation given in Equation 6.16 is quadratic with respect to $\mathbf{s}[k+1]$, the equation can be rewritten as a quadratic function of $\mathbf{s}[k]$ and $\mathbf{u}[k]$ using the moment-linearity conditions for an MLSS with input $\mathbf{u}[k]$. Some algebra is required for this calculation, since the moment-linearity conditions are expressed in terms of vector moments, while Equation 6.16 is in quadratic form. We skip the details of this algebra, and simply present the final expression for $E(J_{k+1}(\mathbf{s}[k+1]) \,|\, \mathbf{s}[k], \mathbf{u}[k])$:

$$\begin{aligned} &E(J_{k+1}(\mathbf{s}[k+1]) \,|\, \mathbf{s}[k], \mathbf{u}[k]) \\ &= \mathbf{s}'[k]W_{ss}[k]\mathbf{s}[k] + \mathbf{s}'[k]W_{su}[k]\mathbf{u}[k] + \mathbf{u}'[k]W_{uu}[k]\mathbf{u}[k] \\ &+Z_s[k]\mathbf{s}[k] + Z_u[k]\mathbf{u}[k] + L[k], \end{aligned} \tag{6.17}$$

where

$$\begin{aligned} W_{ss}[k] &= \frac{1}{2}\text{UNVEC}(\widehat{Y}_2[k+1]H_{2,2}) + \frac{1}{2}\text{UNVEC}(\widehat{Y}_2[k+1]H_{2,2})' \\ W_{su}[k] &= \text{UNVEC}(\widehat{Y}_2[k+1]G_{2,2}) \\ W_{uu}[k] &= \frac{1}{2}\text{UNVEC}(\widehat{Y}_2[k+1]D_{2,2}) + \frac{1}{2}\text{UNVEC}(\widehat{Y}_2[k+1]D_{2,2})' \\ Z_s[k] &= \widehat{Y}_2[k+1]H_{2,1} + Y_1[k+1]H_{1,1} \\ Z_u[k] &= \widehat{Y}_2[k+1]D_{2,1} + Y_1[k+1]D_{1,1} \\ L[k] &= \widehat{Y}_2[k+1]H_{2,0} + Y_1[k+1]H_{1,0} + Y_0[k+1] \\ \widehat{Y}_2[k+1] &= \text{VEC}(Y_2[k+1]), \end{aligned}$$

and where *vec* and *unvec* are defined as follows:

- VEC concatenates the rows of a matrix into a row vector.
- UNVEC places blocks of a row vector as rows in a matrix, with the length of the blocks chosen to match the proper dimension of the resulting matrix.

We note that the matrix $\begin{bmatrix} W_{ss}[k] & \frac{1}{2}W_{su}[k] \\ \frac{1}{2}W'_{su}[k] & W_{uu}[k] \end{bmatrix}$ is positive semi-definite and symmetric.

Finally, substituting Equations 6.17 and 6.7 into Equation 6.14, we find that

$$\begin{aligned} J_k(\mathbf{s}[k]) = \min_{\mathbf{u}[k]} & \\ & \mathbf{s}'[k]P_{ss}[k]\mathbf{s}[k] + \mathbf{s}'[k]P_{su}[k]\mathbf{u}[k] + \mathbf{u}'[k]P_{uu}[k]\mathbf{u}[k] \\ & +Q_s[k]\mathbf{s}[k] + Q_u[k]\mathbf{u}[k] + R[k], \end{aligned} \tag{6.18}$$

where

$$\begin{aligned} P_{ss}[k] &= W_{ss}[k] + P_{ss} \\ P_{su}[k] &= W_{su}[k] + P_{su} \\ P_{uu}[k] &= W_{uu}[k] + P_{uu} \\ Q_s[k] &= Z_s[k] + Q_s \\ Q_u[k] &= Z_u[k] + Q_u \\ R[k] &= L[k] + R, \end{aligned}$$

and where $\begin{bmatrix} P_{ss}[k] & \frac{1}{2}P_{su}[k] \\ \frac{1}{2}P'_{su}[k] & P_{uu}[k] \end{bmatrix}$ is symmetric and positive definite.

Notice that the minimization problem that we solve to find the optimal policy at time k (Equation 6.18) is identical in form to the minimization that we used to find the optimal policy at time T. Thus, we immediately find that the optimal policy at time k is

$$\mathbf{u}^*[k](\mathbf{s}[k]) = \frac{1}{2}P_{uu}[k]^{-1}(-P'_{su}[k]\mathbf{s}[k] - Q'_u[k]), \tag{6.19}$$

and that the corresponding optimal cost is

$$J_k(\mathbf{s}[k]) = \mathbf{s}'[k]Y_2[k]\mathbf{s}[k] + Y_1[k]\mathbf{s}[k] + Y_0[k], \tag{6.20}$$

where

$$Y_2[k] = P_{ss}[k] - \frac{1}{4}P_{su}[k]P_{uu}[k]^{-1}P'_{su}[k]$$
$$Y_1[k] = Q_s[k] - \frac{1}{2}Q_u[k]P_{uu}[k]^{-1}P'_{su}[k]$$
$$Y_0[k] = -\frac{1}{4}Q_u[k]P_{uu}[k]^{-1}Q_u[k]' + R[k].$$

Again, we can straightforwardly verify that $Y_2[k]$ is symmetric and positive definite.

Since $J_k(\mathbf{s}[k])$ takes the same quadratic form as $J_{k+1}(\mathbf{s}[k+1])$, and since $J_T(\mathbf{s}[T])$ has this same form, the procedure used to determine $J_k(\mathbf{s}[k])$ can be applied iteratively to find the optimal cost at the initial stage, or $J_0(\mathbf{s}[0])$, and the optimal policy over all time-steps. Thus, we have completed the minimum quadratic cost control algorithm for MLSS.

6.5 Examples of Miminum Quadratic Cost Control of MLSS

In this section, we present two simple examples of minimum quadratic cost control of MLSS. These examples give an indication of the range of control problems that can be analyzed using our methods, and they explore some interesting dynamics observed in controlled MLSS. Later in the thesis, we will consider a couple more examples of minimum quadratic cost control, in the context of flow networks.

6.5.1 Controlling Linear Systems with Random Parameters

We consider the following MLSS with length-2 state vector and length-2 input vector:

$$E(\mathbf{s}[k+1] \mid \mathbf{s}[k]) = \begin{bmatrix} 1.35 & 0 \\ 0 & 1.35 \end{bmatrix} \mathbf{s}[k] + \alpha[k]\mathbf{u}[k], \quad w.p.\ 0.4 \tag{6.21}$$
$$\begin{bmatrix} 0.8 & 0.2 \\ 0.1 & 0.6 \end{bmatrix} \mathbf{s}[k] + \alpha[k]\mathbf{u}[k], \quad w.p.\ 0.6, \tag{6.22}$$

where $\alpha[k]$ is 1 with probability 0.9 and 0 otherwise. This system is a generalization of a standard linear system, in that the next state is chosen randomly among several different linear combinations of the current state and input.

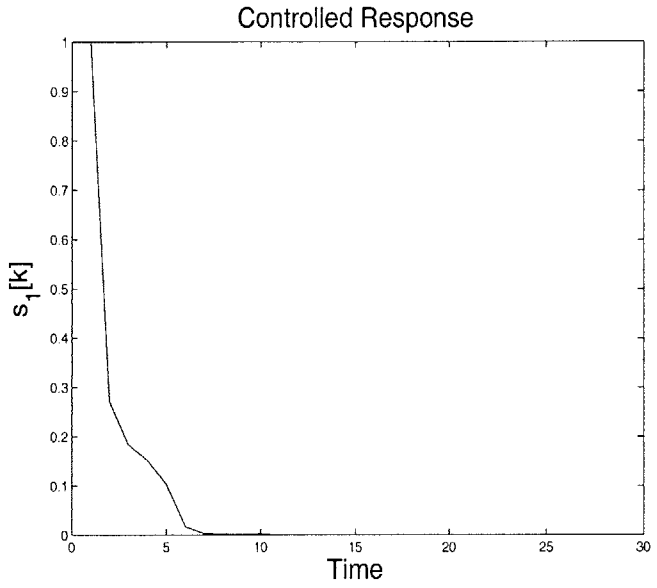


Figure 6.2: $s_1[k]$ is plotted during a 30 time-step simulation of the MLSS, for which the input has been chosen according to the optimal policy. Both state variables are assumed to initially equal 1.

We use the control algorithm developed above to find the optimal input policy over a time horizon of 30 steps, assuming an incremental cost $C[k] = s_1^2[k] + s_2^2[k]$ at each time-step. The value of the first state variable in a typical simulation of the controlled system is shown in Figure 6.2. The simulation suggests that the state variable $s_1[k]$ converges to 0.

The optimal policy for the control input into this MLSS becomes stationary (time-invariant) at an infinite time horizon (i.e., at many time-steps before the final time-step considered in the dynamic program). For this example, the stationary policy is

$$\mathbf{u}^* = \begin{bmatrix} -0.64 & -0.07 \\ -0.07 & -0.56 \end{bmatrix} \mathbf{s}[k]. \tag{6.23}$$

It is interesting to compare the dynamics of this MLSS when the control input Equation 6.23 is applied with the dynamics of the system when no input is applied. Interestingly, the mean response of this system is unstable (divergent) without input, but the optimal control input stabilizes the mean response. In Figure 6.3, we compare the response of the system with and without the input.

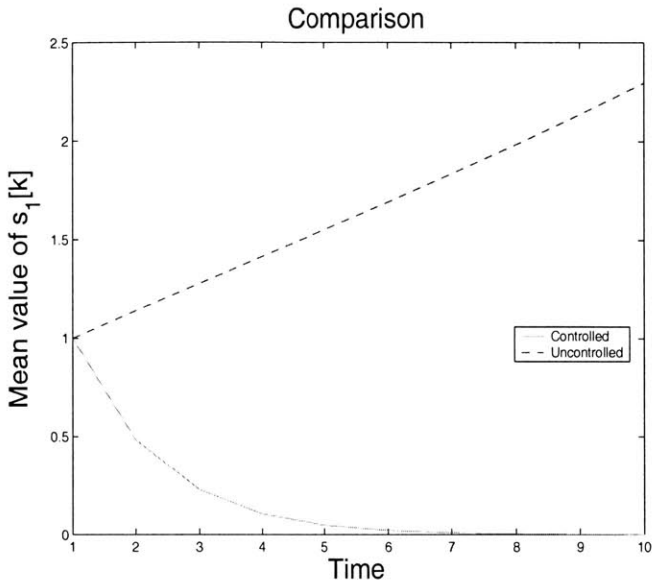


Figure 6.3: The mean response of the MLSS with the stationary optimal policy applied is compared with the mean response with no control input used. The controlled system is stable, while the uncontrolled one is not. The initial values of the state variables in both cases are assumed to be $s_1[0] = s_2[0] = 1$.

Although a stationary optimal policy exists for the particular MLSS considered here, a slightly modified version of the model does not have a stationary policy. In particular, if the state is updated as

$$E(\mathbf{s}[k+1] \,|\, \mathbf{s}[k]) = \begin{bmatrix} 1.5 & 0 \\ 0 & 1.5 \end{bmatrix} \mathbf{s}[k] + \alpha[k]\mathbf{u}[k], \quad w.p.\, 0.4 \tag{6.24}$$

$$\begin{bmatrix} 0.8 & 0.2 \\ 0.1 & 0.6 \end{bmatrix} \mathbf{s}[k] + \alpha[k]\mathbf{u}[k], \quad w.p.\, 0.6, \tag{6.25}$$

then a stationary optimal policy does not exist, and in fact simulations show that the state itself does not converge even if the (non-stationary) optimal policy is applied. Regardless of what policy is used, it turns out that the variance in the state variables increases with time, so that the state does not converge. The non-existence of a stationary optimal policy reflects an interesting distinction between quadratic control in MLSS and in purely linear systems: in linear systems, a stationary optimal policy always exists, but the same is not true for certain MLSS, for which the optimal control gain diverges in the infinite time horizon limit. This phenomenon has been observed, in particular, for linear systems with

random parameters [11], and for jump-linear systems [139]. In these systems, it is found that there are strict conditions on the parameters of the system that guarantee existence of a stationary optimal policy, and otherwise the system is sufficiently "uncertain" to prevent development of a stationary optimal policy. This boundary between systems for which stationary optimal policies can and cannot be found is referred to as the *uncertainty threshold*. We leave it to future work to determine if MLSS have a similar notion of an uncertainty threshold, and, if so, to quantify the threshold.

6.5.2 Controlling the Infinite-Server Queue

We pursue minimum quadratic cost control of the infinite-server queue with random service probabilities (Example 2.4.5). We modify the original example from Section 2.4.5 to allow some control of the input process into the queue. In particular, assume that the number of jobs entering the queue at each time-step is a Poisson random variable, with a mean that can be controlled in response to the measured state. Our objective in controlling the input process is to minimize the average squared deviation between the actual number of jobs in the queue and a target number of jobs (which we have for purposes of illustration chosen to be equal to the average number of jobs in the original example). That is, we develop the optimal controller, assuming an incremental cost $C[k]$ equal to the square of the difference between $s_1[k]$ and the target 21.8.

It is straightforward to check that the described control problem can be analyzed in our framework, so we do not detail this formulation and only present a few results of the analysis. In this example, we find a stationary optimal policy (i.e., we find that the backwards-in-time DP recursion for the optimal policy reaches a fixed point). The stationary policy specifies that the input $u[k]$ at time k should be determined from the number of jobs $s[k]$ in the queue as $u[k] = 19.8 - 0.52s[k]$. Thus, as we might expect, the average number of jobs entering the queue should be reduced if the number of jobs in the queue increases, and should be increased if this number is small.

Figure 6.4 shows simulations of the queue in which the stationary policy is applied to determine the input at each time-step. In addition to the simulations, the figure shows the target number of jobs in the queue, as well 2 "standard-deviation" intervals around this target, which can be found from the DP cost functions. The simulations and analysis show that the controlled queue does track the target number of jobs more closely than the uncontrolled queue described in Section 2. In particular, the root mean square deviation

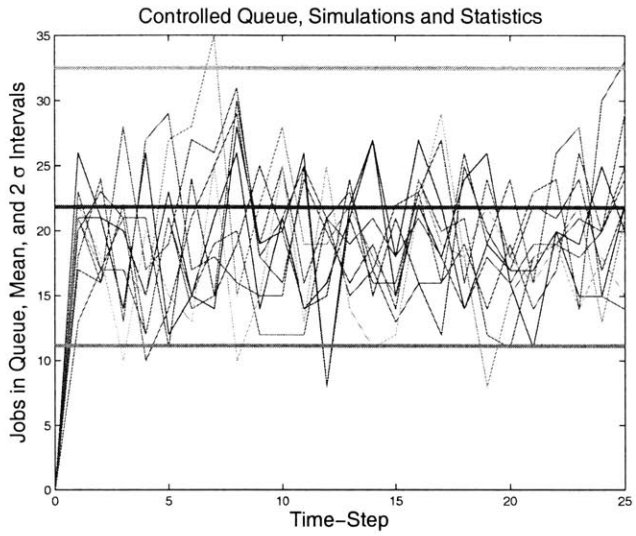


Figure 6.4: Simulations of the infinite-server queue example with optimal input rate control are shown. Along with the simulations, mean and 2σ intervals around the mean are shown. The figure shows that controlling the input rate can be used to reduce the squared deviation of the actual queue length from a target length.

of the actual queue length from the target drops from 6.5 to 5.2.

Interestingly, the average number of jobs in the controlled queue is slightly less than the target (19.8 vs. 21.8). Because the queue is occasionally prone to low service probabilities, the control algorithm guards against excessively large numbers of jobs in order to minimize the expected squared deviation. Thus, the control algorithm aims for a slightly lower number of jobs on average than the target. This bias is borne out in the simulations of the queue.

So far, we have considered control of the input rate into the infinite-server queue. Alternately, the actual number of jobs entering the queue at each time step can be controlled. The optimal policy (i.e., the policy that minimizes squared deviation from a target queue length) for the number of jobs allowed to enter the queue has been derived. Simulations of the queue length, as well as calculations of the expected queue length and two standard deviation intervals around mean, are shown for this case (Figure 6.5). The figure shows that the additional precision afforded by direct control of the number of jobs entering the queue can be used to decrease the expected squared deviation from the target, to 4.3.

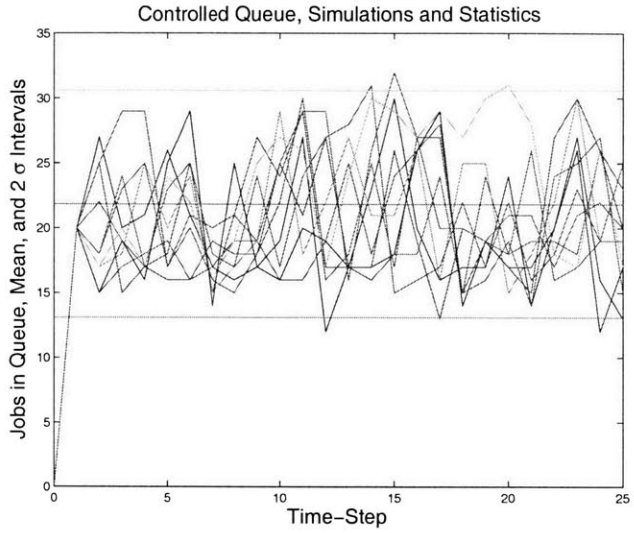


Figure 6.5: The infinite-server queue is simulated assuming that the number of jobs entering the queue at each time step is optimally controlled. Along with the simulations, mean and 2σ intervals around the mean are shown. The figure shows that control of the actual number of entering jobs allows closer tracking of input rates than control of the input rates.

6.6 Future Work: Control Using Only Output Measurements

An important aim of future work for us is to develop methods for control that use only partial state information (in particular, moment-linear observations of the state), rather than the state itself. For linear systems, minimum quadratic cost control from partial state information is tractable because of a *separability theorem*, which allows for separate implementation of a minimum mean-square-error state estimator and a quadratic cost controller that uses the state estimate as to determine its input (see, e.g., [20]). As noted in [20], however, the minimum quadratic cost control problem has no analytical solution if the linear system parameters constitute an i.i.d. random process. Since the linear system with random parameters can be formulated as an MLSS, we expect that an analytical solution for optimal control from partial state information cannot be developed. Nevertheless, if we constrain our estimator, or our estimator and controller, to be linear, we can perhaps develop an optimal controller that uses only moment-linear observations of the state. We leave it to future work to explore whether controllers of this form can be developed, and whether a notion of separability can be shown.

Chapter 7

The Influence Model

We briefly introduced the influence model in Chapter 3 as an example of an MLSN. In this chapter, we explore the influence model in some more detail. Our goals in considering the influence model here are the following:

- To highlight the influence model as a special subclass of MLSN, for which moments and cross-moments of state variables specify (individual or joint) status probabilities, and for which the moment-linearity conditions can be interpreted in terms of conditional probabilities.
- To revisit the original analysis of the influence model in [9] and [10] from a different viewpoint, and to catalog several new results on its analysis.
- To explore further the relationship of the influence model with other models, and to briefly introduce some of its potential applications.
- To better understand the MLSN modeling framework through this detailed case study of the influence model.

7.1 Motivation and Related Literature

The influence model, originally introduced in the the thesis [9] and in the article [10], seeks to represent, in an analytically tractable way, certain stochastic interactions that may occur among the components of a network. The influence model comprises a network of interacting Markov chains, in which the evolution of each chain can be influenced not only by the present status of that chain but also the statuses of neighboring chains in the network (Figure 7.1).

At a broad level, the influence model is motivated by a need for network models that

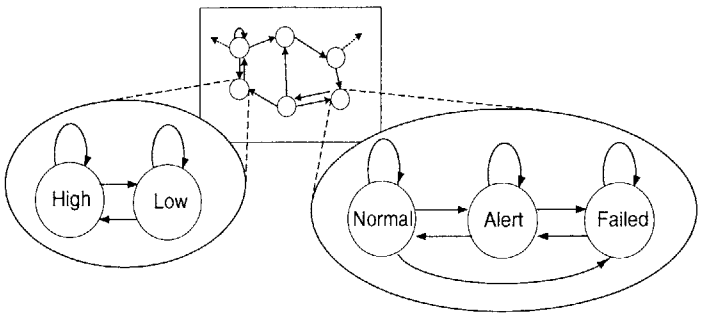


Figure 7.1: As depicted here, the influence model was introduced in [9] as a network of interacting Markov chains.

allow us to understand the causes and characteristics of the global dynamic behavior of networks. Modern engineered networks (e.g., power, communication, and transportation networks) very often are multi-faceted, incoporating physical components, data communications channels, and human elements. Though individual components of such networks can often be modeled well, it remains difficult to assemble or aggregate these component models in such a way that global dynamics can be analyzed or understood. MLSN models, and the influence model in particular, are specially structured to allow inference of global network characteristics. We believe that such structured models—while limited in their modeling capabilities—are valuable as tractable representations of some essential dynamics that occur in networks. As such, we believe that they may provide a good context for studying several aspects of engineered network behavior, including failure and event propagation, estimation, control, and design. Our development in this chapter will show how these aspects of network dynamics can be pursued in the context of the influence model. Through examples, we will also seek to clarify the types of interactions that can be represented using the influence model structure, and to introduce potential specific applications for the model.

The influence model connects with literature from several areas of research. We briefly discuss some of these connections here.

The influence model is one example of a *stochastic cellular automaton*—i.e., a finite state-space Markov model with dynamics embodied in interactions among sites in a network. There is an extensive literature on stochastic cellular automata in general, and on their potential applications in modeling natural and engineered systems (e.g., [137, 138]). Such automata come in many varieties, with different specifications of local dynamics and inter-

action structure, and different approaches to analysis. A recent example is provided by the paper [117], which determines steady-state probabilities of configurations in a stochastic automaton defined on a line, with arbitrary nearest-neighbor interactions.

A class of systems referred to as *interacting particle systems* (or *infinite particle systems*) includes models such as the voter model [71],[46], the contact process [60], and the stochastic Ising model [25],[58], all of which bear similarities to specific forms of our influence model. The standard setup for each of these interacting particle systems is as follows. The network is generally an infinite multi-dimensional lattice, with each lattice site being in one of two possible statuses at any instant of time. Each site also has an "alarm clock" that strikes randomly with an exponential waiting time. When the clock strikes, the site switches to the opposite status. The firing rate of the clock at any given time depends on the current statuses of the site and its neighbors. The differences among the preceding models lie in how the rates of the clocks depend on the concerned statuses [49].

Of the above models, the one that most resembles a specific form of our influence model is the *voter model*, also called the *invasion process*, introduced independently in [71] and [34]. In [34] the invasion process was proposed as a model for spatial conflict of different species. The firing rate of the alarm clock at each site of this model is proportional to the number of neighbors that are currently in the opposite status. Thus, the more neighbors a site has with the opposite status, the faster it is likely to switch to the opposite value. If the statuses of all the neighbors agree with that of a given site, then the status of that site will not change.

A specialization of our influence model is the *binary influence model*, in which each site is allowed to have one of only two statuses (whereas in the general influence model the chains can vary in order and structure from site to site), and with the further restriction that transitions between these statuses at a given site can only occur under the influence of neighbors that have the opposite status. The resulting structure is quite similar to the voter model or invasion process, but with two notable differences. First, the binary influence model evolves in discrete rather than continuous time, so the structure of the network graph (for instance, whether it is ergodic or not) comes more strongly into play in determining whether all sites reach a consensus status. Second, our binary influence model is formed with finite but arbitrarily connected and arbitrarily weighted graphs, whereas the voter model literature largely focuses on uniformly weighted infinite graphs and/or graphs with regular structure, such as the lattice [71, 90], the torus [38, 39], undirected and uniformly weighted graphs [46], or infinite translation-invariant graphs [95]. An exception

is [79], in which there are versions of the invasion process model that are much more like the binary influence model (or natural generalizations of it to the case of more than two statuses per site), although still in continuous time.

Discrete-time stochastic process models for graphical (network) interactions are also studied under the heading of *dynamic Bayesian networks* (DBNs) [105, 56]. The influence model is a particular DBN, in which the interactions among the nodes have a special quasi-linear structure. DBNs are of particular interest to us, because questions of state (and parameter) estimation have been studied in their context. Our methods for linear estimation of MLSN suggest novel approaches for state estimation in certain DBNs. For instance, the state update of a *factorial HMM*—a particular DBN in which each node evolves independently [57]—is a special case of the influence model update. If the observations of a factorial HMM are moment-linear with respect to the state (which is usually the case, since observations are typically Gaussian whose mean is a linear function of the global state), then MLSS estimators can be applied to the factorial HMM.

Another interesting related system is the *Markov Chain Monte Carlo* model. Good introductions to this model are given in [24] and [110]. In this model, sites are chosen at random or according to a prespecified order, and they are updated in a probabilistic way, based on the statuses of their neighbors. The update rules are chosen so that the steady-state distribution yields a Markov random field. Markov Chain Monte Carlo and Markov random field models have found considerable use in image-processing and other applications, [64].

The *stochastic automata network* is yet another model consisting of interacting finite-state stochastic components. The book [129], citing [12] and related literature, describes a form in which the automata are interacting Markov processes. Two specific types of interactions among the automata are considered: functional transitions, in which the rate of a transition in one automaton is dependent on the state of another automaton; and synchronizing events, in which a transition in one automaton forces a transition in another one. The evolution of the joint state of all the automata is described by a master Markov process with number of states equal to the product of the number of states for the constituent automata. It is shown in [12] that the generator matrix for this large master Markov process can be written using Kronecker products of square matrices whose sizes equal the orders of the constituent automata. The special structure of the generator matrix is then used to efficiently calculate the steady-state expected joint status of the sites in the network. As will be described later in this article, we too use a formalism based on Kronecker products to determine the joint statuses of groups of sites in the influence model.

Yet another branch of research that potentially relates to the influence model focuses on simulating partial differential equations using interactions on discrete lattices. For example, the lattice-gas model simulates the Navier-Stokes equations using a hexagonal lattice representation of the system [120]. The model consists of particles possessing 'momentum' that travel along the edges of the lattice and interact probabilistically, redistributing momentum when they come in contact with each other.

7.2 The Influence Model

This section briefly reviews the basic development of the influence model given in [9] and [10].

The influence model comprises a network of n interacting nodes or vertices or *sites*. Site i assumes one of a finite number m_i of possible *statuses* at each discrete-time instant. The status of site i at any time k is represented by an m_i-component *status vector*, which is an indicator vector containing a single 1 in the position corresponding to the present status, and 0 everywhere else:

$$\mathbf{s}_i'[k] = [\,0 \cdots 0\,1\,0 \cdots 0\,] \,.$$

(The prime denotes matrix transposition.)

We are concerned with the evolution of sites' statuses at discrete time-steps. The update of the statuses constitutes a Markov process, in that their joint probability distribution at the next time step is independent of past statuses, given the current statuses of all sites. In particular, updating the status of the ith site in the influence model can be thought of as involving the following three stages:

- Stage 1: The site i randomly selects one of its neighboring sites in the network—or selects itself—to be its *determining site* for Stage 2; site j is selected with probability d_{ij} (so $d_{ij} \geq 0$ and $\sum_j d_{ij} = 1$).
- Stage 2: The present status $\mathbf{s}_j'[k]$ of the determining site j fixes the probability vector $\mathbf{p}_i'[k+1]$ that will be used in Stage 3 to randomly choose (or "realize") the next status of site i (so $\mathbf{p}_i'[k+1]$ is a row vector with nonnegative entries that sum to 1). Specifically, if site j is selected as the determining site, then $\mathbf{p}_i'[k+1] = \mathbf{s}_j'[k]A_{ji}$, where A_{ji} is a

fixed *row-stochastic* $m_j \times m_i$ matrix, i.e., a matrix whose rows are probability vectors, with nonnegative entries that sum to 1. We call the matrices A_{ji} the *local status-evolution matrices* of the influence model. The net effect of these first two stages is the same as if the next status of site i is realized according to the probability vector $\mathbf{p}_i'[k+1] = \sum_{j=1}^{N} d_{ij}\mathbf{s}_j'[k]A_{ji}$.

- Stage 3: The next status $\mathbf{s}_i'[k+1]$ is realized in accordance with the $\mathbf{p}_i'[k+1]$ computed at Stage 2.

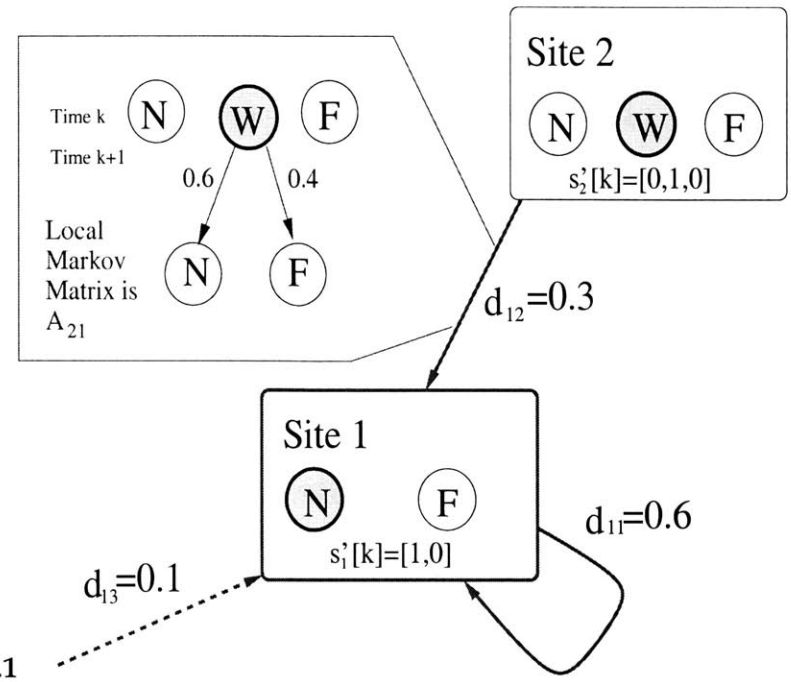
All sites are updated simultaneously, and independently, in this way.

In words, each site updates its status by randomly choosing a determining site, and then generating a new status based on a probability vector specified according to the current status of this determining site. We think of each site as being "influenced" by its neighboring sites, since the current statuses of these neighbors specify the probabilities for the next status of this site of interest.

We have introduced d_{ij} here as the *probability* that site i chooses site j as its determining site. Alternately, the probability d_{ij} can be interpreted as the *weight* given to site j in specifying the probability vector for the next status of site i. In this interpretation, each site j provides a probability vector $\mathbf{s}_j'[k]A_{ji}$, which depends on the current status of site j. The next status of site i is realized from a weighted combination of these probability vectors, namely $\sum_{j=1}^{n} d_{ij}\mathbf{s}_j'[k]A_{ji}$.

In Figure 7.2, we illustrate the update rule for a single site (Site 1) at a particular time-step. In this influence model, Site 1 may choose itself as its determining site (with probability $d_{11} = 0.6$), may choose Site 2 as its determining site (with probability $d_{12} = 0.3$), or may choose Site 3 (with probability $d_{13} = 0.1$), at each time-step.

Consider the case where Site 1 chooses Site 2 as its determining site. Then the probability distribution for the next-status of Site 1, which may be either Normal (N), or Failed (F), is specified by the current status of Site 2, which may be Normal (N), Warning (W), or Failed (F). For example, if Site 2 is in status W, as depicted in the figure, the next status of Site 1 is N with probability 0.6 and F with probability 0.4. In the notation of the influence model, the probability vectors for the next status of Site 1, given that Site 2 is its determining site, are contained in rows of the matrix A_{21}. In this example, the second row of this matrix, which corresponds to the W status for Site 2, is $[0.6, 0.4]$).



Example 7.1

Figure 7.2: The update rule for a single site in an influence model. The status symbols N, W, and F denote Normal, Warning, and Failed respectively.

The probabilities d_{ij} with which determining neighbors are selected in effect *define* the network associated with the influence model (in that they define the way in which sites interact with each other). We assemble these probabilities in an $n \times n$ row-stochastic matrix $D = [d_{ij}]$, called the *network matrix*. Meanwhile, the row-stochastic matrices A_{ij} show how statuses of influenced sites are updated, and so are called the *local status-evolution (or transition) matrices*. An influence model is completely specified by a single network matrix along with local status evolution matrices A_{ij}, $1 \leq i \leq n$, $1 \leq j \leq n$.

It is useful to construct a graphical representation for an influence network. To do so, recall the common association of a directed graph $\Gamma(X)$ with an $n \times n$ matrix X: this graph has n nodes and a directed edge of weight x_{ij} from node i to node j if and only if the entry x_{ij} of X is nonzero. For our purposes, it is somewhat more natural to use the graph of D', namely $\Gamma(D')$, which we call the *network influence graph* or simply the *network graph*. This graph has a directed edge of weight d_{ij} *from* site j *to* site i precisely when $d_{ij} > 0$, and the weight of this directed edge is the probability that site i will select site j to determine its next status. The sum of the weights of the edges directed into a site is therefore always 1.

Example 7.2

Consider an influence model with one site. The network matrix for such an influence model is necessarily $D = d_{11} = 1$. *Thus, the single site is always self-determined, according to a local status-evolution matrix* A_{11}. *In particular, the probability vector for the time-*$(k+1)$ *status of the site is specified by the time-*k *status, as* $\mathbf{s}_1'[k]A_{11}$. *The update rule for the influence model is identical to the status update of a Markov chain with transition matrix* A_{11} *(see Section 2.4.2). This example shows that a Markov chain can be viewed as an influence model with one site.*

Example 7.3

We consider an influence model with two sites. The first site can assume two statuses, called Normal (N) and Failed (F). The status vector $\mathbf{s}_1'[k] = [1,0]$ *indicates the first, or Normal, status, while* $\mathbf{s}_1'[k] = [0,1]$ *indicates the second, or Failed, status. The second site can assume three statuses, called Normal (N), Warning (W), and Failed (F), indicated by status vectors* $\mathbf{s}_2'[k] = [1,0,0]$, $\mathbf{s}_2'[k] = [0,1,0]$, *and* $\mathbf{s}_2'[k] = [0,0,1]$, *respectively. The network matrix for this example is given to be*

$$D = \begin{bmatrix} 0.95 & 0.05 \\ 0.3 & 0.7 \end{bmatrix}. \tag{7.1}$$

Thus, both sites in the model are likely to determine their own next statuses, though Site 1 *influences Site* 2 *more frequently than Site* 2 *influences Site* 1.

The local status-evolution matrices for this example are

$$A_{11} = \begin{bmatrix} 0.95 & 0.05 \\ 0.5 & 0.5 \end{bmatrix} \tag{7.2}$$

$$A_{21} = \begin{bmatrix} 1 & 0 \\ 0.5 & 0.5 \\ 0 & 1 \end{bmatrix}$$

$$A_{12} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$A_{22} = \begin{bmatrix} 0.9 & 0.1 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.3}$$

Conceptually, the model evolves as follows. The status of Site 1 *is usually updated by self-influence;*

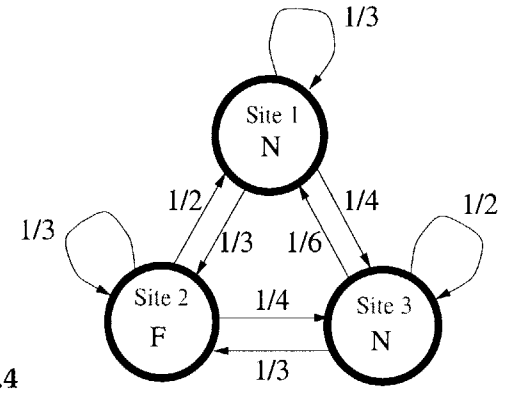
this status transitions back and forth between the Normal and Failed statuses in this case. In contrast, the status of Site 2 remains the same or decays (i.e., goes from Normal to Warning or Warning to Failed) if it is determined by itself. Site 2 requires the occasional influence from Site 1 to return to a Normal status.

Time-Step	Site 1	Site 2
0	F	F
1	F	F
2	N	F
3	F	F
4	F	F
5	N	F
6	N	F
7	N	F
8	N	N

Figure 7.3: Eight time-steps in a simulation of the two-site example are shown. Both sites are initially in Failed status.

Figure 7.3 shows a simulation of this influence model. Both sites are initially in a Failed status. In this instance, 8 time-steps are required for both sites to simultaneously recover to Normal status. As seen in Figure 7.3, Site 1 recovers first, through self-influence, and Site 2 is subsequently repaired through influence from Site 1.

The influence model can be generalized to allow time-varying parameters (i.e., a time-varying network matrix and time-varying status evolution matrices). The resulting stochastic model is a time-varying MLSN. The generalization to this time-varying case is straightforward, so we do not consider it in any further detail. Though the extension to a model with time-varying parameters direct, we note that such generalizations can exhibit some interesting dynamics that are not present in time-invariant influence models.



Example 7.4

Figure 7.4: The network graph for the three-site example is depicted.

7.2.1 The Homogeneous Influence Model

In a *homogeneous influence model,* the weights of influences among sites (i.e., the d_{ij}) can be arbitrary, but the local interactions are uniform throughout the network. That is, a homogeneous influence model is one in which each site has the same number of possible statuses $m_i = m$, and the local status-evolution matrices are identical throughout the network ($A_{ij} = A$ for all i, j). The homogeneous influence model is of interest to us because it is specially tractable, as later development will show, and it is also closer to other models that have been considered in the literature, as noted earlier. Although a homogeneous influence model is highly structured, we believe that it can potentially represent some networks of interest, such as networks in which the interacting components are indistinguishable in some sense (e.g., DNA sequences, which are strings of the same four nucleotide base pairs).

We repeat the example of the homogeneous influence model from Section 3.3.1, see Figure 7.4. In this example, each site can be in one of two possible statuses, Normal (N) or Failed (F). The common local status-evolution matrix for this example is $A = \begin{bmatrix} 0.99 & 0.01 \\ 0.05 & 0.95 \end{bmatrix}$. *Simulations of the status of Site* 1 *are shown in Figure 7.5.*

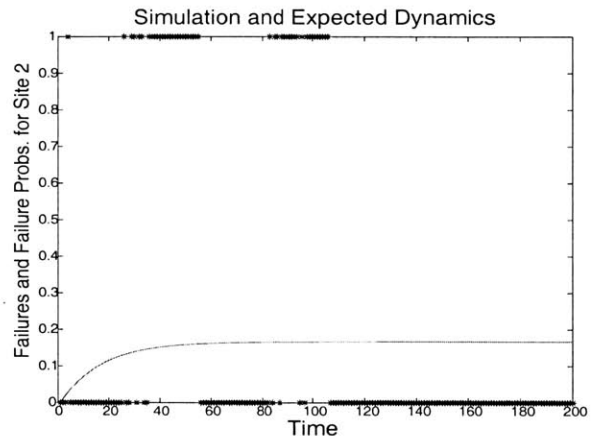


Figure 7.5: This figure shows the failure probability for Site 2 during 200 time-steps, given that all sites are initially normal. A simulation of the status of Site 2 is also shown. (A "*" at the top of the graph indicates a Failed site, and a "*" at the bottom indicates a Normal site.)

7.3 The Influence Model as an MLSN

To reformulate the influence model as an MLSN, we form a state vector $\mathbf{s}[k]$ by stacking (concatenating) the status vectors of the n sites:

$$\mathbf{s}[k] \triangleq \begin{bmatrix} \mathbf{s}_1[k] \\ \vdots \\ \mathbf{s}_n[k] \end{bmatrix} \tag{7.4}$$

Since the next state $\mathbf{s}[k+1]$ is generated solely based on the time k statuses, the sequence $\{\mathbf{s}[k]\}$ is a Markov process. Further, from our specification of the influence model, we can see that the state process is stationary. Thus, to verify that the sequence $\{\mathbf{s}[k]\}$, constitutes an MLSS, it only remains to check for rth moment linearity of the state update for each r.

7.3.1 Verification of the First Moment Linearity Condition

We seek to show that the conditional expectation

$$E(\mathbf{s}[k+1] \,|\, \mathbf{s}[k]) = \begin{bmatrix} E(\mathbf{s}_1[k+1] \,|\, \mathbf{s}[k]) \\ \vdots \\ E(\mathbf{s}_n[k+1] \,|\, \mathbf{s}[k]) \end{bmatrix} \tag{7.5}$$

is linear with respect to $\mathbf{s}[k]$.

To do so, let's consider the conditional expectation $E(\mathbf{s}_i[k+1] \,|\, \mathbf{s}[k])$ separately for each i. Because $\mathbf{s}_i[k+1]$ indicates the status of site i, its expectation (given $\mathbf{s}[k]$) is the probability vector for the next status of site i (given $\mathbf{s}[k]$). From our formulation of the influence model, we know that this probability vector is given by $(\sum_{j=1}^{n} d_{ij}\mathbf{s}_j'[k]A_{ji})'$. (Note that we have transposed the expression for the probability vector given earlier, since we are specifying a column vector rather than a row vector in the MLSN formulation.) Thus, the conditional expectation for each site's next-status vector is

$$E(\mathbf{s}_i[k+1] \,|\, \mathbf{s}[k]) = \sum_{j=1}^{n} d_{ij} A_{ji}' \mathbf{s}_j[k]. \tag{7.6}$$

Stacking the expectations $E(\mathbf{s}_i[k+1] \,|\, \mathbf{s}[k])$ yields

$$E(\mathbf{s}[k+1] \,|\, \mathbf{s}[k]) = H'\mathbf{s}[k], \tag{7.7}$$

where

$$H \triangleq \begin{bmatrix} d_{11}A_{11} & \cdots & d_{n1}A_{1n} \\ \vdots & & \vdots \\ d_{1n}A_{n1} & \cdots & d_{nn}A_{nn} \end{bmatrix} \triangleq D' \otimes \{A_{ij}\}. \tag{7.8}$$

Equation 7.7 verifies that the first moment linearity condition holds for the state sequence of the influence model. The matrix H, which specifies the first moment recursion for the system, is denoted the *influence matrix*[1]. The notation we have defined for H is a generalization of the familiar Kronecker product notation for a pair of matrices. Specifically, the

[1]Incidentally, we specify the first moment linearity condition as $E(\mathbf{s}[k+1] \,|\, \mathbf{s}[k]) = H'\mathbf{s}[k]$ rather than $E(\mathbf{s}[k+1] \,|\, \mathbf{s}[k]) = H\mathbf{s}[k]$ in order to maintain the notation used in [9].

(i,j)th block in H is the product of the ith row, jth column entry of the matrix D' (hence d_{ji}) with the matrix A_{ij}.

We note that the matrix H completely specifies the update of the influence model: given the current state, conditional probabilities for each site's next status are contained in the vector $H'\mathbf{s}[k]$. Thus, it is not surprising that the network matrix D' and the local interaction matrices, which also specify the update of the model, can be extracted from the influence matrix.

Example 7.5
Again consider the two-site influence model of Example 7.3. The influence matrix for this influence model is

$$H = \begin{bmatrix} 0.95 \begin{bmatrix} 0.95 & 0.05 \\ 0.5 & 0.5 \end{bmatrix} & 0.3 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ 0.05 \begin{bmatrix} 1 & 0 \\ 0.5 & 0.5 \\ 0 & 1 \end{bmatrix} & 0.7 \begin{bmatrix} 0.9 & 0.1 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 0 & 1 \end{bmatrix} \end{bmatrix} \tag{7.9}$$

7.3.2 Higher Moment Linearity Conditions

To verify the rth moment linearity condition, we seek to express the conditional rth (vector) moment $E(\mathbf{s}[k+1]^{\otimes r} \mid \mathbf{s}[k])$ as an affine function of $\mathbf{s}[k], \ldots, \mathbf{s}[k]^{\otimes r}$. To do so, we first compute conditional expectations for rth joint statuses at time $k+1$, given $\mathbf{s}[k]$, and then assemble these conditional expectations to form (a permuted and reduced form of) the conditional rth moment.

Recall from Chapters 3 and 4 that higher moment linearity conditions of MLSN, including the influence model, can be expressed in many different forms. The form considered here is chosen to facilitate certain analyses concerning asymptotics and settling properties of the influence model. An alternate form for the higher conditional vector moments is given in [9].

Our development in this section, as well as in the following section on settling times of the influence model, requires a wealth of notation. For convenience, a list of these notations is given in Table 7.1.

Notation	Brief Description
$\mathbf{v}, \mathbf{w}$	groupings (lists of sites)
$\mathbf{s}_{\mathbf{v}}[k]$	joint status vector for $\mathbf{v}$
$\mathbf{i}$	index for joint status vector entries
$ess(\mathbf{v})$	essential site list of $\mathbf{v}$
$\mathbf{w}_{\mathbf{v}}[k]$	determining site list for $\mathbf{v}$
$\mathbf{s}_{[r]}[k]$	influence state vector
$\mathbf{s}_{(r)}[k]$	extended influence state vector
$Q(\mathbf{v})$	mapping between $\mathbf{s}_{\mathbf{v}}[k]$ and $\mathbf{s}_{ess(\mathbf{v})}[k]$
$H_{i,j}$	moment recursion matrix
$H_{i,j}(\mathbf{v}, \mathbf{w})$	part of moment recursion matrix
$H_{(i)}$	extended recursion matrix (for influence model)
$H_{(1)} = H_{1,1} = H'$	first (extended) recursion matrix

Table 7.1: We list the notations used in the subsequent development. A brief description of each symbol is also given. Full definitions for each can be found in the text.

7.3.2.1 Expressions for Conditional Moments and Cross-Moments of Status Vectors

Generally, construction of the conditional rth moment requires determination of conditional expectations of all rth joint status vectors (i.e., joint status vectors of all n^r rth groupings). For the influence model, it turns out that only joint status vectors of groupings of distinct sites need be considered, since these joint status vectors are sufficient to construct a reduced representation of the rth moment linearity condition[2].

We introduce the following notation for our development:

- Recall from Chapter 3 that an rth-order grouping $\mathbf{v}$ is a listing of r sites: that is, it is an r-vector whose components $v_1, \ldots, v_r$ are each in $1, \ldots, n$.
- Recall that the joint status vector for the grouping $\mathbf{v}$ is $\mathbf{s}_{\mathbf{v}}[k+1] = \mathbf{s}_{v_1}[k+1] \otimes \mathbf{s}_{v_2}[k+1] \otimes \ldots \otimes \mathbf{s}_{v_r}[k+1]$.)
- A grouping $\mathbf{v}$ is called a *distinct grouping* if all the sites in the grouping are distinct. Note that only groupings of n or fewer sites can be distinct groupings.
- Consider the joint-status vector $\mathbf{s}_{\mathbf{v}}[k]$ of a grouping $\mathbf{v}$ of r sites (henceforth called an rth joint status vector). This joint-status vector has length $\prod_{i=1}^{r} m_{v_i}$. We use a vector

[2]For $r > n$, there are no groupings of r distinct sites, so no rth groupings need be considered in checking the rth moment linearity condition. This reflects that the nth vector moment specifies all influence model statistics, so that higher moments are irrelevant.

notation to index the elements of this vector. In particular, we identify the entry in $\mathbf{s}_{\mathbf{v}}[k]$ corresponding to the product of entry i_1 of status vector $\mathbf{s}_{v_1}[k]$, entry i_2 of status vector $\mathbf{s}_{v_2}[k]$, etc., by the list $\mathbf{i} = (i_1, i_2, \ldots i_r)$. We denote this product as *entry* $\mathbf{i}$ of the joint status vector $\mathbf{s}_{\mathbf{v}}[k]$.

- The expected value of the joint status of a grouping is called the *expected joint status* of that grouping.
- Consider a grouping $\mathbf{v}$. Now construct a second list of sites $\widehat{\mathbf{v}}$ in the following way: first, enter the first site in $\mathbf{v}$ as the first site in $\widehat{\mathbf{v}}$; then sequentially scan the sites in $\mathbf{v}$, and add any site that is not already in $\widehat{\mathbf{v}}$ to the end of that list. We call this vector, which contains all distinct sites in $\widehat{\mathbf{v}}$, the *essential site list* of $\mathbf{v}$. We use the notation $\widehat{\mathbf{v}} = ess(\mathbf{v})$ to specify the essential site list of $\mathbf{v}$. Note that an essential site list is a distinct grouping.
- Consider an essential site list $\widehat{\mathbf{v}}$ which contains r or fewer sites. In general, there are multiple groupings with r sites which have essential site list $\widehat{\mathbf{v}}$. Call the set of such groupings $\mathcal{W}_r(\widehat{\mathbf{v}})$.

Example 7.6
Consider an influence model with three sites, and consider the two groupings $\{3, 1, 3\}$ *and* $\{3, 3, 1\}$. *The essential site lists of both these groupings is* $\{3, 1\}$.

Let's first discuss why joint status vectors of distinct groupings of r sites (henceforth called distinct rth joint status vectors) are sufficient for specifying a reduced representation for the rth moment linearity condition. According to the discussion in Chapter 4, we need the entries of the distinct rth joint status vectors to be entries of $\mathbf{s}[k]^{\otimes r}$, and this is clearly true. For the distinct rth joint status vectors to serve as a reduced representation, all rth joint status vectors must be expressible as linear functions of rth and lower distinct joint status vectors. To see why this can be done, consider the joint status vector for an arbitrary grouping $\mathbf{v}$ of r sites. Each entry of this vector is a product of one or more entries from each of the status vectors of the sites in $ess(\mathbf{v})$. Since each site's status vector is a $0 - -1$ indicator, these products are either 0 or they are products of individual entries from the status vectors of sites in $ess(\mathbf{v})$—i.e., they are entries of $\mathbf{s}_{ess(\mathbf{v})}[k]$. Thus, we see that the joint status vector $\mathbf{s}_{\mathbf{v}}[k]$ is a linear function of $\mathbf{s}_{ess(\mathbf{v})}[k]$, where $ess(\mathbf{v})$ is a grouping of no more than r sites. From here on, we use the notation

$$\mathbf{s}_{\mathbf{v}}[k] = Q(\mathbf{v})\mathbf{s}_{ess(\mathbf{v})}[k] \tag{7.10}$$

to specify this linear function. The explicit general construction of the matrices $Q(\mathbf{v})$ is

tangential to our development, so we leave it to Appendix B. We mention one special case. If the sites listed in $\mathbf{v}$ are distinct, then $ess(\mathbf{v}) = \mathbf{v}$, and $Q(\mathbf{v})$ is an identity matrix. From the above discussion[3], it is clear that the distinct joint status vectors can be used to construct a reduced representation for $\mathbf{s}[k]^{\otimes r}$.

Example 7.7

Consider an influence model with two sites, each of which can take on two statuses, and consider the length-8 joint status vector $\mathbf{s}_{\{2,1,2\}}[k]$. *The preceding development suggests that* $\mathbf{s}_{\{2,1,2\}}[k]$ *is a linear function of* $\mathbf{s}_{ess(\{2,1,2\})}[k] = \mathbf{s}_{\{2,1\}}[k]$. *For this simple example, we can construct the linear transformation by inspection:*

$$\mathbf{s}_{\{2,1,2\}}[k] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{s}_{\{2,1\}}[k]. \tag{7.11}$$

Let's now show how to compute conditional expectations for time-$(k+1)$ distinct joint status vectors, given $\mathbf{s}[k]$. In particular, consider the joint status of a *distinct grouping* $\mathbf{v}$ at time $k+1$. Recall that the time-$(k+1)$ statuses of the sites in $\mathbf{v}$ are each independently determined from randomly (and independently) chosen sites in the network. It is useful for us to specify the determining sites for the r sites in $\mathbf{v}$ at this time-step by another length r-component vector $\mathbf{w}_{\mathbf{v}}[k]$, which we call the *determining vector* of $\mathbf{v}$. The conditional expectation for the time-$(k+1)$ joint status of $\mathbf{v}$ given the current state can be rewritten by conditioning further on the determining vector of $\mathbf{v}$:

$$E(\mathbf{s}_{\mathbf{v}}[k+1] \,|\, \mathbf{s}[k]) \tag{7.12}$$

$$= \sum_{\mathbf{w}} E(\mathbf{s}_{\mathbf{v}}[k+1] \,|\, \mathbf{s}[k], \mathbf{w}_{\mathbf{v}}[k] = \mathbf{w}) P(\mathbf{w}_{\mathbf{v}}[k] = \mathbf{w} \,|\, \mathbf{s}[k]) \tag{7.13}$$

$$= \sum_{\mathbf{w}} E(\mathbf{s}_{\mathbf{v}}[k+1] \,|\, \mathbf{s}[k], \mathbf{w}_{\mathbf{v}}[k] = \mathbf{w}) P(\mathbf{w}_{\mathbf{v}}[k] = \mathbf{w}). \tag{7.14}$$

[3] We can also show that the joint status vector of any grouping of r sites can be expressed as a linear function of distinct joint status vectors of *exactly* r sites. In [9], this insight is used to develop moment recursions that involve only rth-order statistics.

In Equation 7.12, the notation $\sum_{\mathbf{w}}$ refers to the sum over all possible determining vectors (i.e., all possible groupings of r sites). Also, we note that the last expression in Equation 7.12 follows from the second expression because the choice of determining sites is independent of the current state vector.

In Equation 7.12, $P(\mathbf{w_v}[k] = \mathbf{w})$ is the probability that site v_1 is determined by site w_1, site v_2 is determined by w_2, and so forth. In terms of influence model parameters, this probability is

$$P(\mathbf{w_v}[k] = \mathbf{w}) = \prod_{j=1}^{r} d_{v_j w_j}. \tag{7.15}$$

Now consider $E(\mathbf{s_v}[k+1] \,|\, \mathbf{s}[k], \mathbf{w_v}[k] = \mathbf{w})$. This expectation can be written as

$$\begin{aligned} &E(\mathbf{s_v}[k+1] \,|\, \mathbf{s}[k], \mathbf{w_v}[k] = \mathbf{w}) \\ &= E(\otimes_{j=1}^{r} \mathbf{s}_{v_j}[k+1] \,|\, \mathbf{s}[k], \mathbf{w_v}[k] = \mathbf{w}). \end{aligned} \tag{7.16}$$

Since each site in $\mathbf{v}$ is updated independently according to its determining site's status, Equation 7.16 can be rewritten as

$$\begin{aligned} &E(\mathbf{s_v}[k+1] \,|\, \mathbf{s}[k], \mathbf{w_v}[k] = \mathbf{w}) \\ &= \otimes_{j=1}^{r} E(\mathbf{s}_{v_j}[k+1] \,|\, \mathbf{s}[k], \mathbf{w_v}[k] = \mathbf{w}) \\ &= \otimes_{j=1}^{r} E(\mathbf{s}_{v_j}[k+1] \,|\, \mathbf{s}_{w_j}[k], (\mathbf{w_v}[k])_j = w_j), \end{aligned} \tag{7.17}$$

where $(\mathbf{w_v}[k])_j$ refers to the jth entry in the vector $\mathbf{w_v}[k]$. The expectation $E(\mathbf{s}_{v_j}[k+1] \,|\, \mathbf{s}_{w_j}[k], (\mathbf{w_v}[k])_j = w_j)$ is simply the probability vector for the next-status of site v_j, given the current status of its determining site w_j, and so can be written in terms of influence model parameters as

$$E(\mathbf{s}_{v_j}[k+1] \,|\, \mathbf{s}_{w_j}[k], (\mathbf{w_v}[k])_j = w_j) = A'_{w_j, v_j} \mathbf{s}_{w_j}[k] \tag{7.18}$$

Substituting this expectation into 7.17 leads to

$$\begin{aligned} E(\mathbf{s_v}[k+1] \,|\, \mathbf{s}[k], \mathbf{w_v}[k] = \mathbf{w}) &= \otimes_{j=1}^{r} A'_{w_j, v_j} \mathbf{s}_{w_j}[k] \\ &= (\otimes_{j=1}^{r} A'_{w_j, v_j})(\otimes_{j=1}^{r} \mathbf{s}_{w_j}[k]) \\ (\otimes_{j=1}^{r} A'_{w_j, v_j}) \mathbf{s_w}[k]. & \end{aligned} \tag{7.19}$$

Next, we use Equation 7.10 to rewrite the expectation in Equation 7.19 solely in terms of time-k joint statuses of distinct groupings, as

$$E(\mathbf{s}_{\mathbf{v}}[k+1] \,|\, \mathbf{s}[k], \mathbf{w}_{\mathbf{v}}[k] = \mathbf{w}) = (\otimes_{j=1}^{r} A'_{w_j,v_j}) Q(\mathbf{w}) \mathbf{s}_{ess(\mathbf{w})}[k]. \tag{7.20}$$

Finally, substituting Equation 7.20 and Equation 7.15 into Equation 7.12, we find that

$$E(\mathbf{s}_{\mathbf{v}}[k+1] \,|\, \mathbf{s}[k]) = \sum_{\mathbf{w}} (\otimes_{j=1}^{r} A'_{w_j,v_j}) Q(\mathbf{w}) \mathbf{s}_{ess(\mathbf{w})}[k] \prod_{j=1}^{r} d_{v_j w_j}. \tag{7.21}$$

Equation 7.21 shows how the conditional expectation of the time-$(k+1)$ joint status of any distinct rth grouping can be expressed as a linear function of time-k joint statuses of distinct groupings of r or fewer sites. This is the essential linearity property that allows us to find recursions for vector moments (and hence joint status probabilities) in the influence model.

Example 7.8

Again consider the two-site influence model of Examples 7.3 and 7.5. Let's consider the conditional expectation $E(\mathbf{s}_{\{2,1\}}[k+1] \,|\, \mathbf{s}[k])$. *According to Equation 7.21, this expectation can be written as*

$$\begin{aligned} &E(\mathbf{s}_{\{2,1\}}[k+1] \,|\, \mathbf{s}[k]) \\ &\quad = d_{21} d_{11} (A'_{12} \otimes A'_{11}) Q(\{1,1\}) \mathbf{s}_1[k] + d_{21} d_{12} (A'_{12} \otimes A'_{21}) \mathbf{s}_{\{1,2\}}[k] \\ &\quad + d_{22} d_{11} (A'_{22} \otimes A'_{11}) \mathbf{s}_{\{2,1\}}[k] + d_{22} d_{12} (A'_{22} \otimes A'_{21}) Q(\{2,2\}) \mathbf{s}_2[k]. \end{aligned} \tag{7.22}$$

7.3.2.2 Conditional Moments of the State Vector

The expressions for conditional expectations of distinct rth joint statuses can be stacked into a reduced form of the rth moment linearity condition. In particular, we define the rth *influence state vector* $\mathbf{s}_{[r]}[k]$ as a stacking of all rth distinct status vectors in lexicographic

order:[4]

$$\mathbf{s}_{[r]}[k] = \begin{bmatrix} \mathbf{s}_{\{1,\ldots,r\}} \\ \vdots \\ \mathbf{s}_{\{r,\ldots,1\}} \end{bmatrix}. \tag{7.23}$$

Note that the rth influence state vector contains status vectors for $\frac{n!}{(n-r)!}$ groupings.

Since conditional expectations of rth joint status vectors at time $k+1$ are linear with respect to rth and lower joint status vectors at time k, the conditional expectation for the time $k+1$ rth influence state vector, given the time k state, has the form

$$E(\mathbf{s}_{[r]}[k+1] \mid \mathbf{s}[k]) = \sum_{j=1}^{r} H_{r,j}\mathbf{s}_{[j]}[k]. \tag{7.24}$$

We call the matrices $H_{r,j}$ the *influence recursion matrices*. We call the part of the matrix $H_{r,j}$ that multiplies the status vector of the distinct grouping $\widehat{\mathbf{w}}$ and contributes to the expected joint status of the distinct grouping $\mathbf{v}$ as $H_{r,j}(\mathbf{v}, \widehat{\mathbf{w}})$. In other words, $H_{r,j}(\mathbf{v}, \widehat{\mathbf{w}})$ specifies the contributions to the next-joint status probabilities of $\mathbf{v}$ form the current status of $\widehat{\mathbf{w}}$; this contribution is effected in the circumstance that the determining site list for $\mathbf{v}$ has essential site list given by $\widehat{\mathbf{w}}$. The matrix $H_{r,j}(\mathbf{v}, \widehat{\mathbf{w}})$ is constructed by considering the summands in Equation 7.21 for which the essential site list $ess(\mathbf{w})$ is the distinct grouping $\widehat{\mathbf{w}}$, as shown in the following equation:

$$H_{r,j}(\mathbf{v}, \widehat{\mathbf{w}}) = \sum_{\mathbf{w} \in \mathcal{W}_r(\widehat{\mathbf{w}})} \left(\prod_{j=1}^{r} d_{v_j w_j}\right)(\otimes_{j=1}^{r} A'_{w_j,v_j})Q(\mathbf{w}) \tag{7.25}$$

Equation 7.25 specifies each part of the recursion matrices, and thus we can construct the moment linearity conditions for the influence model. Thus, we have specified the rth moment linearity condition for the influence model.

Example 7.9

Again consider the two-site influence model of Examples 7.3, 7.5, and 7.8. From Equation 7.22, we

[4]Note that we use the same notation for the rth influence state vector as for the permuted rth-order state vector defined in Chapter 3, although the influence state vector does not contain joint statuses of non-disjoint groupings. More precisely, the rth influence state vector is an example of an rth reduced state vector, in permuted form. Our notation and terminology is meant to simplify the presentation while capturing the essence of the analysis.

can infer the matrices $H_{r,j}(\mathbf{v}, \widehat{\mathbf{w}})$ *for* $\mathbf{v} = \{1, 2\}$. *For instance, we find that*

$$H_{r,j}(\{1,2\},\{1\}) = d_{21}d_{11}(A'_{12} \otimes A'_{11})Q(\{1,1\}). \tag{7.26}$$

7.3.3 Basic Analysis: Influence Moment Recursions

As with all MLSN, the moment-linear structure of the influence model permits linear recursions for computing moments and cross-moments of the status vectors at each time-step in terms of the moments and cross-moments of the initial status vectors. Because the status vectors of the influence model are $0 - -1$ indicator vectors, their moments and cross-moments actually specify the probabilities that a site or sites have certain statuses at a given time-step.

Using the standard analysis for MLSN, the rth *influence vector moment* (i.e., the expectation of the rth influence state vector) at time $k+1$ can be expressed in terms of the first r influence vector moments at time k, as

$$E(\mathbf{s}_{[r]}[k+1]) = \sum_{i=1}^{r} H_{r,i}E(\mathbf{s}_{[i]}[k]). \tag{7.27}$$

By applying equations of the form (7.27) iteratively, the rth influence vector moments—which contain all relevant rth moments and cross-moments of status vectors—can be found from the initial values of the first r influence vector moments. We call Equation 7.27 the rth influence moment recursion.

The moment recursions defined by Equation 7.27 allow us to find the joint probability that a group of sites assumes a certain set of statuses. For example, consider the vector $E(\mathbf{s}_{[r]}[k])$. The vector $s_{[r]}[k]$ contains joint statuses for all rth distinct groupings. For a particular rth distinct grouping $\mathbf{v}$, the entry in $E(\mathbf{s}_{\mathbf{v}}[k])$ (which is contained in $E(\mathbf{s}_{[r]}[k])$) indexed by the vector $\mathbf{i} = (i_1, \ldots, i_r)$ equals the joint probability that sites $v_1, \ldots, v_r$ have statuses $i_1, \ldots, i_r$, respectively, at time k.

Interestingly, the nth influence vector moment $E(\mathbf{s}_{[n]}[k])$ contains a probability vector for the joint status of all n sites. Thus, the expectation $E(\mathbf{s}_{[n]}[k])$ completely specifies the joint p.m.f. for the state (and state vector) of the influence model at time k. Since $E(\mathbf{s}_{[n]}[k])$ can be found using the first n influence moment recursions, we see that moment recursions of order greater than n are not of interest for the influence model.

As with general MLSS and MLSN, the first r influence moment recursions can be concatenated into a single recursion. To do so, we define the rth *extended influence state vector* or simply rth *extended state vector* as

$$\mathbf{s}_{(r)}[k] = \begin{bmatrix} \mathbf{s}_{[r]}[k] \\ \vdots \\ \mathbf{s}_{[1]}[k] = \mathbf{s}[k] \end{bmatrix} \tag{7.28}$$

Next, stacking the first r moment recursions (Equation 7.27) leads to

$$E(\mathbf{s}_{(r)}[k+1]) = \tilde{H}_{(r)} E(\mathbf{s}_{(r)}[k]), \tag{7.29}$$

where

$$\tilde{H}_{(r)} = \begin{bmatrix} H_{r,r} & H_{r,r-1} & \dots & H_{r,1} \\ 0 & H_{r-1,r-1} & \dots & H_{r-1,1} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & H_{1,1} \end{bmatrix}. \tag{7.30}$$

Invoking Equation 7.29 iteratively, the rth *extended moment vector* (i.e., the expectation of the rth extended state vector) at time k can be expressed explicitly in terms of the rth extended moment vector at the initial time:

$$E(\mathbf{s}_{(r)}[k]) = \tilde{H}_r^k E(\mathbf{s}_{(r)}[0]). \tag{7.31}$$

In the preceding development, we have introduced the influence model and described its basic analysis. In the following sections, we discuss several aspects of influence model dynamics, estimation, and control. Our discussion is not mean to be exhaustive by any means (e.g., convergence is not discussed at all; the reader is referred to [9] for a disussion). Instead, we present a few new results and extensions, and highlight some ideas that we find interesting and potentially applicable. In the same vein, we also often limit our discussions to illustrative examples, rather than presenting the most general forms of the result.

7.4 Settling Times

The *settling time*, or time required for a stable system to approach its steady-state, is often a useful characterization of a system's transients. For the influence model, the moment recursions are known to be stable, in the sense that the vector moments approach steady-states (which do not depend on the initial condtion), under fairly general conditions on the network matrix and local status evolution matrix (see [9]). The required conditions for stability can further be phrased in terms of the class structure of influence model graphs, and hence can be checked easily. When the vector moments are stable, settling times are well-defined; these settling times specify how quickly the joint-pmfs of groups of r sites approach their steady-state values.

More specifically, it is shown in [9] that (given the appropriate general conditions on the network and local status evolution matrices), each moment recursion has a single unity eigenvalue[5]. The eigenvector corresponding to this unity eigenvalue is known to specify the steady-state moment vector (which contains steady-state status or joint-status probabilities) for the recursion, [9]. The remaining eigenvalues of the recursion have magnitudes that are strictly less than 1. As discussed in [9], the *subdominant relevant eigenvalue* (i.e., the non-unity eigenvalue of largest magnitude with eigenvector that can be excited by a valid set of initial conditions) of each of these recursion matrices is the key determinant of its settling time. In [9], Asavathiratham conjectures that the subdominant relevant eigenvalues of all moment recursions of a particular influence model are identical. If true, this proposition would be quite a useful result, since it suggests that the settling characteristics of all joint status probabilities can be inferred solely from the eigenanalysis of the first moment recursion, with a dimension equal to the sum of the number of statuses at each site. Here, we prove this subdominant relevant eigenvalue conjecture for the homogeneous influence model and show a counterexample for the general model.

7.4.1 Homogeneous Influence Model: Proof of Conjecture

We prove the preceding conjecture by showing that the subdominant relevant eigenvalue of the rth extended moment recursion $\tilde{H}_{(r)}$ is the subdominant eigenvalue of the local interaction matrix A, for all r. For convenience, assume that D and A specify ergodic

[5]The moment recursions are phrased in a different form in [9] than in Section 7.3 here, but the essentials remain the same (as we will show in some more detail for the homogeneous influence model).

Markov chains. In this case, it's easy to check that the unity eigenvalue of $\tilde{H}_{(r)}$ is unique.

Note that the eigenvalues of the extended recursion matrices $\tilde{H}_{(r)}$ are the union of the eigenvalues of its diagonal blocks, $H_{1,1}, \ldots, H_{r,r}$ (because $\tilde{H}_{(r)}$ has a block-upper triangular form with diagonal blocks given by $H_{1,1}, \ldots, H_{r,r}$). For the homogeneous influence model, the eigenvalues of these diagonal blocks can be straightforwardly related to eigenvalues of the local status interaction matrix A.

In particular, we express the matrix $H_{i,i}$ in terms of $(A')^{\otimes i}$, as follows. We first invoke Equation 7.25 to find the joint status recursion matrix between two distinct ith groupings $\mathbf{v}$ and $\mathbf{w}$:

$$H_{i,i}(\mathbf{v}, \mathbf{w}) = (\prod_{j=1}^{i} d_{v_j w_j})(A')^{\otimes i}, \tag{7.32}$$

where we have used the fact that $\mathbf{v}$ and $\mathbf{w}$ are both ith groupings, and that the the model is homogeneous, to simplify the expression. Next, we form the matrix $H_{i,i}$ by assembling these joint status recursion matrices for distinct ith groupings (in lexicographic order according to both row and column). Since each joint status recursion matrix is a product of a scalar and the matrix $(A')^{\otimes i}$, the recursion matrix $H_{i,i}$ can be written as

$$H_{i,i} = D_{(i)} \otimes (A')^{\otimes i}, \tag{7.33}$$

where $D_{(i)}$ contains the probabilities $\prod_{j=1}^{i} d_{v_j w_j}$ for distinct ith groupings, arranged in lexicographic ordering, row-wise and column-wise. We note that the matrix $D_{(i)}$ can also be constructed by removing rows and columns that are indexed by non-distinct groupings from the matrix $D^{\otimes i}$. We call $D_{(i)}$ the ith network matrix.

Example 7.10

Consider a two-site, two-status influence model, with network matrix

$$D = \begin{bmatrix} 0.9 & 0.1 \\ 0.3 & 0.7 \end{bmatrix} \tag{7.34}$$

and local interaction matrix

$$A = \begin{bmatrix} 0.75 & 0.25 \\ 0.4 & 0.6 \end{bmatrix}. \tag{7.35}$$

The first network matrix is simply $D_{(1)} = D$. *To construct the second network matrix, we first generate the matrix*

$$D^{\otimes 2} = \begin{bmatrix} 0.81 & 0.09 & 0.09 & 0.01 \\ 0.27 & 0.63 & 0.03 & 0.07 \\ 0.27 & 0.03 & 0.63 & 0.07 \\ 0.09 & 0.21 & 0.21 & 0.49 \end{bmatrix}. \tag{7.36}$$

We then remove the rows and columns that are indexed by non-disjoint groupings—i.e., the first and fourth rows and columns—from $D^{\otimes 2}$ *to construct the second network matrix:*

$$D_{(2)} = \begin{bmatrix} 0.63 & 0.03 \\ 0.03 & 0.63 \end{bmatrix} \tag{7.37}$$

Note that $H_{1,1} = H' = D \otimes A'$. Thus, the eigenvalues of $H_{1,1}$ are the Cartesian product of the eigenvalues of D with those of A'. The dominant eigenvalue of $H_{1,1}$ is 1, since the dominant eigenvalues of D and A' are both 1. As shown in [9], the eigenvalues of $H_{1,1}$ that are products of the unity eigenvalue of A' with the eigenvalues of D are not relevant. Thus, the first candidate for the subdominant relevant eigenvalue (i.e., the remaining subdominant eigenvalue of largest magnitude) of $H_{1,1}$ is the subdominant eigenvalue of A', or equivalently the subdominant eigenvalue of A. It is discussed in [9] why this eigenvalue is relevant. Briefly, relevance can be argued as follows: we can show that the components of the right eigenvector of $H_{1,1}$ corresponding to this eigenvalue that are associated with each status vector sum to zero; hence, a valid initial condition for the expected state, for which each expected status is a probability vector, can be modified by a component in the direction of this eigenvector and remain valid. Thus, we see that the subdominant relevant eigenvalue of $H_{1,1}$ is the subdominant eigenvalue of A.

Because of the nested (upper-triangular) structure of $H_{(r)}$, the relevant eigenvalues of $H_{1,1}$ are also relevant eigenvalues of $H_{(r)}$, so that the largest relevant eigenvalue of $H_{(r)}$ is at least as large as the subdominant eigenvalue of A. Next, we show that the dominant relevant eigenvalues of the matrices $H_{i,i}$, $2 \leq i \leq r$, are less than the subdominant eigenvalue of A in magnitude, so that subdominant eigenvalue of $\tilde{H}_{(r)}$ is the subdominant eigenvalue of A.

Since $H_{i,i}$ is the Kronecker product of $D_{(i)}$ and $(A')^{\otimes i}$, its eigenvalues are the Cartesian product of the eigenvalues of $D_{(i)}$ and of $(A')^{\otimes i}$. Note that the eigenvalues of $(A')^{\otimes i}$ are

products of r (not-necessarily-distinct) eigenvalues of A. Thus, the only eigenvalue of $(A')^{\otimes i}$ with magnitude greater than that of the subdominant eigenvalue of A is the unity eigenvalue. Also, it is easy to check that the eigenvalues of $D_{(i)}$ have magnitudes that are strictly less than 1. Thus, the only eigenvalues of $H_{i,i}$ that can have magnitude as large as the subdominant eigenvalue of A are the eigenvalues of $D_{(i)}$. We prove that all such eigenvalues are not relevant to the rth moment recursion. To do so, we show that the left eigenvectors of $\tilde{H}_{(r)}$ corresponding to these eigenvalues are necessarily perpendicular to any valid rth extended state vector. This orthogonality is proved indirectly, as described below. Before describing these details, however, we present an example that shows the relevant and irrelevant eigenvalues of a homogeneous influence model.

	Relevant Eigenvalues	**Irrelevant Eigenvalues**
$H_{1,1}$	1, 0.35, 0.21	0.6
$H_{2,2}$	0.231, 0.231, 0.21, 0.21, 0.0808, 0.0735	0.66, 0.6

Table 7.2: The relevant and irrelevant eigenvalues of $H_{1,1}$ and $H_{2,2}$ are shown. The eigenvalues of the extended second moment recursion matrix $H_{(2)}$ are the union of the eigenvalues of $H_{1,1}$ and $H_{2,2}$. We see that the largest subdominant relevant eigenvalue of $H_{(2)}$ is the largest subdominant relevant eigenvalue of $H_{1,1} = H_{(1)}$, which is also the subdominant eigenvalue of A.

Example 7.11

Again consider the influence model introduced in Example 7.10. The relevant and irrelevant eigenvalues of $H_{1,1}$ and $H_{2,2}$ for this model are shown in Table 7.2. From the table, we see that the subdominant relevant eigenvalue of both the first- and second-moment recursions (e.g., of $\tilde{H}_{(1)} = H_{1,1}$ and $\tilde{H}_{(2)}$ is the subdominant relevant eigenvalue of $H_{1,1}$, or 0.35. This is also the subdominant eigenvalue of A.

Since the subdominant eigenvalues of the first- and second-extended moment recursions are identical, we expect the settling times for these recursions to be comparable. For illustration, we compare the dynamic response of the probability that site 1 is in its first status (Figure 7.6) with dynamic response of the probability that both sites are in their first statuses (Figure 7.7). These figures suggest that the settling times of individual site status probabilities and joint status probabilities are identical, reflecting that the first- and second-moment recursions have the same subdominant eigenvalue.

We return to the details of the proof. To prove that the eigenvectors of $\tilde{H}_{(r)}$ corresponding to eigenvalues that are also eigenvalues of the $D_{(i)}$, $1 \leq i \leq r$ are not relevant, we define a square matrix $\tilde{D}_{(r)}$ that captures all network-level dynamics (i.e., all interactions specified by network matrix-based site selection) of the rth moment recursion. The number of rows

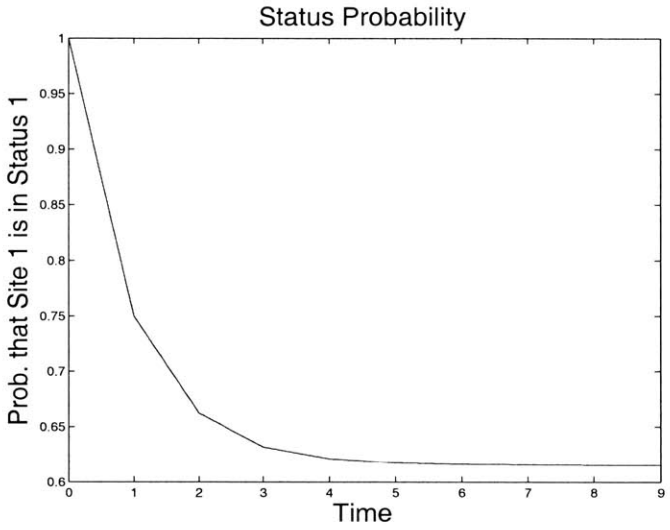


Figure 7.6: The probability that Site 1 is in its first status is shown for 10 time-steps. We assume that both sites are originally in their first statuses.

(and columns) of $\tilde{D}_{(r)}$ is assumed to be the number of distinct groupings of r and fewer sites for the influence model of interest. These rows and columns, which are indexed by groupings, are assumed to be ordered according to increasing grouping size, and in lexicographic order by grouping within a grouping size. The entry with column indexed by $\mathbf{v}$ and row indexed by $\mathbf{w}$ (called column $\mathbf{v}$ and row $\mathbf{w}$ from now on) is the probability that grouping $\mathbf{w}$ is determined by any grouping $\widehat{\mathbf{v}}$ such that $ess(\widehat{\mathbf{v}}) = \mathbf{v}$. Thus, note that the entry at row $\mathbf{w}$ and column $\mathbf{v}$ is

$$d(\mathbf{w}, \mathbf{v}) = \sum_{\widehat{\mathbf{v}}\ s.t.\ ess(\widehat{\mathbf{v}})=\mathbf{v}} \prod_{i=1}^{q} d_{w_i, \widehat{v}_i}, \tag{7.38}$$

where q is the size of the grouping $\mathbf{w}$. From this definition, we see that $\tilde{D}_{(r)}$ is a block-lower-triangular matrix, with diagonal blocks given by the matrices $D_{(1)}, \ldots, D_{(r)}$. We denote the off-diagonal block of $\tilde{D}_{(r)}$ relating all groupings of size i and j by $D_{(i,j)}$. Note that $\tilde{D}_{(r)}$ is a stochastic matrix, since each distinct ith grouping, $1 \leq i \leq r$, is determined with certainty from some other grouping of at most r (in fact, at most i) sites. Thus, each row of $\tilde{D}_{(r)}$ sums to 1. Now consider the eigenvalues of $\tilde{D}_{(r)}$. These eigenvalues are the union of the eigenvalues of $D_{(1)}, \ldots, D_{(r)}$. Thus, one of the eigenvalues of $\tilde{D}_{(r)}$ is 1, and the remaining eigenvalues are strictly less than 1 in magnitude. Since $\tilde{D}_{(r)}$ is a stochastic

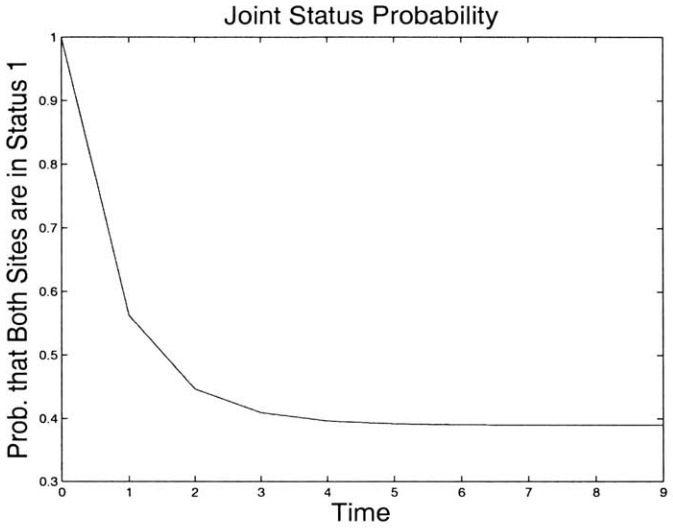


Figure 7.7: The probability that Sites 1 and 2 are both in their first statuses is shown for 10 time-steps. We assume that both sites are originally in their first statuses. This settling time of this joint status probability is similar to the settling time of the individual status probability shown in Figure 7.6, reflecting that the subdominant relevant eigenvalue of the first and second moment recursions are identical.

matrix, its right eigenvector corresponding to the unity eigenvalue is a vector of all ones. Thus, all left eigenvectors of $\tilde{D}_{(r)}$ corresponding to the remaining (non-unity) eigenvalues are orthogonal to the vector of all ones—i.e., they sum to zero. This insight regarding the left eigenvectors of $\tilde{D}_{(r)}$ is used below to prove the eigenvalues of $\tilde{H}_{(r)}$ that are eigenvalues of $D_{(i)}$, $1 \leq i \leq r$, are irrelevant.

Consider the left eigenvector of $\tilde{H}_{(r)}$ corresponding to a particular eigenvalue λ, which is a particular non-unity eigenvalue of $D_{(1)}, \ldots, D_{(r)}$ (and hence also a non-unity eigenvalue of $\tilde{D}_{(r)}$). Say that the left eigenvector of $\tilde{D}_{(r)}$ corresponding to the eigenvalue λ is $\mathbf{c}'$. For convenience, we use the notation $\mathbf{c}' = \begin{bmatrix} \mathbf{c}'_1 & \ldots & \mathbf{c}'_r \end{bmatrix}$ to distinguish the parts of the left eigenvector that are identified with groupings of different sizes. We claim that the vector $\mathbf{c}'_H = \begin{bmatrix} \mathbf{c}'_1 \otimes \mathbf{1}'_m & \mathbf{c}'_2 \otimes \mathbf{1}'_{m^2} & \ldots & \mathbf{c}'_r \otimes \mathbf{1}'_{m^r} \end{bmatrix}$ is a left eigenvector of $\tilde{H}_{(r)}$ with the same corresponding eigenvalue. To see why, consider the product $\mathbf{1}'_{m^i} H_{i,j}(\mathbf{w}, \mathbf{v})$ (where $\mathbf{w}$ is an ith distinct grouping, $\mathbf{v}$ is a jth distinct grouping, and $i \geq j$). For $j = i$, this product is $\mathbf{1}'_{m^i}(\prod_{j=1}^{i} d_{w_j v_j})(A')^{\otimes i} = (\prod_{j=1}^{i} d_{w_j v_j})\mathbf{1}'_{m^i}$. For $j \neq i$, note that $H_{i,j}(\mathbf{w}, \mathbf{v}) = \sum_{\widehat{\mathbf{v}}\, s.t.\, ess(\widehat{\mathbf{v}})=\mathbf{v}} \prod_{i=1}^{q} d_{w_i, \widehat{v}_i} \left[(A')^{\otimes i}\right]_{\widehat{\mathbf{v}}'}$ where $\left[(A')^{\otimes i}\right]_{\widehat{\mathbf{v}}}$ is a $m^i \times m_j$ matrix with columns that are a subset of the columns of $(A')^{\otimes i}$. Thus, we find that $\mathbf{1}'_{m^i} \left[(A')^{\otimes i}\right]_{\widehat{\mathbf{v}}} = \mathbf{1}'_{m^j}$, and so

$\mathbf{1}'_{m^i} H_{i,j}(\mathbf{w}, \mathbf{v}) = (\sum_{\hat{\mathbf{v}}\, s.t.\, ess(\hat{\mathbf{v}})=\mathbf{v}} \prod_{i=1}^{q} d_{w_i,\hat{v}_i})\mathbf{1}'_{m^j}$. Next, consider $(\mathbf{c}'_i \otimes \mathbf{1}'_{m^i})H_{i,j}$. Using the expressions above, we see that the product can be rewritten as $\mathbf{c}'_i D_{(i,j)} \otimes \mathbf{1}'_{m^j}$. Thus,

$$\mathbf{c}'_H \begin{bmatrix} 0 \\ \vdots \\ 0 \\ H_{j,j} \\ \vdots \\ H_{r,j} \end{bmatrix} = \left(\sum_{i=j}^{r} \mathbf{c}'_i D_{(i,j)} \right) \otimes \mathbf{1}'_{m^j} = \lambda \mathbf{c}'_j \otimes \mathbf{1}'_{m^j}, \tag{7.39}$$

since $\mathbf{c}'$ is a left eigenvector of the block-lower-triangular matrix $\tilde{D}_{(r)}$. Assembling Equations 7.39 for $j = 1, \ldots, r$, we find that $\mathbf{c}'_H$ is a left eigenvector of $\tilde{H}_{(r)}$, with eigenvalue λ. In this way, we find that the left eigenvectors of $\tilde{H}_{(r)}$ corresponding to eigenvalues that are also eigenvalues of $D_{(1)}, \ldots, D_{(r)}$ can be written in terms of the left eigenvectors of $\tilde{D}_{(r)}$ in the manner specified above. Finally, consider the product $\mathbf{c}'_H \mathbf{s}_{(r)}$, where $\mathbf{s}_{(r)}$ is any valid rth extended state vector. Since all status and joint status vectors are indicator vectors, $\mathbf{c}'_H \mathbf{s}_{(r)}$ equals $\mathbf{c}'\mathbf{1}$. For any $\mathbf{c}'_H$ corresponding to a non-unity eigenvalue, $\mathbf{c}'_H \mathbf{s}_{(r)} = \mathbf{c}'\mathbf{1} = 0$. Thus, no valid rth extended state vector have a component in the direction of these eigenvectors, and so we have proved that the non-unity eigenvalues of $D_{(1)}, \ldots, D_{(r)}$ are not relevant eigenvalues of the rth moment recursion matrix. Thus, the largest relevant subdominant eigenvalue of the rth moment recursion matrix is the subdominant eigenvalue of A. Q.E.D.

7.4.2 Non-Homgeneous Influence Model: Counterexample to Conjecture

The relevant subdominant eigenvalue conjecture is not generally true for the influence model. For instance, consider a two-site model with the following influence matrix:

$$H_{(1)} = H_{1,1} = H' = \begin{bmatrix} 0.5 \begin{bmatrix} 0.99 & 0.01 \\ 0.01 & 0.99 \end{bmatrix} & 0.5 \begin{bmatrix} 0.01 & 0.99 \\ 0.99 & 0.01 \end{bmatrix} \\ 0.5 \begin{bmatrix} 0.99 & 0.01 \\ 0.01 & 0.99 \end{bmatrix} & 0.5 \begin{bmatrix} 0.01 & 0.99 \\ 0.99 & 0.01 \end{bmatrix} \\ \begin{bmatrix} 0 & 1 \end{bmatrix} & \end{bmatrix} \tag{7.40}$$

We can straightforwardly check (using Matlab) that the subdominant relevant eigenvalue of $H_{(1)}$ is 0. Meanwhile, the subdominant relevant eigenvalue of $H_{(2)}$ is -0.4802. Thus,

the magnitude of the subdominant relevant eigenvalue of $H_{(2)}$ is larger than that of $H_{(1)}$, and so the subdominant relevant eigenvalue conjecture is untrue. We thus expect that the second moment recursion settles to steady-state more slowly than the first moment recursion. All relevant and irrelevant eigenvalues for $H_{(1)}$ and $H_{(2)}$ are listed in Table 7.3

	Relevant Eigenvalues	**Irrelevant Eigenvalues**
$H_{1,1}$	1, 0, 0	0
$H_{2,2}$	−0.4802, 0, 0, 0, 0, 0	0.5, 0

Table 7.3: The relevant and irrelevant eigenvalues of $H_{1,1}$ and $H_{2,2}$ are shown. The eigenvalues of the extended second moment recursion matrix $H_{(2)}$ are the union of the eigenvalues of $H_{1,1}$ and $H_{2,2}$. We see that the largest subdominant relevant eigenvalue of $H_{(2)}$ is larger than the largest subdominant relevant eigenvalue of $H_{1,1} = H_{(1)}$, and hence the relevant subdominant eigenvalue conjecture is false.

It is worthwhile to explore this example a bit further, to understand why the first-moment recursion settles to steady-state more quickly than the second-moment recursion. In fact, we have structured $H_{1,1}$ so that the first moment recursion settles exactly to its steady-state in two time-steps. To see why, consider any valid $E(\mathbf{s}[0])$. From the symmetries of $H_{1,1}$, we see that $E(\mathbf{s}[1]) = H_{1,1}E(\mathbf{s}[0])$ has the form $\begin{bmatrix} \alpha \\ 1-\alpha \\ \alpha \\ 1-\alpha \end{bmatrix}$. Next, we can easily check that $E(\mathbf{s}[2]) = H_{1,1}E(\mathbf{s}[1]) = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}$, and that $E(\mathbf{s}[k]) = E(\mathbf{s}[2])$ for $k > 2$.

Now consider the second-moment recursion. The matrix $H_{2,2}$ specifies aspects of the settling dynamics that are not present in the first-moment recursion. Note that $H_{2,2}$ describes the influence model update in the case when the two sites are determined by different sites—i.e., when both sites are self-determined, or each site is determined by the other. In this case, the next state is related to the current one as follows: if both sites have the same status at the current time, the two sites have different statuses at the next time with high probability, and vice versa. Thus, if the two sites are updated by different sites (i.e., if $H_{2,2}$ describes the dynamics of the update) for several time-steps, the influence model oscillates back and forth between a same-status configuration and a different-status configuration with high probability, and so does not reach steady-state. Since there is a non-zero probability that the sites are consecutively updated by different sites during any finite number

of time-steps, the second moment of the state does not converge in a finite number of time-steps. Figure 7.8 shows the probability that Site 1 is in its first status as a function of time, given that both sites are initially in their first statuses. Meanwhile, Figure 7.9 shows the probability that Sites 1 and 2 are both in their first statuses, given the same initial condition. These figures verify that the first moment of the state settles in 2 time-steps, while the second moment does not.

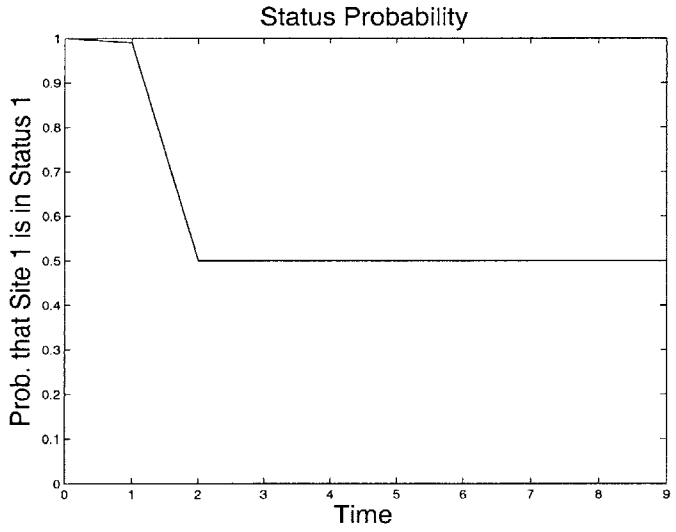


Figure 7.8: The probability that Site 1 is in its first status is shown for 10 time-steps. We assume that both sites are originally in their first statuses. As indicated in the figure, the status probabilities for individual sites reach steady state in two time-steps, reflecting that the subdominant relevant eigenvalue of $\tilde{H}_{(1)}$ is 0.

7.5 Statistics Across Time Steps

The techniques developed in Chapter 4 can be used to find statistics *across* time-steps for the influence model. These statistics specify joint probabilities for the statuses of one or more sites at several time-steps. In turn, these joint probabilities can be used to find conditional probabilities for individual or joint statuses of sites, given the statuses of these and/or other sites at other time-steps. As an example, we have found the conditional status probability response of a single site in an influence model for failures. In particular,

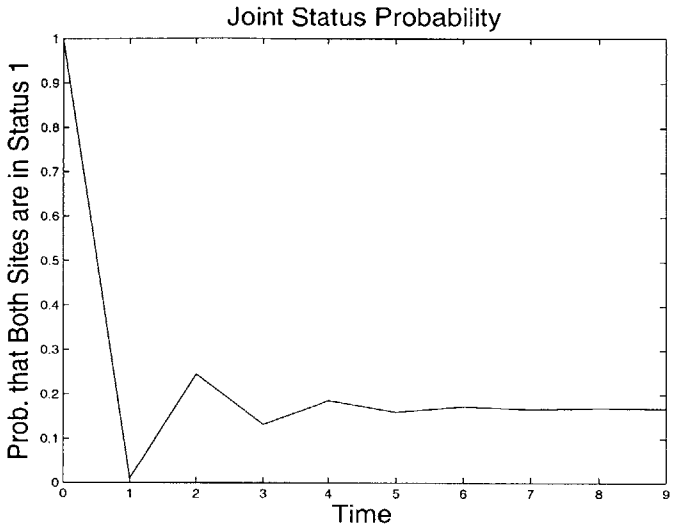


Figure 7.9: The probability that Sites 1 and 2 are both in their first statuses is shown for 10 time-steps. We assume that both sites are originally in their first statuses. This joint status probability does not exactly reach steady-state in a finite number of time-steps, reflecting that the subdominant relevant eigenvalue of $\tilde{H}_{(2)}$ is not zero.

we consider a four-site model, with $D = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}$ and $A = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix}$.

Figure 7.10 shows the probability that site 1 is Normal at each time-step, given that it is initially Normal. The figure is generated assuming that the model is operating at steady-state at the initial time. The figure shows that the status probability returns to its steady-state value more quickly in the four-site model than in a single-site model with the same local status evolution matrix.

7.6 State Estimation in the Influence Model

In this section, we discuss methods for estimating the state vector of an influence model from a sequence of observations. Our study of state estimation in the influence model is motivated by a prevalent interest in estimation in stochastic automata models, including *hidden Markov models* (HMMs) and more general *dynamic Bayesian networks* (DBNs)

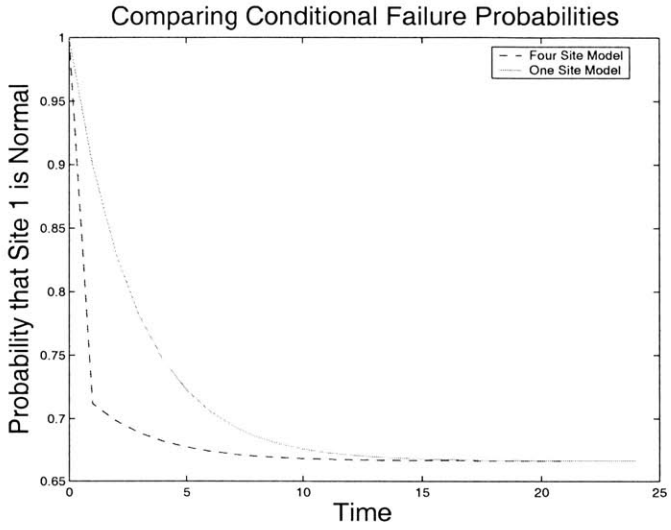


Figure 7.10: The failure tendency of a single site in a four-site failure model is compared with the failure tendency of a one-site failure model with the same status evolution matrix. In particular, we compare the conditional probabilities that the site of interest is Normal at each time-step given that it is initially Normal, for the two models. For the model with four sites, we assume that the model is a priori operating in steady-state at the initial time.

[116, 56]. While the methods for state estimation developed for these models can be applied (directly or indirectly) to the influence model, we are especially interested in adapting those methods, or developing new methods, that exploit the special structure of the influence model to facilitate estimation. In this section, we discuss two methods for Maximum Likelihood (ML) state estimation for the model. The first is a direct approach, based on calculating and maximizing the probabilities for site statuses (or joint statuses) given an observation sequence. The second method uses the LMMSE estimation procedure for MLSS to approximate status probabilities, and then generates estimates from these approximate probabilities.

7.6.1 The Direct Approach to Estimation

ML *filtering* for the influence model—i.e., ML estimation of a site status (or joint status) at a particular time from a sequence of observations up to that time—is considered. In particular, say that we observe the statuses of sites in a grouping $\mathbf{v}$ at times $k = 0, 1, \ldots T$. Our goal is to estimate the status of a site i at time T that is not in the grouping $\mathbf{v}$, given

the observations.

By viewing the dynamic evolution of the influence model as the status evolution of a large *master* Markov chain (which specifies the evolution of the joint status of all sites in the influence model), we can rephrase the state estimation problem in the notation of HMMs. More specifically, consider the joint status for the grouping $\omega \stackrel{\triangle}{=} \{1, \ldots, n\}$ (i.e., the grouping of all sites) at time k. Since this joint status specifies the state vector of the MLSS at time k, the sequence $\{\mathrm{s}_\omega[k]\}$ constitutes a Markov process; in fact, as discussed in our original development of the influence model, $\mathrm{s}_\omega[k]$ is an indicator status vector for a Markov chain with $\prod_{i=1}^{n} m_i$ statuses. Using the notation of [9], we denote the transition matrix for this master Markov chain as G. For the purpose of estimation, we view this finite-state Markov chain as the underlying Markov chain of a HMM (see [116] for details on HMM).

In general, observations of HMM are chosen stochastically from a finite symbol set at each time, with the probabilities of each output symbol specified according to the status of the underlying Markov chain at that time. In our case, we choose the output symbol to be the joint status (equivalently, joint status vector) of the sites in the observed grouping $\mathbf{v}$. Thus, note that the observation at time k is specified deterministically in terms of the underlying state (equivalently, the joint status vector of all sites) at time k, and so these observations are valid as HMM observations.

Based on this formulation, we can use the *forward algorithm* of the HMM to compute probabilities for the joint statuses of all sites at time T, given observations of the sites in $\mathbf{v}$ at times $0, 1, \ldots, T$. By marginalizing these joint status probabilities, we can compute status probabilities for the site i, given the observation sequence. The ML estimate of the status of site i is the one corresponding to the largest of these probabilities.

Example 7.12

We have applied the estimation procedure to a three-site example, with $D = \begin{bmatrix} \frac{1}{2} & \frac{1}{3} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix}$ *and* $A = \begin{bmatrix} 0.99 & 0.01 \\ 0.05 & 0.95 \end{bmatrix}$. *Figure 7.11 shows the conditional probability that Site* 2 *is failed at each time-step, given observations of Site* 1 *up to that time-step. For this particular simulation, the ML estimate for Site* 2 *matches the actual status of the site* 188 *out of* 200 *times.*

It is important to note that the recursive forward algorithm requires us to compute, and store, probabilities for joint statuses of all n sites, so that the computational complexity of

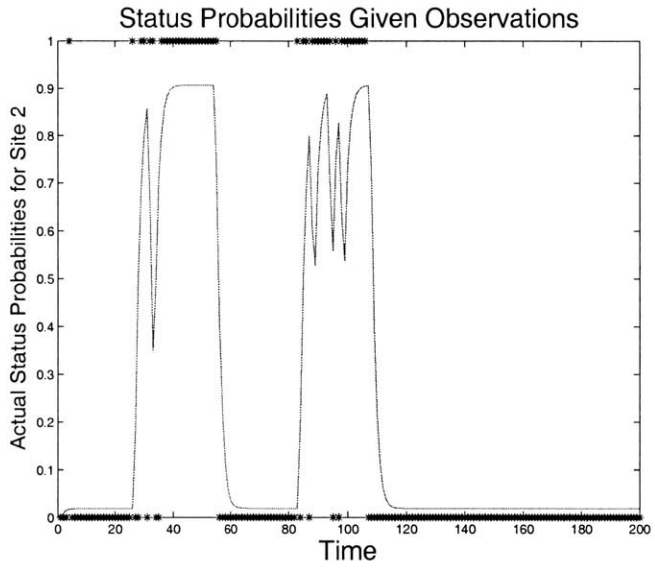


Figure 7.11: A three-site failure model is considered. The conditional probability that site 2 is failed at each time-step given a sequence of observations of site 1 up to that time-step is shown. The conditional probabilities are compared with the actual status of site 2 (with a $*$ at the top of the graph corresponding to a failure, and a $*$ at the bottom corresponding to a Normal status). For this particular state and observation sequence, the ML estimate for the status of site 2 matches the actual site status at 188 out of 200 times.

the algorithm grows exponentially in the number of sites. Because of the special structure of the observations in our example, it turns out that the forward algorithm can be rewritten in terms of the (conditional) joint status probabilities of only the unobserved sites (those not in $\mathbf{v}$), in which case the complexity is exponential in the number of unobserved sites rather than the total number of sites. A recursive estimation algorithm that incorporates this simplification is discussed in the Master's thesis [59].

The large computational complexity of the forward algorithm is a significant hurdle to its implementation, and is worth further study. An explanation for why an exponential growth in the computational complexity with the number of sites cannot usually be avoided is given in [59]. Gomez-Uribe's thesis also explores how the graph structure of the influence model can be exploited to simplify the computation in some cases. These simplifications may often be quite valuable in making influence model state estimation feasible.

While we have discussed an ML filtering algorithm for the influence model here, smoothing and state sequence estimation algorithms can be developed in similar fashion, as discussed in [59].

7.6.2 Indirect Approach to Estimation, Based on LMMSE Filtering

The LMMSE estimation techniques developed for MLSS can be used in the context of the influence model to approximate conditional probabilities for site statuses, given sequences of observations. The results can then be used to approximately determine the ML estimate for hidden site statuses. The advantage of such an indirect approach to estimation is that the computational complexity of the algorithm is polynomial, rather than exponential, with respect to the number of sites in the model.

First, let's verify that our filtering set-up can be reformulated in terms of an MLSS with observations, and interpret the result of the LMMSE estimation techniques for this formulation. As we have discussed previously, the state sequence for the influence model constitutes an MLSS. Note that the observations at each time k—namely, the statuses (equivalently, status vectors) of a subset of the sites in the model—can be obtained by choosing some elements of the state vector, and so are a linear function of the state. Thus, the observations at each time are moment-linear with respect to the corresponding state, and the observed influence model is an LMMSE with observations.

Suppose the LMMSE estimate for the influence model state vector is determined based on the formulation described in the previous paragraph. To interpret this LMMSE estimate, first consider the (not-necessarily-linear) minimum mean square error (MMSE) estimate for the influence model state vector, given the observation sequence. It is well known (see, e.g., [21]) that the MMSE estimate for the state vector is the expected value for the state vector, given the observations. Thus, since the state vector of the influence model comprises a 0–1 indicator vector, the MMSE estimate specifies the status probabilities for each site, given the vector of observations. The LMMSE estimate may be considered an approximation to the MMSE estimate, as the best mean-square error estimate that is linear with respect to the observations. Thus, the LMMSE estimator allows us to approximate the probabilities of site statuses, given the sequence of observations. These approximate probabilities can then be used to approximately generate the ML estimate.

Example 7.13
We again consider the failure model introduced in Example 7.12. Figure 7.12 shows LMMSE-based approximations for the probability that site 1 *is failed, given observations of the status of site* 2. *Interestingly, the estimates for this failure probability are very close to the actual values of the failure probability, which were shown in Figure 7.11. The approximate ML estimates generated from these approximate status probabilities also correctly predict the hidden status correctly at* 188 *out of* 200 *times.*

The LMMSE estimator-based approximation may be a compelling approach for ML estimation in large influence models, for which direct computation of status probabilities is infeasible. More research is needed to better understand whether or not LMMSE estimator-based techniques can usefully predict actual status probabilities for a broad range of model parameters.

7.7 Models with Random Parameters

An influence model can be modified so that the parameters of the model (the network and status-evolution matrices) are chosen stochastically at each time-step. If these model parameters are chosen independently of the previous states of the model, and of previous parameter choices, then the state update can be formulated as an MLSN, as follows: the conditional rth vector moment for the next state, given the current state, can be rewritten by conditioning further on the model parameters; doing so, and using the independence

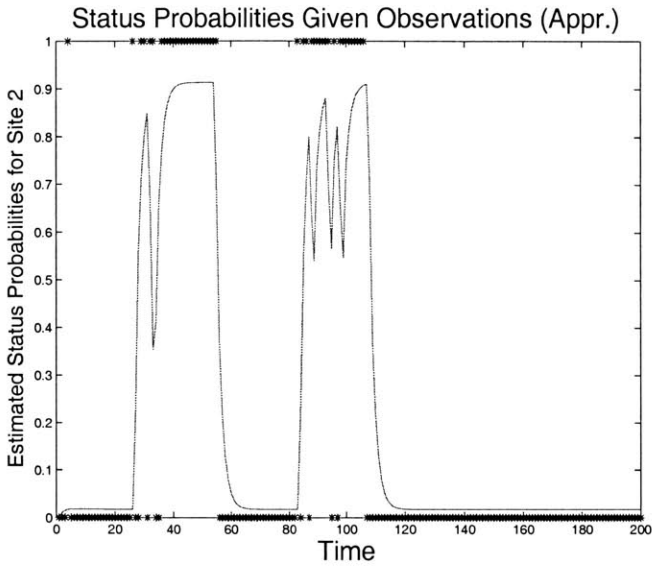


Figure 7.12: We approximate the probability that site 1 *is failed at each of* 200 *time-steps, given observations of the status of site* 2 *up to that time-step. The LMMSE estimator is to generate this approximation. The actual status of site* 1 *at each time-step is also shown.*

assumption, this conditional expectation can be written as linear function of the current rth extended state vector. Alternately, if the parameter choices form are Markov and independent of the past history of the system, it turns out that the state update can still be formulated as an MLSN, albeit one with a larger state space.

In the following example, we consider a specific influence-type model with randomly chosen parameters. In particular, consider a system with n sites, each of which can assume m statuses. At a time-step k, the state of the system (i.e., the statuses of all sites in the network) is updated as follows:

- One site is randomly chosen for updating, independently of all past history of the system. The probability that site i is chosen is p_i.
- Given that site i is chosen for updating, the time-$(k+1)$ status of site i is determined as in an influence model. That is, site i chooses a determining site j according to the probabilities d_{ij}. The time-$(k+1)$ status of site i is then chosen based on the probabilities specified by the row in the local status evolution matrix A corresponding to the time-k status of site j.

- All sites except site i maintain their time-k statuses at time $k+1$.

This example can be viewed as an influence model with parameters that are chosen randomly at each time-step. Given that a site i is updated at a particular time, the update rule for the example is identical to the update rule for an influence model: all sites except i in this influence model choose themselves as their determining site and copy their previous status, while site i is updated as a typical site in an influence model. The influence matrix for this influence model, which we denote $H(i)$, has the form

$$H(i) = \left[\begin{array}{cc|c|cc} I & 0 & d_{i1}A & 0 & 0 \\ 0 & \ddots & \vdots & 0 & 0 \\ 0 & 0 & d_{ii}A & 0 & 0 \\ 0 & 0 & \vdots & I & 0 \\ 0 & 0 & d_{in}A & 0 & \ddots \end{array}\right]. \tag{7.41}$$

In Equation 7.41, the part of the matrix $H(i)$ that relates the next status of the site i to the current status of a site j has the form $d_{ij}A$ for each j. Meanwhile, the submatrices of $H(i)$ that relate the next-statuses of any site other than i to their own previous statuses are identity matrices, and those that relate the next statuses of sites other than i to other sites' previous statuses are zero matrices.

Let's show that the example is indeed an MLSN. Since the site to be updated at each time-step is chosen independently of the history of the system, and the statuses of the sites are updated as in an influence model, given this choice, the state of the system constitutes a Markov process. Thus, it only remains to check the moment-linearity conditions to show that the system is an MLSN.

Consider the first conditional moment for the next state, given the current state. Conditioning on the site X that is chosen for updating, we find that this expectation is given by

$$\begin{aligned} E(\mathbf{s}[k+1] \,|\, \mathbf{s}[k]) &= \sum_{i=1}^{n} E(\mathbf{s}[k+1] \,|\, \mathbf{s}[k], X=i) P(X=i \,|\, \mathbf{s}[k]) \\ &= \sum_{i=1}^{n} P(X=i) H'(i) \mathbf{s}[k] \\ &= \left(\sum_{i=1}^{n} p_i H'(i) \right) \mathbf{s}[k]. \end{aligned} \tag{7.42}$$

Thus, the first conditional moment for the next state given the current state is a linear function of the current state $\mathbf{s}[k]$, and so this system satisfies a first-moment linearity condition.

Similarly, the model can be shown to satisfy higher-moment linearity conditions. Given that site i is updated at time k, we know that the next-status of each site is determined according to the update law of a basic influence model. Call the rth extended recursion matrix for this eqivalent influence model $H_r(i)$. In this notation, the conditional expectation for the r^{th} *extended state vector* $\mathbf{s}_{(r)}[k]$ for the model of interest (which is assumed to be defined in the same way as for the influence model) is given by

$$E(\mathbf{s}_{(r)}[k+1] \,|\, \mathbf{s}[k]) = \left(\sum_{i=1}^{n} H_r(i) p_i \right) \mathbf{s}_{(r)}[k]. \tag{7.43}$$

Example 7.14
The dynamics of an influence model in which all sites are updated simultaneously (called a basic influence model in this section, for clarity) differ in some interesting ways from the dynamics of a model in which sites are chosen randomly for updating. In some cases, we can relate the dynamics of an influence model and a variant in which sites are chosen randomly for updating. For instance, let's consider the following two models:

- *A (homogeneous) basic influence model with network matrix D, and local status-evolution matrix A.*
- *A model in which a single site is chosen for updating at each time, with the probability that each site is chosen given by $p = \frac{1}{n}$. Once a site has been chosen, it is updated in the same manner as the homogeneous influence model above (i.e., it choose a determining site according to the appropriate row of D, and the current status of this determining site specifies the updated site's next-status according to the appropriate row of A).*

Let's call the first-moment recursion matrix for the basic influence model $H_{1,1}$. Then, applying Equation 7.42, we find that the recursion matrix for the second model is $\widehat{H}_{1,1} = pH_{1,1} + (1-p)I$. We can straightforwardly check that the steady-state status probabilities for individual sites are identical for the two models. We can also check that the eigenvalues of the two recursions are related as follows: if λ is an eigenvalue of $H_{1,1}$, then $p\lambda + (1-p)$ is an eigenvalue of $\widehat{H}_{1,1}$. Note that, if the number of sites n is large, all the eigenvalues of $\widehat{H}_{1,1}$ will be near 1; this reflects that dynamics become slow if the model has many sites, because only one site is updated at each time. Joint status probabilities for the basic influence model can be also be related to those of the model in which sites

are chosen randomly for updating. For the sake of brevity, we do not pursue these connections any further.

For example, consider two n site homogeneous models, with the same network dynamics and status evolution matrix. in which the sites that are chosen for updating are updated in the same fashion. For the second model, assume that each site is chosen for updating with equal probability $p = \frac{1}{n}$. In this case, the first moment recursions for the two examples are simply related. In particular, let's say that the basic influence model has first recursion matrix (which is the transpose of the influence matrix) given by $H_{1,1} = D \otimes A'$. Then, applying Equation 7.42, we find that the recursion matrix for the second model is $\widehat{H}_{1,1} = pH_{1,1} + (1-p)I$. We can straightforwardly check that the steady-state status probabilities for individual sites are identical for the two models. We can also check that the eigenvalues of the two recursions are related as follows: if λ is an eigenvalue of $H_{1,1}$, then $p\lambda + (1-p)$ is an eigenvalue of $\widehat{H}_{1,1}$. In similar fashion, joint status probabilities for the basic influence model can be related with those of the model in which sites are chosen randomly for updating. For the sake of brevity, we do not pursue these connections any further.

The reader may be wondering about our motivation for considering models with randomly chosen parameters. Our broad aim is to expose more general types of interactions that can be represented using influence-type models. More specifically, however, stochastic automata models in which single sites or groups of sites are chosen for updating at each time are used in several contexts—including in Markov-Chain Monte Carlo (MCMC) and stochastic optimization algorithms [110], and in modeling DNA sequence evolution [8]. While the specifics of our models differ somewhat from these models, it may be useful in our context to also allow for such site selection mechanisms. We also believe that influence models with stochastic parameters may be valuable in representing systems with random component failures or changes.

7.7.1 Models with More General Network-Level Dynamics

In the basic influence model, we assume that each site chooses a determining site independently of the other sites (and of the past state history of the model) at each time-step. A model in which concurrent determining-site choices are correlated (i.e., a model in which the determining site choice of one site affects the determining probabilities of other sites) can also be formulated as an MLSN. In fact, such a model can be viewed as a special case of an influence model in which the network parameters are chosen randomly.

Specifically, consider a model with n sites, in which each site has a single determining site at each time-step (like the influence model). Define $\delta[k]$ to be an n-component vector that lists the possible determining sites for each site at time k. There are at most n^n possible lists $\delta[k]$. For an influence model, each element in the list (i.e., the determining site for each site) is chosen independently at time k. Here, we more generally assume that each possible list $\delta[k]$ of determining sites is chosen with some arbitrary, but time-invariant, probability at each time-step (but still independently of the choices at other time-steps). We denote the probability that the list $\delta[k]$ is $\widehat{\delta}$ by $d(\widehat{\delta})$. Once the determining sites at a time k have been chosen, we assume that the time-$(k+1)$ statuses of sites are found in the same way as for an influence model: each site independently generates its next status from the current status of its determining site according to a vector of probabilities specified by the appropriate row of a local status evolution matrix.

To see why this model can be viewed as an influence model in which parameters are randomly chosen at each time-step, consider generation of the time-$(k+1)$ state from the time-k state *and* the list of determining sites $\delta[k]$. This update is the same as the update for a basic influence model, in which each site is updated with probability 1 by the determining site listed in $\delta[k]$, according to a particular local status evolution matrix. Thus, the update at each time-step in the generalized model can be viewed as follows: first, the list $\delta[k]$ is chosen probabilistically, independently of all past dynamics of the system; each possible $\delta[k]$ then specifies the parameters of an influence model, which is used to determine the next-status of each site.

The rationale given in the previous paragraph touches on an interesting feature of influence-type models. Given the determining sites of some or all the sites in the model at each time-step, the update of the model can still be viewed as the updating procedure for a particular influence model. Thus, even given additional information about the determining site choices at each time-step, the model remains tractable as a time-varying influence model.

We also note an interesting property concerning settling times of some influence-type models with correlated network interactions. In particular, consider *homogeneous* models of this sort—i.e., models in which the local status evolution matrices are identical throughout the system. With a bit of algebra, we can show that the extended recursion matrices for these models can be written in similar form to the extended recursion matrices for the homogeneous influence model. Using this form for the extended moment recursions, we can show that the largest relevant subdominant eigenvalue associated with the rth moment

recursion is the same—specifically, the largest subdominant eigenvalue of the local status evolution matrix A—for all r. Thus, the subdominant eigenvalue conjecture, which characterizes aspects of the settling behavior, holds for these homogeneous models.

Above, we have viewed general network-level dynamics as allowing for correlations among determining-site selections. In some models, such correlations can also be viewed as special structures enforced on the determining site selection process. For instance, one can imagine a hierarchical model, in which groups of sites select other groups for updating, and individual sites subsequently select individual determining sites within these groups. For a hierarchical model, recursions for status and joint status probabilities can be specified at several levels of aggregation. We leave the further study of hierarchical or other influence-type models with generalized network dynamics for future work.

It is also worthwhile to mention that variations and generalizations of the local status evolution rules that maintain the influence model structure can be developed. This may be a valuable direction for future work.

7.8 An Influence Model Generalization for the HMM

Hidden Markov Models (HMMs) have been usefully applied in several research disciplines, including speech processing and biological sequence classification (e.g., [116, 16]). In this section, we show that an HMM can in fact be represented using an influence model with two sites, and then discuss how to generalize the HMM from this influence model representation[6]. We are motivated to consider this connection between the influence model and Hidden Markov Models, because we believe that our analyses of the influence model can possibly provide some new insight regarding the dynamics of HMMs. We also believe that the influence model generalizations may prove valuable for some applications of HMMs.

[6]Incidentally, we earlier showed (in the context of ML estimation for the influence model) that an influence model with site observations can be viewed as an HMM with large state-space, though with the special structure of the influence model obscured. Thus, HMMs can be represented as influence models, and vice versa.

7.8.1 Reformulation of the Standard HMM as an Influence Model

We consider an HMM with m_h hidden, or underlying, Markov statuses and m_o observations. We assume that the underlying Markov chain has transition matrix $A_{h,h}$. In our HMM, given that the underlying Markov chain is in state i at time k, the output symbol at time $k+1$ is determined[7] according to the probabilities contained in row i of the $m_h \times m_o$ row-stochastic matrix $A_{h,o}$. We call $A_{h,h}$ the *state update matrix* and $A_{h,o}$ the *output generation matrix.*

This HMM can be reformulated as an influence model with two sites. The length-m_h status vector $\mathbf{s}_h[k]$ of the first site h indicates the status of the underlying Markov chain in the HMM. Meanwhile, the length-m_o status vector $\mathbf{s}_o[k]$ of the second site o, indicates the output symbol of the underlying HMM. The two sites are updated as follows:

- Site h chooses itself as its determining site, with probability $d_{hh} = 1$. The pmf for the next-status of site h is given by $\mathbf{s}'_h[k]A_{h,h}$.
- Site o chooses site h as its determining site, with probability $d_{oh} = 1$. The next-status of site o is determined according to the pmf $\mathbf{s}'_h[k]A_{h,o}$.

Since the statuses of sites h and o are updated in exactly the same way as the underlying status and output of the HMM, respectively, this influence model is equivalent to the HMM.

Note that the network and influence matrices for the two-site influence model are as follows:

- The network matrix is

$$D = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}. \tag{7.44}$$

[7] In the literature on HMMs, the generated output symbol is typically associated with the time-step k rather than the time-step $k+1$. We choose to associate the output with the next time-step because the reformulation of the model as an influence model is a little clearer. Though we change the notation, we still pursue the analyses that are of interest for the standard HMM; available output measurements in these analyses are renumbered to match with the new notation.

- The influence matrix is

$$H = \begin{bmatrix} A_{h,h} & A_{h,o} \\ 0 & 0 \end{bmatrix}. \tag{7.45}$$

The first-moment recursion for the influence model allows us to determine the probability of each possible hidden state and output symbol in the HMM at each time-step in terms of the initial pmfs for the hidden state and output. Similarly, the second-moment recursion allows computation of the joint probability of the time-k hidden state and output symbol.

Example 7.15
Consider an HMM consisting of an underlying Markov chain with two statuses, and three possible output symbols. For this example, the state update matrix is given by

$$A_{h,h} = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix}. \tag{7.46}$$

Also, the output generation matrix is

$$A_{h,o} = \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.1 & 0.2 & 0.7 \end{bmatrix}. \tag{7.47}$$

In the influence model representation for this example, the hidden site can take on two statuses, while the output site can take on three statuses. The influence matrix for this representation is

$$H = \begin{bmatrix} \begin{bmatrix} 0.8 & 0.2 \\ 0.1 & 0.9 \end{bmatrix} & \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.1 & 0.2 & 0.7 \end{bmatrix} \\ 0 & 0 \end{bmatrix} \tag{7.48}$$

7.8.2 A Generalization of the HMM

We generalize the HMM by allowing the next hidden state and next output to each depend on both the current hidden state and output. In particular, our generalization is a two-site influence model with an arbitrary network matrix D, rather than the particular network matrix given in Equation 7.44. In this generalization, the hidden site (which can still take on m_h statuses) and output site (which can take on m_o statuses) of the influence model are updated at each time-step as follows:

- The hidden site picks itself as its determining site with probability d_{hh}, in which case the next-status is determined using the local status evolution matrix $A_{h,h}$. Alternately, the hidden site chooses to be updated from the output site with probability $d_{ho} = 1 - d_{hh}$, in which case the next-status is determined using the $m_o \times m_h$ row-stochastic matrix $A_{o,h}$.

- The output site can be updated by the hidden site with probability d_{oh}, in which case its next-status is determined using the local status evolution matrix $A_{h,o}$. Alternately, the output site can be updated by itself with probability $d_{oo} = 1 - d_{oh}$. In this case, the next-status is determined using the $m_o \times m_o$ stochastic matrix $A_{o,o}$.

The influence matrix for this generalized HMM is given by

$$H = \begin{bmatrix} d_{hh}A_{h,h} & d_{oh}A_{h,o} \\ d_{ho}A_{o,h} & d_{oo}A_{o,o} \end{bmatrix}. \tag{7.49}$$

Note that the standard HMM is recovered from this generalization by setting d_{ho} and d_{oo} to zero.

Example 7.16

We extend Example 7.15 to give some insight into the possible value of a generalization of this sort. In particular, assume that this example HMM is modified as follows:

- *In the original HMM, the hidden site always chooses itself as its determining site, and evolves according to a certain stochastic matrix $A_{h,h}$ (given in Equation 7.46). In the new example, we assume that the hidden site is updated from its own previous status with probability* 0.9*, but is "corrupted" by the output site with probability* 0.1*. When the hidden site is corrupted, it assumes the first status at the next time-step if either the first or second output symbol is displayed at the current time-step, and assumes the second status if the third output symbol is displayed.*

- *In the original HMM, the output site is always updated by the hidden site. In the new example, we assume that the output site only "polls" the hidden site occasionally—say independently with probability 0.4 at each time step—and otherwise maintains its previous status.*

This new example is a two-site influence model that generalizes the HMM. The influence matrix for this example is

$$H = \begin{bmatrix} 0.9 \times \begin{bmatrix} 0.8 & 0.2 \\ 0.1 & 0.9 \end{bmatrix} & 0.4 \times \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.1 & 0.2 & 0.7 \end{bmatrix} \\ 0.1 \times \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} & 0.6 \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{bmatrix} \tag{7.50}$$

7.9 Some Possible Applications for the Influence Model

We believe that the influence model is potentially applicable in several contexts. While we have not pursued these applications in sufficient detail to warrant their detailed presentation, we briefly list these applications as directions for future study.

- Influence model analogs for certain RC and RLC circuits can be developed. These analogs can potentially be used to efficiently partition and aggregate these circuits.
- The influence model can potentially be used to represent data sources. Binary Markov chains are commonly used to a models for data; the influence model can display richer temporal dynamics than a single Markov chain, and hence may be valuable for representing data sources.
- As we discussed in the introduction of the thesis, the influence model can potentially serve as a crude model for the evolution of DNA sequences.
- The influence model may be useful for representing interactions that occur in cell signaling networks (see, e.g., [74] for analysis of the dynamics of a particular cell signaling network). In cell signaling networks, local interactions between particle types lead to complex global network dynamics; the influence model can perhaps capture these local interactions, and hence facilitate analysis of global dynamics.

7.10 Beyond MLSN Analysis

We mention several further studies of the influence model that are tangential to our MLSS-based development, but may be interesting to the reader.

- In [121], we consider the effect of a network's structure on its dynamic evolution in the context of an influence model for failures. Specifically, tree graphs formed by connecting randomly-located vertices in the plane using various connection rules are used as network graphs for an influence model. We initiate *failure events* in the influence models for failure by setting one site in the model to the Failed status, and then track the dynamics of failure events. Simulations and analysis show that characteristics of these failure events are indeed related to the structure of the underlying network graph. For example, the integrated size of a failure event (the total number of sites failed over the duration of a failure event) has a power law distribution when the degree of a randomly chosen vertex in the network has a power law distribution. However, we also find that the role of the network graph in the dynamics of the failure graph is modulated by the characteristics of the local interactions in the network (in this case, the repair capabilities of interacting sites in the model).

- In [122], we study resource allocation in networks in the context of an influence model for failures. A resource allocation problem in which *repair resources* are used in the network to mitigate failure events is considered. Assuming a linear resource cost and linear expected failure cost, we are able to characterize failure occurrences and costs incurred in the network for several different resource allocation methods (such as optimal vs. heuristic resource allocations, static vs. adaptive allocations, and spatially homogeneous vs. inhomogeneous allocations). We find that the methods are similar in that each attempts to balance resource and failure costs. However, the methods differ in their ability to achieve this balance. A comparison between a dynamic, heuristic resource allocation and a dynamic, optimal resource allocation is particularly interesting. As would be expected, the optimal method is more efficient in balancing costs and so achieves a lower average cost. Suprisingly, however, large-cost events occur more frequently in the optimal design than the heuristic one (because the optimal design does not allocate resources to prevent large but rare events, which do not increase the total cost significantly). Also, the optimal design can sometimes be more highly sensitive to network parameter errors than the heuristic allocation method. This comparison highlights, albeit in a very simple setting, some of the possible advantages and drawbacks of optimal resource allocation schemes in networks.

- We have developed approximate analysis methods for some useful global network characteristics of an influence model. In particular, probability distributions for the number of sites in a certain status at a certain time may be important measures of a network's behavior, but these distributions are difficult to obtain exactly (the computation required to find the distributions is exponential in the number of sites in the model). We have constructed computationally feasible mean-field approximations for these distributions. Simulations and analysis suggest that these approximations are accurate for a fairly large set of influence models. Further, by subdividing the graph into multiple regions and considering the number of sites in a certain status in each region, better approximations for these distributions can be constructed, albeit at higher computational cost. Also of interest, in the limit when certain parameters become small (e.g., failure and repair probabilities in the failure model), the exact distribution for the number of failures can be obtained. Finally, we note that the moments of the distribution of the number of failures can be found indirectly from the moment recursions.

Chapter 8

An Aggregate Model for the Dynamics of Air Traffic Systems

In this chapter, we introduce an MLSN flow model for aircraft counts in regions in an Air Traffic System (ATS). Through the development, we intend to show one example of how a tractable LSN model can be extracted from a more detailed probabilistic description of an *actual* system—in this case, an ATS. In developing a model for a specific application, we will also touch on some issues that we have not yet considered, such as parameter estimation in MLSN models (based on data from the U.S. ATS in this example). The MLSN analyses will then be used to study some questions of interest concerning dynamics and control of an ATS, and also to verify (and identify shortcomings of) the modeling approach.

Unlike the previous chapters, our development here is driven by the application of interest, rather than the general MLSN development. In particular, we begin by giving some general motivations for our development. We then introduce a detailed stochastic description of an ATS based on Poisson processes. Using this detailed description, we construct a stochastic dynamic model for aircraft counts in regions of an airspace. As an example, the developed model is used to represent Center counts in the United States ATS. We also discuss parameter determination in the model, present some analyses of the model, and evaluate our methodology. Two extensions of the basic model —a hierarchical model that represents aircraft counts in regions of various sizes at multiple time scales, and a model that incorporates stochastic disturbances such as thunderstorms —are described. Also, the MLSN control techniques are applied to develop some basic insights about Traffic Flow Management (TFM) at boundaries between regions in an ATS. Finally, we briefly discuss the model in the more general context of MLSN and flow models.

8.1 Introduction and Motivation

At any time, thousands of aircraft are in flight across the world. For many reasons, including uncertain take-off times and unpredictable weather, the locations and behavior of these aircraft at a given instant in time cannot be exactly predicted in advance. Because of this intrinsic uncertainty in any Air Traffic System (ATS), we believe that a stochastic approach to modeling an ATS is valuable.

In addition to uncertainty, a second hurdle in describing and understanding an ATS is its complexity. In the U.S. ATS, for example, as many as 5000 aircraft may be in the air at once, flying along different routes among several hundred airports. Furthermore, the dynamics of each aircraft may be affected by numerous events, including control directives and weather. Because of the complexity of such an ATS, it is sometimes impractical— and often not useful— to track and predict the location of each aircraft in making global decisions about the management of the system. An aggregate description of the ATS may be more tractable and effective in this situation.

Based on these general motivations, we develop an aggregate dynamic stochastic model for an ATS, in which the numbers of aircraft in regions of the airspace are tracked at discrete time-steps. These aircraft counts change with time in the model because of stochastic flows—in particular, the aircraft in each region move to contiguous regions or leave the system with some probability during each time-step. As an example, we model the numbers of aircraft in each Center in the U.S. ATS at discrete time-steps of one and ten minutes.

Some aspects of the uncertainty in ATS's have been studied in the literature. For example, the distributions of departure, enroute, and arrival delays of aircraft have been characterized [6]. Also, queueing models for the arrival of aircraft at airports have been developed [135] [29]. In particular, the article [135] assumes a Poisson process description for aircraft arriving at an airport, and computes the average delay incurred due to the constraints on the landing aircraft. In [29], a more accurate description of the process of aircraft arriving at an airport is considered, and is used to estimate landing delays. The effects of uncertainties in weather prediction on air traffic flow have also been considered [50]. Yet another area in which uncertainty has been considered is in the modeling of airport surface traffic [7]. One aspect of the airport surface traffic, the departure operation of an airport, has been characterized using a queueing model [115]. A queueing model has also been used to study delay cost optimization at hub airports [23]. Recently, a deterministic aggregate model for an ATS has been developed and analyzed in [102]. Detailed deterministic mod-

els for an ATS, which track the location of each aircraft, have been used to study optimal methods for Traffic Flow Management (TFM) [22].

As far as the authors know, a *stochastic* dynamic model for the global behavior of the ATS has not previously been presented in the literature. As in [102], our model dynamically tracks the numbers of aircraft in regions of the airspace. In contrast to [102], however, we use stochastic models for both the flow of aircraft into the airspace, and for the movement of these aircraft between regions. In addition to the general motivation of exploring stochastic and aggregate descriptions for the ATS, we believe that our model has some specific practical uses. Potential applications of the model include the following:

- Our model may allow quantification of uncertainties in predicted aircraft counts (such as Center or Sector counts in the U.S. ATS). The regional aircraft counts predicted by the model will differ from actual counts, both because of intrinsic uncertainties in the ATS and because of the aggregation in the modeling process. The structure of our model allows us to explicitly compute the degree of uncertainty in the predictions, and to evolve these computations dynamically in time.
- Our model may allow rapid calculation of the behavior of an ATS under many different circumstances (e.g., different initial conditions or different flow patterns due to weather events). We expect that our model can be used to rapidly identify scenarios that may lead to violations in Sector or Center capacities.
- The aggregate model may provide a good framework for studying TFM. The effects over time of a TFM restriction can be more easily determined with an aggregate dynamic model than with a detailed model of an ATS. We are particularly interested in developing algorithms for placing Miles-in-Trail (MIT) and Minutes-in-Trail (MINIT) restrictions in the context of the aggregate model.

As the model is introduced and analyzed in this article, we will occasionally discuss the possible value of the model in achieving these three aims.

8.2 An Aggregate Stochastic Model for an ATS

In this section, we formulate a stochastic model for aircraft counts in regions of an ATS. First, a Poisson process description of the flow of aircraft in an ATS is presented. We then

discuss some difficulties in directly using this detailed description of air traffic flow to analyze the dynamics of the ATS. Motivated by these difficulties, we construct the aggregate stochastic model and consider its relationship with the detailed description.

8.2.1 Poisson Process Description of an ATS

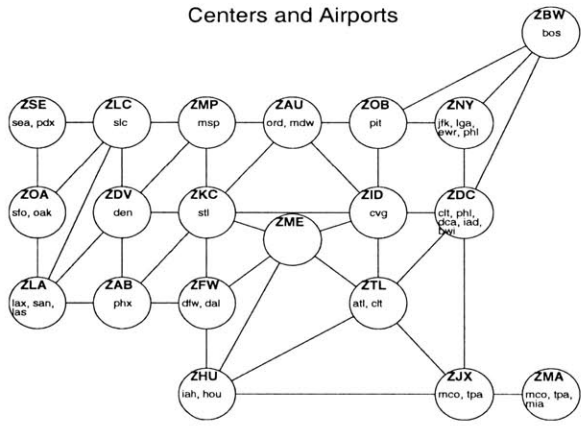


Figure 8.1: A network representation of the 20 Centers in the United States. Two Centers are connected in this plot if they are contiguous in the U.S. ATS. Also, the major airports in each Center are listed in smaller font. If an airport is close to the boundary of two Centers, it is listed in both Centers.

The airspace of an ATS is typically subdivided into regions, to facilitate the control and management of aircraft in the airspace. For example, the U.S. airspace is composed of 20 $Centers$, as shown in Figure 8.1. Aircraft depart from airports distributed among the various Centers, follow routes through the airspace, and arrive at other airports.

Consider an ATS with n Centers. (Although Centers in the U.S. ATS are specific regions of the U.S. airspace, we use the word Center more generally to denote a region of interest in the airspace.) In our stochastic description for the ATS, we assume that the departures of aircraft from each airport are governed by an independent Poisson process with a (possibly) time-varying rate. Later in the article, we will verify from historical data that such a Poisson process description for departures is reasonable.

We also assume that the routes taken by, and the destinations of, the departing aircraft are stochastically independent. This assumption means that the route and destination of a par-

ticular departing aircraft does not provide information about the routes and destinations of other departing aircraft. This assumption constitutes an oversimplified representation for the actual flows in the ATS: for example, we might expect that the departing aircraft that are destined for a particular airport roughly follow a periodic schedule, so that the departure of one such aircraft does provide some information about routes and destinations of other aircraft. However, we use the assumption because it allows us to tractably represent flows in the ATS, without worrying about the particular details of departure schedules for aircraft. We also assume that the cruising speed of each aircraft is constant, and that the cruising speed of different departing aircraft are independent.

Now consider the flows of aircraft among Centers in this Poisson process description of an ATS. For notational convenience in this analysis, we introduce a fictitious Center labeled "0". Aircraft which depart from airports in a Center are said to flow from Center 0 into that Center, while aircraft arriving at an airport in a Center are said to flow from that Center into Center 0. The aircraft flows among Centers in this description can be characterized:

- The aggregate departures of aircraft from all airports in a Center (and their consequent injections into the airspace) are governed by a Poisson process. To see why, note that the aggregate departures in a Center comprise a *merging* of the departures from each airport in the Center, which are each governed by an independent Poisson process. The result of such a merging is well-known to be governed by a Poisson process [53]. We denote the (in general time-varying) rate of the Poisson process governing departures in Center i (or equivalently, flows from Center 0 to Center i) by $\lambda_{0i}(t)$.

- *Aggregate boundary crossings*, or movements of aircraft across a particular boundary from one Center to another one, are also governed by a Poisson process. To see why, first consider the aircraft that depart from a particular airport, fly along a particular route to a destination airport, and have a certain cruising speed. The departures of these aircraft are governed by a Poisson process, since these departures are a *splitting* of all the departures from the airport of interest [53]. Now consider any boundary between two Centers along the route taken by these aircraft. The boundary crossings of these aircraft are also governed by a Poisson process, since each aircraft crosses the boundary after a fixed delay following departure. Finally, the aggregate boundary crossings are the merging of boundary crossings along several routes at several different cruising speeds, each of which are governed by a Poisson process. Thus, the aggregate boundary crossings are governed by a Poisson process. We denote the rate at which aircraft cross a boundary from Center i to Center j at time t by $\lambda_{ij}(t)$.

- Using the same reasoning as for boundary crossings, we find that the aggregate arrivals of aircraft in a Center (i.e., the arrivals of aircraft at all airports in a Center) are governed by a Poisson process. The rate of the aggregate arrivals in Center i (or equivalently, flows from Center i to Center 0) at time t is denoted $\lambda_{i0}(t)$.

- The number of aircraft $s_i(t)$ in Center i at time t is a Poisson random variable. To see why, again consider the aircraft that depart from a particular airport, fly along a particular route to a destination airport, and have a certain cruising speed, and consider a particular Center along the route traveled by these aircraft. The number of these aircraft that are in the Center of interest at time t is equal to the number that entered the Center between times $t-\widehat{t}$ and t, where $\widehat{t}$ is the (fixed) amount of time needed for the aircraft to pass through the Center. Thus, this number equals the number of boundary crossings into the Center over a time interval $\widehat{t}$. Since these boundary crossings are governed by a Poisson process, the number of these aircraft in the Center is known to be a Poisson random variable. Finally, the total number of aircraft $s_i(t)$ is found by summing Poisson random variables of this sort, and so is Poisson.

Even though we can compute Center count and boundary-crossing statistics using the Poisson process description for the ATS, this description is difficult to use directly for many computations of interest, because aircraft statistics along each particular route must be computed and stored separately. For example, if the total departure rate in a Center is changed in the model (perhaps to reflect the occurrence of inclement weather in that Center), the flows along all routes leaving from each airport in the Center must be recomputed. For similar reasons, the *dynamics* of Center statistics are difficult to determine. For example, let's say that we wish to determine the distribution for the number of aircraft in a Center at some time in the future, given information about the current state of the ATS. The Poisson process description can be used to compute this distribution only if the exact locations of every aircraft in the airspace and the statistics of the possible departures along each route from each airport are explicitly modeled. For applications in which dynamics potentially need to be recomputed for many sets of model parameters, such as control or optimal design applications, such computationally intensive calculations may be infeasible. Thus, we are motivated to develop a simpler aggregate stochastic model for the ATS.

8.2.2 An Aggregate Dynamic Model for Center Counts

The state variables in our aggregate model are the numbers of aircraft in each Center, tracked at discrete times. Let ΔT be the *time-interval* of the model. Thus, the number of aircraft in each Center is tracked at the times $k\Delta T$, $k = 0, 1, 2, \ldots$ We denote the number of aircraft in Center i at time $k\Delta T$ as $s_i[k]$. Our goal is to develop a model that describes the time-evolution of the state variables $s_i[k]$.

First, between time-steps k and $k+1$, the state variables can change because of aircraft entering each Center upon departure from airports. In our aggregate model, the number of aircraft that depart from airports in Center i, $1 \leq i \leq n$, between times $k\Delta T$ and $(k+1)\Delta T$ is modeled as a Poisson random variable $U_{0i}[k]$, with mean denoted by $\lambda_{0i}[k]$. In addition to the flows into the ATS due to departures at airports, aircraft may change Centers, or leave a Center through arrival at an airport. In our aggregate model, we envision each aircraft in a Center as moving to another Center or arriving at an airport with some probability during a time-step. In particular, we assume that each aircraft in Center i independently travels to Center j (or leaves the airspace for $j = 0$) between time-steps k and $k+1$ with probability $p_{ij}[k]$. We denote the total number of aircraft that flow from Center i to Center j between times k and $k+1$ by $U_{ij}[k]$. For small enough ΔT, it can be shown that the conditional distribution for the flow $U_{ij}[k]$ given the Center count $s_i[k]$ is well-approximated by a Poisson random variable, with mean $p_{ij}[k]s_i[k]$ [119]. (If ΔT is larger, the $U_{ij}[k]$ must be represented using dependent binomial random variables; the same analyses of the model can be completed in this case, albeit with a little extra computation.) Thus, we have modeled the flows of aircraft among Centers in the airspace, as well as the flows of aircraft arriving at airports.

Now that we have characterized the flows of aircraft in our model, the state variable update can be specified by accounting for the number of aircraft entering and leaving each Center i between times k and $k+1$:

$$s_i[k+1] = s_i[k] - \sum_{j=0, j\neq i}^{n} U_{ij}[k] + \sum_{j=0, j\neq i}^{n} U_{ji}[k], \tag{8.1}$$

This update rule defines the temporal evolution of our aggregate stochastic model. The dynamics of the model are depicted pictorially in Figure 8.2.

In our application of the aggregate model, it is not Equation 8.1 that we propagate forwards in time. Instead, we propagate expectations and variances of the $s_i[k]$, using equations that

are derived from Equation 8.1, and that have a very simple structure. The details are given in Section 3.2.

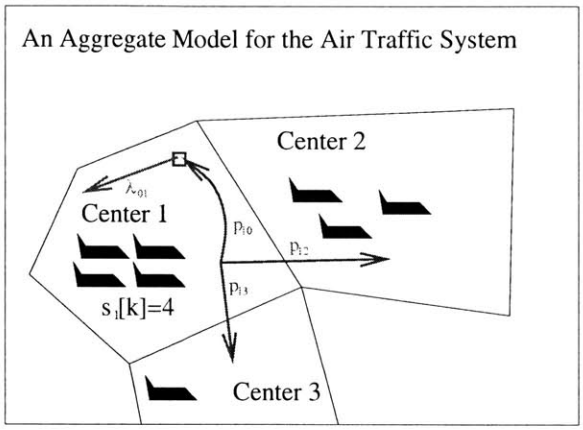


Figure 8.2: This figure describes the dynamics of our aggregate stochastic model for the ATS. Aircraft enter Centers according to Poisson processes. Also, during an interval of time, each aircraft in a Center may move to another Center or leave the system, with some probability. We are interested in tracking the number of aircraft in each Center in this model.

Our aggregate model is closely related to the detailed Poisson process description of the ATS discussed in Section 2.1. In the detailed description, the departures of aircraft in each Center i are governed by a Poisson process with rate $\lambda_{0i}(t)$. Thus, between times $k\Delta T$ and $(k+1)\Delta T$, the number of departing aircraft in Center i is a Poisson random variable, with mean $\int_{k\Delta T}^{(k+1)\Delta T} \lambda_{0i}(t)dt \approx \lambda_{0i}(k\Delta T)\Delta T$. If the mean of $U_{0i}[k]$ in the aggregate model is chosen to be $\lambda_{0i}[k] = \lambda_{0i}(k\Delta T)\Delta T$, the statistics of the departures in each time-interval in the aggregate model and detailed description are essentially identical. Unlike departures, the flows among Centers in this aggregate model cannot be made identical to the flows in the detailed description of the ATS (e.g., an aircraft in the aggregate model can flow from Center i to Center j and then return to Center i with some small probability in the aggregate model, while an aircraft in the detailed description follows a route and so would not revisit a Center). However, the probabilities $p_{ij}[k]$ in the aggregate model can be set so that the flows in the aggregate model match the flows in the detailed description in an average sense.

To do so, consider the number of aircraft $s_i[k]$ in Center i at some time-step k in the detailed description of the ATS. It is reasonable to expect that, on average, a certain fraction (possibly 0) of these aircraft will travel to each other Center, or will exit the ATS, during a

time interval ΔT. At time-step k (time $k\Delta T$), there are on average $\lambda_{ij}(k\Delta T)\Delta T$, $0 \leq j \leq n$, aircraft that flow to Center j during the next time interval (this includes aircraft exiting the system through arrival at airports, which corresponds to $j = 0$). Furthermore, there are on average $\bar{s}_i(k\Delta T)$ aircraft in Center i. Thus, we might expect that a fraction $\frac{\lambda_{ij}(k\Delta T)}{\bar{s}_i(k\Delta T)}\Delta T$, $0 \leq j \leq n$, of the aircraft in Center i will travel to j between time $k\Delta T$ and $(k+1)\Delta T$. By setting probability $p_{ij}[k]$ in the aggregate model equal to $\frac{\lambda_{ij}(k\Delta T)}{\bar{s}_i(k\Delta T)}\Delta T$, we obtain the same average fraction of aircraft traveling from Center i to Center j as in the detailed model. We can also show (with a little algebra) that the average number of aircraft in Center i, as well as the average number of aircraft that flow from Center i to Center j, are essentially the same for the aggregate and detailed descriptions at each time-step if the $p_{ij}[k]$ are chosen in this way.

8.3 Parameter Determination, Analysis, and Verification

In this section, we pursue three important questions regarding the stochastic model developed for an ATS:

1. How can the parameters of the model be determined from data?
2. How can the model be analyzed, and why is this analysis useful?
3. Does the model accurately represent some aspects of the behavior of ATS?

Throughout the discussion in this section, examples from the U.S. ATS are used to illustrate our methodology.

8.3.1 Parameter Determination

Our aggregate model for Center counts requires three sets of parameters: the time-interval ΔT, the average number of departures $\lambda_{0i}[k]$ in each Center i at time-step k, and the probabilities that aircraft in Center i, $1 \leq i \leq n$, go to Center j, $0 \leq j \leq n$, during time-step k. Because the detailed Poisson process description of an ATS represents the movement of actual aircraft more precisely than the aggregate model, it is more natural for us to estimate parameters of the detailed description from historical data first, and subsequently infer the

parameters of the aggregate model. Thus, we focus on estimating the parameters of the detailed description— namely, the mean number of aircraft in each Center ($\bar{s}_i(t)$); and the average departure flow rates, arrival flow rates, and flow rates between Centers ($\lambda_{ij}(t)$). Assuming that the ATS is operating under typical conditions, these parameters can be estimated from historical data. Once these parameters have been found, we can choose a time interval ΔT and approximate the aggregate model's parameters as described in the Section 2, i.e., by setting $\lambda_{0i}[k] = \lambda_{0i}(k\Delta T)\Delta T$ and $p_{ij}[k] = \frac{\lambda_{ij}(k\Delta T)}{\bar{s}_i(k\Delta T)}\Delta T$.

First, consider the mean number of aircraft $\bar{s}_i(t)$ in Center i at time t. In general, we allow this expectation to vary with time in our framework; realistically, we might expect that the mean number of aircraft in each Center would change slowly throughout a single day under typical operating conditions, but would be nearly identical when compared at a certain time over several days. As a simple first attempt at modeling the U.S. ATS, we use a constant value for mean number of aircraft in each Center. These average numbers of aircraft are estimated from actual Center counts during 500 minutes in the afternoon and evening of a particular day, September 6, 2000. For example, we find that, on average, 127.6 aircraft are present in the Seattle Center during this time interval.

The second set of parameters that are necessary for the analysis of the model are the rates $\lambda_{ij}(t)$ of aircraft flow from Center i to Center j (or, for $i = 0$ or $j = 0$, the flow rates into or out of the Center due to departures and arrivals). Like the mean parameters, the rate parameters $\lambda_{ij}(t)$ can be computed by using historical data on the numbers of aircraft that cross each boundary, and that enter into and depart from the system at airports. As with Center count averages, the flow rates across boundaries are expected to vary slowly throughout the course of a day. In our simulations, we have used a crude model in which a single flow rate is estimated based on historical data from September 6th, 2000. For simplicity's sake, these flow rates have been computed assuming that each aircraft flies along the shortest path from its origin to its destination. A more accurate model would require careful measurement of flow rates; here, we are interested in the modeling methodology rather than the accuracy of the specific model, so a more careful computation of the flow rates is not pursued. The average flow rates computed from data and used to construct the model are shown in Figure 8.3.

Under normal operating circumstances, it is reasonable that mean numbers of aircraft and expected flow rates can be computed from data. Thus, the model can employ these average statistics, combined with actual information on the state of the ATS, to simulate and analyze the future behavior of the network. However, under unusual conditions (due to

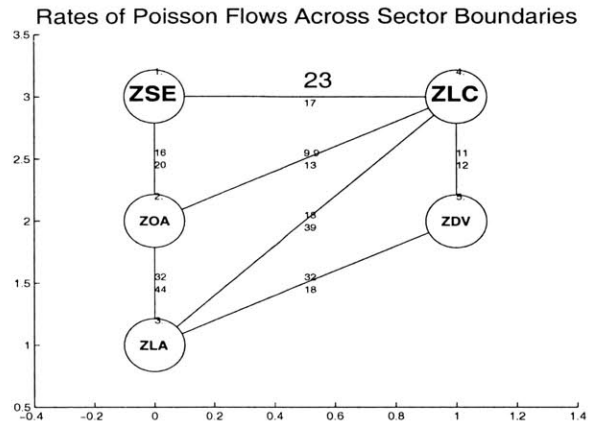


Figure 8.3: The expected flow rates between Centers (in aircraft per hour) are shown for five of the Centers in U.S. ATS. These average flow rates were computed using data from September 6, 2000. The top number on each branch show the average number of aircraft moving from the lower-numbered Center to the higher-numbered Center, while the bottom number on each branch shows the average flow in the reverse direction.

bad weather or other disturbances, for example), average behavior most likely cannot be deduced from historical data. In fact, in these aberrant situations, we would hope to compute these averages through the modeling process rather than using them as parameters in the model! In particular, we expect that information about a disturbance can be used to change certain model parameters locally (i.e, near the affected Center), and in turn the model analysis can be used to quantify the behavior of the disturbed ATS.

The final parameter of the model, the time-interval ΔT, should be chosen small enough to capture the fluctuations of interest in the dynamics of the ATS. However, if ΔT is chosen to be too small, unnecessary computation is introduced in the analysis of the model; if ΔT is chosen to be too large, some dynamics of interest may not be modeled, and also the use of a discrete model for the dynamics may introduce significant error. To choose a time-step ΔT for our model for the U.S. ATS, we looked at plots of the average numbers of aircraft in Centers during time intervals of various durations (Figure 8.4). Based on these plots, we believe that a time-step of less than or equal to 10 minutes is sufficient to capture the dynamics of interest in the ATS.

Once the mean Center counts, expect flow rates, and time-step have been determined, the probability $p_{ij}[k]$ that a randomly chosen aircraft in Center i goes to Center j between $k\Delta T$ and $(k+1)\Delta T$ can be computed for each i and j. Some of these probabilities are shown in

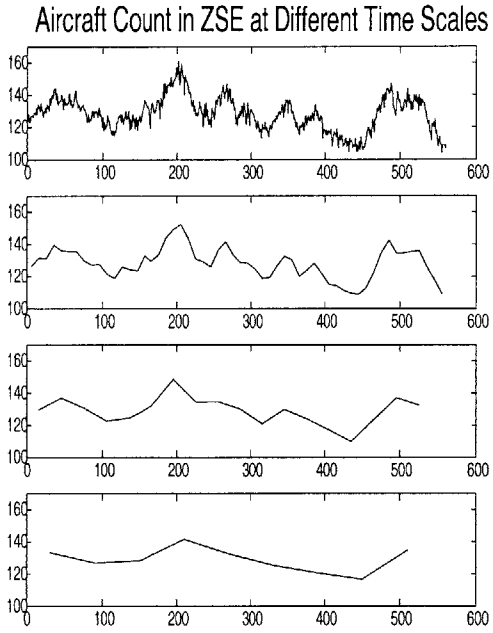


Figure 8.4: The time-evolution of the number of aircraft in the Seattle Center (ZSE) during the afternoon and evening of September 6th, 2000, is shown, plotted at four different time scales. In particular, the top plot shows the number of aircraft that were actually present in ZSE, measured at one-minute intervals. The second, third, and fourth plots show the average numbers of aircraft present in ZSE during each ten-minute interval, thirty-minute interval, and one-hour interval, respectively. Ten-minute averages of Center counts accurately approximate the actual Center counts (almost always to within 5 aircraft), suggesting that 10 minutes is a fine enough resolution to capture dynamics of interest in the U.S. ATS.

Figure 8.5.

8.3.2 Analysis

Once the parameters of our model have been determined from historical data, the model can be analyzed to gain insight into the behavior of the ATS. In particular, given the numbers of aircraft in each Center at the initial time, moments and cross-moments of the numbers of aircraft in each Center at future times can be computed. Thus, we can predict expected future Center counts and the possible variability in these Center counts using the model. In turn, the expected response and variability in Center counts can be used to iden-

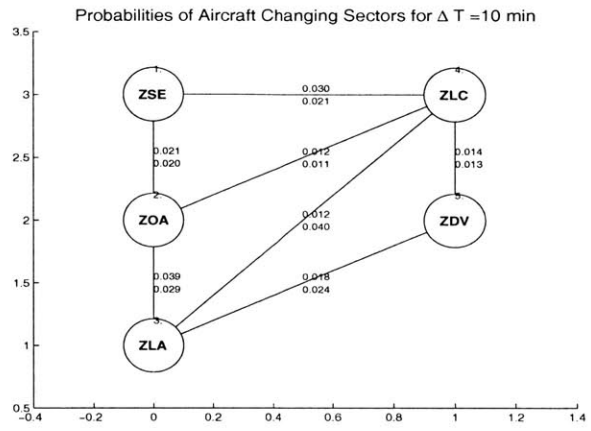


Figure 8.5: This plot shows the probability that an aircraft in the aggregate model moves from one Center to another during one time-interval. The upper probability on each branch is the probability that an aircraft from the lower-numbered Center moves to the higher-numbered Center. (In our example, the probabilities do not change with time, since the mean numbers of aircraft and flow rates do not change with time. More generally, however, these probabilities may depend on the time-step k.)

tify regions of the airspace that may be prone to excessive traffic. These regions could then be studied more carefully to determine whether or not capacity excesses would actually occur.

To compute the expected numbers of aircraft in each Center at each time-step, given initial conditions, it is helpful to redefine the model specified by Equation 8.1 in vector notation. Consider the following definitions:

- Define the *state vector* at time-step k to be

$$\mathbf{s}[k] = \begin{bmatrix} s_1[k] \\ \vdots \\ s_n[k] \end{bmatrix}$$

.

- Define the elements of the $n \times n$ *transition matrix* $P[k]$ as follows:

$$\text{for } 1 \leq i \leq n, \quad P_{ii}[k] = (1 - \sum_{j=0, j \neq i}^{n} p_{ij}[k])$$
$$\text{for } 1 \leq i \leq n,\ 1 \leq j \leq n, i \neq j, \quad P_{ij}[k] = p_{ji}[k].$$

- Define the *transition vector* to be

$$\beta[k] = \begin{bmatrix} \lambda_{0,1}[k] \\ \vdots \\ \lambda_{0,n}[k] \end{bmatrix}.$$

Our goal is to determine the conditional expectation for the state vector $\mathbf{s}[k]$ given the initial state vector $\mathbf{s}[0]$, or $E(\mathbf{s}[k]|\mathbf{s}[0])$. In fact, these conditional expectations can be found through the following linear recursion:

$$E(\mathbf{s}[1]|\mathbf{s}[0]) = P[0]\mathbf{s}[0] + \beta[0]$$

and

$$E(\mathbf{s}[k+1]|\mathbf{s}[0]) = P[k]E(\mathbf{s}[k]|\mathbf{s}[0]) + \beta[k], \quad k \geq 1. \tag{8.2}$$

An outline for the derivation of Equation 8.2 is given in the Appendix. A simulation of the number of aircraft in the Seattle Center ZSE is compared with the expected number (conditioned on the initial Center counts) of aircraft in ZSE in Figure 8.6. The conditional expected number of aircraft in ZSE depends upon the initial Center counts at the beginning of the simulation, but eventually approaches the steady-state expected number (which is a parameter of the model).

In addition to providing a prediction for the behavior of the ATS, we believe that the recursion for mean Center counts is valuable because it allows us to determine the sensitivity of Center counts to various parameters in the model, including initial conditions and steady-state average flow rates. The sensitivity analysis of the model is not presented in any further detail here, but we note that sensitivities of the expected state vector to parameter changes can be deduced from Equation 8.2.

In addition to the conditional mean of each Center count, higher moments and cross moments of the numbers of aircraft in each Center (conditioned on an initial state) can be

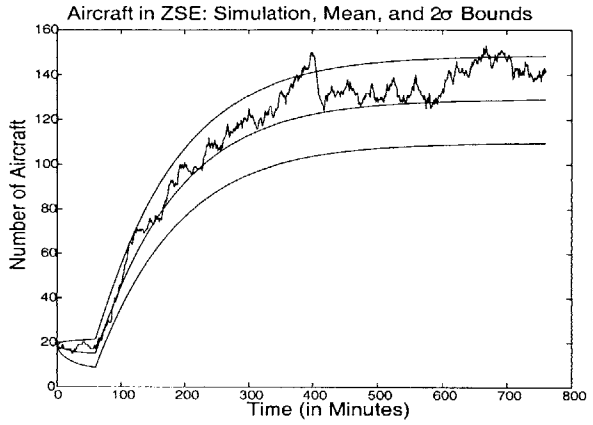


Figure 8.6: This plot shows a simulation of a flow model for the U.S. ATS. In particular, the number of aircraft in ZSE is simulated at one-minute intervals, over a duration of 760 minutes. In addition to the flow model simulation of the number of aircraft in ZSE, the expected number of aircraft in ZSE, conditioned on the initial counts of all Centers, and 2-standard deviation bands around this expected number are plotted. The initial Center counts in this simulation are based on actual data of Center counts at approximately 5:00 AM PDT on September 6th, 2000. We have assumed that there are no departures from airports in ZSE until 6:00 AM, and then departures commence at a nominal daytime rate, explaining the sudden jump in the aircraft count at 6:00 AM.

computed through a linear recursive procedure. We have developed the recursion for the second-order moments and cross-moments. The analysis of the second-order moments is straightforward, but the resulting expression for the moments at each time-step is not very illustrative. Thus, we only briefly describe the analysis in the Appendix. In Figure 8.6, the computation of the second moments is used to develop 2σ bounds on the number of aircraft in ZSE.

8.3.3 Verification of the Model

First, we would like to show that a Poisson process description for the ATS is adequate. Our methodology is founded on the assumption that departures from airports are described accurately by Poisson processes. At the time scales of interest to us (on the order of a few minutes to a few hours), the data that we have explored suggests that departures from large airports are indeed essentially Poisson in nature. For example, we plot a histogram of the number of daytime departures during one-minute intervals at Chicago

O'Hare Airport (ORD) in Figure 8.7, and find that the departures in a minute are well modeled by a Poisson random variable. Also, we find that the number of aircraft departing in any given minute is independent of the number of aircraft departing at other times. Both these properties suggest that the departure process is well represented as Poisson, at a one-minute granularity. We have found similar behavior at other airports, and for time-scales of greater than one minute. At smaller airports, a Poisson process description with a fixed mean is not accurate, but we believe that a time-varying Poisson process description could be reasonable; alternately, the aggregate departure process from several small airports can perhaps be modeled as a Poisson process. At any rate, the departures from large airports contribute most significantly to air traffic, so we believe that a Poisson process description for departures is justified. To further investigate our Poisson process frame-

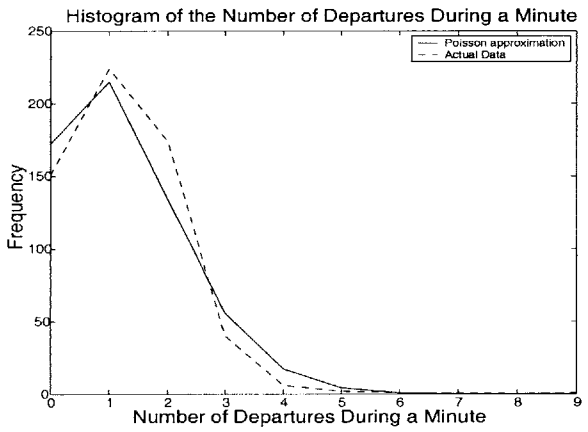


Figure 8.7: An empirical distribution (histogram) of the number of aircraft departing from an airport (ORD) during minute intervals is shown. The empirical distribution is based on data from ORD over 600 consecutive minutes; over this time interval, the average departure rate from the airport remained approximately constant. Also, the empirical distribution is compared with a Poisson distribution with the same mean. The comparison suggests that the number of aircraft departing from ORD in minute intervals are indeed well-represented by a Poisson random variable.

work, the distribution of the number of aircraft in a Center is also studied. For example, an empirical distribution of the number of aircraft in the Seattle Center (ZSE) is compared with a Poisson distribution of the same mean in Figure 8.8. The variances of the empirical and fit distributions are close (117.6 and 127.6, respectively), and the two distributions are similar in shape. The statistics of other Centers are similar. Thus, we believe that a Poisson random variable model is a good description for the number of aircraft in a Center.

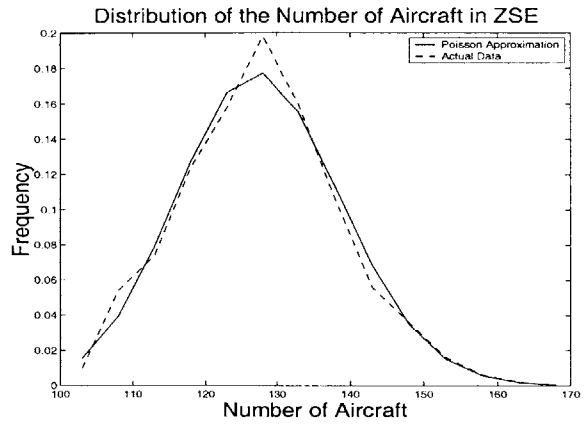


Figure 8.8: An empirical distribution for the number of aircraft in ZSE is shown, and is compared with a Poisson random variable of the same mean. The empirical distribution is generated based on 500 observations at one-minute intervals. The sample variance of the empirical distribution and the variance of the Poisson approximation are similar.

Second, we explore whether our model, which is motivated by the Poisson process description of an ATS but is not identical to it, can accurately represent the dynamics of the U.S. ATS. We can check whether or not our model is consistent with a Poisson process description of the ATS. For example, the standard deviation in the steady-state aircraft count predicted by the model can be compared with the standard deviation if the count were modeled by a Poisson random variable of the same mean (in accordance with the Poisson process description of the ATS). In our example, the standard deviation for the number of aircraft in ZSE predicted by our model is 9.7, while the standard deviation predicted by a Poisson random variable representation is 11.3. In general, we find that the standard deviations for Center counts predicted by our model are slightly smaller than, but close to, the standard deviations predicted by the Poisson process description of an ATS.

Another approach for verifying the model is to check whether or not the behavior of the model matches actual Center count data. In Figure 8.9, the number of aircraft in ZSE during a 12-hour period on September 6th, 2000 is traced. Furthermore, the model prediction for mean and standard deviation for the number of aircraft in ZSE, given the initial numbers of aircraft in all Centers, is plotted for comparison; these are the same as in Figure 8.6. The actual data largely falls within two standard deviations of the predicted mean, suggesting that the model provides a good description of the dynamics of the ATS. The most noticeable difference between the model prediction and the actual data is the slower set-

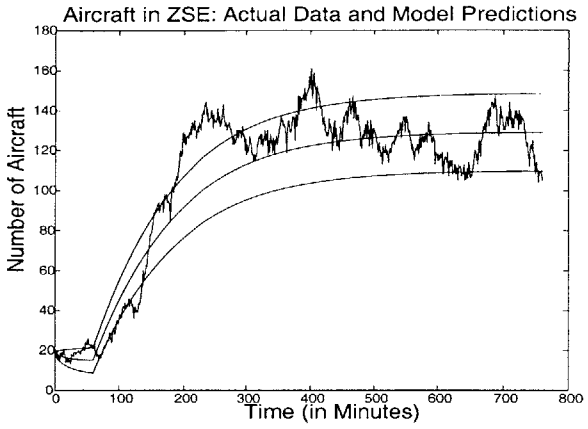


Figure 8.9: The actual number of aircraft in ZSE at 760 consecutive one-minute time steps is compared with the mean number of aircraft predicted by our model, using the actual Center counts at all Centers at the time of the first data point (5:00 AM PDT, September 6th, 2000) as the initial conditions for the model. In our model, we have assumed that aircraft departures in ZSE begin at 6:00 AM. This plot also includes the 2σ bounds on the aircraft count in ZSE predicted by our model. The actual data is largely contained within two standard deviations of the predicted mean, suggesting that the model predictions for the mean and standard deviation are both reasonable. The one noticeable difference between the actual data and the model prediction is the sluggishness in the model's transient as compared to the data.

tling time to the steady-state of the prediction. Based on our explorations, we believe that this sluggishness in the model comes about because aircraft in the model do not spend fixed amounts of time in Centers, and instead may be delayed for lengthy time periods with some (small) probability.

In our comparison of the model prediction with historical data, we chose to track the Center counts beginning early in the morning of September 6th, even though the parameters of the model are based only on afternoon and evening Center counts. It is reassuring that the model can predict the Center counts from this initial condition, albeit imperfectly. After all, the primary goal of our methodology is to develop a *dynamic* model—one which can track the transient behavior of the ATS (such as the increase in traffic in the airspace in the morning), as well as the spatial and temporal correlations in Center counts.

8.4 Extensions of the Basic Model

8.4.1 Extension 1: A Hierarchical Model

Real ATS naturally have a hierarchical structure: for example, Centers in the U.S. ATS are further subdivided into smaller regions, called *Sectors*. Modeling of parts of the U.S. ATS at a Sector level is often valuable, both because many air traffic control and flow management decisions are made at the Sector level, and because a model with a finer spatial granularity can predict the future behavior of the system more accurately. A model for Sector counts also may require a finer time-sampling, because of the occurrence of significant dynamics over shorter time intervals. On the other hand, parts of the ATS can sometimes be aggregated in a model, reducing the computational complexity of the analysis while still correctly representing the parts of the system of interest.

Our modeling methodology is well-suited for describing a system at various spatial and temporal granularities. For example, consider an ATS with three levels of representation (again, we adopt the terminology for the U.S. ATS in our model):

1. Some regions in the airspace are modeled at a Sector level. Thus, the number of aircraft in each Sector in these regions is represented. These Sector counts are tracked at intervals of ΔT.
2. Some regions in the airspace are modeled at a Center level. The Center counts in these regions are tracked less often, at intervals of $f\Delta T$, for some positive integer f.
3. Some regions of the airspace are not modeled at all, and aircraft counts in these regions are not tracked (though aircraft flows to and from these regions are still incorporated in the model).

Figure 8.10 shows such a hierarchical model for the ATS.

A system with these three levels of representation can be modeled in our framework, as follows:

- First consider the update of Sector counts, in regions where Sector-level dynamics are represented. Sector counts are updated at intervals of ΔT. Each of these Sectors has

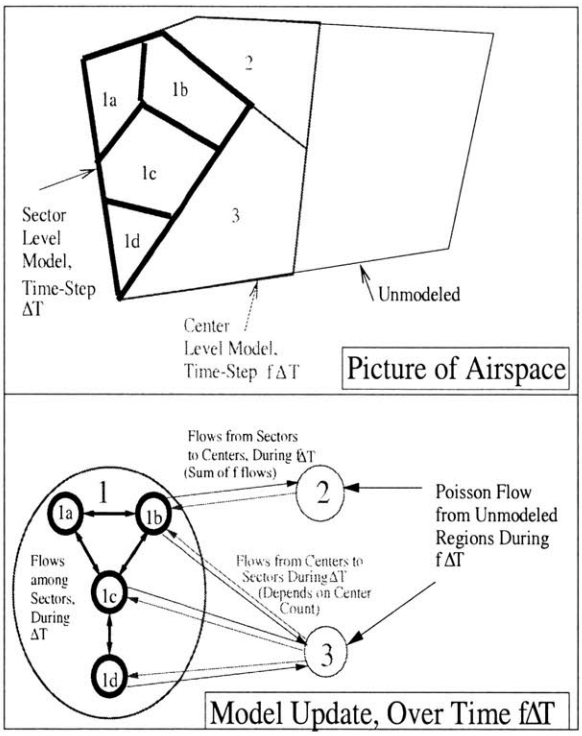


Figure 8.10: This figure shows a hierarchical model for the ATS which incorporates dynamics at various levels of aggregation and time-scales. The upper portion of the figure is a drawing of the airspace, while the lower portion describes the equivalent model.

flows into and out of other parts of the airspace. In general, a Sector may have flows to and from other modeled Sectors, regions modeled at the Center level, unmodeled regions of the airspace, and the exterior of the system (i.e., flows to and from airports within the Sector). The flows out of a Sector in an interval ΔT are computed based on probabilities that aircraft flow to each other region, and out of the airspace (which can be found by deducing the rates of traffic flow among the appropriate regions of the airspace using historical data). Meanwhile, flows into a Sector are generated in different ways in the simulation. Flows from the exterior of the system, and from unmodeled regions, are modeled as Poisson processes. Flows from regions modeled at a Center level are determined based on the most recently updated counts in these Centers (along with flow probabilities), and flows from other modeled Sectors are determined based on these Sector's counts at the previous time-step along with flow probabilities.

- Aircraft counts in regions modeled at the Center level are updated at intervals of $f\Delta T$.

The flows out of each Center to other Centers, unmodeled regions of the airspace, and the exterior of the system, are determined based on the current count of the Center and probabilities of aircraft going to other regions of the airspace. The flows out of a Center to modeled Sectors are found by summing the appropriate flows into the Sector (which have already been generated) over f intervals of duration ΔT. Next, consider flows into these Centers. Flows from the exterior of the airspace and from unmodeled regions are generated as Poisson processes. Flows from other Centers are generated as in the basic model, using flow probabilities and counts in these Centers. Finally, the flows from modeled Sectors to Centers are found by summing the appropriate flows out of these Sectors over f time-steps.

8.4.2 Extension 2: A Model with Stochastic, Flow-Altering Disturbances

The U.S. ATS is subject to disturbances that change rates of aircraft flow in parts of the network. Many of these flow-altering disturbances, which are often inclement weather events in parts of the airspace, cannot accurately be predicted in advance. Furthermore, although the disturbance event may directly affect only a small part of the airspace, the resulting changes in flows and Sector/Center counts may propagate throughout the network. Since our model for the U.S. ATS is stochastic, we can naturally incorporate stochastic disturbances that alter flows in the model. By computing the expected behavior and variability of Center counts and flows in the model, regions of the airspace that may be prone to capacity excesses due to the weather events can be identified. In turn, the model may suggest improved methods for managing traffic flow in response to weather disturbances.

We model local perturbations as changes to the nominal parameters, as shown in Figure 8.11. In our approach, multiple disturbances, each of which occur independently with some probability, can affect flows in a ATS. Given that a particular set of disturbances has occurred, we can calculate statistics of Center counts with our basic model, using the appropriate set of model parameters (which are modified from their nominal values based on the particular disturbances that have occurred). In turn, we can calculate statistics of Center counts without prior knowledge of the disturbances, by scaling the predicted statistics for each set of disturbances with the probability that these disturbances occur, and then summing these scaled statistics. In this way, the dynamics of an ATS that is subject to stochastic disturbances can be modeled and analyzed.

One possible shortcoming of this approach for modeling stochastic disturbances is the

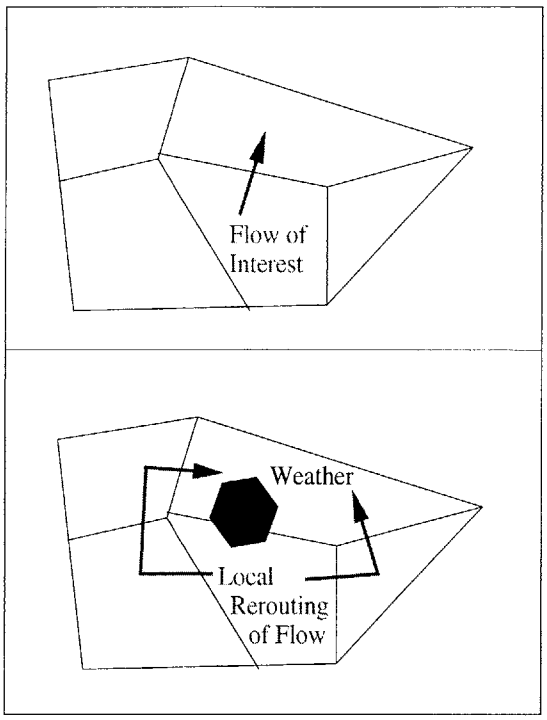


Figure 8.11: Local perturbation of flows due to a weather event is depicted. In this case, the flow of interested is rerouted in two different directions, leading to different flow rates across each boundary.

computational complexity resulting from the large number of disturbances that may need to be considered. (For example, if there are 10 different weather events that may or may not be present on a given day, we must consider $2^{10} = 1024$ possible combinations of disturbances.) Given certain special conditions on the location of disturbances, the computational complexity can sometimes be reduced by considering the change in the system's dynamics due to each disturbance separately, and then combining these individual responses.

8.5 Conclusions and Future Work on the ATS Model

In this article, we have proposed an aggregate stochastic model for an ATS. Some analyses of the model that may eventually prove useful for predicting and controlling flows in an ATS have been discussed. Also, some verification of our modeling framework has been

attempted, and two extensions of the basic model have been outlined.

We believe that aggregate stochastic models provide a promising description of the ATS, but more study is required to gauge the value of these models in improving understanding and operation of the ATS. To better undertand the benefits and drawbacks of using aggregate and stochastic models, we plan to compare our model with other deterministic and detailed stochastic models for the ATS. We are also interested in using the model as a framework for developing Traffic Flow Management (TFM) algorithms.

8.6 The Air Traffic Model as an MLSN: Mathematical Details

8.6.1 The Moment-Linearity Conditions

Equation 8.2 shows how the expected numbers of aircraft in each Center at each time-step can be calculated recursively, given the initial numbers of aircraft in all Centers. We give a conceptual justification of Equation 8.2, without concerning ourselves with a detailed proof. In particular, we explain how the recursion in Equation 8.2 comes about, without concerning ourselves with the vector notation of the recursion. To do so, consider $E(s_i[k+1] \mid \mathbf{s}[k]$, the expected number of aircraft in a Center i at time $k+1$, given all Center counts at time k. This expectation can be found by adding and subtracting the expected flows into and out of Center i, respectively, from $s_i[k]$. Note that the expected number of aircraft that flow from Center i to a Center j (or out of the system for $j = 0$) is given by $s_i[k]p_{ij}[k]$, the number that flow from a Center j to Center i is $s_j[k]p_{ji}[k]$, and the number that depart from airports in Center i is $\lambda_{0i}[k]$. Thus, the conditional expectation for the number of aircraft in Center i is

$$\begin{aligned} &E(s_i[k+1] \mid \mathbf{s}[k]) \\ &= s_i[k] - \sum_{\substack{j=0 \\ j \neq i}}^{n} s_i[k]p_{ij}[k] + (\sum_{\substack{j=1 \\ j \neq i}}^{n} s_j[k]p_{ji}[k] + \lambda_i[k]), \end{aligned} \tag{8.3}$$

which is a linear function of the time-k Center counts. Finally, by taking the expectation of Equation 8.3 with respect to the time-k Center counts $\mathbf{s}[k]$, given the initial Center counts

s[0], we find that

$$\begin{aligned} & E(s_i[k+1] \mid \mathbf{s}[0]) \\ & = E(s_i[k] \mid \mathbf{s}[0]) - \sum_{\substack{j=0 \\ j \neq i}}^{n} E(s_i[k] \mid \mathbf{s}[0]) p_{ij}[k] \\ & + (\sum_{\substack{j=1 \\ j \neq i}}^{n} E(s_j[k] \mid \mathbf{s}[0]) p_{ji}[k] + \lambda_i[k]). \end{aligned} \tag{8.4}$$

Thus, we see that the expected number of aircraft in Center i at time $k+1$ given s[0] can be written as a linear function of the expected Center counts at time k given s[0].

We also briefly discuss why second moments and cross-moments of state variables can be found using linear recursions given the initial state s[0]. The recursion for the second moments and cross-moments is derived analogously to the recursion for the expectations of state variables. For example, consider $E(s_i^2[k+1] \mid \mathbf{s}[k])$, the expected value of the square of the number of aircraft in a Center i at time $k+1$ given the Center counts at time k. Substituting the expression for $s_i[k+1]$ given in Equation 8.1 into this expectation gives

$$\begin{aligned} & E(s_i^2[k+1] \mid \mathbf{s}[k]) \\ & = E((s_i[k] - \sum_{\substack{j=0 \\ j \neq i}}^{n} U_{ij}[k] + \sum_{\substack{j=0 \\ j \neq i}}^{n} U_{ji}[k])^2 \mid \mathbf{s}[k]) \end{aligned} \tag{8.5}$$

To continue the analysis, note that $\sum_{\substack{j=0 \\ j \neq i}}^{n} U_{ij}[k]$ and $\sum_{\substack{j=0 \\ j \neq i}}^{n} U_{ji}[k]$ are two independent Poisson random variables given the time-k Center counts s[k], with means $\sum_{\substack{j=0 \\ j \neq i}}^{n} s_i[k] p_{ij}[k]$ and $\sum_{\substack{j=1 \\ j \neq i}}^{n} s_j[k] p_{ji}[k] + \lambda_i[k]$, respectively. Since the second moment of a Poisson random variable is a quadratic function of its mean, we find after some algebra that the expectation $E(s_i^2[k+1] \mid \mathbf{s}[k])$ is a quadratic function of the state variables at time k (i.e., the expectation can be written as a sum of second- and lower-order powers of the state variables, such as $s_i^2[k]$, $s_i[k]s_j[k]$, $s_i[k]$, etc.). Taking the expectation of Equation 8.5 with respect to s[k], we find that the second moment of $s_i[k+1]$ (given s[0]) is a linear function of second and lower moments and cross-moments of the time-k state variables (also given s[0]). In fact, it turns out that all second moments and cross-moments of time-$(k+1)$ state variables can be written as linear functions of second and lower moments and cross-moments of time-k state variables. Thus, the second-order statistics of the model can be found using linear recursions. It is clear that higher moment recursions can be developed in similar fashion,

so we do not consider them further here.

8.6.2 Connection between the Air Traffic Model and Infinite Server Queues

It turns out that our model for an ATS can be viewed as a discrete-time network of infinite server queues, by reasoning as follows. From a queueing theory viewpoint, we can describe each aircraft in a Center as being served independently at each time-step, with a probability $\sum_{\substack{j=0 \\ j \neq i}}^{n} p_{ij}[k]$ that service is completed at time-step k. In this interpretation, an aircraft that has completed service in Center i during time k then enters Center j (or leaves the system, for $j = 0$) by time $k+1$ with probability $\frac{p_{ij}[k]}{\sum_{\substack{j=0 \\ j \neq i}}^{n} p_{ij}[k]}$. Thus, we see that this model indeed can be envisioned as an open network of infinite server queues, in which served jobs (aircraft) are independently stochastically routed to other servers (regions), or out of the system.

Since our model can be viewed as a network of infinite-server queues, we briefly review relevant literature on such models. An introduction to discrete-time infinite-server queueing networks can be found in [40]. Infinite-server queueing networks with memoryless servers and Poisson input streams, which constitute a type of Jackson network with state dependent service rates, are particularly amenable to analysis [79]. For instance, the steady-state and transient joint distribution for the queue lengths in the network can be deduced [79]. Methods for characterizing (both transient and steady-state) first- and second-order statistics of queue lengths are well-known, and a characteristic function-based method for inferring higher moments as a function of time has also been developed [88]. Our approach serves to reformulate these higher-moment computations as temporal linear recursions, and to provide a framework for further analysis (e.g., estimation and control) of these models.

8.6.3 Our Model: An Airplane-Location Viewpoint

We have been concerned with tracking numbers of aircraft in regions of the airspace as a function of time. As an alternative, it is interesting to track the locations (regions) of aircraft with time. Once an aircraft in our model has entered the airspace (through departure), its location in the airspace is governed by a Markov chain that is independent of the dynamics of the other aircraft. More precisely, let's say that we specify a status to each aircraft in

the airspace at each time, which indicates the location of that aircraft—either the region containing the aircraft or the exterior of the airspace (which indicates that the aircraft has left the airspace through arrival). In our model, the status of each aircraft subsequent to take-off is governed by an independent Markov chain, since the next-status (location) of each aircraft given the entire past history of the system is realized based on a probability vector specified by the current status of that aircraft. In this formulation, the exterior of the airspace is represented as an absorbing state, and aircraft are not tracked after they land (aircraft that enter the aircraft are viewed as being "new")[1].

One possible use of this aircraft-location model is to incorporate correlations between aircraft routes in our model. Instead of viewing each aircraft as being governed by an independent Markov chain, we can think of the aircraft as influencing each others' statuses, and hence obtain an influence model generalization for the original aircraft-location model. We may be able to develop a more realistic model for the aircraft flows in this manner. We note, however, that an aircraft-location model is likely to be considerably more computationally intensive than a aggregate count model.

Another advantage of the aircraft-location viewpoint is that it suggests connections between with other models in the literature. For instance, cell locations of mobile users in cellular wireless networks are often modeled using finite-state Markov chains (e.g., [66]). Thus, we see that a cellular network with many users can be represented in the same way as the air traffic system, and the MLSS analyses can perhaps provide useful characterizations for this network.

[1] If we modify the original model slightly, we can use a representation with a finite number of aircraft that are located in regions in the air or on the ground, and move through these regions according to Markov processes.

Chapter 9

Conclusions and Future Work

9.1 Summary and Conclusions

In this thesis, we have introduced a class of discrete-time Markov models, called moment-linear stochastic systems (MLSS), that are specially structured so successive moments and cross-moments of state variables at each time can be found from equal and lower moments and cross-moments at the previous time, using linear recursions. We have shown that MLSS provide a framework for representing several common models, and that interesting network dynamics can be captured using MLSS.

We have exploited the special quasi-linear structure of MLSS to analyze their dynamics. In particular, we have not only shown how to compute MLSS state statistics, but also developed methods for developed qualitative features of state and moment asymptotes. We have also developed optimal linear estimators and quadratic-cost controllers for MLSS. These analyses constitute a significant generalization of analyses for discrete-time linear state-space systems, and are applicable to a variety of models of interest (e.g., jump-linear systems, infinite server queues, and hidden Markov models).

We have pursued two examples of MLSS—namely, a network model called the influence model, and a model for aggregate flows in an air traffic network—in some detail, both to illustrate our analyses and to explore the applicability of our approach. Our MLSS reformulation of the influence model leads to several new analyses of this model, including a proof for a conjecture of [9] on the settling time of state statistics and an efficient suboptimal estimator for the model. Our air traffic network model is a first attempt at end-to-end stochastic representation of air traffic flows, and is also novel in that aggregate flows are captured. This model holds promise as a framework for developing Traffic Flow Management (TFM) algorithms for the United States Air Traffic System.

We believe that MLSS models will prove to be useful tools for characterizing the dynamics

of complex networks and systems. As our examples show, MLSS can potentially represent myriad interactions that occur in systems of interest, yet they are structured enough to allow for significant analysis—not only of dynamics, but also of estimation and control. Further, MLSS allow us to analyze network dynamics at multiple levels of detail (e.g., moments of different orders, or cross-moments characterizing multiple components), with the cost of computation growing gracefully with the level of detail required.

Also, MLSS are valuable in that they provide a common framework for representing several types of interactions. We expose a simple concept, moment linearity, which can be viewed as underlying the dynamics several different stochastic systems. One significant advantage of considering such a common framework is that analyses that are well-known for one system can be applied to another. For instance, our formulation shows that linear estimation techniques—which are well-known in the context of linear and jump-linear systems—can be also applied be applied in the context of infinite-server queues and certain stochastic automata, such as the influence model. Another potential advantage of representing several types of interactions in the MLSS framework is that we can study systems which combine several of these interactions.

As a whole, MLSS are compelling structured stochastic models, which potentially have several applications and yet are amenable to significant analysis.

9.2 Future Work

Our aim in introducing MLSS has been to scan the range of their potential applications, and to introduce several analyses. In taking such a broad approach, we have to some extent sacrificed deeper study of particular applications and analyses. Much future work is needed to fill out the theoretical developments presented here, and to gauge the value of the MLSS framework in several application areas. Here, we list some significant directions for future work.

1. We believe that development of a general graphical representation of an MLSN is an important direction for future work. A graphical representation of an MLSN would illustrate the interaction structure of the various networks that we consider, and might also allow quick identification of qualitative features of these networks' dynamics. A graphical representation might also help in understanding, in a general sense, the types of

network dynamics (e.g., flows and influences) that can be captured in the MLSS framework. One significant challenge in developing a graphical representation of MLSN will be to account for the redundancies in the higher-moment recursions. The development of influence model graphs in [9] may provide some guidelines for the types of results that we can hope to derive from MLSN graphs, though (as described in Chapter 3) the interaction structures of MLSN are in general more complicated than those of an influence model. We mention that, to develop graphs that capture interactions among groups of sites or nodes, the notation of hypergraphs could be useful [19].

2. General results concerning asymptotics of the state distribution, including distributional convergence, are an important goal of future work. In Chapter 4, we outlined an approach for proving distributional convergence for a particular MLSS, the infinite-server queue with random service probabilities. Several other examples that we have described in the thesis also have state variables that seemingly converge in distribution (e.g., the air traffic and road traffic networks), and a general approach for proving distributional convergence may provide valuable insights into the dynamics of these examples. Just as in the infinite-server queue, convergence results for general Markov chains ([104]) are possibly applicable in the general case, because the moment recursions can perhaps be tied with the premises of these results. However, a significant challenge in applying these methods is the variety in the allowed state spaces (e.g., continuous-valued, discrete-valued, and indicator states) among our examples; this variation requires caution in applying general Markov chain results because notions of, e.g., recurrence and periodicity, which are required to derive these results, are tricky to verify in general.

3. As discussed in Chapters 5 and 6, optimal control from observations, rather than from the entire state, is an objective of future work. Because the separability of the optimal observer and controller that facilitates analysis of linear systems does not generalize to MLSS, optimal control from observations is likely a difficult problem. Nevertheless, we believe that at least good suboptimal partial-information controllers can be designed for MLSS, and that these will highlight interesting tradeoffs between estimation and control.

4. We can analyze the dynamics of the example MLSN that we have introduced, regardless of the particular network structure of the interactions. However, characteristics of these dynamics depend strongly on the network structure, so it is important for us to better understand the role of the network structure in the systems' dynamic evolution. More generally, for all MLSS, we would like to relate dynamic characteristics

to model parameters (e.g., eigenvalues of the recursion matrices), and to relate these model parameters to physical or graphical characteristics of the modeled systems. We have presented some basic results that relate dynamic characteristics with model parameters (e.g., results on settling times in the influence model, which invoke general characterizations of the subdominant eigenvalues of the recursion matrices), but have not explicitly considered graph-based analyses. We believe that far more can be done in both these directions for general MLSS and MLSN.

5. An important goal of future work is to develop tractable nonlinear generalizations of MLSS. The modeling power of an MLSS is limited by its quasi-linear structure, and hence many stochastic systems of interest cannot be represented as MLSS. Perhaps certain nonlinearities can be introduced to MLSS models, while preserving some or all of their tractability. One interesting approach for generalizing MLSS is to enforce moment-linearity conditions of a particular order, but to allow lower moments at the next time to depend on current statistics up to that order (e.g., to only enforce that first and second moments of the next state each depend on first and second moments of the current state).

6. We would like to consider the use of MLSS and MLSN in modeling systems with multi-faceted dynamics. Many real systems incorporate several disparate types of dynamics: for instance, a power network has an underlying communications network associated with it, and air traffic dynamics are modulated by random weather phenomena. MLSS and MLSN may prove useful in modeling such systems, because they can naturally capture several types of stochastic interactions.

7. The influence model is amenable to much further development. One glaring need is to pursue potential applications of the models in further detail. A potential application area of interest is in the modeling of social and economic networks[1]. In terms of theoretical studies of the model, parameter estimation and control are two topics that deserve more consideration.

8. We believe that our aggregate model for an air traffic system will be useful for developing control for air traffic flows. In particular, we plan to develop methods for Traffic Flow Management (TFM) in the framework of the aggregate model, using the quadratic controllers of MLSS developed in the thesis. Another important goal of future work is to incorporate weather phenomena in the model.

[1] We are grateful to Yannis Ioannides for an illuminating conversation on this topic (see, e.g., [73]).

9. Finally, and perhaps most importantly, we need to apply all the developed analyses to the many examples of MLSS considered in the thesis, and to delve further into relevant application areas to motivate these analyses. Through these applications, we may be able to better appraise the scope, and value, of MLSS.

Appendix A

Background Review

A.1 Expectations

This thesis is deeply concerned with *expectations*, or averages, of functions of random variables. We refer the reader to [21] and [119], for good introductions to expectations. Here, we only present some terminology on expectations that is used throughout the thesis, and clarify a notational issue about conditional expectations that is pertinent to our development.

In general, we use the notation $E(f(\mathbf{s}))$ to represent the expectation of the vector function $f()$ of a collection (vector) of random variables $\mathbf{s}$.

In the thesis, we derive methods for finding *moments* and *cross-moments* of random variables. These are expectations of powers, or products of powers, of random variables. In particular, the rth *moment* of a state variable s is the expectation of the rth power of s (where r is assumed to be positive and integral). Thus, the notation $E(s^r)$ is used to represent the rth moment of s. Meanwhile, an rth cross-moment is the expectation of the product of positive integral powers of two or more state-variables, such that the sum of the powers of all the state variables is r. For instance, for random variables s_1, s_2, and s_3, $E(s_1^2 s_2^3 s_3)$ is a sixth cross-moment.

The notion of a conditional expectation is widely used in our development. A conditional expectation is the average of a function of one collection of random variables, given another collection of random variables. We refer the reader to [21] for a thorough introduction to conditional expectation.

It is important for us to mention that our notation on conditional expectations is sloppy in one respect. To do so, consider the conditional expectation of one random variable s_1 given another random variable s_2. Formally, one could define this conditional expectation in two

ways ([20]). First, one could consider the expectation of s_1 given that s_2 takes on a particular value, say s. This conditional expectation, which is typically denoted as $E(s_1 \mid s_2 = s)$ is a function of the non-random variable s. Alternately, we can consider the conditional expectation of s_1 given the random variable s_2, which is the function $E(s_1 \mid s_2 = s)$ evaluated as $s = s_2$. Note that this expectation, typically denoted $E(s_1 \mid s_2)$, is itself a random variable—in partiular, a function of the random variable s_2. To avoid introducing dummy variables, we consistently use notation of the form $E(s_1 \mid s_2)$ in the thesis, though we sometimes view this quantity as a function of a deterministic variable rather than a random variable.

A.2 Discrete-Time Markov Processes

The models that we consider are instances of discrete-time stochastic processes, and in particular discrete-time Markov processes. A *discrete-time stochastic process* is a collection of random vectors indexed by a *discrete time index*, or set of non-negative integers. We refer to random vector indexed by the *time-step* or *time* k as the *state* of the process at time k. The book [113] provides a good introduction to discrete-time stochastic processes.

A discrete-time Markov proces is a discrete-time stochastic process, in which the dependence of the *next state* (i.e., the state at the next time) on the current and past history of the system is limited to the current state. That is, the conditional distribution of the next state given the entire state sequence up to the current time depends only on the current state.

A.3 Kronecker Products and Permutation Matrices

Kronecker products provide a useful notation for several applications, including in the solution of linear matrix equations and in the analysis of hierarchically-defined systems (see, e.g., [72]). A Kronecker product of two matrices is a larger matrix which contains all pairwise products of entries in the matrices. In particular, let's consider a matrix an $m \times n$ matrix $D = \begin{bmatrix} d_{11} & \dots & d_{1n} \\ \vdots & & \vdots \\ d_{m1} & \dots & d_{mn} \end{bmatrix}$, and an $p \times r$ matrix A. Then the Kronecker product of D

and A, denoted $D \otimes A$, is given by

$$D \otimes A = \begin{bmatrix} d_{11}A & \dots & d_{1n}A \\ \vdots & & \vdots \\ d_{m1}A & \dots & d_{mn}A \end{bmatrix}. \tag{A.1}$$

Note that $D \otimes A$ has dimension $mp \times nr$.

Chapter 4 of [72] list many properties of the Kronecker product. One property that we use in our development is the *mixed-product property*. The mixed-product property states that the matrix $(AB)\otimes(CD)$—where A, B, C, and D are appropriately-dimensioned matrices—can be rewritten as $(A \otimes C)(B \otimes D)$.

In expressions or equalities involving Kronecker products, operators that rearrange the entries in a vector or matrix are often useful. For instance, for vectors $\mathbf{s}_1$ and $\mathbf{s}_2$, consider the two Kronecker products $\mathbf{s}_1 \otimes \mathbf{s}_2$ and $\mathbf{s}_2 \otimes \mathbf{s}_1$. These two Kronecker products are each vectors that contain the same entries, but in different orders (except in the special case in which one or both of $\mathbf{s}_1$ and $\mathbf{s}_2$ are scalars). Thus, to relate the two Kronecker products, it is useful to construct an operator that rearranges the entries in a vector. It is well-known that multiplication by a *permutation matrix* serves to rearrange entries in vectors, or rows or columns in a matrix. A permutation matrix is one in which each row and column contains a single unity entry, and otherwise contains only zeroes. Permutation matrices are commonly used to map between different Kronecker product representations; the reader is asked to see [72] for details.

One further concept that is useful in some of our Kronecker-product formulations is *lexicographic ordering*. In particular, we sometimes find it useful to order items that are indexed by vectors of integers. More specifically, we assume that each item is indexed by a (say) m-component vector, and that each component of this index vector is an integer between (say) 1 and n. A group of items indexed in this fashion is said to be listed in lexicographic order if, in considering the index vector as a numeral in base n (with digits unconventionally labeled beginning with 1 rather than 0), the value of the indices increase down the list of items. For instance, say that 4 items are indexed by 2-element vectors, each of which can take on the values 1 and 2. Then the items are organized in lexicographic order if the indices down the list are $\{1,1\}$, $\{1,2\}$, $\{2,1\}$, and $\{2,2\}$, respectively.

Appendix B

Mappings between State and Moment Representations

At several points in the thesis, we argued that alternate representations for (first or higher) state and moment vectors could be related by linear transformations, but did not explicitly construct these transformations. In this appendix, we construct these mappings between state representations. To do so, it is helpful for us to develop some further notation concerning Kronecker products:

Definition B.1

Consider two partitioned vectors $\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_{n(x)} \end{bmatrix}$ *and* $\mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_{n(y)} \end{bmatrix}$. *Denote the length of vector* $\mathbf{x}_i$ *as* $m_i(x)$ *and denote the length of* $\mathbf{y}_i$ *as* $m_i(y)$. *Thus, the length of* $\mathbf{x}$ *is* $m(x) = \sum_{i=1}^{n(x)} m_i(x)$ *and the length of* $\mathbf{y}$ *is* $m(y) = \sum_{i=1}^{n(y)} m_i(y)$. *We define the* block Kronecker product *of* $\mathbf{x}[k]$ *and* $\mathbf{y}[k]$ *as follows:*

$$\mathbf{x} \boxtimes \mathbf{y} = \left[\begin{array}{c} \mathbf{x}_1 \otimes \mathbf{y}_1 \\ \vdots \\ \mathbf{x}_1 \otimes \mathbf{y}_{n(y)} \\ \hline \vdots \\ \hline \mathbf{x}_{n(x)} \otimes \mathbf{y}_1 \\ \vdots \\ \mathbf{x}_{n(x)} \otimes \mathbf{y}_{n(y)} \end{array}\right]. \tag{B.1}$$

As proven in the following theorem, the block Kronecker product of $\mathbf{x}$ and $\mathbf{y}$ is a permutation of the standard Kronecker product of $\mathbf{x}$ and $\mathbf{y}$.

Theorem B.2

There is a permutation matrix P such that $\mathbf{x} \boxtimes \mathbf{y} = P(\mathbf{x} \otimes \mathbf{y})$. This permutation matrix is precisely specified in the proof of the theorem.

Proof. First, consider the vector $\begin{bmatrix} \mathbf{x}_1 \otimes \mathbf{y}_1 \\ \vdots \\ \mathbf{x}_1 \otimes \mathbf{y}_{n(y)} \end{bmatrix}$. We claim that this vector is a permutation of the vector $\mathbf{x}_i \otimes \mathbf{y}$. To see why, let's rewrite $\mathbf{x}_i \otimes \mathbf{y}$ as $\begin{bmatrix} x_{i,1} \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_{n(y)} \end{bmatrix} \\ \vdots \\ x_{i,m_i(x)} \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_{n(y)} \end{bmatrix} \end{bmatrix}$, where $x_{i,j}$ is the jth element of the vector $\mathbf{x}_i$. Note that the vector $\mathbf{x}_i \otimes \mathbf{y}$ has length $m_i(x)m_i(y)$. Now consider multiplying $\mathbf{x}_i \otimes \mathbf{y}$ with a matrix $P_i(l,j)$, which has $m_j(y)$ rows and $m_i(x)m_i(y)$ columns. Assume that the block of columns of $P_i(l,j)$ indexed by $(l-1)m(y) + \sum_{k=1}^{j-1} m_k(y) + 1, \ldots, (l-1)m(y) + \sum_{k=1}^{j} m_k(y)$ is the identity matrix, while the remainder of the matrix is all zeros. It is easy to check that $P_i(l,j)(\mathbf{x}_i \otimes \mathbf{y})$ equals $x_{i,l}\mathbf{y}_j$. Assembling these equalities, we find that

$$P_i \mathbf{x}_i \otimes \mathbf{y} = \left[\begin{array}{c} x_{i,1}\mathbf{y}_1 \\ \vdots \\ x_{i,n}\mathbf{y}_1 \\ \hline \vdots \\ \cdots \\ x_{i,1}\mathbf{y}_{n(y)} \\ \vdots \\ x_{i,m_i(x)}\mathbf{y}_{n(y)} \end{array}\right] = \begin{bmatrix} x_i \otimes \mathbf{y}_1 \\ \vdots x_i \otimes \mathbf{y}_n, \end{bmatrix} \tag{B.2}$$

where $P_i \triangleq \left[\begin{array}{c} P_i(1,j) \\ \vdots \\ P_i(m_i(x),j) \\ \hline \\ \vdots \\ \\ \hline P_i(1,n(y)) \\ \vdots \\ P_i(m_i(x),n(y)) \end{array}\right]$. From our definition for $P_i(l,j)$, it is clear that P_i is a permutation matrix.

Assembling Equations B.2 for $i = 1, \ldots, n$ into a single column vector, we find that

$$\mathbf{x} \boxtimes \mathbf{y} = P \begin{bmatrix} \mathbf{x}_1 \otimes \mathbf{y} \\ \vdots \\ \mathbf{x}_{n(y)} \otimes \mathbf{y} \end{bmatrix} = P(\mathbf{x} \otimes \mathbf{y}), \tag{B.3}$$

where $P \triangleq \begin{bmatrix} P_1 & 0 & 0 & 0 \\ 0 & P_2 & 0 & 0 \\ \vdots & & \ddots & \\ & & & P_n \end{bmatrix}$. Thus, we have shown that $\mathbf{x} \boxtimes \mathbf{y} = P(\mathbf{x} \otimes \mathbf{y})$, where P is a permutation matrix. □

Theorem B.2 can be used to relate the rth state vector with the rth permuted state vector of an MLSN, as discussed in the following theorem.

Theorem B.3

The rth state vector $\mathbf{s}[k]^{\otimes r}$ is a permutation of the rth permuted state vector $\mathbf{s}_{[r]}[k]$. The permutation matrix P_r is explicitly constructed in the proof.

Proof. We prove the theorem by induction, in much the same way as in the appendix of [9]. First, note that $\mathbf{s}_{[1]}[k] = P_1 \mathbf{s}[k]$, where $P_1 = I_m$, by definition. Next, assume that $\mathbf{s}_{[i]}[k] = P_i s[k]^{\otimes i}$ for some permutation matrix P_i. Now consider $\mathbf{s}_{[i+1]}[k]$. Note that $\mathbf{s}_{[i+1]}[k] = \mathbf{s}[k] \boxtimes \mathbf{s}_[i][k]$, where $\mathbf{s}[k]$ is partitioned into status vectors, and $\mathbf{s}_[i][k]$ is partitioned into

ith joint status vectors. From Theorem B.2, we know that $\mathbf{s}[k] \boxtimes \mathbf{s}_{[i]}[k]$ can be written as $\widehat{P}_i(s[k] \otimes \mathbf{s}_{[i]}[k])$ for a permutation matrix $\widehat{P}_i$, and we know how to construct $\widehat{P}_i$. Thus,

$$\begin{aligned} \mathbf{s}_{[i+1]}[k] &= \widehat{P}_i(s[k] \otimes \mathbf{s}_{[i]}[k]) && \text{(B.4)} \\ &= \widehat{P}_i(s[k] \otimes (P_i \mathbf{s}[k]^{\otimes i})) && \text{(B.5)} \\ &= (\widehat{P}_i \otimes P_i)\mathbf{s}[k]^{\otimes(i+1)}. && \text{(B.6)} \end{aligned}$$

$\widehat{P}_i \otimes P_i$ is the Kronecker product of two permutation matrices and so is itself a permutation matrix. Thus, we see that $\mathbf{s}_{[i+1]}[k] = P_{i+1}\mathbf{s}[k]^{\otimes(i+1)}$ for the permutation matrix $P_{i+1} \triangleq (\widehat{P}_i \otimes P_i)$. Thus, we have shown how to construct the permutation matrix relating the rth state vector to the rth permuted state vector. □

The algorithm for computing MLSS statistics across time-steps invokes a linear relation between the $r_1 + r_2$th extended state vector and the Kronecker product of the r_1th and r_2th extended state vectors. In the following theorem, we explicitly construct this linear relation.

Theorem B.4

The $(r_1 + r_2)$th extended state vector can be expressed as $\mathbf{s}_{(r_1+r_2)}[k] = L_{r_1,r_2}(\mathbf{s}_{(r_1)}[k] \otimes \mathbf{s}_{(r_2)}[k])$. In proving this relation, we recursively compute L_{r_1,r_2}.

Proof. Let's first relate $\mathbf{s}_{(r_1+r_2)}[k]$ to $\mathbf{s}_{(r_1)}[k] \boxtimes \mathbf{s}_{(r_2)}[k]$, where $\mathbf{s}_{(r_1)}[k]$ is partitioned as $\begin{bmatrix} \mathbf{s}[k]^{\otimes r_1} \\ \hline \vdots \\ \hline \mathbf{s}[k]^{\otimes 1} \\ 1 \end{bmatrix}$, and $\begin{bmatrix} \mathbf{s}[k]^{\otimes r_2} \\ \hline \vdots \\ \hline \mathbf{s}[k]^{\otimes 1} \\ 1 \end{bmatrix}$. This block Kronecker product comprises a listing of $(r_1 + 1)(r_2 + 1)$ Kronecker products, each of which are Kronecker powers of $\mathbf{s}[k]$. Let's consider a matrix

$L_{r_1,r_2}(i)$, with as many columns as there are entries in $\mathbf{s}_{(r_1)}[k] \boxtimes \mathbf{s}_{(r_2)}[k]$, and $m^{r_1+r_2-i+1}$ rows. Assume that the columns of $L_{r_1,r_2}(i)$ are partitioned to correspond with the partitions of $\mathbf{s}_{(r_1)}[k]$ and $\mathbf{s}_{(r_2)}[k]$. Let the partition of $L_{r_1,r_2}(i)$ corresponding to block $r_1(i-1)+1$ of $\mathbf{s}_{(r_1)}[k] \boxtimes \mathbf{s}_{(r_2)}[k]$ (i.e., the block which is the Kronecker product of the ith partition of $\mathbf{s}_{(r_1)}[k]$ with the first partition of $\mathbf{s}_{(r_2)}[k]$) be the identity matrix. Assume that the remaining entries of $L_{r_1,r_2}(i)$ are 0. Then

$$L_{r_1,r_2}(i)\mathbf{s}_{(r_1)}[k] \boxtimes \mathbf{s}_{(r_2)}[k] = s[k]^{(r_1+r_2-i+1)} \tag{B.7}$$

.

Also, consider a matrix $L_{r_1,r_2}(end)$ with $\sum_{j=0}^{r_2} m^j$ rows, and as many columns as there are entries in $\mathbf{s}_{(r_1)}[k] \boxtimes \mathbf{s}_{(r_2)}[k]$. Assume that the columns of $L_{r_1,r_2}(end)$ are partitioned in the same way as the entries in $\mathbf{s}_{(r_1)}[k] \boxtimes \mathbf{s}_{(r_2)}[k]$. Assume that the final r_2+1 partitions of $L_{r_1,r_2}(end)$ (which correspond to the prodicts of the final entry of $\mathbf{s}_{(r_1)}[k]$ with all partitions of $\mathbf{s}_{(r_2)}[k]$, and total $\sum_{j=0}^{r_2} m^j$ columns) are together the identity matrix. Also assume that all other entries of $L_{r_1,r_2}(end)$ are 0. Then

$$L_{r_1,r_2}(end)\mathbf{s}_{(r_1)}[k] \boxtimes \mathbf{s}_{(r_2)}[k] = \begin{bmatrix} \mathbf{s}_{(r_2)}[k] \\ \vdots \\ \mathbf{s}[k] \\ 1 \end{bmatrix} = \mathbf{s}_{(r_2)}[k] \tag{B.8}$$

By concatenating the products specified by Equation B.7, for $1 \leq i \leq r_2$, as well as the product specified in Equation B.8, into a single matrix-vector product, we find that

$$\begin{bmatrix} L_{(r_1,r_2)}(1) \\ \vdots \\ L_{(r_1,r_2)}(r_1) \\ L_{(r_1,r_2)}(end) \end{bmatrix} \mathbf{s}_{(r_1)}[k] \boxtimes \mathbf{s}_{(r_2)}[k] = \mathbf{s}_{(r_1+r_2)}[k] \tag{B.9}$$

Finally, from Theorem B.2, $\mathbf{s}_{(r_1)}[k] \boxtimes \mathbf{s}_{(r_2)}[k] = P(\mathbf{s}_{(r_1)}[k] \otimes \mathbf{s}_{(r_2)}[k])$ for a permutation matrix P. Thus, substituting, we find that

$$\mathbf{s}_{(r_1+r_2)}[k] = L_{r_1,r_2}(\mathbf{s}_{(r_1)}[k] \otimes \mathbf{s}_{(r_2)}[k]), \tag{B.10}$$

where

$$L_{r_1,r_2} = \begin{bmatrix} L_{(r_1,r_2)}(1) \\ \vdots \\ L_{(r_1,r_2)}(r_1) \\ L_{(r_1,r_2)}(end) \end{bmatrix} P(\mathbf{s}_{(r_1)}[k] \otimes \mathbf{s}_{(r_2)}[k]). \tag{B.11}$$

Thus, we have proved the theorem, and shown how to construct the linear transformation matrix L_{r_1,r_2}.

□

In Section 4.2.6, we argued that the rth primary state vector $\widehat{\mathbf{s}}_{[r]}[k]$ can be written as a linear function of the rth state vector $\mathbf{s}[k]^{\otimes r}$. (Recall that $\widehat{\mathbf{s}}_{[r]}[k]$ contains state variable products for each rth grouping primary grouping of state variables—i.e., each ordered list of r state variables for which the indices of the state variables are increasing along the list. These groupings ae assembled in lexicographic ordering in forming $\widehat{\mathbf{s}}_{[r]}[k]$.)

Theorem B.5
In this theorem, we specify the linear transformation L_r for which $\widehat{\mathbf{s}}_{[r]}[k] = L_r\mathbf{s}[k]^{\otimes r}$.

Proof. We compute this linear transform using an iterative process. To do so, first note that $\widehat{\mathbf{s}}_{[1]}[k] = L_1\mathbf{s}[k]^{\otimes 1}$, where $L_1 = I_m$. Next, assume that we have found the L_i for which $\widehat{\mathbf{s}}_{[i]}[k] = L_i\mathbf{s}[k]^{\otimes i}$. To develop an expression for $\widehat{\mathbf{s}}_{[i+1]}[k]$, consider $\mathbf{s}[k] \otimes \widehat{\mathbf{s}}_{[i]}[k]$. For convenience, partition the vector $\widehat{\mathbf{s}}_{[i]}[k]$ as $\begin{bmatrix} \widehat{\mathbf{s}}_{<1>}[k] \\ \vdots \\ \widehat{\mathbf{s}}_{<m>}[k] \end{bmatrix}$, where $\widehat{\mathbf{s}}_{<j>}[k]$ comprises the elements of

$\widehat{\mathbf{s}}_{[i]}[k]$ with ith primary groupings that begin with the state variable j. In this notation,

$$\mathbf{s}[k] \otimes \widehat{\mathbf{s}}_{[i]}[k] = \begin{bmatrix} & \begin{bmatrix} \widehat{\mathbf{s}}_{<1>}[k] \\ \vdots \\ \widehat{\mathbf{s}}_{<m>}[k] \end{bmatrix} \\ s_1[k] & \\ & \vdots \\ & \begin{bmatrix} \widehat{\mathbf{s}}_{<1>}[k] \\ \vdots \\ \widehat{\mathbf{s}}_{<m>}[k] \end{bmatrix} \\ s_m[k] & \end{bmatrix} \tag{B.12}$$

Also, note that $\widehat{\mathbf{s}}_{[i+1]}[k]$ can be rewritten in this notation, as

$$\widehat{\mathbf{s}}_{[i+1]}[k] = \begin{bmatrix} & \begin{bmatrix} \widehat{\mathbf{s}}_{<1>}[k] \\ \vdots \\ \widehat{\mathbf{s}}_{<m>}[k] \end{bmatrix} \\ s_1[k] & \\ & \begin{bmatrix} \widehat{\mathbf{s}}_{<2>}[k] \\ \vdots \\ \widehat{\mathbf{s}}_{<m>}[k] \end{bmatrix} \\ s_2[k] & \\ & \vdots \\ & \begin{bmatrix} \widehat{\mathbf{s}}_{<m>}[k] \end{bmatrix} \\ s_m[k] & \end{bmatrix} \tag{B.13}$$

To specify a map between $\mathbf{s}[k] \otimes \widehat{\mathbf{s}}_{[i]}[k]$ and $\widehat{\mathbf{s}}_{[i+1]}[k]$, we note that

$$s_j[k] \begin{bmatrix} \widehat{\mathbf{s}}_{<j>}[k] \\ \vdots \\ \widehat{\mathbf{s}}_{<m>}[k] \end{bmatrix} = L_i(j) s_j[k] \begin{bmatrix} \widehat{\mathbf{s}}_{<1>}[k] \\ \vdots \\ \widehat{\mathbf{s}}_{<m>}[k] \end{bmatrix}, \tag{B.14}$$

where $L_i(j)$ has the form $[0, \mathbf{I}]$ and so truncates $\begin{bmatrix} \widehat{\mathbf{s}}_{<1>}[k] \\ \vdots \\ \widehat{\mathbf{s}}_{<m>}[k] \end{bmatrix}$ to $\begin{bmatrix} \widehat{\mathbf{s}}_{<j>}[k] \\ \vdots \\ \widehat{\mathbf{s}}_{<m>}[k] \end{bmatrix}$.

Assembling equations of the form B.14, we find that

$$\widehat{\mathbf{s}}_{[i+1]}[k] = \begin{bmatrix} L_i(1) & 0 & \dots \\ 0 & \ddots & 0 \\ \vdots & 0 & L_i(m) \end{bmatrix} \mathbf{s}[k] \otimes \widehat{\mathbf{s}}_{[i]}[k]. \tag{B.15}$$

From our assumption, $\widehat{\mathbf{s}}_{[i]}[k] = L_i \mathbf{s}[k]^{\otimes i}$, so

$$\begin{aligned} \widehat{\mathbf{s}}_{[i+1]}[k] &= \begin{bmatrix} L_i(1) & 0 & \dots \\ 0 & \ddots & 0 \\ \vdots & 0 & L_i(m) \end{bmatrix} (\mathbf{s}[k] \otimes L_i \mathbf{s}[k]^{\otimes i}) \\ &= L_{i+1} \mathbf{s}[k]^{\otimes (i+1)}, \end{aligned} \tag{B.16}$$

where

$$L_{i+1} = \begin{bmatrix} L_i(1) & 0 & \dots \\ 0 & \ddots & 0 \\ \vdots & 0 & L_i(m) \end{bmatrix} (I_m \otimes L_i). \tag{B.17}$$

Thus, we have constructed the mapping L_r such that $\widehat{\mathbf{s}}_{[r]}[k] = L_r \mathbf{s}[k]^{\otimes r}$, through an iterative process. □

In Section 4.2.6, we also argued that the rth state vector $\mathbf{s}[k]^{\otimes r}$ can be written as a linear function of the rth primary state vector $\widehat{\mathbf{s}}_{[r]}[k]$—i.e., that $\mathbf{s}[k]^{\otimes r} = M_{r,r} \widehat{\mathbf{s}}_{[r]}[k]$ for some $M_{r,r}$.

Theorem B.6
In this theorem, we specify the linear transformation $M_{r,r}$ for which $\widehat{\mathbf{s}}_{[r]}[k] = M_{r,r} \mathbf{s}[k]^{\otimes r}$.

Proof. Like the mapping L_r from $\mathbf{s}[k]^{\otimes r}$ to $\widehat{\mathbf{s}}_{[r]}[k]$, the reverse mapping $M_{r,r}$ can be constructed iteratively. However, this iterative construction of $M_{r,r}$ turns out to require rather unpleasant notation; we believe that an explicit, non-iterative construction of $M_{r,r}$ is clearer, so we present this direct construction. We note that the other mappings described in this appendix can also be generated through direct construction, and intend for this development to clarify this approach.

The number of rows in $M_{r,r}$ is the length of $\mathbf{s}[k]^{\otimes r}$, or m^r. The number of columns in $M_{r,r}$ is the length of $\widehat{\mathbf{s}}_{[r]}[k]$, which is equal to the number of rth primary groupings, or $\binom{m+r-1}{r}$. For convenience, let us first index the rows and columns in lexicographic order according to their groupings, and specify the matrix $M_{r,r}$ in this notation.

Consider the element of $\mathbf{s}[k]^{\otimes r}$ indexed by $(i_1, \ldots, i_r)$. This element of $\mathbf{s}[k]^{\otimes r}$ is also necessarily an element of $\widehat{\mathbf{s}}_{[r]}[k]$. In particular, this element is identical to the element of $\widehat{\mathbf{s}}_{[r]}[k]$ indexed by $(\widehat{i}_1, \ldots, \widehat{i}_r)$, where $(\widehat{i}_1, \ldots, \widehat{i}_r)$ is constructed by sorting $(i_1, \ldots, i_r)$ in increasing order. From now on, we use the notation $(\widehat{i}_1, \ldots, \widehat{i}_r) = prim(i_1, \ldots, i_r)$ to represent the sorting of the grouping $(i_1, \ldots, i_r)$ into a primary grouping. Since the element $(i_1, \ldots, i_r)$ of $\mathbf{s}[k]^{\otimes r}$ is equal to the element $(\widehat{i}_1, \ldots, \widehat{i}_r)$ of $\widehat{\mathbf{s}}_{[r]}[k]$, row $(i_1, \ldots, i_r)$ of $M_{r,r}$ can ve constructed to have a single unity entry at location $(\widehat{i}_1, \ldots, \widehat{i}_r)$, and to be zero otherwise. Thus, we can construct each row of $M_{r,r}$ so that

$$\mathbf{s}[k]^{\otimes r} = M_{r,r}\widehat{\mathbf{s}}_{[r]}[k]. \tag{B.18}$$

In order to simplify the construction of $M_{r,r}$, it is useful tohave a scalar numeric index for the columns of $M_{r,r}$. (In particular, we can then contruct each row of $M_{r,r}$ by identifying the numeric column index of the unity entry in the row.

To use a numeric index rather than a grouping index for the columns of $M_{r,r}$, recall that the columns are indexed by primary groupings, arranged in lexicographic order. Thus, the number of the column indexed by the primary grouping $(\widehat{i}_1, \ldots, \widehat{i}_r)$ can be found by computing the number of primary groupings which fall before this grouping in lexicographic order. The number of such groupings is $\sum_{\alpha_1=1}^{\widehat{i}_1} \sum_{\alpha_2=\alpha_1}^{\widehat{i}_2} \cdots \sum_{\alpha_r=\alpha_{r-1}}^{\widehat{i}_r-1} 1$, so the numeric index for the grouping index $(\widehat{i}_1, \ldots, \widehat{i}_r)$ is $\left(\sum_{\alpha_1=1}^{\widehat{i}_1} \sum_{\alpha_2=\alpha_1}^{\widehat{i}_2} \cdots \sum_{\alpha_r=\alpha_{r-1}}^{\widehat{i}_r-1} 1\right) + 1$. □

Consider an influence status vector $\mathbf{s}_{\mathbf{w}}[k]$ for an rth grouping $\mathbf{w}$, as well as the influence status vector $\mathbf{s}_{\widehat{\mathbf{w}}}[k]$ for its essential site list $\widehat{\mathbf{w}}$. We argued in Section 7.3.2.1 that $\mathbf{s}_{\mathbf{w}}[k]$ can be written as $Q(w)\mathbf{s}_{\widehat{\mathbf{w}}}[k]$ for some mapping matrix $Q(\mathbf{w})$.

Theorem B.7

Here, we explicitly construct the matrix $Q(\mathbf{w})$ such that $\mathbf{s}_{\mathbf{w}}[k] = Q(w)\mathbf{s}_{\widehat{\mathbf{w}}}[k]$.

Proof. Let's say that the essential site list $\widehat{\mathbf{w}}$ has a length of $\widehat{r}$ (i.e., contains $\widehat{r}$ sites). Note that $\widehat{r} \leq r$. Denote the elements of $\mathbf{w}$ by $w_1, \ldots, w_r$, and the elements of $\widehat{\mathbf{w}}$ by $\widehat{w}_1, \ldots, \widehat{w}_r$. In this notation, $Q(\mathbf{w})$ is a matrix with $\prod_{i=1}^{r} m_{w_i}$ rows and $\prod_{i=1}^{\widehat{r}} m_{\widehat{w}_i}$ columns. We index the rows of $Q(\mathbf{w})$, and the elements of $\mathbf{s}_{\mathbf{w}}[k]$, lexicographically according to a length r vector $(\alpha_1, \ldots, \alpha_r)$, where $\alpha_i in 1, \ldots, m_{w_i}$. It's easy to check that the element of $\mathbf{s}_{\mathbf{w}}[k]$ indexed by $(\alpha_1, \ldots, \alpha_r)$ is unity if and only if site w_1 is in status α_1 at time k, site w_2 is in status w_2 at time k, etc, and is zero otherwise. In similar fashion, we index the columns of $Q(\mathbf{w})$ and the elements of $\mathbf{s}_{\widehat{\mathbf{w}}}[k]$, lexicographically according to a length $\widehat{r}$ vector $(\widehat{\alpha}_1, \ldots, \widehat{\alpha}_{\widehat{r}})$, where $\widehat{\alpha}_i \in 1, \ldots, m_{\widehat{w}_i}$.

A little further notation is required in order to construct $Q(\mathbf{w})$. Specifically, consider each site $\widehat{w}_i$, $1 \leq i \leq \widehat{r}$, in the essential site list. Note that there is at least one site in the grouping $\mathbf{w}$ which is $\widehat{w}_i$. In general, let $\mathcal{G}(i)$ list the locations along the vector $\mathbf{w}$ such that the site specified at the location is $\widehat{w}_i$.

Finally, to construct $Q(\mathbf{w})$, consider the case where the element $(\widehat{\alpha}_1, \ldots, \widehat{\alpha}_{\widehat{r}})$ of $\mathbf{s}_{\widehat{\mathbf{w}}}[k]$ is 1. Then site $\widehat{w}_i$ is in status $\widehat{\alpha}_i$, for $1 \leq i \leq \widehat{r}$. Also, exactly one entry of the vector $\mathbf{s}_{\mathbf{w}}[k]$ is 1 (and the remaining elements are 0). This unity entry is indexed by $(\alpha_1, \ldots, \alpha_r)$, where, for each $1 \leq i \leq \widehat{r}$ and $j \in \mathcal{G}(i)$, it is true that $\alpha_j = \widehat{\alpha}_i$ for $1 \leq i \leq \widehat{r}$. That is, we require that each site in $\mathbf{w}$ equal to $\widehat{w}_i$ has the same status $\widehat{\alpha}_i$ as $\widehat{w}_i$. Thus, by setting column $(\widehat{\alpha}_1, \ldots, \widehat{\alpha}_{\widehat{r}})$ of $Q(\mathbf{w})$ to 0 except at element $(\alpha_1, \ldots, \alpha_r)$, which we set to 1, we can ensure that $\mathbf{s}_{\mathbf{w}}[k] = Q(\mathbf{w})\mathbf{s}_{\widehat{\mathbf{w}}}[k]$.

We note that a bit of work is required to assign numeric, rather than vector, indices to the elements of $Q(\mathbf{w})$. We leave this (straightforward but somewhat tedious) mapping to the reader.

□

Appendix C

Scope of the First-Moment Recursion, Vector Case

We consider the proof of the result introduced in Section 2.3, in the case of m-component state vector $\mathbf{s}[k]$. Conceptually, the proof in the vector case is quite similar to the scalar case, so we only give an outline, and point out significant differences between the scalar and vector case.

Recall that we assume the mean of the state $\mathbf{s}[k]$ of a Markov process at a time k is known, but the distribution of $\mathbf{s}[k]$ is otherwise arbitrary. We wish to prove that, in order for the expected state $E(\mathbf{s}[k+1])$ to be uniquely calculable, $E(\mathbf{s}[k+1] \,|\, \mathbf{s}[k])$ must be affine with respect to $\mathbf{s}[k]$—i.e., $E(\mathbf{s}[k+1] \,|\, \mathbf{s}[k]) = H\mathbf{s}[k] + \mathbf{b}$ for some H and $\mathbf{b}$. We can prove this statement, through the following steps:

- First, we consider two different distributions for $\mathbf{s}[k]$, which both have the specified mean $E(\mathbf{s}[k]) \stackrel{\triangle}{=} \bar{\mathbf{s}}$. One of the distributions that we consider is concentrated solely at $\bar{\mathbf{s}}$. The second distribution that we consider is concentrated at $m+1$ points (where m is the number of elements in the state vector) whose *convex hull* in R^m contains $\bar{\mathbf{s}}$ strictly in its interior. We can straightforwardly check that probabilities can be assigned to each of these values for $\bar{\mathbf{s}}$, such that the expected value of $\mathbf{s}[k]$ is $\bar{\mathbf{s}}$. In order for $E(\mathbf{s}[k+1])$ to be calculable, we need that the expected value of the next state in both scenarios to be equal. Equating these expectations, we obtain a linear relation between the conditional expectation of the next-state given $\bar{\mathbf{s}}$ and the conditional expectations of the next-state given each of the other $m+1$ points. That is, in an $m+1$ dimensional axis system consisting of the m state vectors and the conditional expectation of the next state given the current one, we find that these $m+2$ points lie on the same hyperplane.

- Our remaining task is to prove that the conditional expectation for $\mathbf{s}[k+1]$ given any other value for $\mathbf{s}[k]$ lies on the same hyperplane. We use the same general strategy as in the scalar case: we consider a distribution for $\mathbf{s}[k]$ that is concentrated on an arbitrary

point, as well as m of the $m+1$ points considered in second scenario above. Unlike the scalar case, it takes a bit of effort to see that we can choose m of the $m+1$ points so that the new set of $m+1$ points (i.e., these m points and the arbitrarily chosen point) have a distribution concentrated on them such that the mean is $\bar{\mathbf{s}}$. In particular, some thought shows that, for any new point, m of the original points can be chosen so that $\bar{\mathbf{s}}$ is in the convex hull of these points, and hence that we can construct a distribution for $\mathbf{s}[k]$ concentrated on the points that has mean $\bar{\mathbf{s}}$. We then equate the expected value for $\mathbf{s}[k+1]$ if $\mathbf{s}[k]$ has this distribution with the expected value for $\mathbf{s}[k+1]$ if $\mathbf{s}[k]$ is concentrated at its mean $\bar{\mathbf{s}}$. Finally, by invoking the linear relation between the conditional expectations that was found in the bullet above, and doing some algebra, we can prove that the condtional expectation for $\mathbf{s}[k+1]$ given an arbitrary value for $\mathbf{s}[k]$ must lie on the same hyperplane, and hence the result is proven.

Bibliography

[1] G. A. Ackerson and K. S. Fu. On the state estimation in switching environments. *IEEE Transactions on Automatic Control*, AC-15:10–17, Feb. 1970.

[2] N. I. Akhiezer. *The Classical Moment Problem and Some Related Questions in Analysis (trans. N. Kemmer)*. Hafner, New York, 1965.

[3] E. Altman, F. Baccelli, and J. C. Bolot. Discrete-time analysis of adaptive rate control mechanisms. *IFIP Transactions, High Speed Networks and Their Performance*, Oct. 1993.

[4] E. Altman, F. Baccelli, and J. C. Bolot. Discrete-time analysis of rate-based feedback control mechanisms. *http://www-sop.inria.fr/rodeo/personnel/bolot/papers.html*, 1997.

[5] E. Altman and T. Basar. Multiuser rate-based flow control. *IEEE Transactions on Communications*, 46(7):940–949, Jul. 1998.

[6] E. R. Mueller amd G. B. Chatterji. Analysis of aircraft arrival and departure delay characteristics. *Aircraft Technology, Integration, and Operations Technical Forum, Los Angeles, CA*, Oct. 2002.

[7] K. Andersson, F. Carr, E. Feron, and W. Hall. Analysis and modeling of ground operations at hub airports. *3rd USA-Europe Air Traffic Management Seminar, Naples, Italy*, Jun. 2000.

[8] P. F. Arndt, C. B. Burge, and T. Hwa. Dna sequence evolution with neighbor-dependent mutation. *Proceedings of the 6th International Conference on Computational Molecular Biology (RECOMB02), G. Meyer et al eds.*, pages 32–38, 2002.

[9] C. Asavathiratham. *The Influence Model: A Tractable Representation for the Dynamics of Networked Markov Chains, Ph.D. Thesis.* EECS Department, Massachusetts Institute of Technology, Oct. 2000.

[10] C. Asavathiratham, S. Roy, B. C. Lesieutre, and G. C. Verghese. The influence model. *IEEE Control Systems Magazine*, Dec. 2001.

[11] M. Athans, R. Ku, and S. B. Gershwin. Uncertainty threshold principle–some fundamentals of optimal decision-making under dynamic uncertainty. *IEEE Transactions on Automatic Control*, 22(3):491–495, 1977.

[12] K. Atif and B. Plateau. Stochastic automata network for modeling parallel systems. *IEEE Transactions on Software Engineering*, 17(10):1093–1108, Oct. 1991.

[13] E. Bacarelli and R. Cusani. Recursive Kalman-type optimal estimation and detection of hidden Markov chains. *Signal Processing*, 51:55–64, 1996.

[14] A. Baiocchi, N. B. Melazzi, M. Listani, A. Roveri, and R. Winkler. Loss performance analysis of an atm multiplexer loaded with high-speed on off sources. *IEEE Journal on Selected Areas of Communications*, 9:388–393, Apr. 1991.

[15] P. Bak, C. Tang, and N. Wiesenfield. Self-organized criticality. *Physical Review A*, 38:364–374, 1998.

[16] P. Baldi and S. Brunak. *Bioinformatics: the Machine Learning Approach.* MIT Press, Boston, 2001.

[17] S. Basu, T. Choudhury, B. Clarkson, and A. Pentland. Learning human interactions with the influence model. *MIT Media Lab Vision and Modeling TR#539*, June 2001.

[18] M. Beckerman. *Adaptive Cooperative Systems.* John Wiley and Sons, New York, 1997.

[19] C. Berge. *Graphs and Hypergraphs.* North-Holland, New York, 1976.

[20] D. P. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, Belmont, MA, 1995.

[21] D. P. Bertsekas and J. N. Tsitsiklis. *Introduction to Probability.* Athena Scientific, 2002.

[22] D. Bertsimas and S. Stock-Patterson. The air traffic management problem with enroute capacities. *Operations Research*, May-Jun. 1998.

[23] L. Bianco, P. Dell'Olmo, and ed. A. R. Odoni. *New Concepts and Methods in Air Traffic Management.* Springer, New York, 2001.

[24] P. Bremaud. *Point Processes and Queues: Martingale Dynamics.* Springer-Verlag, New York, 1981.

[25] P. Bremaud. *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues.* Springer-Verlag, New York, 1999.

[26] J. W. Brewer. Kronecker products and matrix calculus in system theory. *IEEE Transactions on Circuits and Systems*, 25:772–781, Sep. 1978.

[27] J. M. Carlson and J. Doyle. Highly-optimized tolerance: a mechanism for power laws in designed systems. *Physical Review E*, 60:1412–1427, 1999.

[28] D. E. Catlin. *Estimation, Control, and the Discrete Kalman Filter.* Springer-Verlag, New York, 1989.

[29] H. Chen and Y. Zhao. A new queueing model for aircraft landing process. *AIAA, GNC, AFM, and MST Conference and Exhibit, New Orleans, LA*, Aug. 1997.

[30] W. K. Ching. Markov-modulated poisson processes for multi-location inventory problems. *International journal of production economics*, 52:217–233, Nov. 1997.

[31] W. K. Ching, R. H. Chan, and X. Y. Zhou. Circulant preconditioners for Markov-modulated poisson processes and their applications to manufacturing systems. *SIAM Journal on Matrix Analysis and Applications*, 18:464–481, Apr. 1997.

[32] H. J. Chizeck, A. S. Willsky, and D. Castanon. Discrete-time Markovian-jump quadratic optimal control. *International Journal of Control*, 43(1):213–231, Oct. 1991.

[33] N. J. Chizeck and Y. Ji. Optimal quadratic control of jump linear systems with gaussian noise in discrete-time. *Proceedings of the 27th IEEE Conference on Decision and Control*, pages 1989–1999, Dec. 1988.

[34] P. Clifford and A. Sudbury. A model for spatial conflict. *Biometrika*, 60(3):581–588, 1973.

[35] E. O. R. Costa and E. O. H. Filho. Discrete-time constrained quadratic control of Markovian jump linear systems. *Proceedings of the 35th IEEE Conference on Decision and Control*, 2:1763–1764, Dec. 1996.

[36] O. L. V. Costa. Linear minimum mean square error estimation for discrete-time Markovian jump linear systems. *IEEE Transactions on Automatic Control*, 39:1685–1689, Aug. 1994.

[37] O. L. V. Costa and R. P. Marques. Robust h-2-control for discrete-time Markovian jump linear systems. *International Journal of Control*, pages 11–21, Jan. 2000.

[38] J. T. Cox. Coalescing random walks and voter model consensus times on the torus in z^d. *Annals of Probability*, 17(4):1333–1336, 1989.

[39] J. T. Cox and A. Greven. On the long term behavior of some finite particle systems. *Probability Theory and Related Fields*, 85:195–237, 1990.

[40] H. Daduna. *Queueing Networks with Discrete Time Scale: Explicit Expressions for the Steady-State Behavior of Discrete Time Stochastic Networks*. Lecture notes in computer science. Springer, New York, 2001.

[41] C. F. Daganzo. Requiem for second-order fluid approximations of traffic flow. *Transportation Research Part B*, 29(4):277–286, Aug. 1995.

[42] M. Dahleh, M. A. Dahleh, and G. C. Verghese. *Lecture Notes: Dynamic Systems and Control*. MIT Course Notes Office, 2003.

[43] J. Dai and S. P. Meyn. Stability and convergence of moments for networks and their fluid approximations. *IEEE Transactions on Automatic Control*, Nov. 1995.

[44] W. L. Dekoning. Infinite horizon optimal-control of discrete-time-systems with stochastic parameters. *Automatica*, 18(4):443–453, 1982.

[45] D. D. Dimitrijevic. Inferring most likely queue length from transactional data. *Operations Research Letters*, 19:191–199, Oct. 1996.

[46] P. Donnelly and D. Welsh. Finite particle systems and infection models. *Mathematical Proceedings of the Cambridge Philosophical Society*, 94:167–182, 1983.

[47] F. Dufour and R. J. Elliot. Adaptive control of linear systems with Markov perturbations. *IEEE Transactions on Automatic Control*, 43:351–372, Mar. 1997.

[48] R. Dugad and U. B. Desai. A tutorial on hidden Markov models. *Technical Report No.: SPANN-96.1*, May 1996.

[49] R. Durrett. An introduction to infinite particle systems. *Stochastic Processes and their Applications*, 11:109–150, 1981.

[50] J. E. Evans. Tactical weather decision support to complement 'strategic' traffic flow management for convective weather. 4^{th} *USA/Europe Air Traffic Management R&D Seminar*, Dec. 2001.

[51] Y. Fang and K. A. Loparo. Stabilization of continuous-time jump linear systems. *IEEE Transactions on Automatic Control*, 47:1590–1603, Oct. 2002.

[52] Y. Fang, K. A. Loparo, and X. Feng. Modeling issues for the control systems with communication delays. *Ford Motor Co., SCP Research Report*, Oct. 1991.

[53] R. G. Gallager. *Discrete Stochastic Processes*. Kluwer Academic Publishers, Boston, 1996.

[54] C. W. Gardiner. *Handbook of Stochastic Methods for Physics, Chemistry, and the Natural Sciences*. Springer, Berlin, 1998.

[55] E. Gelenbe and G. Pujolle. *Introduction to Queueing Networks, 2nd ed.* Wiley, 1998.

[56] Z. Ghahramani. Learning dynamic Bayesian networks. *Adaptive Processing of Sequences and Data Structures*, C. L. Giles and M. Gori (eds.), pages 168–197, 1998.

[57] Z. Ghahramani and M. I. Jordan. Factorial hidden Markov models. *Machine Learning*, 1997.

[58] R. J. Glauber. Time-dependent statistics of the ising model. *Journal of Mathematical Physics*, 4:294–307, 1963.

[59] C. Gomez-Uribe. *Partially Observed Influence Models*. S.M. Thesis, EECS MIT, June 2003.

[60] D. Griffeath. The basic contact process. *Stochastic Processes and their Applications*, 11:151–185, 1981.

[61] M. Grigoriu and F. Waisman. Linear systems with polynomials of filtered Poisson processes. *Probabilistic Engineering Mechanics*, 12:97–103, 1997.

[62] G. Grimmett and D. Stirzaker. *Probability and Random Processes, 3rd ed.* Oxford University Press, Oxford, 2001.

[63] D. Gross and C. M. Harris. *Fundamentals of Queueing Theory*. Wiley, New York, 1998.

[64] P. Guttorp. *Stochastic Modeling of Scientific Data*. Chapman and Hall, New York, 1995.

[65] W. Tan H. Wu. Modelling the hiv epidemic: a state-space approach. *Mathematical and Computer Modelling*, 32:197–215, 2000.

[66] B. Hajek, K. Mitzel, and S. Yang. Paging and registration in cellular networks: Jointly optimal policies and an iterative algorithm. *IEEE INFOCOM*, 2003.

[67] J. M. Harrison. The heavy-traffic approximation for single server queues in series. *Journal of Applied Probability*, 10:613–629, 1973.

[68] M. Hasegawa, H. Kishino, and T. Yano. Dating the human-ape split by a molecular clock of mitochondrial dna. *Journal of Molecular Evolution*, 22:160–174, 1985.

[69] N. Hemachandra and Y. Narahari. A mathematical programming approach to optimal Markovian switching of poisson arrival streams to queueing systems. *Queueing Systems*, 36(4):443–461, 2000.

[70] M. Hilleges and W. Weidlich. A phenomenological model for dynamic traffic flow in networks. *Transportation Research Part B*, 29(6):407–431, Dec. 1995.

[71] R. A. Holley and T. M. Liggett. Ergodic theorems for weakly interacting infinite systems and the voter models. *Annals of Probability*, 3:643–663, 1975.

[72] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[73] Y. M. Ioannides. Topologies of social interactions. *Tufts Working Paper*, June 2003.

[74] K. A. Janes, S. Gaudet, J. Kelly, P. K. Sorger, M. B. Yaffe, and D. A. Lauffenburger. Multidimensional quantification and analysis of the apoptosis-survival 'decision' process in ht-29 cells. *CSBi Conference Poster Session*, January 2003.

[75] N. L. Johnson, S. Kotz, and A. W. Kemp. *Univariate Discrete Distributions, 2nd ed.* Wiley, New York, 1992.

[76] T. Kailath. A view of three decades of linear filtering theory. *IEEE Transactions on Information Theory*, 20(2):145–181, Mar. 1974.

[77] R. E. Kalman. A new approach to linear filter and prediction problems. *ASME Transactions*, 82D, 1960.

[78] T. Katayama. Matrix ricatti equation for linear-systems with a random gain. *IEEE Transactions on Automatic Control*, 21(5):770–771, 1976.

[79] F. P. Kelly. *Reversibility and Stochastic Networks*. John Wiley and Sons, New York, 1979.

[80] R. Kindermann and J. L. Snell. *Markov Random Fields and their Applications*. American Mathematical Society, Providence RI, 1980.

[81] A. Klar and R. Wegener. A hierarchy of models for multilane vehicular traffic i: Modeling. *SIAM Journal on Applied Mathematics*, 59(3):983–1001, Mar. 1999.

[82] D. L. Kleinman. Optimal stationary control of linear systems with control dependent noise. *IEEE Transactions on Automatic Control*, AC-14:673–677, 1969.

[83] L. Kleinrock. *Queueing Systems*, volume 1. Wiley, New York, 1975.

[84] V. Krishnamurty and R. J. Elliot. Filters for estimating Markov modulated poisson processes and image-based tracking. *Automatica*, 33(5):821–833, 1997.

[85] P. R. Kumar and P. Varaiya. *Stochastic Systems: Estimation, Identification, and Adaptive Control*. Prentice Hall, Englewood Cliffs, NJ, 1986.

[86] H. J. Kushner. *Heavy Traffic Analysis of Controlled Queueing and Communications Networks*. Springer, New York, 2001.

[87] P. Lancaster. Explicit solutions of linear matrix equations. *SIAM Review*, 12(4), Oct. 1970.

[88] M. Lebah and J. Pellaumail. Transient-behavior for some Jackson networks. *Performance Evaluation*, 17:115–122, Mar. 1993.

[89] J. T. Lewis and R. Russell. An introduction to large deviations for teletraffic engineers. *http://math.uc.edu/ brycw/classes/576/ldtut96.pdf*, Oct. 1996.

[90] T. M. Liggett. *Interacting Particle Systems*. Springer-Verlag, New York, 1985.

[91] M. J. Lighthill and J. B. Whitham. On kinematic waves. i: Flow movement in long rivers. ii: A theory of traffic flow on long crowded roads. *Proceedings of the Royal Society*, A229:281–345, 1955.

[92] G. D. Lin. On weak convergence within a family of life distributions. *Journal of Applied Probability*, 34:823–826, Sep. 1997.

[93] L. Ljung. Asymptotic-behavior of the extended Kalman filter as a parameter estimator for linear-systems. *IEEE Transactions on Automatic Control*, 24(1):36–50, 1979.

[94] K. A. Loparo, M. R. Buchner, and K. Vasuveda. Leak detection in an experimental heat exchanger process: a multiple model approach. *IEEE Transations on Automatic Control*, 36, 1991.

[95] V. A. Malyshev. Breakdown of conservation laws in stochastic cellular automata. *Problemy Peredachi Informatsii*, 27(2):3–8, Apr.-June 1991.

[96] A. Mandelbaum and G. Pats. State-dependent stochastic networks. part i: Approximations and applications with continuous diffusion limits. *Annals of Applied Probability*, 8(2):569–646, May 1998.

[97] P. J. McLane. Linear optimal control of a linear system with state and control dependent noise. *Transactions of the ASME*, pages 34–40, 1972.

[98] K. S. Meier-Hellstern. The analysis of a queue arising in overflow models. *IEEE Transactions on Communications*, 37:367–372, Apr. 1989.

[99] J. M. Mendel. Tutorial on higher-order statistics (spectra) in signal processing and system theory: theoretical results and some applications. *Proceedings of the IEEE*, 3:278–305, Mar. 1975.

[100] J. M. Mendel. Use of higher-order statistics in signal and image processing theory: an update. *Proceedings of the SPIE Conference on Advanced Algorithms and Architectures for Signal Processing III, San Diego, CA*, pages 126–144, 1988.

[101] R. Mendelssohn. Some problems in estimating population sizes from catch-at-age data. *Fishery Bulletin*, 86(4), Oct. 1988.

[102] P. K. Menon, G. D. Sweriduk, and K. D. Bilimoria. A new approach for modeling, analysis, and control of air traffic flow. *AIAA Guidance, Navigation, and Control Conference and Exhibit, Monterey, CA*, Aug. 2002.

[103] A. Messmer and M. Papageorgiou. Automatic-control methods applied to freeway network traffic. *automatica*, 30(4):691–702, Apr. 1994.

[104] S. Meyn and R. Tweedie. *Markov Chains and Stochastic Stability*. http://black.csl.uiuc.edu/ meyn/pages/TOC.html, 1994.

[105] K. Murphy. A brief introduction to graphical models and Bayesian networks. *http://www.ai.mit.edu/ murphyk/Bayes/bnintro.html*, 2003.

[106] G. Muscolino. Linear systems excited by polynomial forms of non-gaussian filtered processes. *Probabilistic Engineering Mechanics*, 10:35–44, 1995.

[107] R. Nagarajan, J. F. Kurose, and D. Towsley. Approximation techniques for computing packet loss in finite-buffered voice multiplexers. *IEEE Journal on Selected Areas of Communications*, 9:368–377, Apr. 1991.

[108] K. Nagel and M. Schreckenberg. A cellular automaton model for highway traffic. *Journal de Physique 1*, 2(12):2221–2229, Dec. 1992.

[109] C. L. Nikias and M. Raghuveer. Bispectrum estimation: a digital signal processing framework. *Proceedings of the IEEE*, 75:869–891, 1990.

[110] J. R. Norris. *Markov Chains*. Cambridge University Press, New York, 1996.

[111] S. Petrone P. Campagnoli, P. Muliere. Generalized dynamic linear models for financial time series. *Applied Stochastic Models in Business and Industry*, 17(1):27–39, Jan.-Mar. 2001.

[112] M. Papgeorgiou. Dynamic modeling, assignment, and route guidance in traffic networks. *Transportation Research Part B-Methodological*, 24(6):471–495, Dec. 1990.

[113] A. Papoulis. *Probability, Random Variables, Stochastic Processes, 3rd ed.* McGraw Hill, Boston, 1991.

[114] W. Paul and J. Baschnagel. *Stochastic Processes from Physics to Finance.* Springer, New York, 1999.

[115] N. Pujet, B. Delcaire, and E. Feron. Input-output modeling and control of the departure process of congested airports. *AIAA Guidance, Navigation and Control Conference, Portland, Oregon*, Aug. 1999.

[116] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, Feb. 1986.

[117] N. Rajewsky and M. Schreckenberg. Exact results for one-dimensional cellular automata with different types of updates. *Physica A*, 245:139–144, Oct. 1997.

[118] P. I. Richards. Shockwaves on the highway. *Operations Research*, 4:42–51, 1956.

[119] S. M. Ross. *A First Course in Probability.* Prentice Hall, Upper Saddle River, NJ, 2002.

[120] D. Rothman and S. Zaleski. *Lattice-Gas Cellular Automata: Simple Models of Complex Hydrodynamics.* Cambridge University Press, New York, 1997.

[121] S. Roy, C. Asavathiratham, B. C. Lesieutre, and G. C. Verghese. Network models: Growth, dynamics, and failure. *34th Annual Hawaii International Conference on Systems Science*, Jan. 2001.

[122] S. Roy, B. C. Lesieutre, and G. C. Verghese. Resource allocation in networks: a case study of the influence model. 35^{th} *Annual Hawaii International Conference on System Sciences*, Jan. 2002.

[123] S. Roy, B. Sridhar, and G. C. Verghese. An aggregate stochastic dynamic model for an air traffic system. *submitted to the 5th Annual Eurocontrol/Federal Aviation Agency Air Traffic Management Research and Development Conference*, June 2003.

[124] S. L. Scott. Bayesian analysis of a two-state Markov modulated poisson process. *Journal of Computational and Graphical Statistics*, 8:662–670, Sep. 1999.

[125] B. Segall and T. Kailath. The modeling of randomly modulated jump processes. *IEEE Transactions on Information Theory*, 21(2), Mar. 1975.

[126] D. L. Snyder. Filtering and detection of doubly stochastic poisson processes. *IEEE Transactions on Information Theory*, IT-18:91–102, Jan. 1972.

[127] D. L. Snyder and M. I. Miller. *Random Processes in Time and Space, 2nd ed.* Springer-Verlag, 1991.

[128] R. V. Sole and S. Valverde. Information transfer and phase transitions in a model of internet traffic. *Physica A*, 289:595–605, Jan. 2001.

[129] W. J. Stewart. *Introduction to the Numerical Simulations of Markov Chains*. Princeton University Press, Princeton NJ, 1994.

[130] J. M. Stoyanov. *Counterexamples in probability*. Wiley, New York, 1997.

[131] G. Strang. *Linear Algebra and Its Applications, 3rd ed.* Saunders, 1988.

[132] P. J. Sullivan. A Kalman filter approach to catch-at-length analysis. *Biometrics*, 48(1):237–257, Mar. 1992.

[133] A. Swami and J. M. Mendel. Time and lag recursive computation of cumulants from a state space model. *IEEE Transactions on Automatic Control*, 35:4–17, 1990.

[134] D. D. Sworder, J. E. Boyd, and R. J. Elliot. Modal estimation in hybrid systems. *Journal of Mathematical Analysis and Applications*, 245:225–247, May 2000.

[135] H. Vandevenne and M. A. Lippert. Evaluation of runway-assignment and aircraft-sequencing algorithms in terminal area automation. *The Lincoln Laboratory Journal*, 7(2):215–238, 1994.

[136] D. J. Watts. *Six Degrees: The Science of a Connected Age*. W. W. Norton and Company, New York, 2003.

[137] S. Wolfram. *Theory and Applications of Cellular Automata: Including Selected Papers, 1983-1985*. World Scientific, 1986.

[138] S. Wolfram. *A New Kind of Science*. Wolfram Media, Inc., 2002.

[139] E. Yaz. A generalization of the uncertainty threshold principle. *IEEE Transactions on Automatic Control*, 35(8):942–944, Aug. 1990.

[140] E. Yaz. Robustness of stochastic-parameter control and estimation schemes. *IEEE Transactions on Automatic Control*, 35(5):637–640, May 1990.

[141] B. Zehnwirth. A generalization of the Kalman filter for models with state-dependent observation variance. *Journal of the American Statistical Association*, 83(401):164–167, Mar. 1988.