OPTIMAL STATE ESTIMATION FOR PARTIALLY OBSERVED BOOLEAN DYNAMICAL SYSTEMS IN THE PRESENCE OF CORRELATED OBSERVATION NOISE

A Thesis

by

LEVI DANIEL MCCLENNY

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Ulisses Braga-Neto
Committee Members,	Edward Doughtery
	Erchin Serpedin
	Erick Moreno-Centeno
Head of Department,	Miroslav Begovic

August 2016

Major Subject: Electrical Engineering

ABSTRACT

Recently, state space signal models have been proposed to characterize the behavior of discrete-time boolean dynamical systems. The current system model is one in which the system is observed in the presence of noise. The existing algorithms, however, rely on an assumption of independent and identically distributed (i.i.d.) white noise processes. The existing recursive MMSE process of estimating a boolean dynamical system (in the presence of i.i.d. noise) is called the Boolean Kalman Filter (BKF). Here we address a different sort of noise, one that is correlated in time to other observation noise, specifically through an AR(1) time series process. In this thesis, we propose modifications to the state-space model that will allow the existing Boolean Kalman Filtering recursive process to adapt to handle time-correlated noise. Additionally, we will propose a modification to the Boolean Particle Filtering approximation to compensate for the same correlated noise AR(1) process.

In addition, this document will address a new software package created in the R programming language that will allow the scientific community easier (and free) access to the algorithms created by the Genomic Signal Processing Lab at Texas A&M University. These algorithms will be explained in this document, with results of the algorithms derived from the use of the package.

DEDICATION

This document is dedicated to my parents. Without their love and support I would never have made it to where I am today.

I also dedicate this document to my friends, who I have leaned on throughout this process in both good times and bad.

"I give it all for them. They give it all for me."

ACKNOWLEDGEMENTS

First and foremost, I would like to thank Mahdi Imani for the unrelenting support in the development of the package and the correlated noise algorithms. Mahdi's expertise guided me throughout the entire process of developing the contents of this document, and without his help I never would have made it through this process.

I would also like to thank Dr. Ulisses Braga-Neto for his patience and expertise when times got tough. Without his guidance I would not have been able to get to a presentable product, and without his faith in me I never would have had the opportunity to perform this research and learn all that I have.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 Boolean Regulatory Networks

Genetic interactions are constantly regulated within the cellular environment, and many, if not all, regulatory networks are essential to cellular function. These complex interactions are the subject of extensive research by the scientific community, and of particular interest are regulatory networks that control essential cellular process, such as cell cycle, stress response, DNA repair, etc. Kauffman introduced the concept of Boolean Regulatory Networks in 1969 [11], which quickly emerged as a very effective model of the complex dynamical systems present in cellular networks. This model consists of genes stable in two transcriptional states - activated or suppressed [12] [5]. These two discrete states can be represented as a boolean 0 (OFF) or 1 (ON), and the interactions between discrete time steps are governed by discrete boolean logical gates. These interactions between states are predefined by the logical gates, therefore Boolean Regulatory Networks are deterministic in nature. The only hint off randomness that might exist is in the initial distribution of the states, which can be modeled using an initial joint probability distribution consisting of all the possible nodes, each of which has its own unique boolean function, and after that the boolean network will step through the states in a deterministic fashion [16].

The boolean network model can be expressed as a deterministic vector time series by

$$\mathbf{X}_k \;=\; \mathbf{f}\left(\mathbf{X}_{k-1}\right), \tag{1.1}$$

for $k = 1, 2, \ldots$, and where $\mathbf{f} : \{0,1\}^d \longrightarrow \{0,1\}^d$ is the *network function* described above, e.g. a vector of *boolean functions* that govern the relationship between

discrete time points.

An example of a deterministic boolean regulatory network that is heavily researched is the p53-mdm2 transcriptional network in the presence of DNA double strand breaks. The gene p53 is key in suppressing unregulated tumor cell replication. It is worth noting that anywhere from 30 to 50 percent of common human cancers can be accredited to the loss of p53 functionality [20]. The activated/deactivated patters for each of the genes $p53, MDM2, ATM$, and $WIP1$ summarize the dynamic in response to a DNA double strand break [1] [13]. In this network, there is clearly a dimensionality $d = 4$ and the state at a given time k can be represented as a boolean vector $\mathbf{X}_k = (ATM_k, p53_k, WIP1_k, MDM2_k)$, where the subscript k is used to express the genes value at time k. The additional input representing the DNA double strand break can be represented with $\mathbf{u}_k = dna\_dsb_k$ for each time point k. Figure 1.1 shows the proposed given the boolean network model proposed in [13].

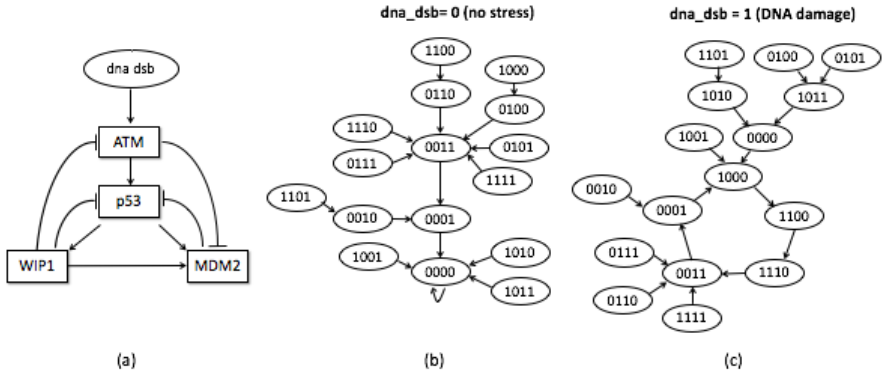

Figure 1.1: p53 Activation/Repression Pathways and State Transitions

In figure 1.1, we see the activation and deactivation interactions between different

genes effecting the production of $p53$ in (a), as well as the boolean state transition trajectory in normal operation (b) and in the event of a DNA double strand break (c). From the interactivity pathway diagram, we see that $MDM2$ has a suppressing effect on $p53$, and the presence of $p53$ can activate $MDM2$. This is referred to as the $p53 - MDM2$ negative-feedback regulatory loop, which is responsible for keeping $p53$ at low levels. However, ATM is capable of suppressing $MDM2$ in the presence of a DNA double strand break, $dna\_dsb = 1$ or figure 1.1.c above. ATM is the transducer gene for *dna_dsb*, which will cause the normal regulation of $p53$ via $MDM2$ to become inactive, due to the suppressing effect that ATM has on $MDM2$. This results in oscillatory behavior of the $p53$ gene through $WIP1$, and $MDM2$. However, given no stress on the network in the form of $dna\_dsb = 1$, we can see that a single attractor, 0000, exists such that all other states are transient. This is shown in figure 1.1.b. However, we can see in the second case, there is a cyclic attractor corresponding to the activation of $p53$, $WIP1$, and $MDM2$. Therefore this boolean network contains two *basins of attraction*, reliant on the status of the DNA damage signal *dna_dsb*. This model can be represented as a connectivity matrix, in which all interactions are represented by a 1, 0, or -1, corresponding to activation, no connectivity, or suppression respectively, as shown below:

$$Q = \begin{array}{c} \\ ATM \\ p53 \\ WIP1 \\ MDM2 \end{array} \begin{array}{c} \begin{array}{cccc} ATM & p53 & WIP1 & MDM2 \end{array} \\ \begin{pmatrix} 0 & 0 & -1 & 0 \\ 1 & 0 & 1 & -1 \\ 0 & 1 & 0 & 0 \\ -1 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

1.2 Probabilistic Boolean Networks

There could potentially be multiple competing functions as a predictor of a given gene, and with this in mind a new idea of how to model the transitions from state to state is proposed in [17]. In the event that there are multiple potential predictors of a gene's next state. At a given time step, any function in a set of potential functions could be selected for the transition to the next step. In order to compensate for the inherent uncertainty in attempting to predict a transition, the approach is to absorb that uncertainty into the predictor itself. The process is to find a series of genes, based off of sample data, that performs somewhat well in predicting a target gene. These are combined into a nonlinear network of boolean functions - meaning a combination of a series of somewhat simple predictors into a larger one capable of predicting the target genes. [17].

Once a decent idea of the genes that can act as predictors is known, a probabilistic boolean model can be created in which the dynamic behavior can be modeled with a Markov chain, allowing us to apply the theory of Markov decision processes to the network [15]. We said above that a PBN is essentially a collection of boolean networks, and at any time point the transition to the next time point acts according to the rules of one of the boolean networks comprising the PBN [16] [17]. This implies that the 'governing' boolean network is chosen at random from a series of boolean functions according to a fixed and predefined probability distribution. In this fashion, the inherant uncertainty is compensated for in network transitions, be it from biological considerations or some other unknown factor. A small probability p is applied that allows each gene to change randomly at any instant to compensate for this biological uncertainty. This creates a system in which all states communicate, in turn creating a completely ergodic Markov chain process with a unique steady state

distribution [15] [18].

1.3 Partially Observed Boolean Dynamical Systems

Models have been proposed to extend the concept of Boolean Networks past compensation of only process noise described above, but also allows us to compensate for the partial observation of a state variable and random process noise. Such a model is originally proposed in [3].

1.3.1 State Model of a Partially Observed Boolean Dynamical System

As described above, the state transition can be modeled in such a way to compensate for the inherent process noise in the transition from state to state due to latent variables and biological interference. We assume this is modeled by a *state process* $\{\mathbf{X}_k; k = 0, 1, \ldots\}$, where $\mathbf{X}_k \in \{0,1\}^d$ is a Boolean vector of size d. The evolution of the state process is governed by the equation

$$\mathbf{X}_k = \mathbf{f}(\mathbf{X}_{k-1}, \mathbf{u}_k 1) \oplus \mathbf{n}_k \quad \text{(state model)} \tag{1.2}$$

for $k = 1, 2, \ldots$. Here, $\mathbf{n}_k \in \{0,1\}^d$ is Boolean transition noise, "$\oplus$" indicates component-wise modulo-2 addition, $\mathbf{u}_k \in \{0,1\}^p$ and $\mathbf{f} : \{0,1\}^{d+p} \longrightarrow \{0,1\}^d$ are the input and network function, respectively, and $\{\mathbf{n}_k, \mathbf{v}_k; k = 1, 2, \ldots\}$ is assumed to be white noise, e.g. it is independent process in time.

1.3.2 Observation Model of a Partially Observed Boolean Dynamical System

The difference between a Boolean Dynamical System and a Probabilistic Boolean Network is that a Boolean Dynamical System operates under the assumption that the system state is only partially observable, and that these observations are taken in the presence of some sort of measurement or sensor noise. In light of this assumption, let $\mathbf{Y}_k$ be the observation of $\mathbf{X}_k$ at time k. This defines an *observation process*

$\{\mathbf{Y}_k; k = 1, 2, \ldots\}$. The observation $\mathbf{Y}_k$ is derived from the state variable $\mathbf{X}_k$ by the following equation:

$$\mathbf{Y}_k = \mathbf{h}(\mathbf{X}_k, \mathbf{v}_k) \quad \text{(observation model)} \tag{1.3}$$

for $k = 1, 2, \ldots$. Here, $\mathbf{n}_k \in \{0, 1\}^d$ is Boolean transition noise.

In the Boolean Kalman Filter section, both the state model and observation model will be combined to yield the proposed signal model for which the Boolean Kalman Filter is the optimal recursive MMSE estimator.

1.4 Partially Observed Boolean Dynamical System Optimal Estimation

It is well known that if a system is linear with gaussean noise, the optimal recursive MMSE estimator is the Kalman Filter [10]. The extended Kalman filter (EKF) came about to handle partially observed nonlinear dynamical systems. The EKF approach is to apply a first order Taylor-series expansion to perform a linearization at each time step, on which a traditional Kalman filter is applied [9]. However, EKF approaches cannot be applied to a boolean dynamical system due the non-differentiable nature of the transition functions. There are a series of derivativeless filters that could be capable of handling the non-differentiable nature of Partially Observed Boolean Dynamical Systems, called Sigma-Point Kalman Filters (SPKF) [19], however the SPKF algorithms are, thusfar, limited to non-discrete distributions. Since the distributions in Partially Observed Boolean Dynamical Systems are discrete in nature, the SPKF theory cannot be applied.

The Boolean Kalman Filter [3] and the Boolean Kalman Smoother [8] are the optimal recursive MMSE estimator of these non-linear, derivativeless Partially Observed Boolean Dynamical Systems. These systems belong to a class of Hidden Markov

Models, in which it is well known that a matrix implementation of a backward-forward procedure is the optimal estimator [2]. The BKF/BKS algorithms share a striking similarity to these backward-forward estimators that are typically used to estimate the state of a HMM. However, it is worth noting that there exists an extra matrix multiplication step that allows for MMSE computation, whereas traditional HMM estimation is in the Maximum A-Posteriori (MAP) sense. This MAP step has a tendency to be computationally intensive for high dimensional state spaces. Also, a unique property of the MMSE in the boolean case provides the MAP in each boolean state variable *seperately*, which is not typically a property of the global MAP estimator.

An approach to Partially Observed Boolean Dynamical System approximation can be found in Sequential Monte-Carlo Methods, which can approximate the equations in the nonlinear prediction step. This approach is more commonly referred to as Particle Filtering [14]. These algorithms are incredibly useful, however they exist only to serve as approximations of the optimal MMSE estimator [4]. More information on the approximation of Partially Observed Boolean Dynamical Systems will come about later in this document.

2. THE PROBLEM OF CORRELATED OBSERVATION NOISE

One of the fundamental assumptions of the Boolean Kalman Filter derived in [3] is that the noise processes for both the state and observation processes $\{\mathbf{n}_k, \mathbf{v}_k; k = 1, 2, \ldots\}$ (discussed in (1.2) and (1.3), combined below in (2.10)) are white noise processes, uncorrelated in time. However, real-world applications could exist in which the noise at each time point is not completely uncorrelated in time, but rather dependent on some auto-regressive process. Here, we develop a modification to the existing Boolean Kalman Filter to allow for time-dependent noise processes - specifically a first-order autoregressive time series function, commonly referred to as an $AR(1)$ *process* .

2.1 Boolean Kalman Filter in the Presence of Correlated Observation Noise

2.1.1 Boolean Kalman Filter Background

Assume that the system is described by a *state process*$\{\mathbf{X}_k; k = 0, 1, \ldots\}$, where $\mathbf{X}_k \in \{0, 1\}^d$ is a Boolean vector of size d — in the present case of a gene regulatory network, the components of $\mathbf{X}_k$ represent the activation/inactivation state, at discrete time k, of the genes comprising the network. The state is observed indirectly through the *observation process* $\{\mathbf{Y}_k; k = 1, 2, \ldots\}$, where $\mathbf{Y}_k$ is a vector of measurements. The state is assumed to be updated and observed at each discrete time through the following nonlinear signal model:

$$\begin{aligned} \mathbf{X}_k &= \mathbf{f}(\mathbf{X}_{k-1}, \mathbf{u}_k) \oplus \mathbf{n}_k \quad \text{(state model)} \\ \mathbf{Y}_k &= \mathbf{h}(\mathbf{X}_k, \mathbf{v}_k) \quad \text{(observation model)} \end{aligned} \tag{2.1}$$

for $k = 1, 2, \ldots$. Here, $\mathbf{n}_k \in \{0,1\}^d$ is Boolean transition noise, "$\oplus$" indicates component-wise modulo-2 addition, $\mathbf{u}_k \in \{0,1\}^p$ and $\mathbf{f} : \{0,1\}^{d+p} \longrightarrow \{0,1\}^d$ are the input and network function, respectively, whereas $\mathbf{h}$ is a general function mapping the current state and observation noise $\mathbf{v}_k$ into the measurement space. The noise processes $\{\mathbf{n}_k, \mathbf{v}_k; k = 1, 2, \ldots\}$ are assumed to be "white" in the sense that the noises at distinct time points are uncorrelated random variables. It is also assumed that the noise processes are uncorrelated with each other and with the initial state $\mathbf{X}_0$.

The optimal filtering problem consists of finding an estimator $\hat{\mathbf{X}}_k = h(\mathbf{Y}_1, \ldots, \mathbf{Y}_k)$ of the state $\mathbf{X}_k$ that minimizes the conditional mean-square error (MSE):

$$\mathrm{MSE}(\mathbf{Y}_1, \ldots, \mathbf{Y}_k) \;=\; E\left[||\hat{\mathbf{X}}_k - \mathbf{X}_k||^2 \mid \mathbf{Y}_k, \ldots, \mathbf{Y}_1\right] \tag{2.2}$$

at each value of $\mathbf{Y}_1, \ldots, \mathbf{Y}_k$.

The BKF provides the minimum MSE state estimator and may be computed exactly in a recursive fashion, as shown in [3]. Briefly, let $(\mathbf{x}^1, \ldots, \mathbf{x}^{2^d})$ be an arbitrary enumeration of the possible state vectors. For each time $k = 1, 2, \ldots$ define the posterior distribution vectors (PDV) $\mathbf{\Pi}_{k|k}$ and $\mathbf{\Pi}_{k|k-1}$ of length 2^d by means of

$$(\mathbf{\Pi}_{k|k})_i \;=\; P\left(\mathbf{X}_k = \mathbf{x}^i \mid \mathbf{Y}_k, \ldots, \mathbf{Y}_1\right) , \tag{2.3}$$

$$(\mathbf{\Pi}_{k|k-1})_i \;=\; P\left(\mathbf{X}_k = \mathbf{x}^i \mid \mathbf{Y}_{k-1}, \ldots, \mathbf{Y}_1\right) , \tag{2.4}$$

for $i = 1, \ldots, 2^d$. Let the *prediction matrix* M_k of size $2^d \times 2^d$ be the transition matrix of the Markov chain defined by the state mode

$$\begin{aligned} (M_k)_{ij} \;&=\; P(\mathbf{X}_k = \mathbf{x}^i \mid \mathbf{X}_{k-1} = \mathbf{x}^j) \\ &=\; P\left(\mathbf{n}_k \;=\; \mathbf{x}^i \oplus \mathbf{f}(\mathbf{x}^j, \mathbf{u}_k)\right) , \end{aligned} \tag{2.5}$$

for $i, j = 1, \ldots, 2^d$. Additionally, given a value of the observation vector $\mathbf{y}$, let the *update matrix* $T_k(\mathbf{y})$, also of size $2^d \times 2^d$, be a diagonal matrix defined by the observation model

$$\left(T_k(\mathbf{Y}_k)\right)_{jj} = p\left(\mathbf{Y}_k \mid \mathbf{X}_k = \mathbf{x}^j\right) \tag{2.6}$$

for $j = 1, \ldots, 2^d$. Finally, define the matrix A of size $d \times 2^d$ via $A = \left[\mathbf{x}^1 \cdots \mathbf{x}^{2^d}\right]$.

The following result, which appears in [3], gives a procedure to compute the MMSE state estimator.

Theorem 1. (Boolean Kalman Filter.) *The optimal minimum MSE estimator* $\hat{\mathbf{X}}_k$ *of the state* $\mathbf{X}_k$ *given the observations* $\mathbf{Y}_1, \ldots, \mathbf{Y}_k$ *up to time* k *is given by*

$$\hat{\mathbf{X}}_k = \overline{E\left[\mathbf{X}_k \mid \mathbf{Y}_k, \ldots, \mathbf{Y}_1\right]}, \tag{2.7}$$

where $\overline{\mathbf{v}}(i) = I_{\mathbf{v}(i)>1/2}$ *for* $i = 1, \ldots, d$. *This estimator and its optimal conditional MSE can be computed by the following procedure.*

1. *Initialization Step: The initial PDV is given by* $(\mathbf{\Pi}_{0|0})_i = P\left(\mathbf{X}_0 = \mathbf{x}^i\right)$, *for* $i = 1, \ldots, 2^d$.

For $k = 1, 2, \ldots,$ *do:*

2. *Prediction Step: Given the previous PDV* $\mathbf{\Pi}_{k-1|k-1}$, *the predicted PDV* $\mathbf{\Pi}_{k|k-1}$ *is given by* $\mathbf{\Pi}_{k|k-1} = M_k\, \mathbf{\Pi}_{k-1|k-1}$.

3. *Update Step: Given the current observation* $\mathbf{Y}_k = \mathbf{y}_k$, *let* $\boldsymbol{\beta}_k = T_k(\mathbf{y}_k)\, \mathbf{\Pi}_{k|k-1}$. *The* updated *PDV* $\mathbf{\Pi}_{k|k}$ *is obtained by normalizing* $\boldsymbol{\beta}_k$ *to obtain a probability measure:* $\mathbf{\Pi}_{k|k} = \boldsymbol{\beta}_k / ||\boldsymbol{\beta}_k||_1$.

4. MMSE Estimator Computation Step: The MMSE estimator is given by

$$\hat{\mathbf{X}}_k \;=\; \overline{A\mathbf{\Pi}_{k|k}} \tag{2.8}$$

with optimal conditional MSE

$$\begin{aligned} &\mathrm{MSE}(\mathbf{Y}_1, \ldots, \mathbf{Y}_k) \\ &\quad = \; || \min\{A\mathbf{\Pi}_{k|k}, (A\mathbf{\Pi}_{k|k})^c\}||_1 \,, \end{aligned} \tag{2.9}$$

where the minimum is applied component-wise, and the complement of a vector $\mathbf{v}$ *is defined by* $(\mathbf{v}^c)_i = 1 - \mathbf{v}_i$*, for* $i = 1, \ldots, d$*.*

2.1.2 Modifications for Correlated Noise

To accommodate the issue of correlated noise, we will augment the existing state space model of the Boolean Kalman Filter to accommodate the new proposed noise time dependency. This will effectively double the number of variables in the state space, all of which will be boolean for this particular application of Bernoulli noise. In order to accommodate for the correlated noise, we will add a parameter $\mathbf{v}_k$ to the state space model that is dependent on $\mathbf{v}_{k-1}$ plus some random Bernoulli perturbation $\boldsymbol{\eta}_k$. This will modify our model from what was previously

$$\begin{aligned} \mathbf{X}_k \;&=\; \mathbf{f}\left(\mathbf{X}_{k-1}, \mathbf{u}_k\right) \;\oplus\; \mathbf{n}_k \quad \text{(state model)} \\ \mathbf{Y}_k \;&=\; \mathbf{X}_k \oplus \mathbf{v}_k \quad\;\; \text{(observation model)} \end{aligned} \tag{2.10}$$

where $\mathbf{v}_k$ was independent and identically distributed noise and $\oplus$ is component-wise modulo-2 addition (for the boolean case), to a process now dependent on the value of $\mathbf{v}_{k-1}$. Our new observation model will therefore be modified to where $\mathbf{v}_k$ is defined

as

$$\mathbf{v}_k \;=\; \mathbf{v}_{k-1} \oplus \boldsymbol{\eta}_k \quad \text{(correlated observation model)} \tag{2.11}$$

Overall, the recursive process that the Boolean Kalman Filter utilizes does not change, however we must re-evaluate what is put into the algorithm, specifically modifying both the predict matrix M_k (2.5) and the update matrix $T_k(\mathbf{y})$ (2.6).

The new state model is given below:

$$\underbrace{\begin{bmatrix}\mathbf{X}_k \\ \mathbf{v}_k\end{bmatrix}}_{\mathbf{Z}_k} = \underbrace{\begin{bmatrix}\mathbf{f}\left(\mathbf{X}_{k-1}\right) \\ \mathbf{v}_{k-1}\end{bmatrix}}_{\mathbf{f}'\left(\mathbf{Z}_{k-1}\right)} \oplus \begin{bmatrix}\mathbf{n}_k \\ \boldsymbol{\eta}_k\end{bmatrix} \tag{2.12}$$

Where $\mathbf{Z}_k$ is our new state space defined as:

$$\begin{aligned}\mathbf{Z}^i &= \left(\mathbf{X}^i, \mathbf{v}^i\right) \\ \mathbf{Z}^j &= \left(\mathbf{X}^j, \mathbf{v}^j\right)\end{aligned} \tag{2.13}$$

Our observation model is now defined by some function $\mathbf{h}'\left(\mathbf{Z}_k\right)$ as shown below.

$$\mathbf{Y}_k \;= \underbrace{\mathbf{X}_k \;\oplus\; \mathbf{v}_k}_{\mathbf{h}'(\mathbf{Z}_k)}, \tag{2.14}$$

This definition of $\mathbf{h}'\left(\mathbf{Z}_k\right)$ shown above (with modulo-2 addition) is specific to Bernoulli noise.

2.1.2.1 The Predict Matrix: M_k

One of the steps in modifying the existing Boolean Kalman Filtering algorithm to compensate for and AR(1) correlated noise process is to augment the state space to include these new boolean variables. This implies that we will add n additional boolean variables to the state space, where n is the number of genes. Therefore, for

the 4 gene p53-mdm2 network, we will have 8 boolean variables to represent each state in the new transition matrix. This implies that the new transition matrix will be 2^{2n} x 2^{2n}. In this new state space, the first n boolean variables will represent the state of the system, and the second n variables will represent the observation noise of the system. These two sets of n variables will be binded together to create the transition matrix M_k (eqn 2.5) in the augmented Boolean Kalman Filter. The formal definition of each of the entries in the new transition matrix is:

$$(M_k)_{ij} = p^{|\mathbf{x}^i - \mathbf{f}(\mathbf{x}^j)|_1} (1-p)^{d-|\mathbf{x}^i - \mathbf{f}(\mathbf{x}^j)|_1} q^{|\mathbf{v}^i - \mathbf{v}^j|_1} (1-q)^{d-|\mathbf{v}^i - \mathbf{v}^j|_1} \tag{2.15}$$

Notice that this takes both the process noise parameter p and the observation noise parameter q into account, whereas the naive Boolean Kalman Filter only accounts for process noise p in the calculation of the matrix M_k.

2.1.2.2 The Update Matrix: $T_k(\mathbf{y})$

The update matrix $T_k(\mathbf{y})$ (eqn 2.6) is utilized in the update step in the recursive Boolean Kalman Filtering process to calculate the posterior distribution of the probability of the states within the state space given a particular observation $\mathbf{Y}_k$. In the naive case, the update matrix was derived in much the same way as the predict matrix, except that it was dependent only on the *observation* noise probability distribution and the observation itself. Since this matrix multiplication step gives the posterior probability of moving to a unique state, as opposed to state-to-state transition like in the predict matrix, this matrix is diagonal in nature. However, in the case of correlated noise, we seek out the possible combinations of state variables and noise that can potentially yield the given observation $\mathbf{Y}_k$. This will yield a vector of 0s and 1s, where the 1s are possible combinations of state and observation noise that

could produce the given observation. Making a diagonal matrix of this vector and applying it in the same fashion as the naive Boolean Kalman Filter will yield the proper posterior distribution of the states for that time point. The formal definition of the diagonal update matrix is given below:

$$(T_k)_{ii} = \prod_{j=1}^{d} \Big[1 - |\mathbf{Y}_k(j) - (\mathbf{X}^i(j) \oplus \mathbf{v}^i(j))|\Big] \tag{2.16}$$

2.1.3 Results

Randomly generated datasets from the p53-MDM2 network transition function (containing process noise) were generated, and time-correlated observation noise was applied (using modulo-2 addition) to simulate time-correlated noisy data. This was used to test the functionality of the Time Correlated BKF in comparison to the naive BKF. Results are shown in this section - shown as average probability of correct prediction, and referred to as performance.

First, a comparison of the the performance between the Time Correlated BKF (T-C BKF) and the Naive BKF, with additive time correlated Bernoulli noise is shown in table 2.1.

		dna_dsb = 0		***dna_dsb = 1***	
Process Noise p	**Observation noise q**	**T-C BKF**	**Naive BKF**	**T-C BKF**	**Naive BKF**
0.01	0.01	0.95	0.41	0.95	0.29
0.01	0.1	0.94	0.56	0.87	0.30
0.1	0.01	0.91	0.28	0.91	0.34
0.1	0.1	0.60	0.15	0.46	0.10

Table 2.1: Performance of the Time Correlated (T-C) BKF and the naive BKF

From this table, it is clear that the Correlated Noise BKF significantly outperforms the naive BKF in every combination of process and observation noise (where observation noise is now the parameter $\boldsymbol{\eta}_k$ shown in the observation model in equation 4.1).

An interesting phenomena to note is the increase of the correct prediction rate of the naive BKF as the observation noise increases. This can be accredited to the increase in randomness as the observation noise parameter approaches $\frac{1}{2}$. Between the range of $\boldsymbol{\eta}_k = 0.4$ to $\boldsymbol{\eta}_k = 0.5$ the T-C BKF and the naive BKF take on very similar prediction rates. This is a logical phenomena, since the randomness in this range of the applied observation noise effectively removes the correlation, closely mimicking the randomness that the naive BKF expects. This is shown in figure 2.1

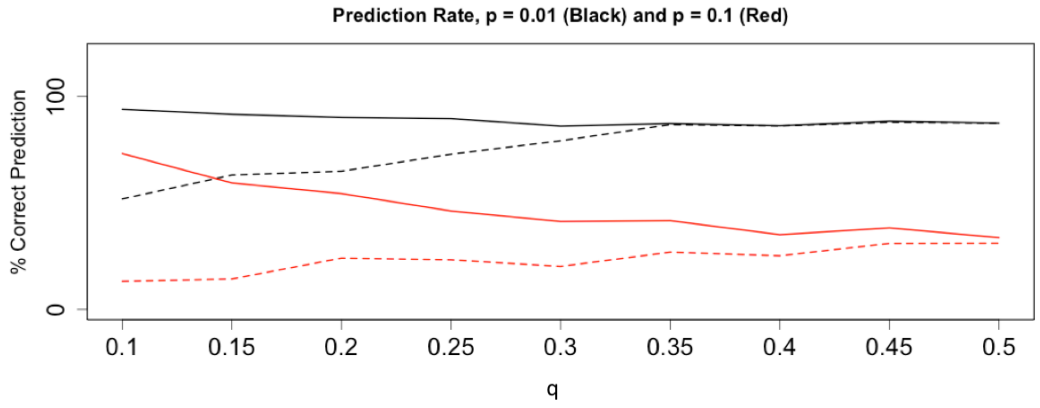

Figure 2.1: Comparison of T-C BKS with naive BKS as Randomness Increases

2.2 Boolean Particle Filter in the Presence of Correlated Observation Noise

2.2.1 Particle Filtering Background

As the number of state variables increases, the computation of the BKF becomes intractable or computationally expensive (due to the large size of transition and update matrices in the BKF algorithm), and one has to resort to approximation methods. A common and popular approach for performing this approximation is called *sequential importance sampling* (SIS).

Considering the full posterior distribution at time $k-1$, $P(\mathbf{X}_{0:k-1} \,|\, \mathbf{Y}_{1:k-1})$, SIS is designed to approximate the posterior distribution at time $k-1$, $P(\mathbf{X}_{0:k-1} \,|\, \mathbf{Y}_{1:k-1})$, with a weighted set of particles from the state space and update particles for an approximation of the posterior distribution at the next time step, $P(\mathbf{X}_{0:k} \,|\, \mathbf{Y}_{1:k})$, in a recursive manner. The SIS method is based on a concept called importance sampling, which is commonly used when the direct sampling of target distribution is difficult. The idea is approximating a target distribution $p(\mathbf{x})$ using sample drawn from proposed distribution $q(\mathbf{x})$, which is much easier than sampling directly from the actual target. The discrepancy created by sampling from the proposed distribution instead of the actual target distribution is compensated by weighing each sample. Assuming $\mathbf{x}^i$ as a particle, the weight can be obtained as $W_i \propto \frac{\pi(\mathbf{x}^i)}{q(\mathbf{x}^i)}$, where $\pi(\mathbf{x})$ is a function that is proportional to $p(\mathbf{x})$ (i.e. $p(\mathbf{x}) \propto \pi(\mathbf{x})$). Thus, importance sampling approximates the posterior distribution at time $k-1$ as:

$$P(\mathbf{X}_{0:k-1} \,|\, \mathbf{Y}_{1:k-1}) \approx \sum_{i=1}^{N} \tilde{W}_{k-1,i}\, \delta_{\mathbf{x}_{0:k-1,i}}\,, \tag{2.17}$$

where N is the number of particles, $\tilde{W}_{k-1,i}$ specifies the normalized weight of the ith particle ($\mathbf{x}_{k-1,i}$) at time step $k-1$ and $\delta_{\mathbf{x}_{0:k-1,i}}$ is a delta function centered at

$\mathbf{x}_{0:k-1,i}$. The key idea of SIS is using the normalized weights $\{\tilde{W}_{k-1,i}\}_{i=1}^{N}$ and particles $\{\mathbf{x}_{k-1,i}\}_{i=1}^{N}$ at time $k-1$ and approximating the posterior distribution at next time step ($P(\mathbf{X}_{0:k} \,|\, \mathbf{Y}_{1:k})$). The following factorization of the proposal distribution declares the basis for obtaining the new particles:

$$\begin{aligned} q(\mathbf{X}_{0:k} \mid \mathbf{Y}_{1:k}) = q(\mathbf{X}_k \mid \mathbf{X}_{0:k-1}, \mathbf{Y}_{1:k}) \\ \times q(\mathbf{X}_{0:k-1} \mid \mathbf{Y}_{1:k-1}) \,, \end{aligned} \tag{2.18}$$

The above equation implies that the next set of particles can be obtained by sampling new particles ($\{\mathbf{x}_{k,i}\}_{i=1}^{N}$) from $q(\mathbf{X}_k \,|\, \mathbf{X}_{0:k-1}, \mathbf{Y}_{1:k})$, known as the proposal distribution. After sampling the current particles, the whole set of particles, $\{\mathbf{x}_{0:k,i}\}_{i=1}^{N}$ is created by putting the obtained particles in current time step next to the previous particles. Note that according to importance sampling, the new weights of particle can be obtained as:

$$W_{k,i} \;\propto\; \frac{P(\mathbf{x}_{0:k,i} \,|\, \mathbf{Y}_{1:k})}{q(\mathbf{x}_{0:k,i} \,|\, \mathbf{Y}_{1:k})} \,, \;\text{for } i = 1, ..., N \,. \tag{2.19}$$

To be able to express equation (2.19) recursively, one can write $P(\mathbf{X}_k \,|\, \mathbf{Y}_k)$ based on $P(\mathbf{X}_{k-1} \,|\, \mathbf{Y}_{k-1})$ as:

$$\begin{aligned} & P(\mathbf{X}_{0:k} \mid \mathbf{Y}_{0:k}) \\ & = \frac{P(\mathbf{Y}_k \mid \mathbf{X}_{0:k}, \mathbf{Y}_{0:k-1}) P(\mathbf{X}_{0:k} \mid \mathbf{Y}_{1:k-1})}{P(\mathbf{Y}_k \mid \mathbf{Y}_{1:k-1})} \\ & = \frac{P(\mathbf{Y}_k \mid \mathbf{X}_{0:k}, \mathbf{Y}_{0:k-1})\, P(\mathbf{X}_k \,|\, \mathbf{X}_{0:k-1}, \mathbf{Y}_{1:k-1})}{P(\mathbf{Y}_k \,|\, \mathbf{Y}_{1:k-1})} \\ & \qquad\qquad \times P(\mathbf{X}_{0:k-1} \,|\, \mathbf{Y}_{1:k-1}) \\ & = \frac{P(\mathbf{Y}_k \mid \mathbf{X}_k)\, P(\mathbf{X}_k \mid \mathbf{X}_{k-1})}{P(\mathbf{Y}_k \mid \mathbf{Y}_{1:k-1})}\, P(\mathbf{X}_{0:k-1} \mid \mathbf{Y}_{1:k-1}) \,. \end{aligned} \tag{2.20}$$

By substituting equations (2.18) and (2.20) into equation (2.19), the weights of particles at time step k can be obtained based on the weights at time $k-1$ as:

$$W_{k,i} \propto \frac{P(\mathbf{Y}_k \,|\, \mathbf{x}_{k,i})\, P(\mathbf{x}_{k,i} \,|\, \mathbf{x}_{k-1,i})}{q(\mathbf{x}_{k,i} \,|\, \mathbf{x}_{0:k-1,i}, \mathbf{Y}_{1:k})}\, W_{k-1,i}\,, \tag{2.21}$$

for $i = 1, ..., N$; where the particles at time k, $\mathbf{x}_{k,i}$, are drawn from proposal distribution ($\mathbf{x}_{k,i} \sim q(\mathbf{X}_k \,|\, \mathbf{X}_{0:k-1}, \mathbf{Y}_{1:k})$). Assuming that the proposal at time k is only a function of the state at time $k-1$, ($q(\mathbf{X}_k \,|\, \mathbf{X}_{0:k-1}, \mathbf{Y}_{1:k}) = q(\mathbf{X}_k \,|\, \mathbf{X}_{k-1})$), equation (2.21) can be rewritten as:

$$\begin{aligned} \mathbf{x}_{k,i} &\sim P(\mathbf{X}_k \mid \mathbf{X}_{k-1} = \mathbf{x}_{k-1,i}), \\ W_{k,i} &\propto P(\mathbf{Y}_k \,|\, \mathbf{x}_{k,i})\, W_{k-1,i}\,, \end{aligned} \tag{2.22}$$

for $i = 1, ..., N$. This Using the weights obtained in equation (2.22), the posterior probability at time k can be approximated as:

$$P(\mathbf{X}_k \,|\, \mathbf{Y}_{1:k}) \approx \sum_{i=1}^{N} \tilde{W}_{k,i}\, \delta_{\mathbf{x}_{k,i}}\,. \tag{2.23}$$

where $\{\tilde{W}_{k,i}\}_{i=1}^{N}$ is obtained by normalizing $\{W_{k,i}\}_{i=1}^{N}$. Note that, the particle filter with dynamic model $P(\mathbf{X}_k \mid \mathbf{X}_{k-1})$ as a importance distribution are known as *bootstrap* filters [6]. We develop our particle filter based on this importance distribution.

After a few iterations of the SIS method, we can reach a state where only few of the particles contain a significant amount of the weight, and the wights of all the other particles are very small. This problem is commonly referred to as the *degeneracy problem*. The intensity of the degeneracy can be measured by using the

effective sample size which is defined as follows:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N}(\tilde{W}_{k,i})^2}, \tag{2.24}$$

Since $\sum_{i=1}^{N} \tilde{W}_{k,i}$ sum to 1, N_{eff} is between 1 and N. The smaller N_{eff}, the more degenerate particles are. The most common way of handling the degeneracy problem is using a 'resampling' process. In the resampling process, the new set of particles will be drawn (with replacement) from the approximate current distribution. Thus, N new particles will be drawn from the current particles based on their weights and they will be the new set of particles to approximate the current posterior distribution. Once the particles are resampled, the process continues recursively as before. Resampling can happen as many times as necessary to maintain the recursive process' integrity throughout the calculation of the approximation of $\mathbf{X}$.

2.2.2 Modifications for Correlated Noise

2.2.2.1 Predicting the Next Time Step

In order to compensate for the correlated noise, the 'step forward' step, which is the SMC (or SIS) approximation of the transition matrix required to calculate the next time point, needed to be modified. This modification takes the existing 'step forward' SIS approach and adds the new dimensions described in our new observation model. This, once again, doubles the number of dimensions of the 'step forward' step, where the first n dimensions are the states of the genes and the second n dimensions are the values of the observation noise. These two halves of the state space are 'stepped forward' in their own individual way, by using the model to step forward the first n gene states and a random Bernoulli perturbation to step forward the next n observation states. The SIS approximation approach allows for calculations in significantly higher dimensionality, due to an approximation of the predicted next step, rather than a direct (and exact) calculation. This approximation approach dramatically reduces the computation time of lower dimensional networks, and enables the approximation of higher-order networks that were previously too dimensionally complex to be calculated in an exact recursive MMSE fashion.

2.2.2.2 Incorporating the Observation

Again, in a similar fashion to the time-correlated Boolean Kalman Filter, the observation at a given time point is included in the particle filtering approximation to generate a posterior probability distribution of the next possible states, from which the most likely next state is the prediction output of the algorithm. This is, again, implemented in a recursive fashion - the same fashion as the time-correlated Boolean Kalman Filter for each observation $\mathbf{Y}_k$.

2.2.3 Results

2.2.3.1 4 Gene Regulatory Network

The Time-Correlated Boolean Kalman Particle Filter results are shown below in table 2.2. Both the T-C BKF and the T-C PF algorithms were ran and an optimal estimate/approximation was derived using the same observations $\mathbf{Y}_k$, to ensure performance difference accuracy between the T-C BKF and the T-C PF with varying levels of particles N. The simulations were ran using the 4-gene p53-MDM2 network described in section 1.1. It is clear that the approximation of the SIS method does under-perform the MMSE estimate of the Time-Correlated Boolean Kalman Filter. We can also see that as the number of particles N increases, the approximation does approach the MMSE value of the T-C Boolean Kalman Filter, but does not attain it. Past $N = 10,000$ the algorithm begins to take a signifiganly longer amount of time to approximate, and the extra computational intensity of increasing the particles could potentially be deemed to not warrant the marginal increase in performance of the T-C Particle Filter.

p	q	**T-C PF** $N = 10$	**T-C PF** $N = 100$	**T-C PF** $N = 1000$	**T-C BKF**
0.01	0.01	0.82	0.88	0.94	0.97
0.01	0.1	0.81	0.88	0.91	0.93
0.1	0.01	0.27	0.65	0.86	0.92
0.1	0.1	0.23	0.42	0.55	0.60

Table 2.2: Performance of the Time Correlated (T-C) BKF v.s. the T-C Particle Filter

It is worth noting that one of the main benefits to this SIS approximation algorithm is how much less computationally intensive the approximation method is than the MMSE method. Figure 2.2 shows, graphically, the difference in computation times for the performance displayed in table 2.2. The T-C BKF is not extremely computationally intensive in this 4 genes correlated noise application, despite the additional dimensionality added by the modifications required to handle the noise correlation. The T-C BKF requires approximately 2.26 seconds to compute the optimal MMSE estimator of $\mathbf{X}$, whereas the T-C Particle Filter (with $N = 1000$) only requires 1.242 seconds to compute and approximation that falls within a few percent error of the optimal estimator.

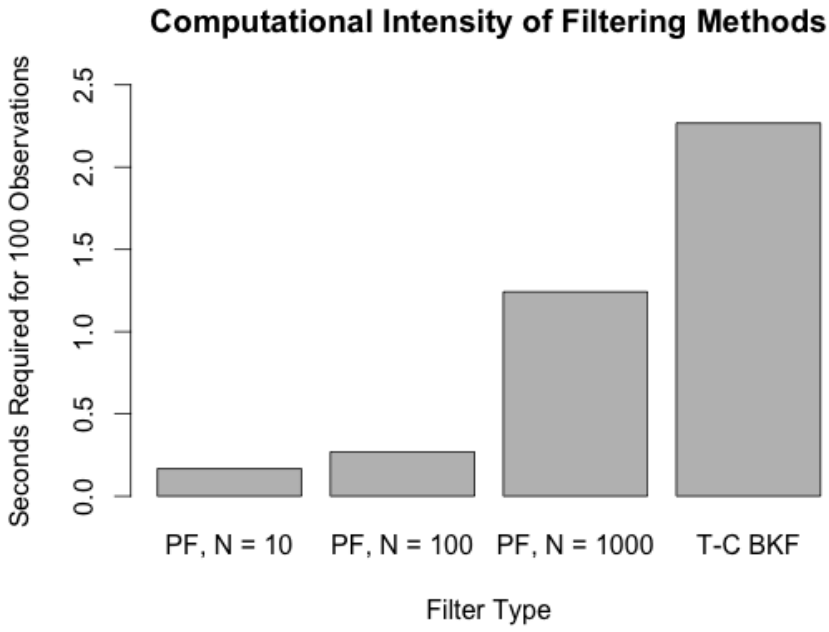

Figure 2.2: Seconds Required for Computation, Various Correlated Filter Types, 100 Observations

2.2.3.2 10 Gene Regulatory Network

It is quickly worth noting that a test of the high-dimensionality application of the Time-Correlated Particle Filter was implemented. The method was used on a 10 gene network with additive correlated noise, and the resulting computation took slightly more than 20 seconds to calculate a single state estimate for a single observation (with $N = 1000$). For a 100 observation data set $\mathbf{Y}_k$, this implies that the computation time would be over 2014 seconds, or around 33 minutes, simply to compute an approximation of the MMSE estimator. The optimal MMSE estimate of this same 10 gene correlated noise network (which would have a predict matrix of size 2^{20} x 2^{20}) proved too much for a 2013 MacBook Air (with 4 Gb of RAM) to compute without crashing. Therefore, it is clear that the value of this algorithm lies in the fact that it at least allows computation of a 10-gene network on this particular machine, whereas the T-C BKF physically failed to run.

More tests of the particle filtering approximation as applied to a 10-gene network (in the presence of i.i.d. noise) will be discussed in section 3.3.

3. R PACKAGE: BOOLFILTER

3.1 Boolean Kalman Filter

3.1.1 Description

The Boolean Kalman Fliter algorithm is intended to generate an optimal MMSE estimate of a partially observed boolean dynamical system. The R package can handle user-defined networks, and is currently capable of handling gaussian and bernoulli noise.

3.1.2 Results

In order to show the Boolean Kalman Filter in action, the p53-MDM2 model is first used to generate gene expression data, with additive Bernoulli noise. The Boolean Kalman Filter output is graphed in figure 3.1 by gene, where the black is the original trajectory $\mathbf{X}_k$, the blue-dashed trajectory is the optimal estimator for an observation noise $q = .01$ and the red-dashed is the optimal estimator for a observation noise value $q = 0.1$.

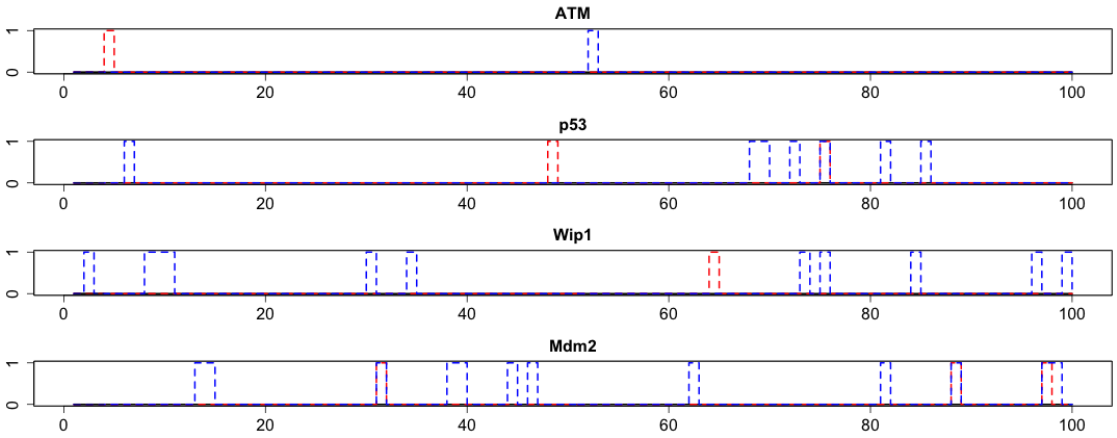

Figure 3.1: Optimal State Estimator Trajectories, p = 0.01

Again we yield an optimal estimate, this time increasing process noise to $p = 0.1$. This is shown in figure 3.2, and we can see yields a slightly more chaotic estimator, however the singleton attractor at 0000 still shows through the noise.

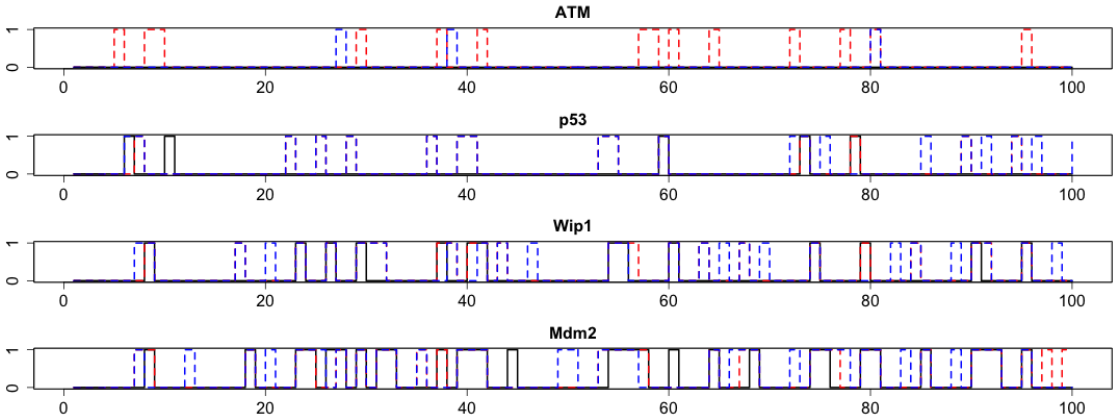

Figure 3.2: Optimal State Estimator Trajectories, p = 0.1

Now, we will activate *dna_dsb* and again yield an optimal estimator. We repeat this experiment for both a low process noise ($p = .01$), shown in figure 3.3, and high process noise ($p = .01$), shown in figure 3.4.

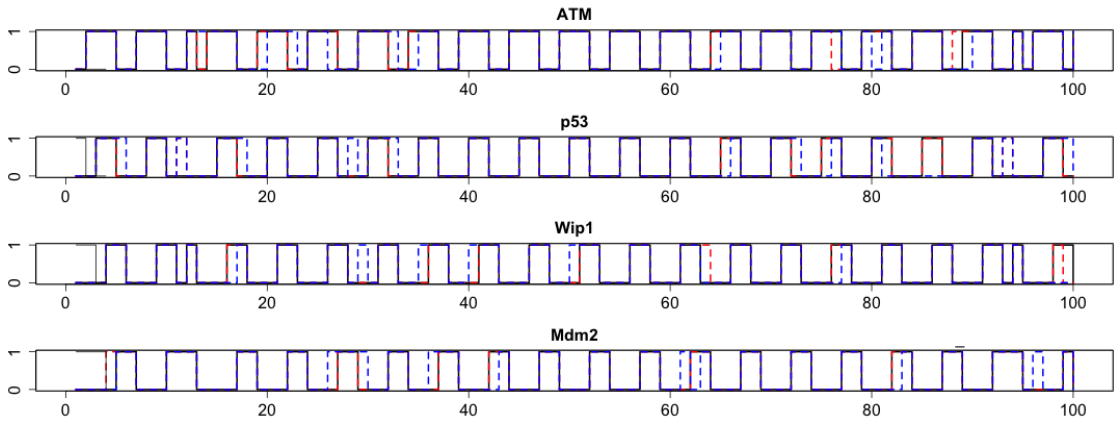

Figure 3.3: Optimal State Estimator Trajectories, p = 0.01, dna dsb = 1

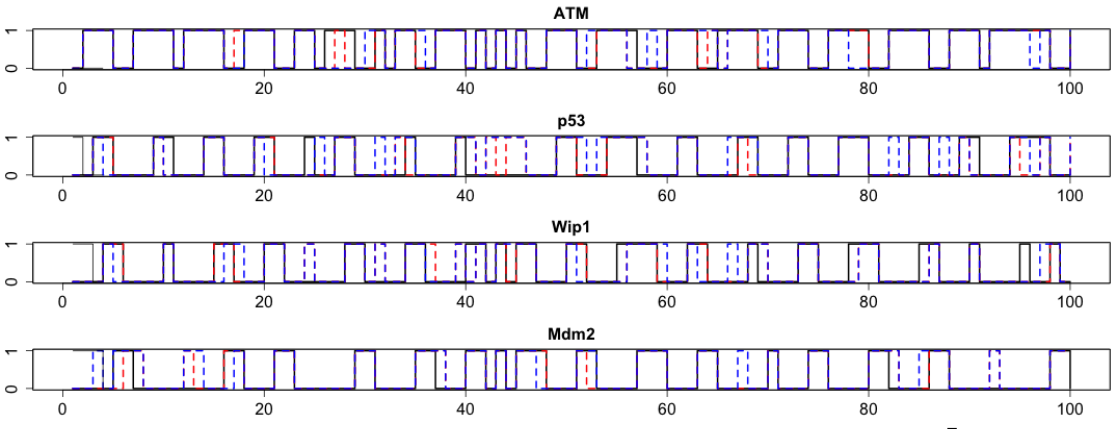

Figure 3.4: Optimal State Estimator Trajectories, p = 0.1, dna dsb = 1

3.2 Boolean Kalman Smoother

3.2.1 Description

The Boolean Kalman Smoother (BKS) [8] implements a backward/forward approach to optimally estimating Partially Observed Boolean Dynamical Systems. The algorithm must be run offline due to the backwards estimation that occurs within the algorithm. The BKS algorithm takes the existing BKF algorithm and runs it both from $k = 1, 2, ..., m-1, m$ and also from $k = m, m-1, ..., 2, 1$, developing an estimation for each time step in both directions, then combining the individual estimates into one final state estimation, which is typically more accurate than the uni-directional Boolean Kalman Filter on its own.

3.2.2 Results

A comparison of the accuracy of the estimates from the uni-directional (forward) Boolean Kalman Filter and the bi-directional Boolean Kalman Smoother is shown in table 3.1. We can see from the table that the Boolean Kalman Smoother outperforms the Boolean Kalman Filter in all noise combinations, both in the presence of a DNA double strand break and not.

A time-series plot of the trajectory estimate (broken down by gene) is shown in figure 3.5 where the black trajectory is $\mathbf{X}_k$, the blue trajectory is the BKF estimate, and the red trajectory is the BKS estimate.

		dna_dsb = 0		***dna_dsb = 1***	
Process Noise p	**Observation noise q**	**BKF**	**BKS**	**BKF**	**BKS**
0.01	0.01	0.96	0.98	0.96	0.98
0.01	0.1	0.93	0.96	0.92	0.96
0.1	0.01	0.95	0.96	0.95	0.96
0.1	0.1	0.66	0.77	0.65	0.77

Table 3.1: Performance of the BKF and the BKS with additive Bernoulli noise

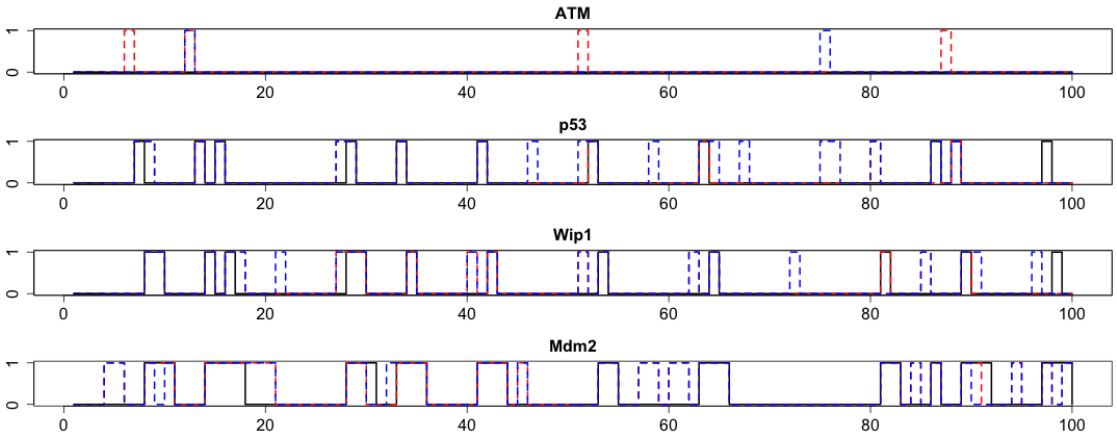

Figure 3.5: Comparison of BKS vs BKF Estimate Trajectories

3.3 Particle Filtering Approximation

3.3.1 Description

The Particle Filtering approximation is a SIS (SMC) approach to approximating the optimal Boolean Kalman Filtering MMSE estimate. The Particle Filter uses SIS methods to approximate a forward transition in time, as opposed to generating a predict matrix, which generates a step forward in time for the Boolean Kalman Filter. The benefit of this approach (which is discussed in depth in section 2.2.1) is that it is significantly faster at generating a close approximation of a smaller network, and allows for computation of higher dimensionality networks in which the BKF might fail (due to hardware limitations) or be computationally inefficient (e.g. taking hours or days to complete).

3.3.2 Results

In this section, the Particle Filtering approximation was implemented on simulated data from the 10-gene mammalian cell cycle regulatory network described in appendix 1. The optimal MMSE Boolean Kalman Filtering algorithm is capable of handling a network of this size (as long as the noise is i.i.d.), however the approximations of the network are quite accurate and run in a fraction of the time. This, again, is one of the major benefits of the Particle Filtering approximation approach, we can utilize it to compute networks very quickly that, if run with the BKF, could take significantly longer to compute. This is shown in the results below. Figure 3.6 shows the difference in computation times for various levels of the N in the particle filter, compared with the optimal estimation of the BKF. Here, we notice that the time of computation for a 10-gene network in the Boolean Kalman Filter takes approximately 140 seconds, or 2.33 minutes. The Boolean Kalman Filter generates a transition matrix of 2^{10} x 2^{10} for this case. Compare this result to the result of

the time-correlated BKF in section 2.2, and we notice that with only 2 additional dimensions the time of computation increases from 2.268 seconds to the 2.33 minutes we see here. This shows the exponential rate of computation time required for additional dimensions. Simply increasing from 8 dimensions to 10 dimensions increases the computation time by around 6100%.

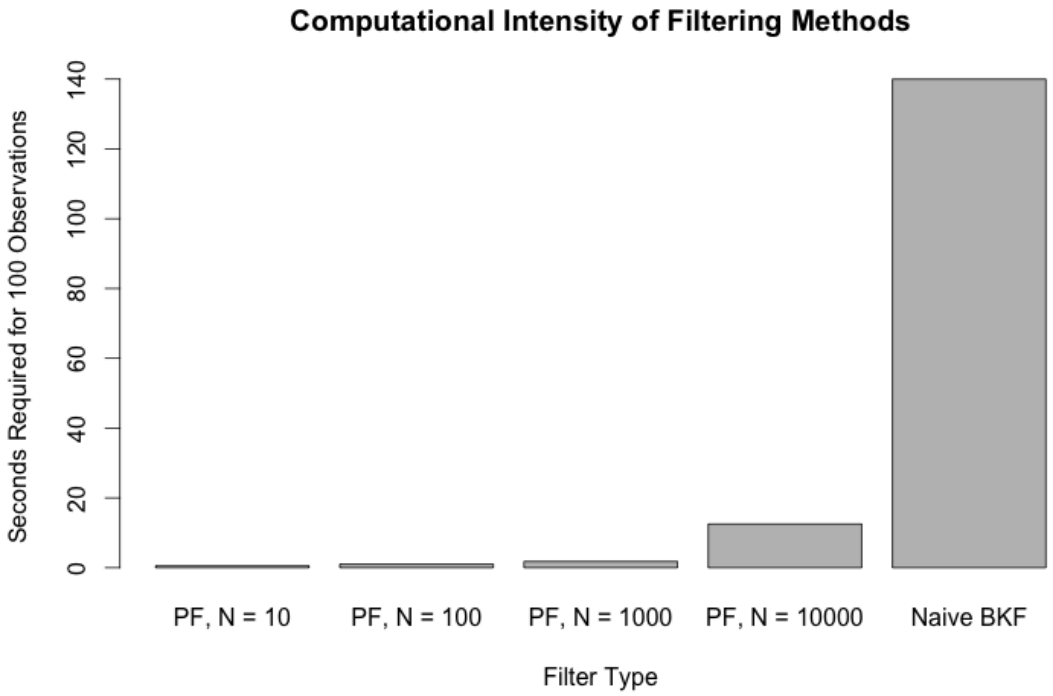

Figure 3.6: Seconds Required for Computation, Various Naive Filter Types, 100 Observations

Shown in table 3.2 are the performance rates for the Particle Filter at various values of N, compared to the optimal estimate from the Boolean Kalman Filter. The results are shown for varying values of process and observation noise. Here, as in previous instances of the naive BKF and PF algorithms, it appears that the observation noise is the main issue when deriving an estimate. Even with higher

process noise, as long as the observation noise is low the estimate (or approximation) turns out to be sufficient.

p	q	**Particle Filter** $N = 10$	**Particle Filter** $N = 100$	**Particle Filter** $N = 1000$	**Particle Filter** $N = 10000$	**Optimal BKF**
0.01	0.01	0.65	0.87	0.90	0.90	0.94
0.01	0.1	0.60	0.80	0.83	0.84	0.85
0.1	0.01	0.32	0.70	0.88	0.90	0.92
0.1	0.1	0.23	0.33	0.36	0.37	0.40

Table 3.2: Performance of the Particle Filter (Varying N) vs. BKF

Another phenomona that is noteworthy here is that the Particle Filtering approximation takes a fraction of the time yet yields approximations that are quite close to the optimal estimate from the Boolean Kalman Filter. This shows the benefits of the PF algorithm addressed earlier in this section, as well as section 2.2, in which it is mentioned that there is a tradeoff between accuracy of the estimate and computational intensity. In some instances, the Particle Filtering approximation might be sufficient, despite a slightly less accurate performance, due to hardware constraints or other factors.

3.4 Multiple Model Adaptive Estimation

3.4.1 Description

MMAE uses a bank of Boolean Kalman Filters running in parallel to approximate various parameters of a model [7]. This process is called systems identification and is very useful in many applications, not limited to genetic networks.

3.4.2 Results

Below is an example of MMAE performing model selection functionality. In this example, network 1 was used to generate data, and networks 2 and 3 were randomly generated connectivity matrices. All 3 networks were fed into the MMAE algorithm and, as shown in figure 3.7, the algorithm determined that network 1 was the proper fit to the data with probability very close to 1.

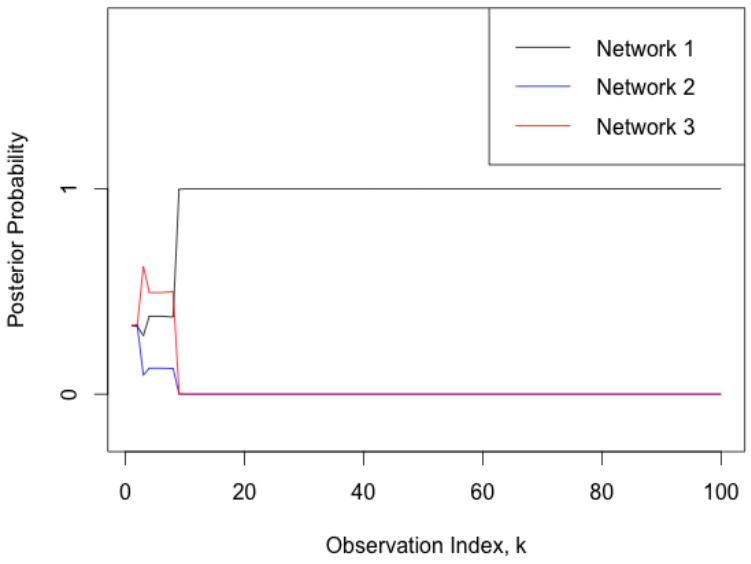

Figure 3.7: Example Output of MMAE Functionality in *BoolFilter*

4. CONCLUSION

4.1 Time-Correlated Boolean Kalman Filter

This document discussed modifications to the naive, i.i.d observation noise Boolean Kalman Filter discussed in [3] to accommodate noise that is correlated in time, specifically by an AR(1) process. This could be described by the new observation process:

$$\mathbf{v}_k = \mathbf{v}_{k-1} \oplus \boldsymbol{\eta}_k \quad \text{(correlated observation model)} \tag{4.1}$$

Where $\boldsymbol{\eta}_k$ is a random Bernoilli perturbation. This modification of the observation process involved augmenting the state space of the Boolean Kalman Filter to allow for the the noise at a given time to be considered a variable of the state space, instead of a random addition to the observation noise process, as in the naive filter. This doubled the number of variables in the state space, since each state variable of the process had its own observation noise value. Once the modifications were made to the state space, the state transition matrix M and the update matrix T could be derived, and the Boolean Kalman Filtering algorithm was run with these new parameters.

Interestingly, as the amount of noise added to the AR(1) process ($\boldsymbol{\eta}_k$) increased, the performance of the naive BKF and the Time-Correlated BKF converged in value. This is due to the fact that with an increasing randomness in the Bernoulli variable added to the AR(1) process ($\boldsymbol{\eta}_k$), the randomness of the (formerly) time correlated noise becomes almost completely random, which is what the naive BKF expects. Therefore, as the parameter value approaches 0.5, the performances of the two filters eventually converge.

4.2 Time-Correlated Boolean Particle Filter

The Naive Boolean Kalman Particle Filter modifications are much the same as the modifications required for the Time-Correlated Boolean Kalman Filter. The Particle Filter is an approximation method, utilizing a sequential Monte-Carlo approach to approximating the optimal solution found by the Boolean Kalman Filter. The modifications were much the same in order to expand the filter to accommodate time-correlated noise. Augmenting the state space of the particle filter and the 'step forward' function of the particles was required to accommodate for the additional state variables. The update step of the algorithm was modified in the same way as the Time-Correlated BKF to accommodate for the additional dimensionality of the added state variables. The particle filter is then run in the same fashion as the naive sense.

The SMC approach to approximation of the optimal solution allows for significantly higher dimensionality. In a test of a 10 gene time-correlated genetic network, the T-C BKF algorithm failed to run entirely, whereas the particle filter did manage to approximate a solution, albeit extremely slowly. However, herein lies the value of the approximation algorithm, it was at least able to derive a solution, whereas the optimal algorithm crashed and failed to run entirely due to the dimensionality.

4.3 R Package: BoolFilter

This document also addressed a new R package, to be called *BoolFilter* that will allow for the easy and free distribution of the work done in the Genomic Signal Processing Lab at Texas A&M on the subject of POBDS. The package contains a means of inputting boolean networks in an easy-to-read boolean format, and contains numerous functions to work with them, including Boolean Kalman Filtering [3], Boolean Particle Filtering [4], and MMAE [7] algorithms.

REFERENCES

[1] Eric Batchelor, Alexander Loewer, and Galit Lahav. The ups and downs of p53: understanding protein dynamics in single cells. *Nature Reviews Cancer*, 9(5):371–377, 2009.

[2] Leonard E Baum, John Alonzo Eagon, et al. An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bulletin of the American Mathematical Society*, 73(3):360–363, 1967.

[3] Ulisses Braga-Neto. Optimal state estimation for boolean dynamical systems. *2011 45th Asilomar Conference on Signals, Systems and Computers*, pages 1050–1054, 2011.

[4] Ulisses Braga-Neto. Particle filtering approach to state estimation in boolean dynamical systems. *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, pages 81–84, 2013.

[5] Adrien Fauré, Aurélien Naldi, Claudine Chaouiya, and Denis Thieffry. Dynamical analysis of a generic boolean model for the control of the mammalian cell cycle. *Bioinformatics*, 22(14):e124–e131, 2006.

[6] Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *Radar and Signal Processing, IEEE Proceedings F*, 140(2):107–113, 1993.

[7] Mahdi Imani and Ulisses Braga-Neto. Optimal gene regulatory network inference using the boolean kalman filter and multiple model adaptive estimation.

2015 49th Asilomar Conference on Signals, Systems and Computers, pages 423–427, 2015.

[8] Mahdi Imani and Ulisses Braga-Neto. Optimal state estimation for boolean dynamical systems using a boolean kalman smoother. *2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 972–976, 2015.

[9] Andrew H Jazwinski. *Stochastic Processes and Filtering Theory*. Courier Corporation, Mineola, NY, 2007.

[10] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[11] Stuart A Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22(3):437–467, 1969.

[12] Stuart A. Kauffman. *The Origins of Order: Self Organization and Selection in Evolution*. Oxford University Press, New York, NY, 1993.

[13] Ritwik K Layek, Aniruddha Datta, and Edward R Dougherty. From biological pathways to regulatory networks. *Molecular BioSystems*, 7(3):843–851, 2011.

[14] Ping Li, Roger Goodall, and Visakan Kadirkamanathan. Estimation of parameters in a linear state space model using a rao-blackwellised particle filter. *Control Theory and Applications, IEEE Proceedings-*, 151(6):727–738, 2004.

[15] Ranadip Pal, Aniruddha Datta, and Edward R Dougherty. Optimal infinite-horizon control for probabilistic boolean networks. *Signal Processing, IEEE Transactions on*, 54(6):2375–2387, 2006.

[16] Ilya Shmulevich, Edward R Dougherty, Seungchan Kim, and Wei Zhang. Probabilistic boolean networks: a rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, 18(2):261–274, 2002.

[17] Ilya Shmulevich, Edward R Dougherty, and Wei Zhang. From boolean to probabilistic boolean networks as models of genetic regulatory networks. *Proceedings of the IEEE*, 90(11):1778–1792, 2002.

[18] Ilya Shmulevich, Edward R Dougherty, and Wei Zhang. Gene perturbation and intervention in probabilistic boolean networks. *Bioinformatics*, 18(10):1319–1331, 2002.

[19] Rudolph Van Der Merwe. *Sigma-point Kalman filters for probabilistic inference in dynamic state-space models*. PhD thesis, Oregon Health & Science University, 2004.

[20] Robert Weinberg. *The Biology of Cancer*. Garland Science, New York, NY, 2013.

APPENDIX 1: Mammalian Cell Cycle Network

A gene connectivity diagram for the Mammalian Cell Cycle network is shown below:

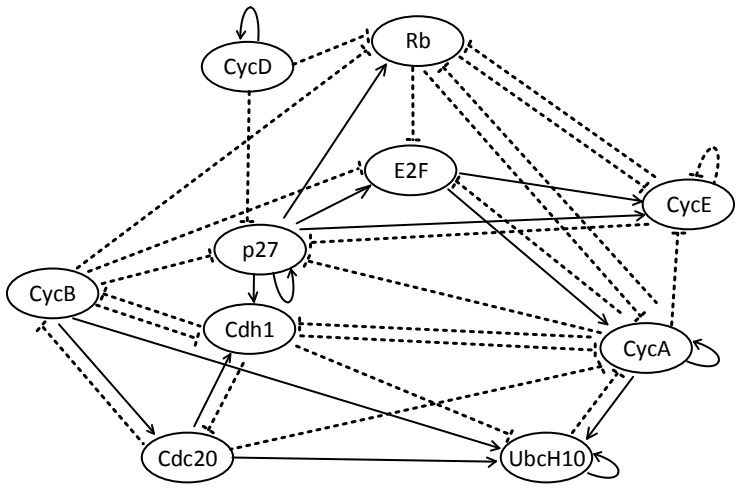

Figure 4.1: Mammalian Cell Cycle

With a connectivity matrix as follows:

$$Q = \begin{array}{c} \\ CycD \\ Rb \\ p27 \\ E2F \\ CycE \\ CycA \\ Cdc20 \\ Cdh1 \\ UbcH10 \\ CycB \end{array} \begin{array}{c} \begin{array}{cccccccccc} CycD & Rb & p27 & E2F & CycE & CycA & Cdc20 & Cdh1 & UbcH10 & CycB \end{array} \\ \begin{pmatrix} 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & -1 & -1 & 1 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & -1 & -1 & -1 & 0 & 0 & 1 & -1 & 1 & 0 \end{pmatrix} \end{array}$$